City2City

Allan Zhou
EECS
University of California, Berkeley

YiDing Jiang
EECS
University of California, Berkeley

Abstract

In this work we explore methods of style transfer between images of different cities. Given a picture taken in one city, we want to generate an image in the style of another city while retaining the semantics of the original image. Without having access to paired data for this problem, we try different approaches to accomplish this task. We investigate using unpaired image-to-image translation models. We also leverage depth data corresponding to images at training time to do image-to-image translation at test time. We assembled image (and depth) datasets from cities such as Paris, Manhattan, and Venice for training each model.

1 Introduction

There are many cities around the world, which share many of the same ingredients – buildings, shops, roads, vehicles and humans. However, many people can effortlessly tell Paris apart from San Francisco or New York when looking at photographs taken in each city, despite the semantic similarities. This is because these individual components have vastly diverse styles and appearances going from one city to another and these appearances are rich with information about the identities of the cities. Indeed, previous work[1] has focused on finding distinctive visual elements between cities. The goal of our project is to utilize these stylistic differences between images from different cities in order to translate images taken from one city into the style of another.

(a) New York City

(b) Paris

(c) Venice

Figure 1: Cities have distinct styles, which we can see through photos.

We hypothesize that most cities share semantic latent representations in the same way that horses and zebras or maps and satellite images share the same latent representation. We want to train a neural network that learns and utilizes this semantic representation to translate images of one city into the style of another.

A natural first approach is to try neural style transfer [2]. We use an input image from Manhattan as ”content,” and an image from Burano, Venice as the ”style.” As the result in Figure 2 shows, this can produce interesting artistic effects but the result is not very photo-realistic. In this paper we will instead explore using conditional Generative Adversasrial Networks [5, 3] to accomplish this task. CycleGAN[6] and pix2pix[4] are two recent approaches that have achieved considerable success in

the domain of style transfer and image to image translation and we will use them to translate images from one city to another, both directly and through the intermediate representation of depth maps.

Figure 2: The result of applying Neural Style Transfer[2] to city translation. Here the Style photo is from Venice, while the Content photo is from Manhattan. The result has an interesting artistic effect, but is not very photo-realistic.

2 Approach

2.1 Direct Translation from Unpaired Data

CycleGAN[6] is a deep architecture that achieves unsupervised image to image translation. The neural network consists of two generators and two discriminators and the objective is to translate images from one domain to the other and back with minimum difference (cycle consistency), akin to an autoencoder. The discriminator is used enforce that the latent from one domain is as close to the distribution of the the other domain as possible. Optimizing both GANs simultaneously and enforcing cycle consistency result in learning mappings in both directions.

One of the major advantage of using CycleGAN is that it does not need the training images to be paired as the discriminator learns the quality metric for the translated images. This is desirable for tasks such as translating images of cities because it is nearly impossible to collect thousands of images from both cities that are pairwise pose-aligned, but there is an abundance of unpaired photos of cities online.

2.2 Leveraging Depth Data

Figure 3: An example Google Street View depth map (left) and its corresponding panorama (right).

Previous work [1] on analyzing the distinctive elements of a city made use of Google Street View [1] imagery. Google Street View allows us to access an enormous quantity of images in each city, and is not biased towards famous city landmarks or tourist attractions the way that Flickr or Google Image Search are. Crucially, Google Street View imagery is accompanied by **depth maps**. This means that although we cannot obtain paired images between two cities, we can get paired data in the form of (image, depth map). In this approach we used this paired data to train image translation models, specifically using the Pix2Pix[4] architecture.

[1] `https://www.google.com/streetview/`

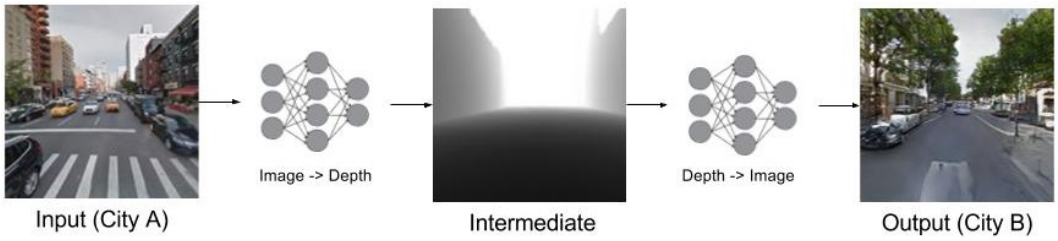

Figure 4: Our city translation process using depth maps. We train 2 models, one that translates an image to a depth map, and another which translates a depth map to an image. Crucially, the two models are trained on data from **different** cities. Combining the two models accomplishes our city translation goal. Note that although the training proces requires depth data, no depth data is needed at evaluation, it is just an intermediate product.

Say we train a model using paired (image, depth map) data collected in City B to translate from depth map to image. We could then plug in a depth map from City A to this model, and the generated image would be in the style of City B, while still containing the original scene structure captured by the depth map. However, this process is not true city to city image translation since we are providing depth data as input for the translation. To resolve the issue of needing always depth maps, we can instead train a separate model to **generate** the depth map from the input image.

Specifically, say we want to translate images from City A into the style of City B. We can train a model that transforms depth maps to images using data from City B, as described previously. In a similar fashion, we can separately train a model that transforms images to depth maps using paired data from City A. We can then chain the former model after the latter model, creating a combined model that translates an image from City A into an image in the style of City B. This whole process is illustrated in Figure 4. Importantly, we use the (image, depth map) paired data during training of each model, but we do not need the depth data at test time–the combined model translates directly from images to images.

3 Data Collection

3.1 Unpaired Images

For the unpaired translation approach, we used Google Street View images from [1] that are taken randomly over the target cities and cropped to show only a portion of the objects in the scene such as the building. Training models with this dataset did not work as we intended as we found that the images remain largely unchanged even after 200 epochs. We found that although the number of images is large, the data are not very suitable for the translation we want to perform. The images are largely taken from generic residential areas or streets that do not contain characteristics particular to the cities. It is even hard for humans to tell the origin of the images. In other words, the training signal is very weak and the network is unable to learn any meaningful signal from the training data. An alternative interpretation would be that the images all look the same so the best option for the generator is to slightly adjust the brightness or simply keep the input unchanged. While supervised learning with deep architecture is robust to noisy training data, we suspect it is not the case for unsupervised generative model. Additionally, this particular dataset was not accompanied by depth data so we could apply the depth map approach here.

As such, we decided to recollect a new dataset for the unpaired approach. From the previous experiment we learned that randomly collected images do not perform well. Ideal training data should contain very unique characteristics of the city – the kind of images that makes people say ”oh, this is definitely Paris”. We decided to leverage the vast amount of images available on Google image and Flickr because these images are much more biased towards images that are representative of the cities because unlike street views, photos do not have images of generic residential areas because they are simply not interesting enough to incentivize photographer to take pictures; therefore, there is strong correlation between these photos and the characteristics of a city. In addition, we wanted cities that have very unique styles so make signal even stronger. In the end, we used Paris, which

Figure 5: Some training data. Top: Burano, Bottom: Paris

is rich with buildings from Renaissance, Baroque and Neoclassicism, and Burano Island of Venice, which is one of the most colorful cities in the world.

We scraped 1500 images from each city from Google Image and Flickr with Selenium and Flickr API. While the data have higher qualities, they are also mixed with images of indoor or night which are not ideal. Therefore, we collected more than 2000 images for both cities and manually cleaned the dataset.

3.2 Street View Depth Data

Google Street View data comes as spherical panoramas in equirectangular projection (see Figure 3), which we needed to scrape and process. We used the library GSVPano[2], which displays the Google Street View panorama nearest to the geographic coordinates (latitude and longitude) that you provide, but we modified it to access panoramas using unique "Panorama IDs" instead. We also used GSVPanoDepth [3], which pulls down the corresponding depth map for a Street View panorama.

Figure 6: Our Street View scraper at work in Manhattan. Each dot marks the location where a panorama has already been scraped.

For any Street View panorama (specified by ID), we can find its geographic neighbor panoramas. Thus the panoramas form a graph structure, and we can run Depth First Search (DFS) to densely scrape the panoramas in a city. We implemented this dense scraper, starting from a central location

[2] https://github.com/spite/GSVPano.js
[3] https://github.com/proog128/GSVPanoDepth.js

and using DFS to scrape all Street View panoramas within a specified radius (we used a radius of 1500 meters). Figure 6 shows our dense scraper at work in Manhattan, where the red dots show locations of panoramas it has scraped so far.

The data arrives as in the form of a spherical panorama, and the equirectangular projection inevitably causes distortion in the image. We further process the panoramas to create undistorted perspective projection images. The fastest way we found of doing this was to use the 3D rendering library ThreeJS[4]. In ThreeJS, for each panorama we created a scene with a camera located **inside** of a sphere. We textured the inside of the sphere with the spherical panorama (or depth map), then set the scene camera to some viewing angle to render the perspective view. For each panorama or depth map, we rendered perspective views in the 6 directions up, down, forward, left, right, and back. An alternative approach, which we found was simpler but slower, was to use a utility that processes panoramas into cubemaps.

As Figure 3 shows, the depth maps were fairly low resolution and did not do a good job of capturing details such as cars or trees. The depth maps mainly captured buildings. Unfortunately, the Street View depth data was often so inaccurate that they did not even capture the building properly. Thus, we manually filtered our collected data by viewing each panorama/depth map pair side-by-side and then deleting pairs where the depth map was highly inaccurate.

4 Experiments and Results

4.1 Unpaired Approach

We used a standard CycleGAN [5] architecture with 9 ResNet Block and trained the network for 200 epochs using the default hyperparameters. The images are center cropped and resized to 128 by 128 pixels. The results can be found in our demo [6].

Figure 7: CycleGAN result on 128x128 images. Top: Paris to Venice and then back. Bottom: Venice to Paris and then back

As you can see, CycleGAN outperforms naive style transferring by a big margin. For example, CycleGAN demonstrates some understanding of semantic structure. It correctly transforms the classical buildings of Paris to colorful buildings of Burano. It also learns that Paris has taller buildings

[4] https://threejs.org/

[5] https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix

[6] https://allanyangzhou.github.io/city2city/results/paris_venice_cyclegan/index.html

compared to Burano as it "grows" the buildings of Burano taller and erases the top portions of the buildings of Paris. Another interesting observation is that the network also translates the streets of Paris to canals and vice versa because both often appear in the center of the photos. It does not however successfully transform the cars to boats perhaps because the geometric change is too large. In addition, the network also tries to add trees going from Burano to Paris and to transform trees into buildings in the other direction.

Note by observing what changes from one city to another reveals what characteristics are unique to the respective cities. Aside from the obvious ones such as the building changes, some changes are less obvious. For example, the change from trees to building is not immediately obvious to humans because trees are usually not the first thing people notice in an photo. We did not realize that until CycleGAN starts to add and remove trees from the photos.

We have also applied the network to frames of videos. Unlike pix2pix, we found the the output of CycleGAN is smooth with respect to input as in there are no sudden changes of colors, and the video looks very coherent. The video can be found in your website as well.

4.1.1 Partial Scale Invariance

It is known that CycleGAN excels at changing the textures and appearances but less so at changing geometries and shapes. While this trait can be a limitation, we can also use it to our advantage. As we know, textures and appearances are less sensitive to scales compared to geometry, which means textures learned at a smaller scale can be used at reasonably larger scale without losing photo realism. On the other hand, convolutional neural network can be applied to image of any sizes in principle; therefore, we applied the CycleGAN trained with 128 by 128 images to images of 256 by 256 images which is 4 times larger than the training data. To our surprise, the translation worked very well.

Figure 8: CycleGAN result on 256x256 images. Top: Paris to Venice and then back. Bottom: Venice to Paris and then back

This is valuable because the training time of 128x128 (4 days / 200 epochs on K80) is at least 3 times faster than that of 256x256 (14 days / 200 epochs on K80) which means it is possible to train

a CycleGAN at a smaller scale and apply it to images of higher resolution. Of course, we suspect the effectiveness varies with the data. The results of 256x256 can be found in our demo [7].

4.2 Depth Map Approach

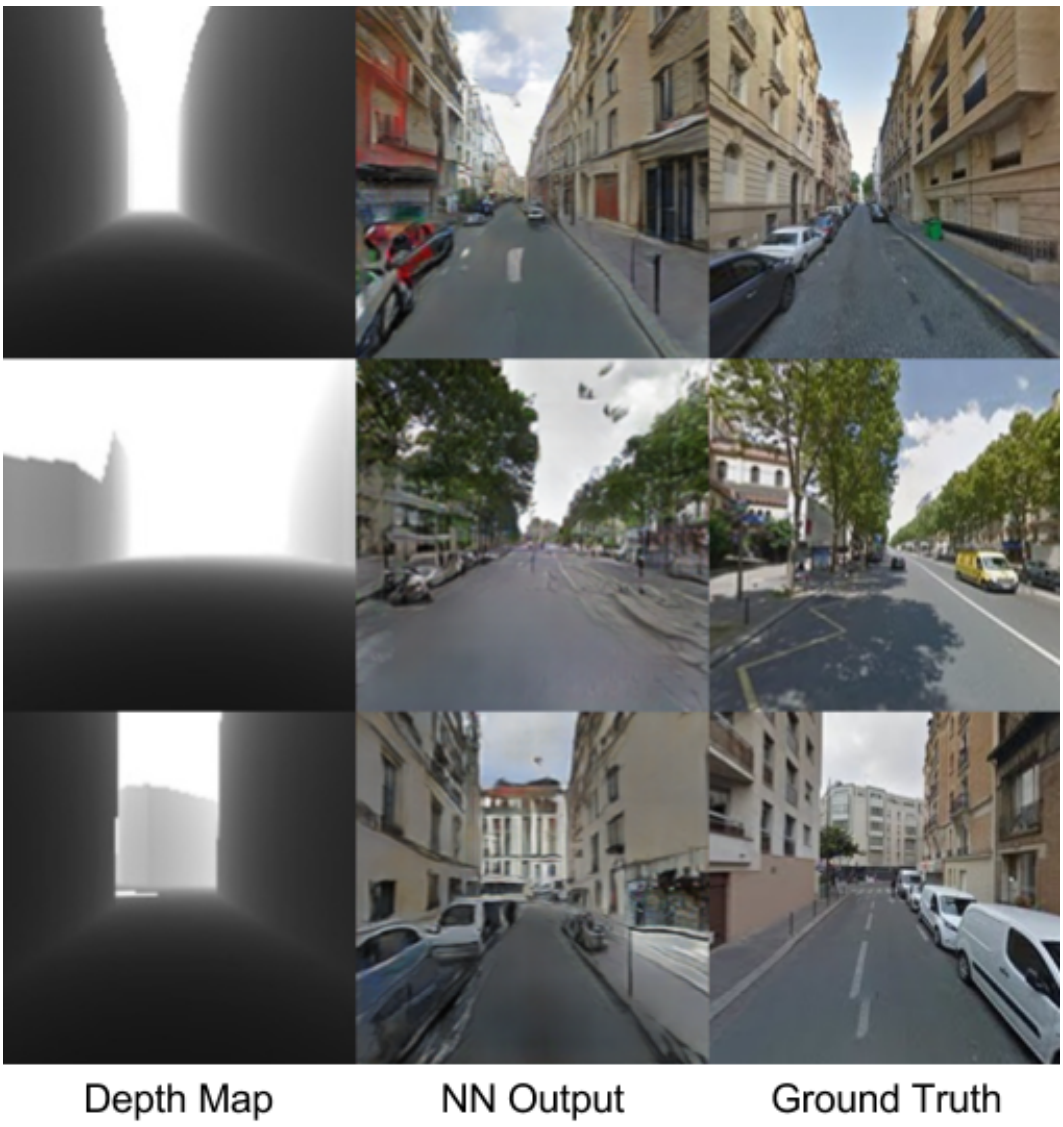

Figure 9: Test results of our depth to image translation model trained on Street View images from Paris. Each row is a different test sample. More comprehensive results on random test data from both Paris and Manhattan can be found at the website.

We used our paired (depth map, image) data from each city to train two types of models. One type of model translated from depth map to image, while the other worked in the opposite direction. We used the Pix2Pix[4] architecture to train these models using data collected in Paris and Manhattan. For each city we had around 1200 training pairs, and each image was 264×264 after all the processing. We found that the default training hyperparameters with the UNet 256 architecture worked fairly well for the depth map to image translation. Our image to depth map translation was similar, but

[7] https://allanyangzhou.github.io/city2city/results/paris_venice_cyclegan_256/test_latest/index.html

we set λ, the weight on the L1 loss, to 50 instead of the default 10. Each model was trained for 200 epochs.

Figure 9 shows some example outputs from the depth map to image models on test data. The results show the model is able to handle different types of inputs, such as an "endless" street (1st row) vs a "dead-end" street (3rd row). Our website contains more complete results on random test data for both the models trained in Paris[8] and Manhattan[9].

Some example test results of the end-to-end process of translating images of Manhattan into the Paris style are shown in Figure 10, and more complete results on random test data are shown at the project webpage[10].

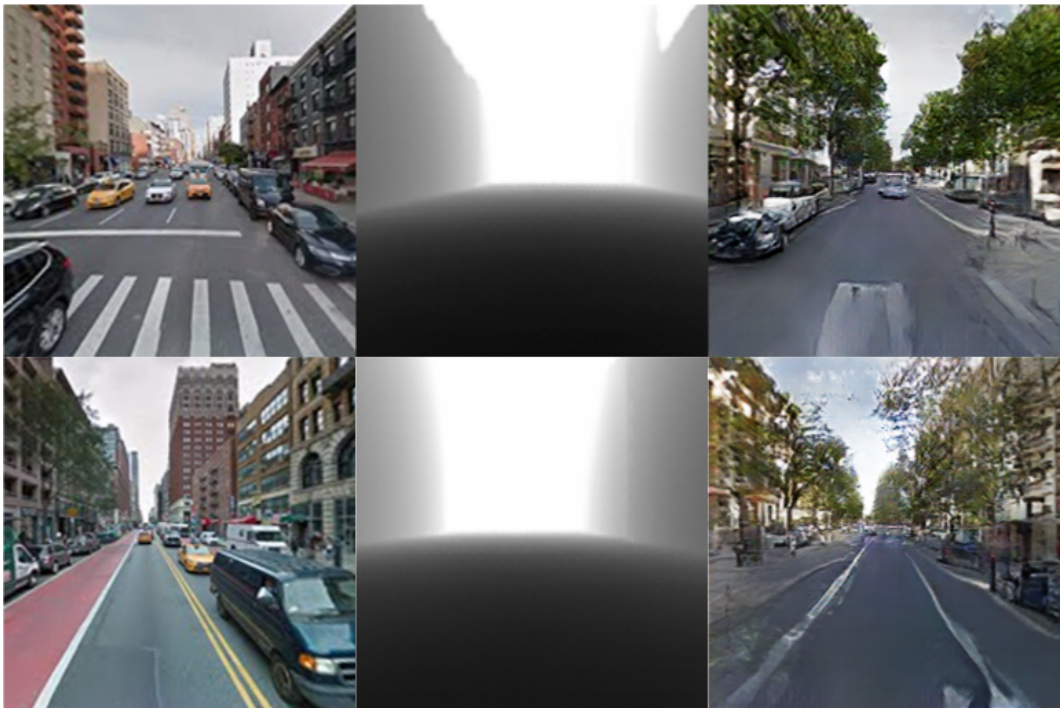

Figure 10: Results of image translation from Manhattan to Paris using the method described in Section 2.2. The left column shows the input images, from Manhattan. The right column shows the final model output, in the style of Paris. The middle column shows the intermediate depth map produced by the image to depth model. More results on random test data are available at the project webpage.

5 Conclusion

In this project, we explored multiple approaches for transferring the style of one city onto an image taken in another city. We tried an unpaired approach using a CycleGAN model. We found that by carefully preparing a datset with images that are very distinct between the two cities, the trained CycleGAN model can produce very good results. In another approach, we leveraged depth data corresponding to Google Street View imagery to train two paired models, and then combined the two models together to achieve city-to-city image translation. This approach has access to much more abundant data and can work even with relatively "boring" images, where the unpaired approach does not.

[8] https://allanyangzhou.github.io/city2city/results/depth2gsv_paris

[9] https://allanyangzhou.github.io/city2city/results/depth2gsv_manhattan

[10] https://allanyangzhou.github.io/city2city/results/manhattan2depth2paris/

References

[1] Carl Doersch, Saurabh Singh, Abhinav Gupta, Josef Sivic, and Alexei Efros. What makes paris look like paris? *ACM Transactions on Graphics*, 31(4), 2012.

[2] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015.

[3] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[4] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint arXiv:1611.07004*, 2016.

[5] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[6] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint arXiv:1703.10593*, 2017.