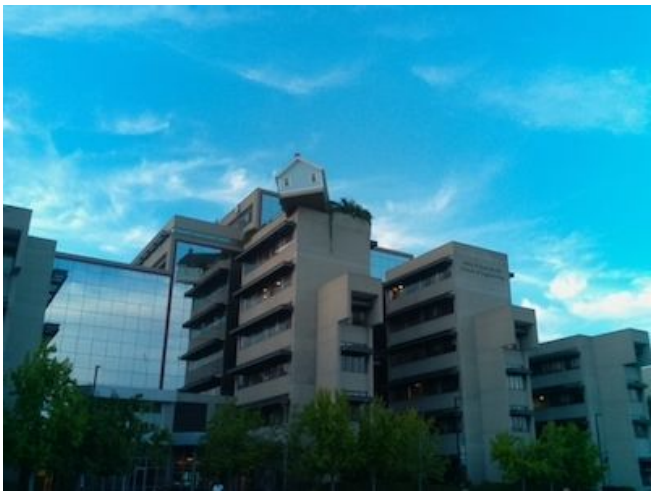
Machine Learning for the Arts
UCSD SPRING 2019
**FINAL PROJECT**

# ECE 188 Final Project

(Neural Style Transfer Camera)



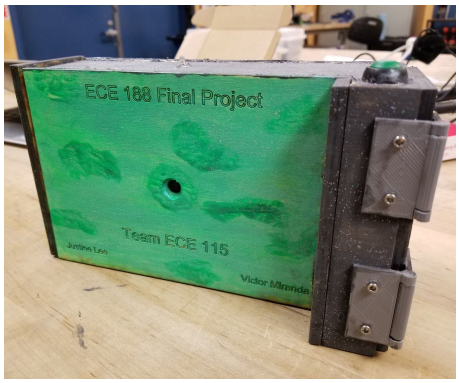
Victor Miranda
Justine Lee

**DESCRIPTION**

Our project involves creating a camera module from scratch to take pictures that will be sent to Datahub to undergo semantic style transfer processing. We tried processing some of our older pictures from project 4 in addition to images captured through our camera with style images found online. Our final results showed some similarities to its original image but some were also hard to distinguish.

**Concept**:

We wanted to refine, enhance, and extend our previous project of generating visuals using Neural Doodle. We were inspired by the Cartoonify Polaroid idea and decided to create our own camera from scratch using a Raspberry Pi. Using the images we capture, we can send it to Datahub to run through the semantic style transfer. Since the camera can save multiple images and store it in its memory, and we did not want to hand segment the images, we decided to use k-means to cluster the colors. However, since we let the program segment the colors, we were not able to decide which color masks to which color, it is random. Our idea was to try out different food images as our style for the pictures we took around campus. We also tried restyling a couple of our images from the previous project to compare the k-means clustering results to when we processed them with our manually painted segmentations. Some of our results came out clear while others were difficult to distinguish. By comparing the content and results, we were able to find similarities.




**Technique**:

The implementation for our project is very similar to how our Generative Visual (project 4) was structured [1]. On top of project 4, we set up a raspberry pi (RPi) with a camera module to take pictures, implemented in Python [5]. In addition, we also set up the RPi with a button and LED for a better user interface when taking pictures. This idea was inspired by the “Cartoonify” project [3][4].

After taking various pictures, manually cleaning them (deleting test photos), and renaming them, we then saved them on datahub with “rsync” which is a utility for efficiently transferring and synchronizing files across networked computers [8]. This would allow us to do all the heavy computation work on datahub with its pre-installed packages and GPU resources (RIP RPi).

Instead of creating our own masking through Microsoft Paint/Mac Paintbrush like we did in project 4, we wanted to achieve maskings computationally to save time. We did this by implementing a k-means segmentation algorithm that masked pictures based on ‘k’ number of segments/clusters, implemented in Python [6]. Lastly, we ran the Neural Doodle algorithm as we did in project 4 in order to achieve the semantic style transfer on our desired content and style images, with the maskings images done through k-means [2].

We also created an encasing for the whole structure with the help and resources of the ECE Makerspace. This includes 3D printed hinges, laser cut wood/acrylic, soldered wires, and a painted outer case.

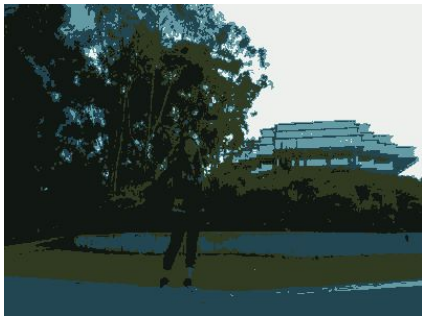
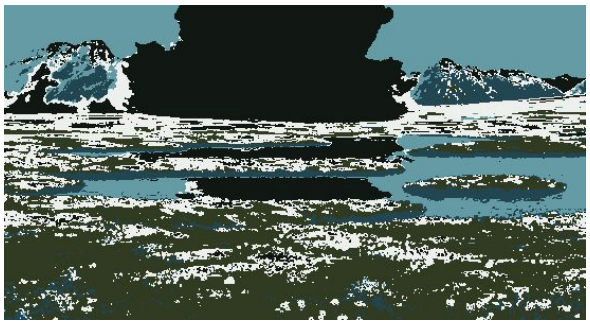
**Process**:

During our development process, we resized images to be smaller so that the file size would be approximately less than 100kB. The camera would take a picture of 640px by 480px with a range size of 100kB to 300kB and we would scale the width to 400px with a size of around 20kB to 40kB, allowing the processing time to decrease.

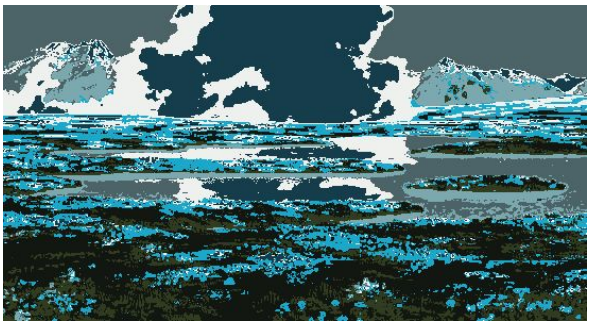
We also played around with the number of clusters for segmentation. For example with the Geisel and Studio Ghibli segmentations shown below, we would start with the initial cluster of 5 then slowly increased the number depending on how well the segmentation appeared on the content image. Although a cluster of 7 would give us more details from Studio Ghibli, it didn’t show much difference on the Geisel image. Cluster of 6 gave our ideal segmentation as the mountains and clouds are better clustered together with the same colors rather than having 2 segments. Our average cluster size was approximately 6, which gave us a decent segmentation where it could still define the original image.

5 Clusters


6 Clusters

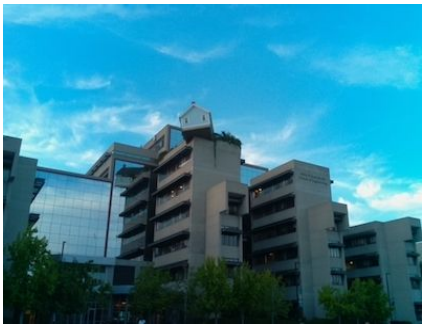
7 Clusters

**Result**:

As mentioned previously, some of our results came out clear while others were difficult to distinguish. You really have to compare images side by side to find any similarities within pictures and too see if Neural Doodle was doing its job.

Fallen Star As Gummy




Here we can see that the buildings are generated as yellow-ish gummy bears, while the sky is generated as the green gummy bears. Neural Doodle is definitely doing its job here as you can clearly distinguish the 3 buildings and the sky which are the main features of the image.

Geisel As Studio Ghibli

Here we can see that the trees on the left are generated as the fluffy clouds , the sky is generated as the flower meadow, and Geisel is generated as the body of water (this one is a bit hard to see). Again, Neural Doodle here did a decent job distinguishing the main features.

Freeway As Fruits

Here we can see that Neural Doodle did not produce a quality results. The generated image on the right seems to be almost a copy of the style image with the layered fruit. However, you can see that the fence being generated is starting to be distinguished. We believe that with a longer training time or more iterations per epoch, we would produce a higher quality result.

Price Center As Sashimi

Here is a very poor example generated by Neural Doodle. It hard to distinguish which exact features are being generated. We believe that it may have to do with how both the content and style images are masked with k-means. Both images have a lot of features in them, and so the segmentation isn't of good quality, thus generating a poor result.

**Reflection**:

In the future, we would like to be able to train and generate for longer periods of time (>1-2 hours) in order to produce higher quality results. There are also many parameters that come into play when generating final results. These include the pixel sizes of the content and style images, the chosen number of k-means clusters, and the number of iterations to run each epoch. With these variable parameters in mind, we just went with what we felt was right in order to produce decent results in an efficient amount of time.

We would also like to be able to connect to datahub on our RPi directly so we don't have to use our laptop as a "middleman." With more time and research, maybe something like an HTTP/API request to send pictures from the RPi to datahub for processing, then back to the RPi after processing. This is so the final results can be stored directly in the RPi and the project can be self-contained.

**REFERENCE**:


[1] Project 4 Generative Visual:
- https://github.com/ucsd-ml-arts/generative-visual-group-ece-115

[2] Neural Doodle:
- https://github.com/alexjc/neural-doodle

[3] Cartoonify Camera:
- https://github.com/danmacnish/cartoonify

[4] RPi with GPIO Pins:
- https://projects.raspberrypi.org/en/projects/the-all-seeing-pi/
- https://gpiozero.readthedocs.io/en/stable/recipes.html

[5] PiCamera:
- https://medium.com/@petehouston/capture-images-from-raspberry-pi-camera-module-using-picamera-505e9788d609

[6] Image Segmentation:
- https://www.analyticsvidhya.com/blog/2019/04/introduction-image-segmentation-techniques-python/

[7] VGG-19 Paper:
- https://arxiv.org/abs/1409.1556

[8] RSYNC Documentation:
- https://linux.die.net/man/1/rsync


**CODE:**

- https://github.com/ucsd-ml-arts/ml-art-final-ece-115

**RESULT:**

- https://github.com/ucsd-ml-arts/ml-art-final-ece-115/results
- https://docs.google.com/document/d/1jnNvAf6oiMX2bmNCnMKpvXF0K6zn5Qc-QuiKX2Uw_Ws/edit?usp=sharing