ART AND MACHINE LEARNING
CMU 2019 SPRING
PROJECT 2

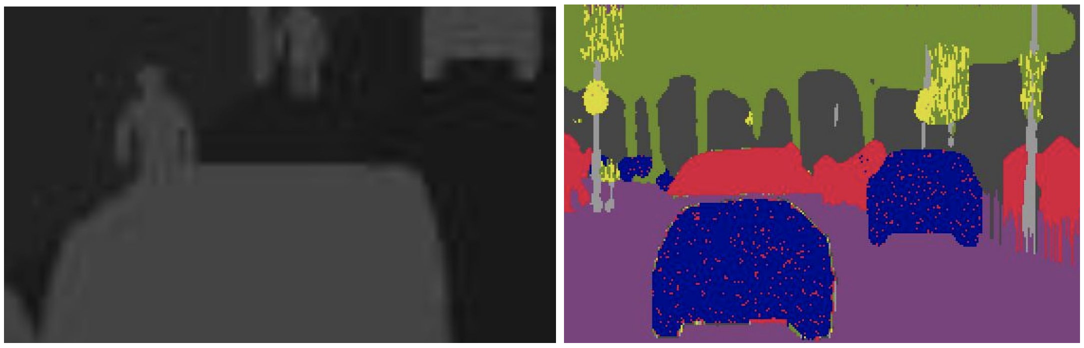
THE CONSTANT FLUX

Group 3
Wooshik Kim
Yejin Lee
Mimi Wang
Shengzhi Wu

<The Wandering City>

Concept: In *The Wandering City*, we attempt to create an imaginary city reflecting our constantly moving life. In our times, people are continuously moving to different countries and cities. Industrializing and globalizing city, old buildings are constantly replaced by new buildings and the city loses its own identity. To emphasize the mobility pattern, we replace mobile vehicles to mobile buildings to emphasize the constant flux in our lives, not able to stay in one place.

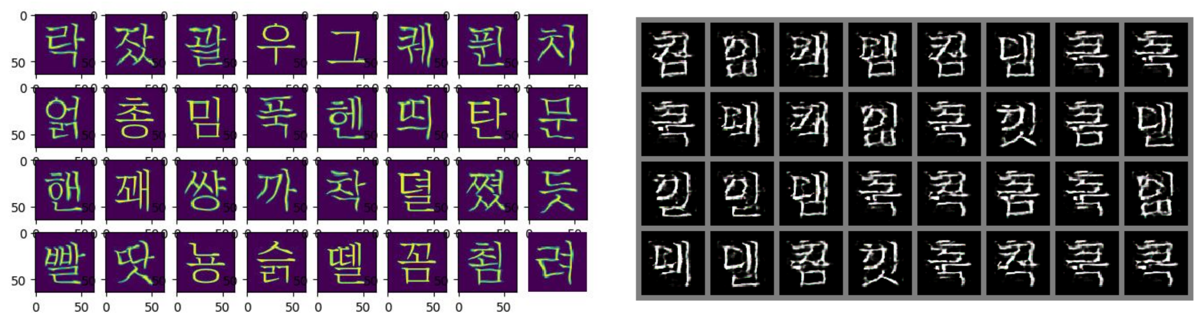
Technique: The techniques used for Cityscape generator is vanilla pix2pixHD. For this project creating our own labels was the most critical bottleneck for this part of the project. There were technical difficulties with creating our own labels. The biggest problem was the edge blurring.



Edge Blurring and Noisy Labels

We need to have a single value for a single object, but these blurred images gave us some troubles. We could change pixel values that are approximately the same to one value, but sometimes the values were noisy as you can see from above. At this point, we didn't have enough time to implement solutions, so the results are not as great.

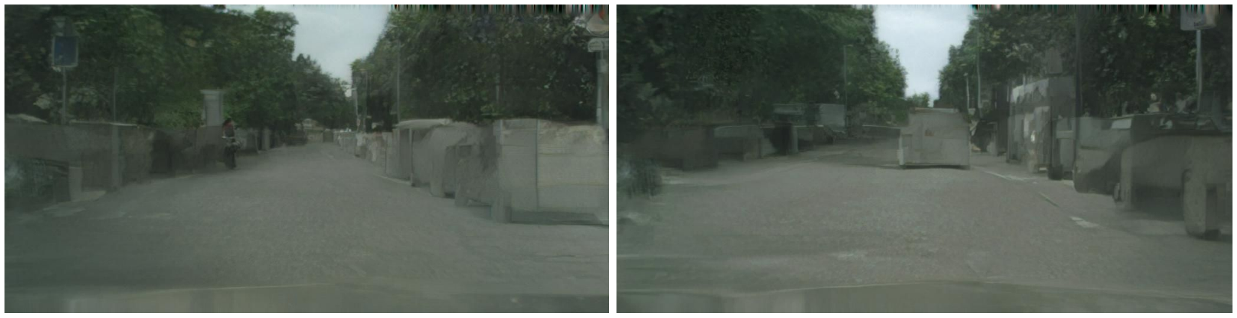
Process:
In order to understand how pix2pix generative adversarial network technique works, we started experimenting with vanilla pix2pix from the professor's repository. The first experiment was to create a new letter from an existing one. We knew creating new letters based on Korean characters would be most promising for multiple reasons. For Korean letters, hangul, vowels, and consonants are combined to form a single character. And this single character is the same size for more all the combinations (which is in thousands). So it is easy to create many image data of the same size. Like the following:



Training Set and Generated Set

We created new letters based on ~2000 training set. The results were interesting. We could find some letters that do not exist in the actual Korean characters, but still similar to it. It was also a good sign for us since it was working as we expected it to.

Next, we moved to pix2pixHD repository. This was more promising one of using pix2pix with cityscape labels to create a generated city. One of the ideas that came up was to generate buildings instead of where cars are supposed to be. We wanted to experiment and express our philosophy about a city and the inhabitants. The first few results are shown below.

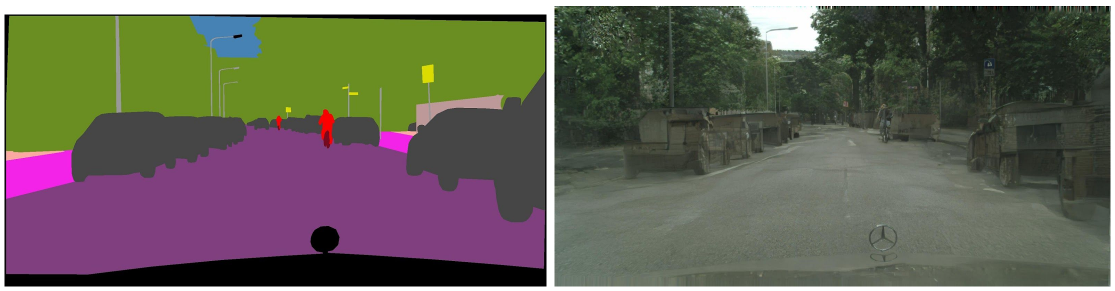


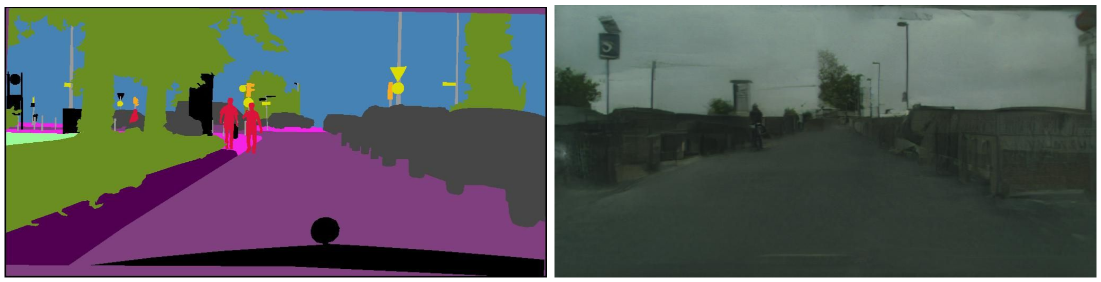
Buildings Instead of Cars and Trees Instead of Buildings

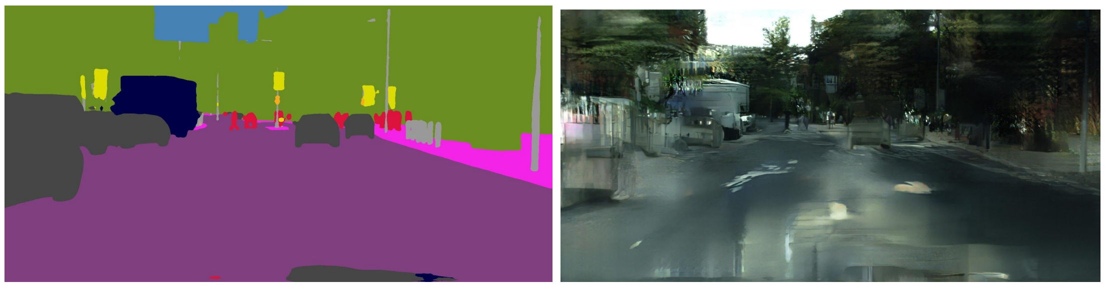
Instead of cars, we see the resemblance of buildings here and there: few panes, some windows, but mostly greyish walls. We wanted to explore this further. We wanted to create a video.

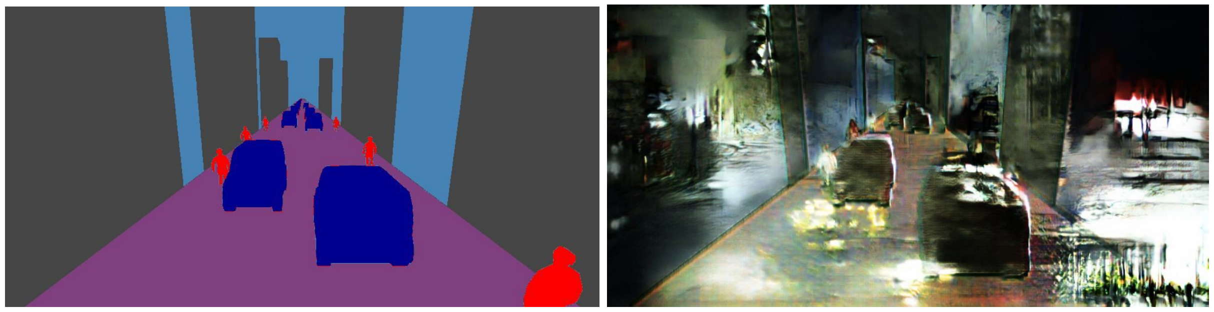
To do so, we had to create our own labels. This was done in two methods. One, we created a rather static sequence of pictures where one car was moving with Adobe Photoshop. Two, we created a more dynamic sequence with Unity Engine.

Creating our own label with Photoshop and Unity

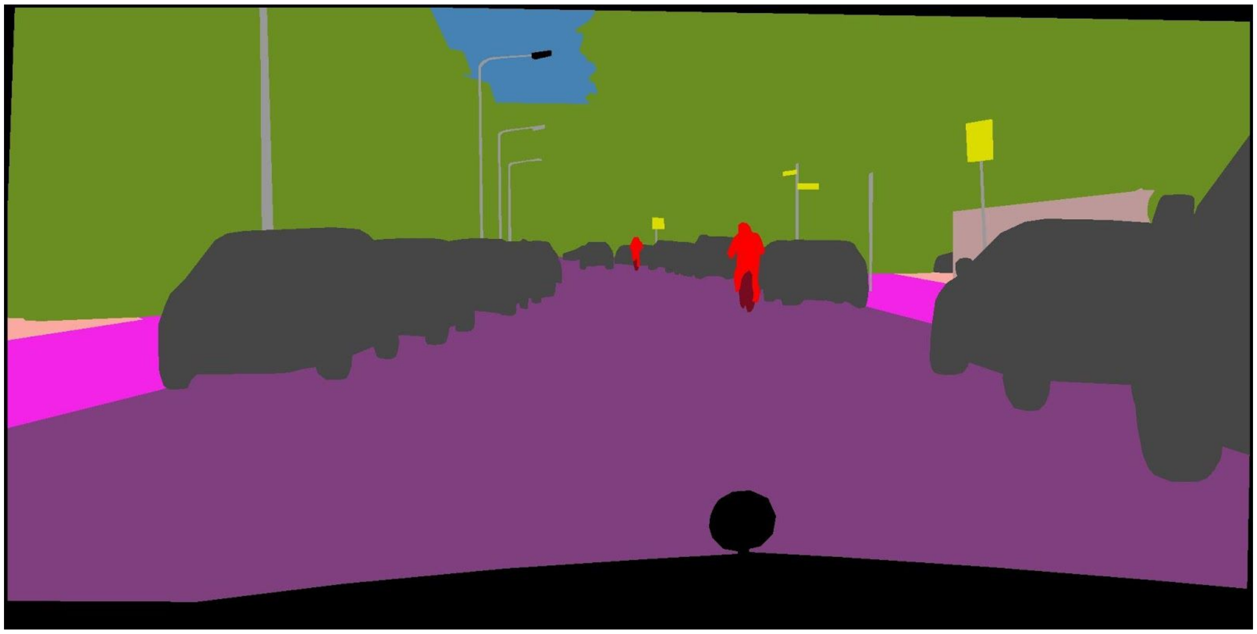

Test1: Cars to Buildings and Buildings to Trees

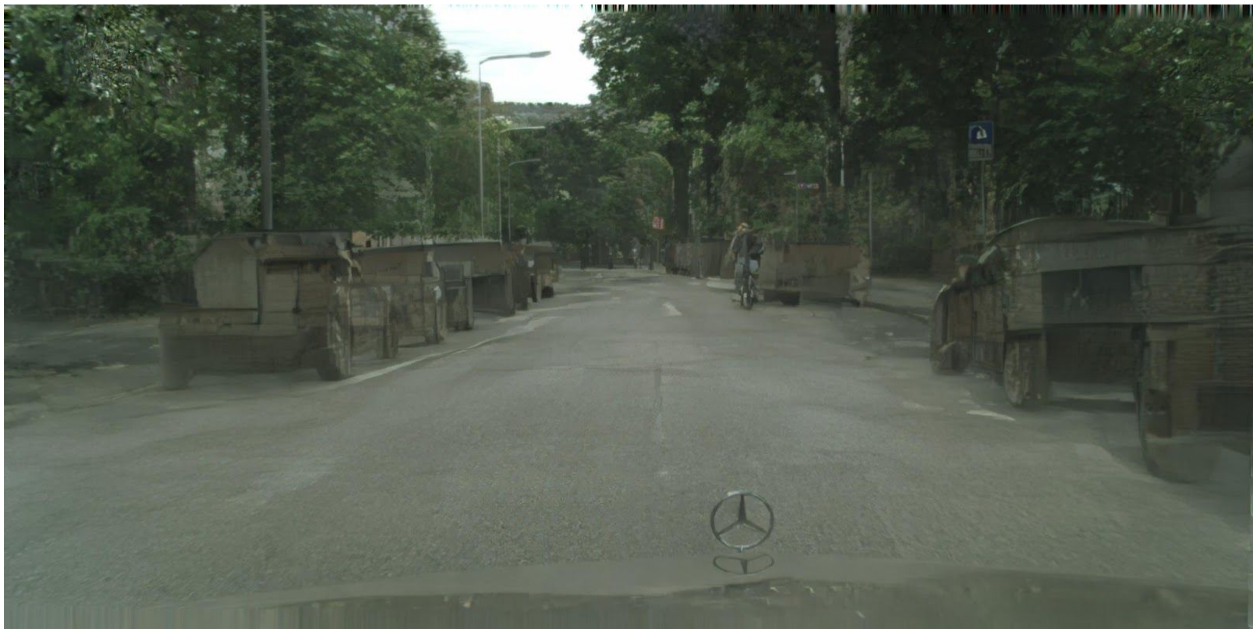

Test2: Cars to Buildings and Buildings to Sky

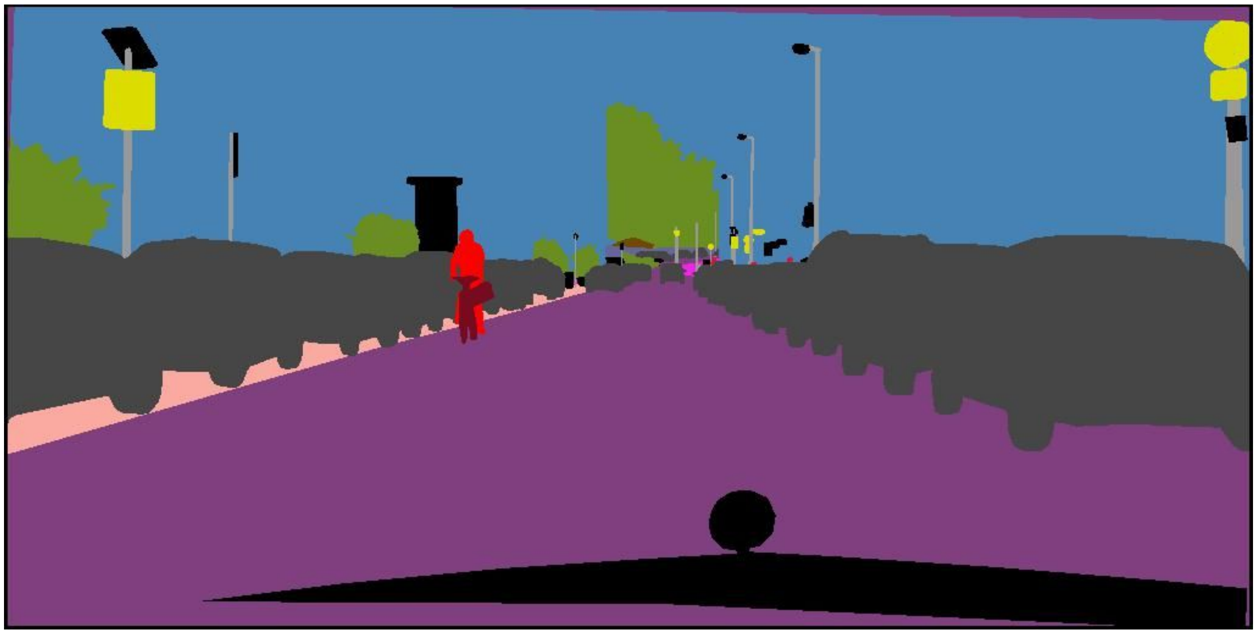

Test3: Own Label with Photoshop (no instance)

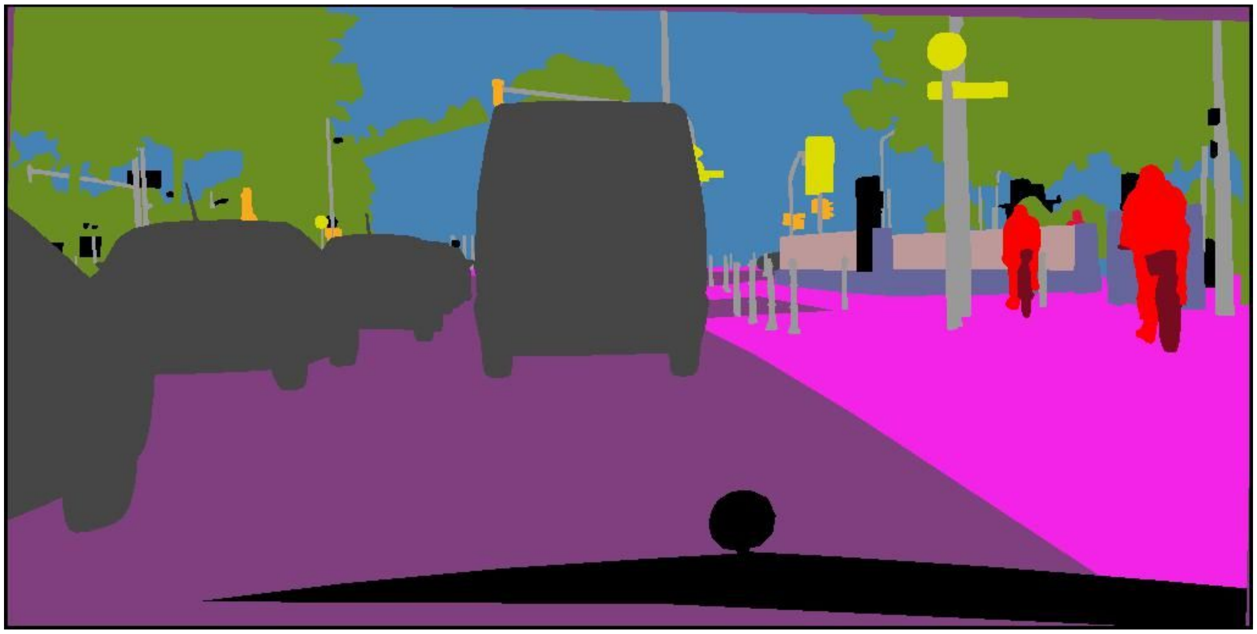

Test4: Own Label with unity

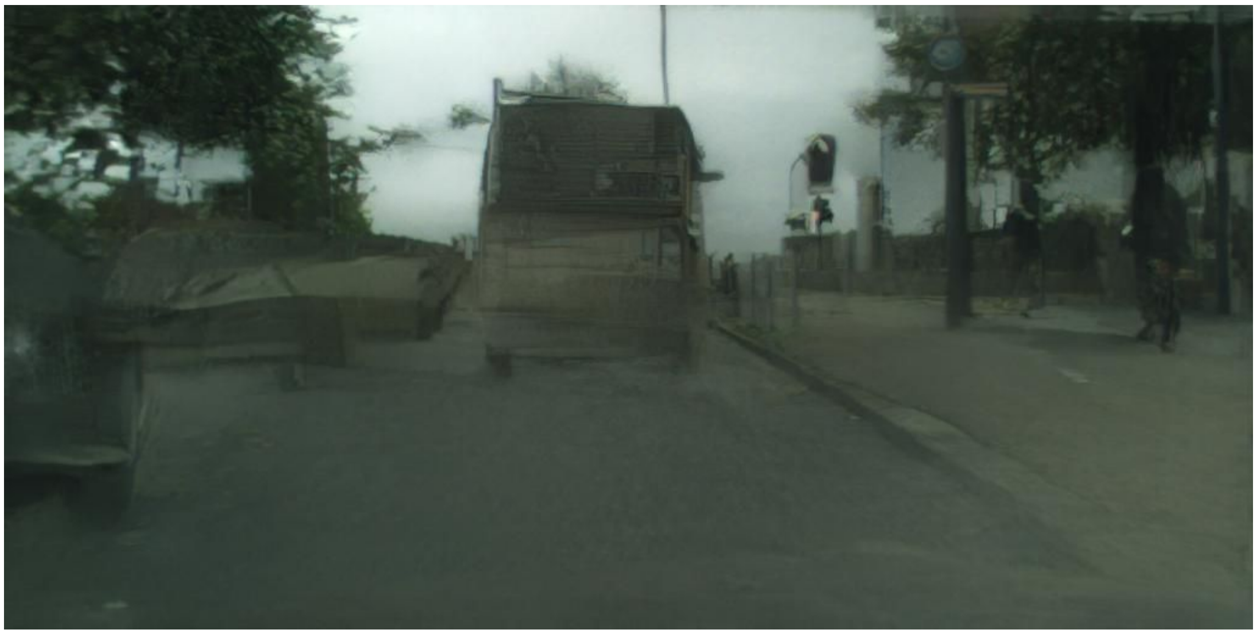
Best Result









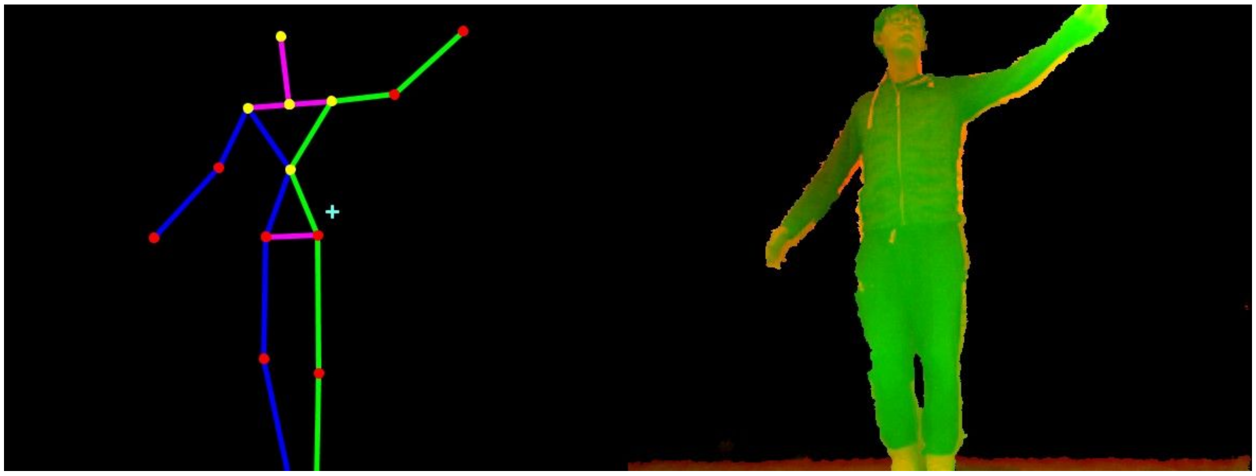


Reflection:
It is still hard to make the moving buildings clearly visible. If we could develop this more, we want to create an animated 3D cityscape with the mobile buildings and make them more visible.

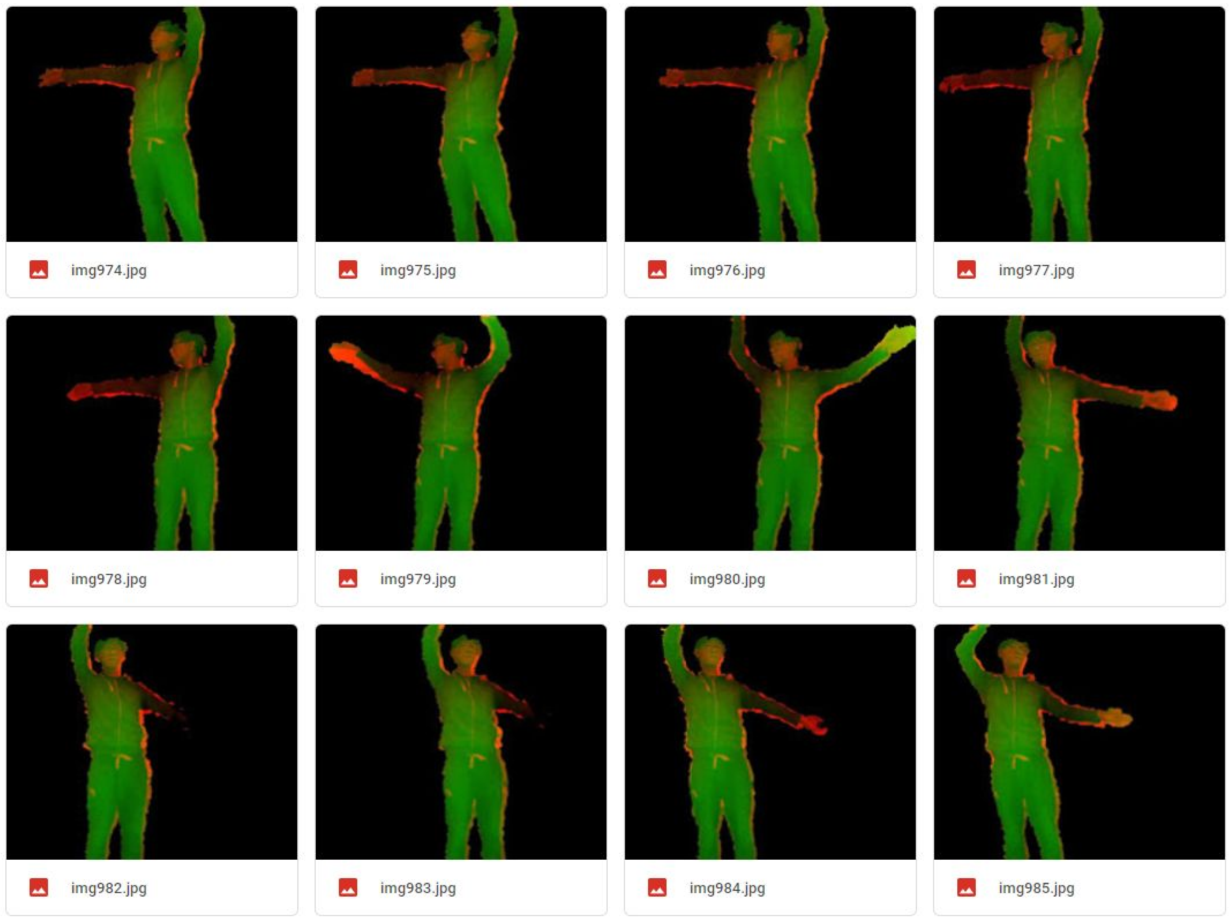
<The Ephemeral Ego>

Kinect Depth Point Cloud Testing

Collecting Data
The process starts from gathering paired data using Kinect Sensor and Processing program. I used processing code to detect skeleton and collect the depth data as well as the brightness of the pixels. Then I encoded the depth data into a image’s red color channel, and the brightness data into the green channel and leave the blue channel empty(0). Simultaneously, the skeleton data is generated on the left, and the skeleton is colored with blue, green, and yellow colors to differentiate the different parts of the body.

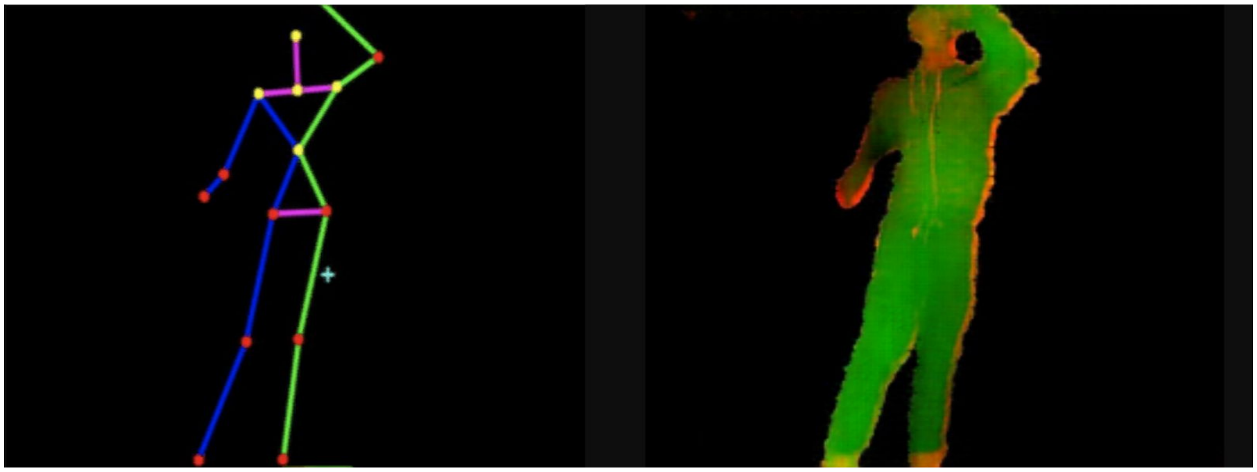


With over 1,000 paired images, I was able to have a dataset that is large enough to execute the Pix2Pix ML algorithm.



Training with Pix2Pix
I used the Tensorflow Pix2Pix algorithm to train with my dataset, and the initial training used 900+ images with 50 epochs, and here is the training results. You can see that the result image doesn't perform well for the face reconstruction, and it is probably due to the low quality of the training dataset, since Kinect data isn't pixel accurate. But it still gives a reasonable result, since the green color represent for the depth data, you can clearly tell that the left leg is greener, which means it is closer, and the left hand is darker, since it is farther in depth. In addition, the texture of cloth is clearly transferred, as well as the general skin and cloth color.

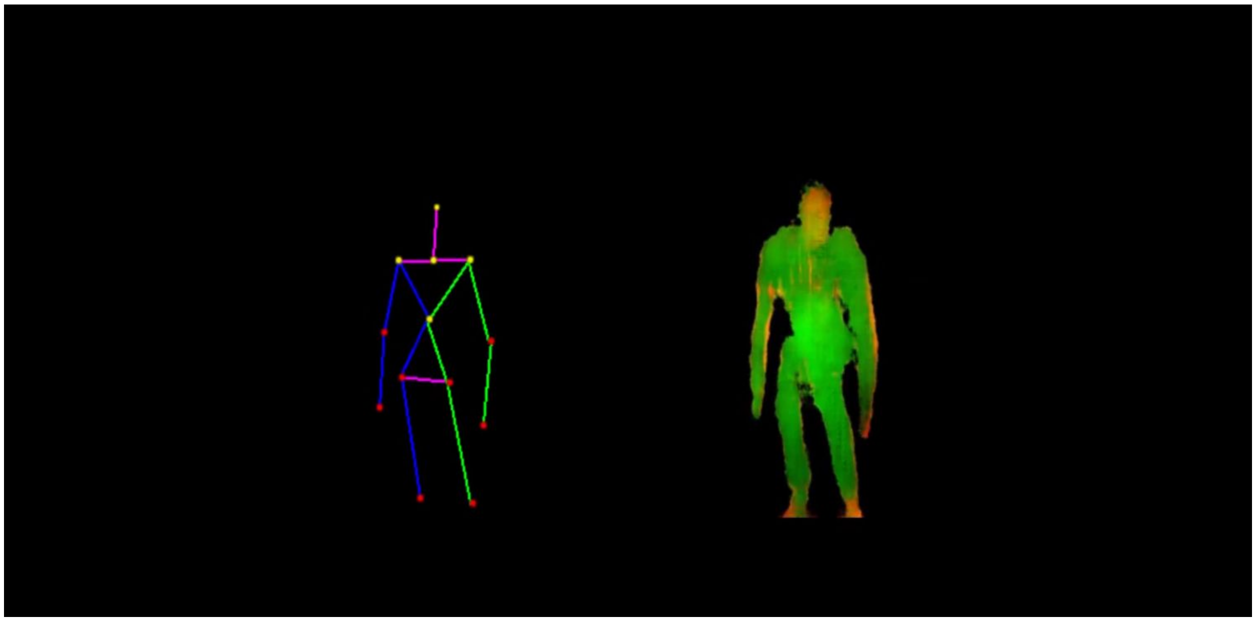
The Pix2Pix code can be found here: https://github.com/affinelayer/pix2pix-tensorflow



Export the model from the checkpoint

I used affinelayer's code to export the checkpoint into a .pict file, the .pict file is basically a weight data with the pre-trained model. The pre-trained model is about 50MB, and with the model, I will be able to generate the images in real time.
https://github.com/affinelayer

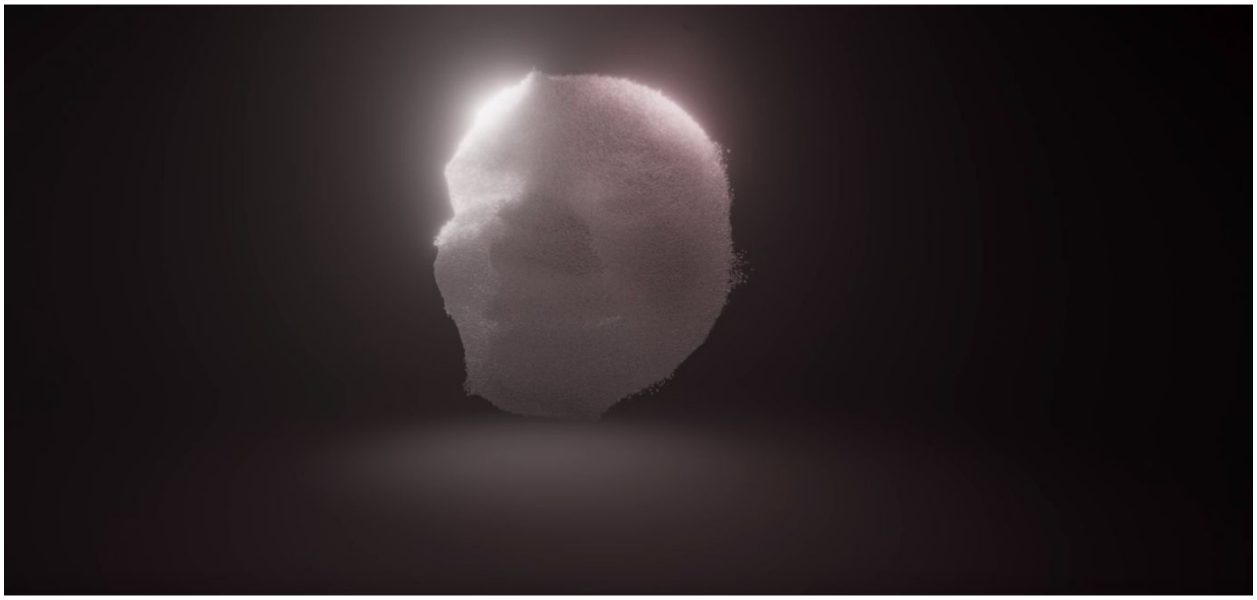
Real-time Generation
I imported the model into Unity, Keijiro's open source project for Unity allows me to use the pre-trained model to generate Pix2pix result in real time, but note that it requires minimal with GTX 1070 GPU to achieve a reasonable framerate, and using my own GTX 1070, I have a average FPS of 22-25. More detail can be found here:
https://drive.google.com/file/d/1TB_3Zi1omtXkLvbZS6STpbV3JtkmZMLY/view?usp=sharing

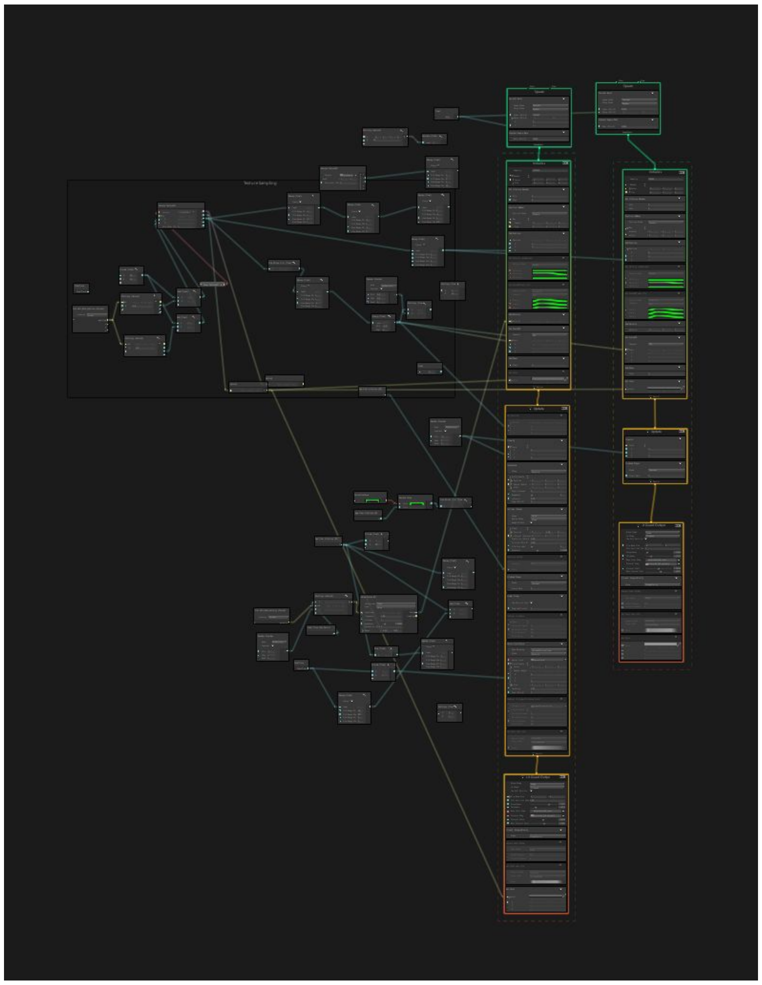


Real time result in Unity

Reconstructing 3D Point Cloud from 2D images
The next step would be reconstructing the 3D point cloud. I used Unity Visual Effect Graph in high definition render pipeline to transfer the pixel data into point cloud, and the Visual Effect Graph (VFX Graph) is a node based programming tool to help game developers to build compelling particle effects with GPU. The reconstruction process took many efforts, and in order to achieve a dynamic artistic effect, I added some special physics simulation. For example, a random gravity between a range is added to each particles, so they will fall down after being spawned. When they collide with ground mesh, a collision force is also implemented to make them look real and believable. Moreover, a small turbulence force simulates the air flowing to make the particles behave more naturally. Besides, a camera movement is added in order to convey the sense of depth.

Finally, I choose a light music as background music, and some special lighting and glitching effect helps the whole scene looks cohesive, and express a feeling of dynamic and evolving. A music rhythm detection program is utilized here to make all the lighting, glitching effect reacts to the music to have a seamless experience. Additionally, I choose some fragments words and sentence of the concept of Oneness from buddhism philosophy, and randomly displayed them on the screen to enhance the feeling of uncertainty.


A test of 3D point cloud reconstruction in Unity using VFX graph, more detail can be found here:
https://drive.google.com/file/d/1hGXH8xWIvwlwNxD8ZvUdpZpqyQ3NwxTh/view?usp=sharing



Other Technical Challenges
Since we want to generate this all in real time, but the Pix2Pix and VFX effect are both GPU heavy, and we don't have a extremely powerful GPU, so we have to separate the computing resources into two separate computer. One computer for Pix2pix, and another computer for generating the point cloud, and we used OSC and Network to stream the data from one computer to the other, while as a tradeoff, it gets a little bit lagging, but in general the delay is acceptable.

EPHEMERAL EGO
ALL ONE
BEINGING

EPHEMERAL EGO
ONENESS
BUDDHADHARMA

EPHEMERAL EGO
ONENESS
BEINGING
BUDDHADHARMA
non being
SINGLE TRUTH
WE ARE ALL ONE

EPHEMERAL EGO
ONENESS
BEINGING
BUDDHADHARMA
non being
SINGLE TRUTH
WE ARE ALL ONE
BUDDHADHARMA

Reference:
Unity Pix2pix real time:https://github.com/keijiro/Pix2Pix
Pix2pix Tensorflow: https://github.com/affinelayer/pix2pix-tensorflow
Background Music:
https://drive.google.com/file/d/1SFZe8zzJl1KnfMgZm3loMdGv-JkB4CNd/view?usp=sharing

Code:
The Wandering City: https://github.com/briankim13/art_ml_project2

Ephemeral Ego
Unity executable for Pix2pix:
https://drive.google.com/drive/folders/1JU7u_nKlnpi1L9cFklR7q70b-UblFxwb?usp=sharing

Processing Code for depth image and skeleton data collection:
https://drive.google.com/file/d/1Wg6gSHtDVhPqJ8-H0gfr-C1qWt6TXDFg/view?usp=sharing

Text file of the Oneness:
https://drive.google.com/file/d/1gZ1snhBRl809eYV9vgpmwmgcWIeMqmpE/view?usp=sharing

RESULT:
https://drive.google.com/drive/u/0/folders/1nmvaA6_x5QZ5cyR9rMmiK6JADihvJcgh

For the final result video of “Ephemeral Ego”, please find it here:
https://drive.google.com/file/d/1STFiPhFUH7Eiz_EoTiG5Yfhk5aRLRdxY/view?usp=sharing