Mini Project:
Emotion Recognition for Art Generation
Artificial Intelligence and Machine Learning Course

Santiago Won Siu
March 2024

MSc Data Science & Artificial Intelligence
University of the Arts London

https://github.com/santiagowonsiu/BetaProject

Table of Contents

I. Abstract

This project explores the integration of pose **classification, 3D environments, and deep learning in image generation** with the aim of eventually being able to convey into a more matured solution that enables personal growth tools through art and technology.

The project's trajectory includes tools like Google's MediaPipe for pose classification, and Generative Adversarial Networks for image transformation, specifically Contrastive Unpaired Translation (CUT), CycleGAN and pix2pix models.

The system's architecture combines front-end interactions with Three.js and a Flask-powered backend, covering the user journey from pose recognition to image generation. Challenges in server setup led to a deep dive into HTTP protocols. OpenCV is integrated for real-time camera functionality, seamlessly capturing user poses for recognition and image generation.

In the end, a GAN (Generative Adversarial Network) was added, and it made the project better and took it further than what was first planned. Training this model with pictures of landscapes and 3D objects helped it learn to create images, but it wasn't easy to adapt the model to run in a web-based system. Getting the GAN to work just right involved careful changes, like understanding in depth the relationships between the different functions.

For the future, there's a chance to make the system better by having it recognize more kinds of poses, improving the GAN's training images, and making it smart enough to generate 3D objects that change based on what it "sees".

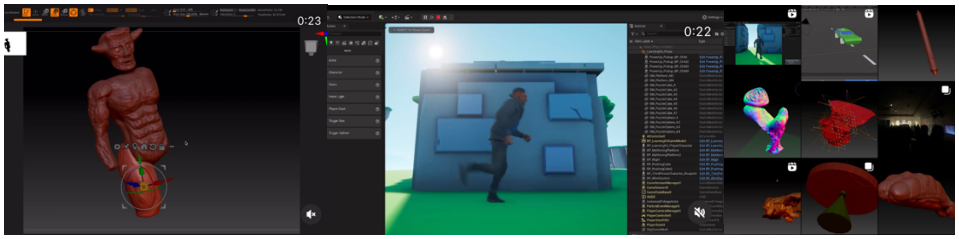
This project is a first step towards using AI and 3D tech to make personal growth and emotion awareness more fun and engaging. However, it could also end up developing towards developing tools for inclusive art creation.

II. Personal Background and Project Pre-Work

Four years ago, I started a sports mentoring platform called Zport (www.instagram.com/zport_official) to support teenage athletes. There, my interest in education and democratizing personal growth tools for any sort of people emerged. However, I had to pause my project because it was not scalable and engaging enough.

Currently, I believe that immersive technologies and gamification could be a game changer towards accomplishing higher impact in this field. Also, I personally have started a growing interest on 3D experiences, which is influencing the models, tools, and libraries in which I would like to gain more expertise.

Therefore, at first, I started going through some learning in 3D design (ZBrush) and game design (Unreal Engine). Nonetheless, the complexity to integrate to web solutions surpassed my knowledge and had to pause it for, perhaps, the next version of this project.



Thus, I decided to go into a more basic and flexible area and embraced Three.js as a starting point.

III. Project learnings

There are four elements of learning I would like to highlight through the process of developing this project

1. To explore machine learning and creative computing, I focus on **covering larger range over depth**. I excel at connecting concepts, so I prioritize a broad learning approach, even if it meant sacrificing depth for now.

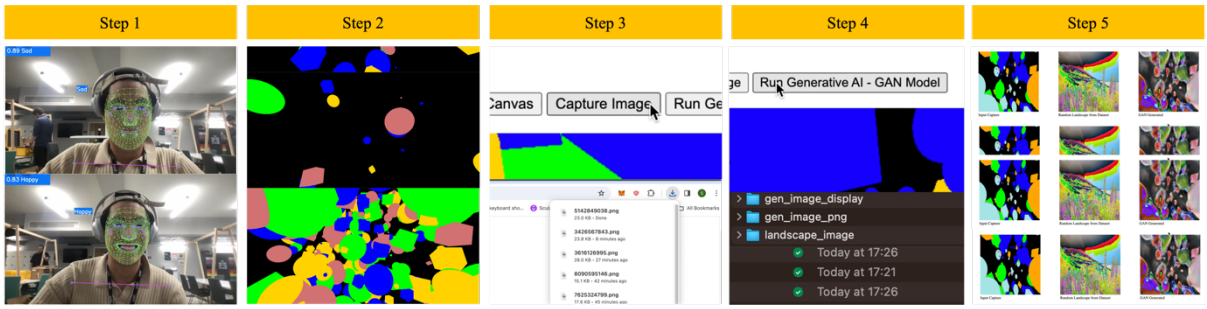
2. It is hard to **make things tangible** and working, but it is also a great strategy to have a **clear progress and feasibility notion**, now it makes more sense why Scrum and working in sprint with deliverables

3. **Integrations** are hard, so it is always better to work on the basics of each component separately and make it **more complex gradually**

4. **GitHub** co-pilot can be great to understand potential end state and break the inertia on next steps of code, but usually documentation and basic tutorial from Youtube can be more effective to set up a properly working project

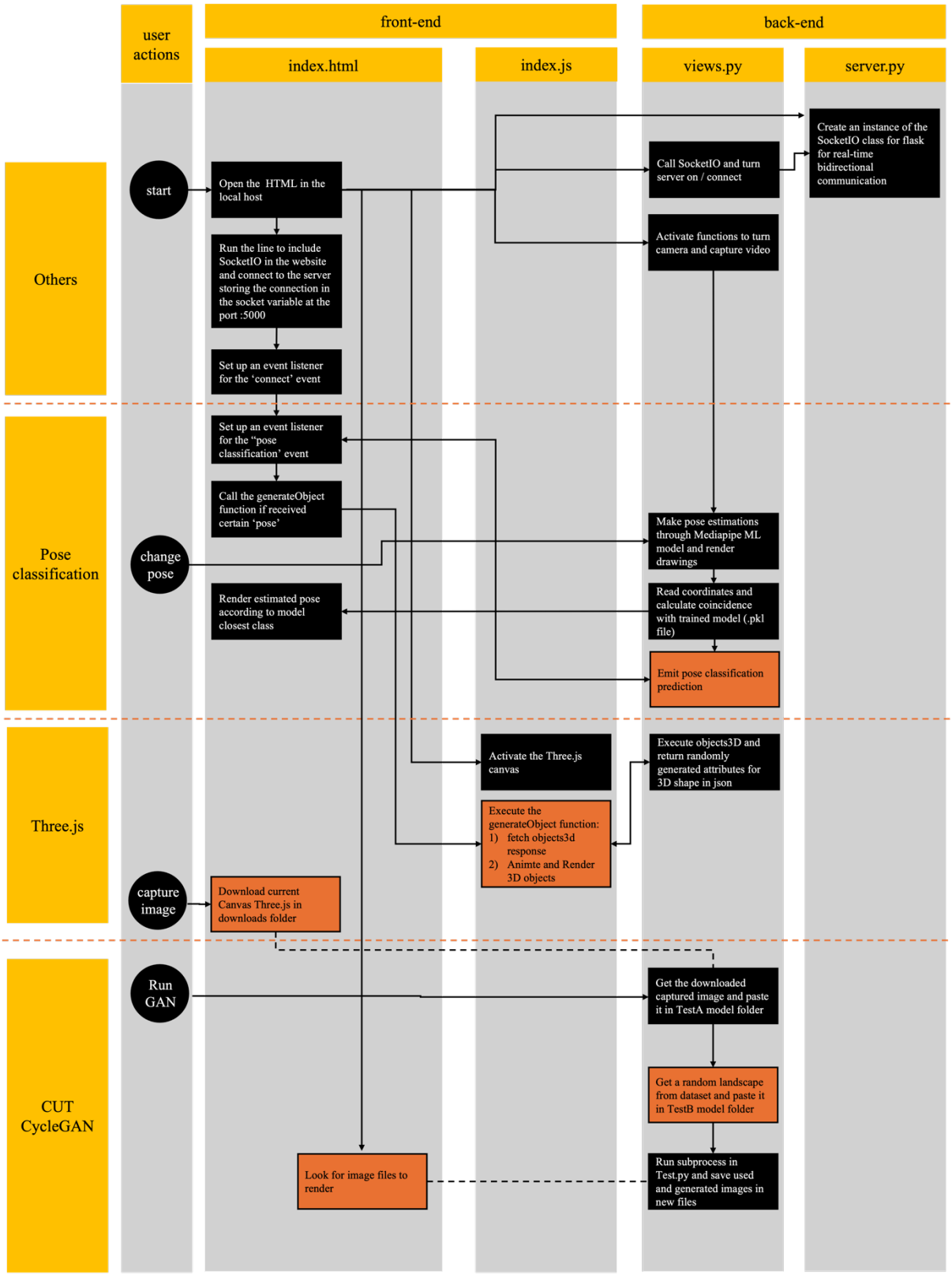
IV. The solution and user journey

The idea of this current project is to be able to play through the interaction on pose classification, 3D objects generation and GAN models to generate new images based on our input. Perhaps, some potential uses could include inclusive tools for art generation or emotions acknowledging through art and technology. The solution has five main steps of experience which are exemplified in the following images:



V. System Architecture

The following system architecture was graphed in order to have a better still high-level understanding of how the solution is built. Key actions are highlighted in orange and doted links mean there is a relevant relation, but there is not a real code action call

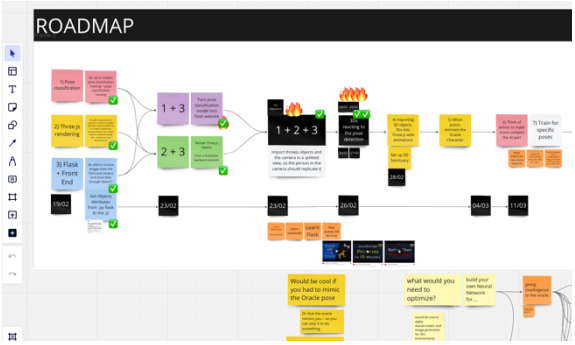


VI. Project Development: process, challenges, and outcomes in implementing Deep Learning Models and Web Development

1. Introduction and planning

Getting started to have something ongoing was a bit of a struggle for this project, going on and off from different part required me to get on top of it by decomposing the elements that I wanted to include and create a roadmap plan in Miro.

And that is how it all started.



2. First phase – pieces

a. The first part of the process implied getting to know and run independently the main pieces of inspiration. That is how I started with the **pose estimation model** and training it to determine how did I want to classify my poses. The input stage was registering into a csv file with the **labeled classes for pose classification**. (Renotte, 2021) (FreeCodeCamp, n.d.)

```
# Concate rows
row = pose_row+face_row

# Append class name
row.insert(0, class_name)

# Export to CSV
with open('coords.csv', mode='a', newline='') as f:
    csv_writer = csv.writer(f, delimiter=',', quotechar='"', quoting=csv.QUOTE_MINIMAL)
    csv_writer.writerow(row)
```

Then, the model was fitted through sklear library and after that the parametres would be stored in a body_language.pkl file. (Renotte, 2021)

```
fit_models = {}
for algo, pipeline in pipelines.items():
    model = pipeline.fit(X_train, y_train)
    fit_models[algo] = model
```

```
fit_models
```

b. After that, it was key to **setup a webcam** working not in JupyterLab and the camera default application, but in a website client. For that, some learning on HTML and Javascript had to be done and **Vite** (front-end build tool) was used. Some adjustments to the code had to be made to properly capture and render the images

c. Next, learning **Three.Js** basics was part of the plan, since there was a lot of complex integration, I opted to only work with the most basic geometric shapes.

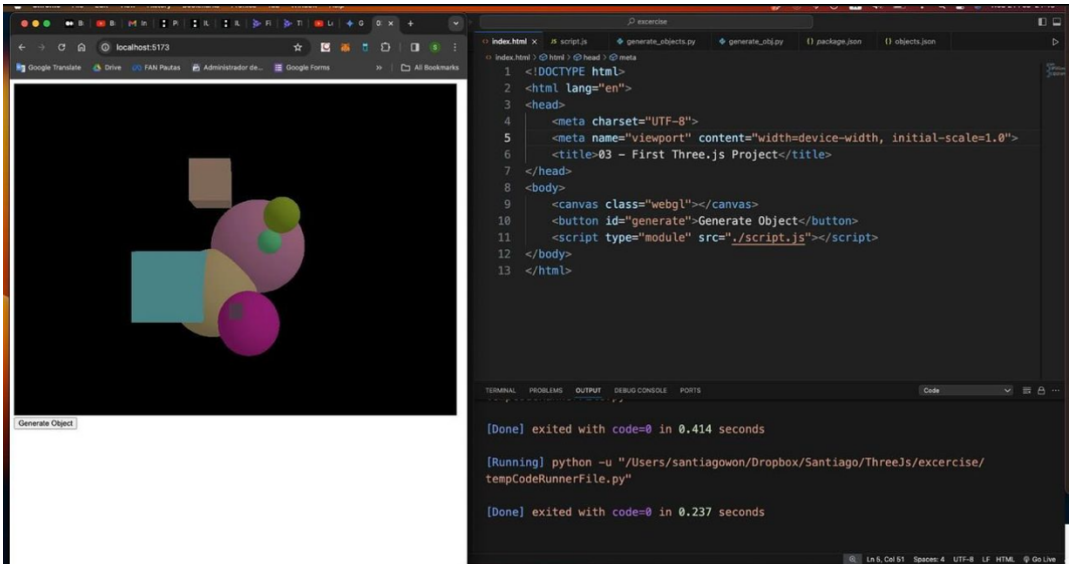
d. Last, I focused on learning how to setup **Flask** project structure and the right environment

3. Second phase – front and back basics

This phase was about being able to connect each proved element to Flask (backend)

a. Starting with the **webcam**, had to go through some video tutorials on the basics on Promises and Fetch functions to be able to interconnect my front and back. This learning also helped to adapt the pose classification model code from .ipynb to the flask code.

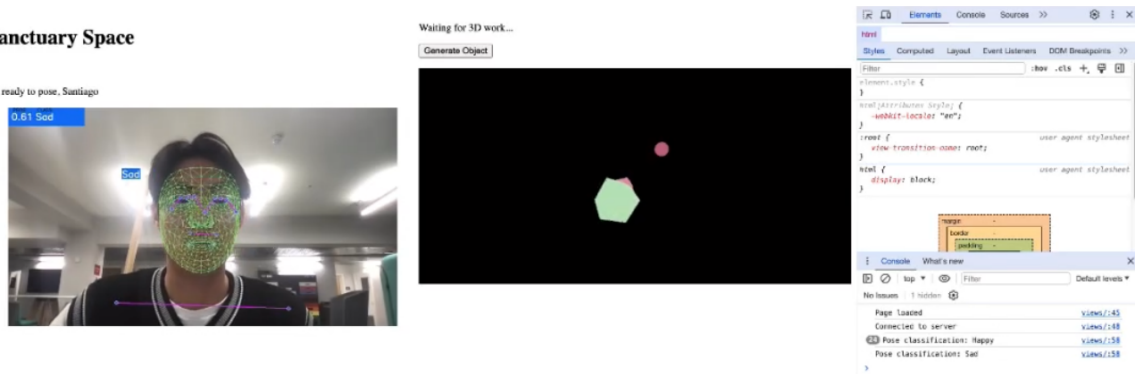
b. Next, I created **python code to generate 3D objects** attributes. The idea was that the 3D objects could be generated based on inputs like size and shape based on algorithms and results from other python functions and had the connection between Javascript **Three.Js and Flask** (Three.Js, n.d.) (Neupane, n.d.)



4. Third phase – setting up the server

This was one of the most challenging steps for me because the amount of code and files was already in a decent amount and it was the first time I really tried to understand what is a server. I got into some videos and consulted the technicians from CCI to get an idea on HTTP, SockeIO and review line by line (TechWithTim, 2021).

After several iterations and revisions, I was able to print the "emotions" (the class of the pose classification model) into the console. Which then was very easy towards linking in to generate the 3D objects.

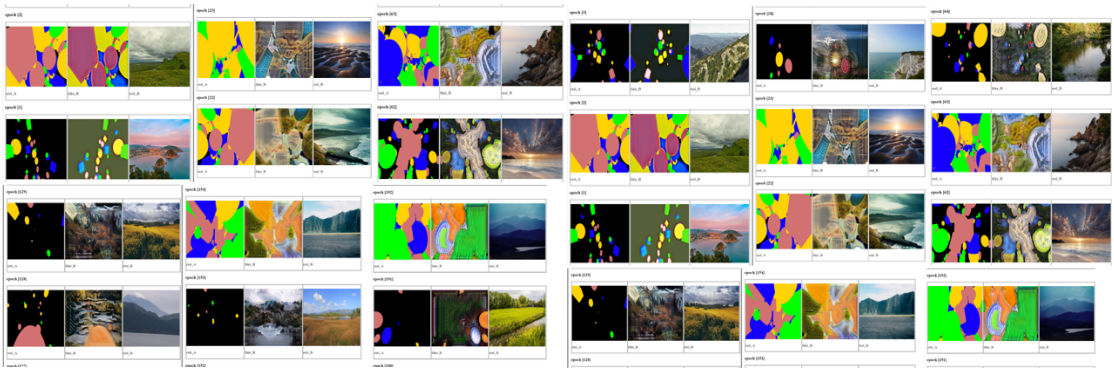


5. Forth Phase – Make it cooler: Including a GAN model

The next phase was about deciding what to do next to make the project more complex. Thus, I decided to **run a GAN model** in my computer, for this I had to understand the code and get through the documentation. (Taesung Park, 2020) (Park, 2021) (Phillip Isola, 2017) (Jun-Yan Zhu*, 2017)

After that, I prepared my **picked dataset** (landscapes) and cropped it to fit the measures of the model (it is not mandatory but improves training since otherwise it will crop it randomly). Then, **the model was trained** for around ~52 hours during the weekend on a dataset of 4,000 landscapes and 4,000 randomly generated captures of 3D Objects through 200 epochs (Rougetet, n.d.)**.**

Graph examples on the model results through the different epochs:



I **selected the 150th** epoch as the input for my model since it appeared to me the most aesthetically pleasing result and closer to the TrainB dataset. It is important to mention that it was hard to identify were should I **replace my ".pth" file of the newly trained model** because the repository had a lot of files, but by getting backwards through the functions it was possible to find it.

Finally, the repository folder was added to my current project's model. The CUT model would run inferences on a specific command, however, to **make it work with my client side website** several modifications had to be made.

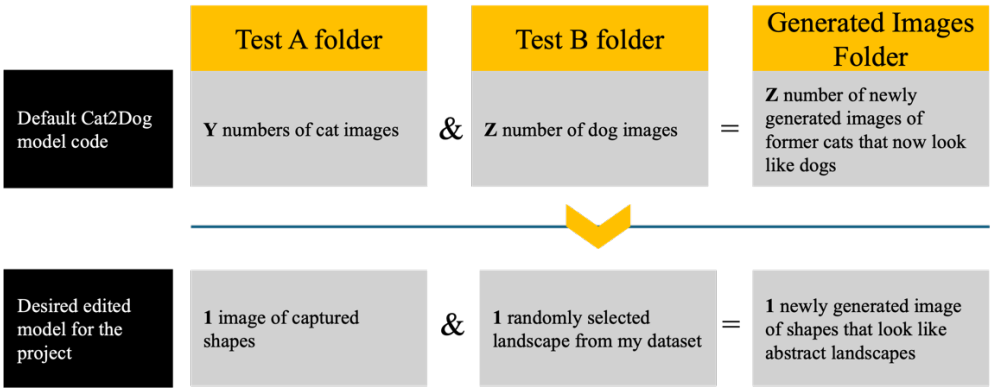
a. All the new **roots** and paths had to be updated since some of the code would expect or save new files in the root of the project, but the actual repository of the GAN was in an inferior layer of folders.
b. All the code was always prioritizing to use **GPU**, since my Mac has a GPU, but is an AMD that can't work for AI, it required several code editions and some hard coding on several files of the repository
c. Some **parameters were hardcoded** to reduce the amount of required files, since actually the repository would work for 5 different model, but I only required one of them (CUT Cat2Dog). Thus, **Namespace** class and opt object were hardcoded inside the Test.py file. I was able to find the right values to hardcode through printing debugging
d. Transformations of formats from tensor to **PIL images** to directly save from Test.py and allow an easy display

** Some other code edits might have been made but lost because I lost the rest of my notes in the attempt to free up some space in my computer. Rookie mistake. Never save notes in the downloads folder again.*

6. Fifth and Last Phase – Make the GAN work taking input from Three.js

In this step, I edited the RunTest function to work with the right inputs. The code of the **Cat2Dog CUT model** was strongly overcoded so some changes had to be made so it would make sense for the project. In essence, so each time the user wants to generate an image it doesn't generate hundreds of new images influenced my test dataset of landscapes, but only picks one randomly (**refer to the run_test function for further detail in the code documentation)**

Graph explanation on the difference between how the folders structure worked originally vs how it had to be modified for the project to work:



Also, I added a button to download a screen capture of my Three.js since Javascript is very restricted towards changing the downloads folder, for security reasons (Test A input). Finally, the generated images were displayed in the website. Unfortunately, I couldn't make it to load live, so the user must refresh the browser to see them.

VII. Evaluation on next steps

The results of this project were quite interesting since when actually operating some users testing it appeared to be amused and briefly engaged.

1. Potential Fixes

While evaluating the solution, some potential fixes popped to enhance the system.

Technical:

When stopping the application or closing the browser **there are still processes running in the background**: the Google Chrome Renderer, which could be a bug with Mediapipe since some errors appeared when using it with the regular camera application.

Also, if the app is stopped, JavaScript code keeps being printed with errors and then if trying to run again the client does not load. As a temporal partial solution, the browser had to be closed and all the processes still running from port :5000 had to be killed to execute the "python app.py" command and be successful

Experience:

Currently, after the GAN generates the new images, they do not appear in the client until it is refreshed, further exploration and knowledge on HTML and sockets could help improve the experience

Also, it would be nice to get an alert when the model is done processing, instead of having to see if new images have been added to the root folder

Improve results:

The GAN model could be retrained with less divergent datasets, images from landscapes where too different in the used dataset, so even there is an aesthetically pleasing output, getting some recognizable pattern of the real world would be interesting

Additionally, I could retrain the model with bigger data for pose classification would be valuable. The current model was only trained with my face and in a same room, so pose classification often fails with other people or spaces

2. Potential Future Work Ideas

Machine Learning

a. Learn more on the neural networks and Torch to get dirtier into the code and model structure

b. Add different pose classification classes to run different function (e.g., clear the Three.js canvas)

c. Do and learn fine tuning for the GAN

d. Train through a Semantics Classification Dataset to assign a color to the captured 3D shapes accordingly (e.g., green is for Trees and Plants textures). Also, it could be used to identify objects in the camera display and replicate an equivalent in 3D with a different artistic meaning

Three.JS

e. Have another canvas of three.js with imprinted AI Generated textures instead of static images

f. Take more input from the Mediapipe pose classification (e.g., create the 3D objects according to the position where the face was detected in the camera display)

g. Try importing more complex 3D animated objects and make them move / react in real time based on Mediapipe readings

Augmented Reality

h. Combine both, the camera render and the Three.Js render… and make it become more immersive by adding new 3D objects that go in line with the identified objects in the environment

VIII. Bibliography

Renotte, N., 2021. *Body Language Decoder.* [Online]
Available at: https://github.com/nicknochnack/Body-Language-Decoder

Taesung Park, A. A. E. R. Z. J.-Y. Z., 2020. *Contrastive Learning for Unpaired Image-to-Image Translation.* [Online]
Available at: https://arxiv.org/pdf/2007.15651.pdf

Park, T., 2021. [Online]
Available at: https://github.com/taesungp/contrastive-unpaired-translation

Phillip Isola, J.-Y. Z. T. Z. A. A. E., 2017. *Image-to-Image Translation with Conditional Adversarial Nets.* [Online]
Available at: https://phillipi.github.io/pix2pix/

Jun-Yan Zhu*, T. P. P. I. A. A. E., 2017. *Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks.* [Online]
Available at: https://junyanz.github.io/CycleGAN/

Rougetet, A., n.d. *Landscape Pictures Dataset.* [Online]
Available at: https://www.kaggle.com/datasets/arnaud58/landscape-pictures

FreeCodeCamp, 2019. *Learn JavaScript - Full Course for Beginners.* [Online]
Available at: https://www.youtube.com/watch?v=PkZNo7MFNFg&t=12s

FreeCodeCamp, n.d. *Responsive Web Design.* [Online]
Available at: https://www.freecodecamp.org/learn/2022/responsive-web-design/

Three.Js, n.d. *Three.js Journey.* [Online]
Available at: https://threejs-journey.com/lessons/introduction

Neupane, A., n.d. [Online]
Available at: https://www.youtube.com/watch?v=7LNl2JlZKHA&t=27s

TechWithTim, 2021. *Python Socket Programming Tutorial.* [Online]
Available at: https://www.youtube.com/watch?v=3QiPPX-KeSc&t=2607s

FreeCodeCamp, n.d. [Online]
Available at: https://www.youtube.com/watch?v=V_xro1bcAuA&t=50668s