User Guide for Identification of 3D Attention Points Project

By Sandeep Tandra, Vineel Gannu

April 17, 2020

This file contains the user guide for

- The Executives
- The Engineers
- The Analysts

1 User guide for Executives

Executives are those who have supervisory or administrative authority in an organization.

1.1 Situation and Project Goals

1.1.1 Situation

- Anxiety disorder is characterized by feelings of intense and persistent anxiety, fear, and nervousness. In general, anxiety treatment involves therapy and psychological counseling.
- Exposure therapy is a type of cognitive behavioral therapy (CBT) that includes in-vivo and imaginary exposures. In-Vivo exposure is costly and it is also difficult for few patients to imagine specific stimuli when they undergo imaginal exposures.

1.1.2 Goals

- Our goal is to create a virtual reality environment with different stimuli and scenes and identify the attention points using the trained model.
- In-Vivo exposure and imaginal exposure can be implemented at a reasonable cost.

1.2 Overview

- Data Analyst can create the dataset and build a model using the data.
- The built model can then be used to predict the attention points and the data can be tracked on dashboard along with other parameters that deal with anxiety disorder.
- On the other hand, the therapists uses the dashboard to monitor the patients behavior.

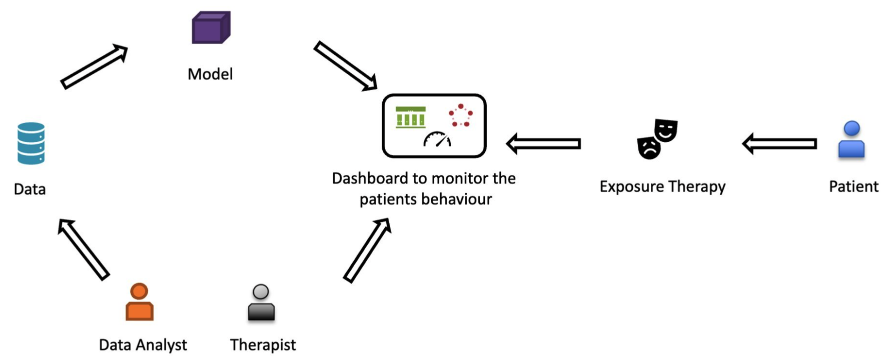


Figure 1: Big Picture

1.3 Approach

- We created the dataset by using a virtual reality environment. The VR environment was built in such a way that it can be used in different anxiety-reduction techniques.
- We developed the predictive model to identify the attention points, that can be used to identify the attention points of the patients when they are exposed to anxiety-reduction techniques.
- Different methods were used to determine and improve the model performance.

1.4 Model Description

- The trained model predicts the 3D Attention points in a virtual environment which has two different images that depict different emotions.
- The model that we trained was a multilayer perceptron which is a feed-forward neural network.

- Fifty-six percent of the data was used for training, whereas twenty-four and twenty percent of the data were used to validate different models and test the trained model respectively.
- The model developed has reasonable predictive capability for the given dataset.

1.5 As-is and To-be processes

1.5.1 As-is process

- Healthcare organizations like Big Health and Medigold Health provides digital therapy for the patients. The current services provided by these organizations does not include VR based therapy.

1.5.2 To-be process

- VR based behavioral therapy with the digital therapy can be leveraged.
- Companies like Psious that uses VR therapy has higher revenue when compared to companies that follow conventional therapy methods.
- Leveraging VR based behavioral therapy boosts the organization's revenue allowing us to make a good deal with the organizations that does not include VR therapy.

1.6 Recommendations

- We can implement the model on the anxiety-reduction methods such as exposure therapy (a technique in behavior therapy) for better performance.
- We can implement the model on the self-driving cars which can monitor the attention of the driver.
- The model can also be implemented in VR based educational applications where the teacher can monitor the students remotely.

2 User guide for Engineers

Engineers in a project deal with the technical part of the project. We have below phases in our project.

- Dataset creation
- Feature selection
- Data loading and pre-processing
- Model creation

- Model training
- Model evaluation

2.1 Dataset Creation

This phase of the project is totally programmed in C# using Unity and Tobii eye tracking software.

- Step 1. Create a virtual reality 3D scene with few objects in it.
- Step 2. Integrate tobii eye tracker software with unity software.
- Step 3. Participants are asked to look into VR scene, so that their eye gaze positions are tracked using tobii eye tracker.
- Step 4. Object positions and the gaze position are captured and saved into a text file.
- Step 5. Data saved in text file is loaded into csv file and position of each coordinate of different objects is considered as one feature.

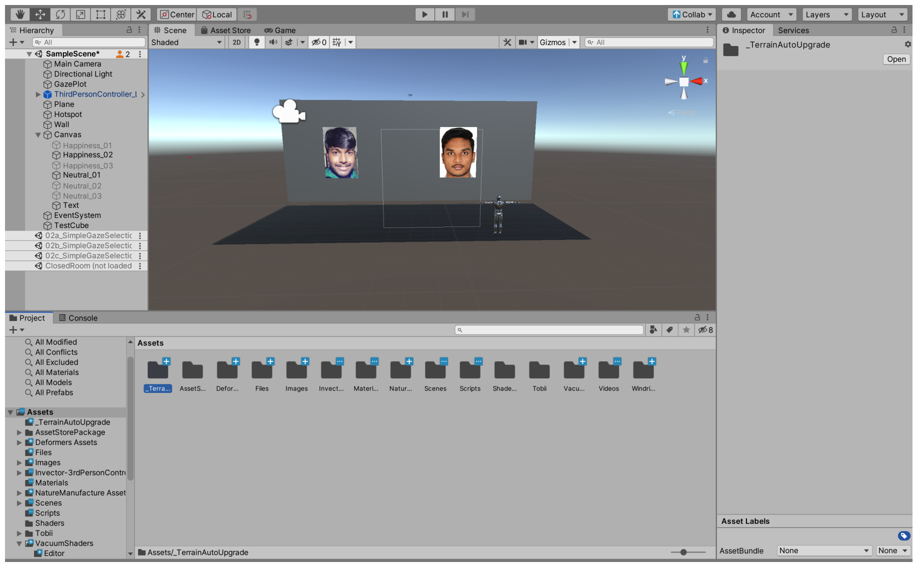
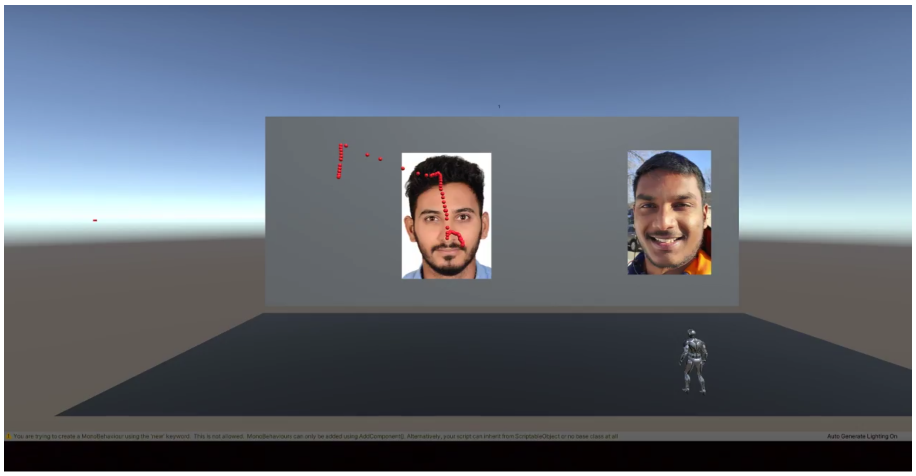


Figure 2: Unity in developer mode

2.2 Feature Selection

- Step 1. Now we have a csv file with 45 features but we have redundant and same valued features in the dataset which will not impact the prediction. So, we will find those and remove those features.



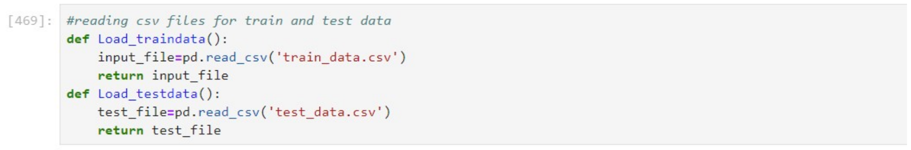
```
You are trying to create a MonoBehaviour using the 'new' keyword.  This is not allowed.  MonoBehaviours can only be added using AddComponent(). Alternatively, your script can inherit from ScriptableObject or no base class at all
```

Figure 3: VE displayed to the participant

- Step 2. After removing them we have 30 features and 1000 samples in our dataset.

2.3 Data Loading and Preprocessing

- Step 1. We have values in different range and also we have negative values which will be normalized to train model.
- Step 2. After training the model, predicted values are again de-normalized to get actual coordinate positions.



```
[469]: #reading csv files for train and test data
       def Load_traindata():
           input_file=pd.read_csv('train_data.csv')
           return input_file
       def Load_testdata():
           test_file=pd.read_csv('test_data.csv')
           return test_file
```

Figure 4: Loading data

Module to RUN: **Load_traindata(),Load_testdata(), data_normalize()**

2.4 Data visualization

- Step 1. Data visualization steps are performed to check the correlation, positions in the 3D coordinate system.

Module to RUN: **scatter_Plot, correlation_matrix(), pair_plot()**

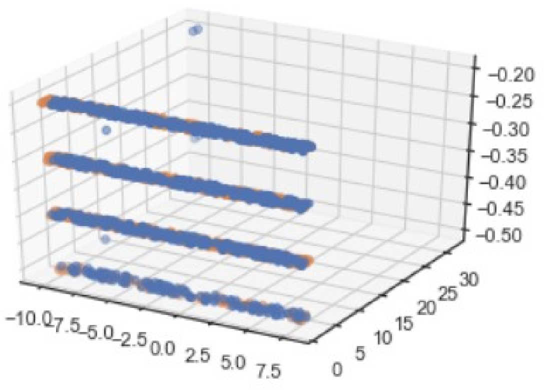


Figure 5: Before Normalization

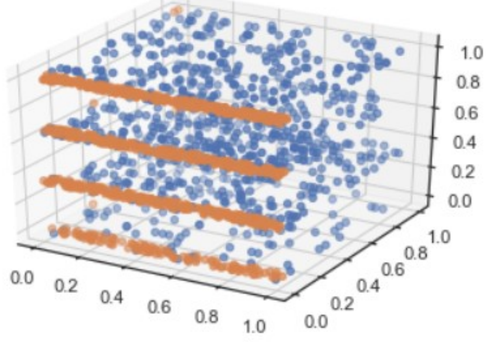


Figure 6: After Normalization

2.5 Model Creation

- Step 1. We are going to use multi layer perceptron to predict multi target labels.
- Step 2. Model is build with one input layer with 30 input neurons, 2 hidden layers with 16,8 neurons respectively and output layer with 3 neurons.
- Step 3. We also added dropout layers between each layers to overcome overfitting.
- Step 4. At each layer activation function used is tanh.
- Step 5. Optimization function used is Adam with parameter values learning_rate=0.0001, beta_1=0.88, beta_2=0.911, amsgrad=False and other

default values.

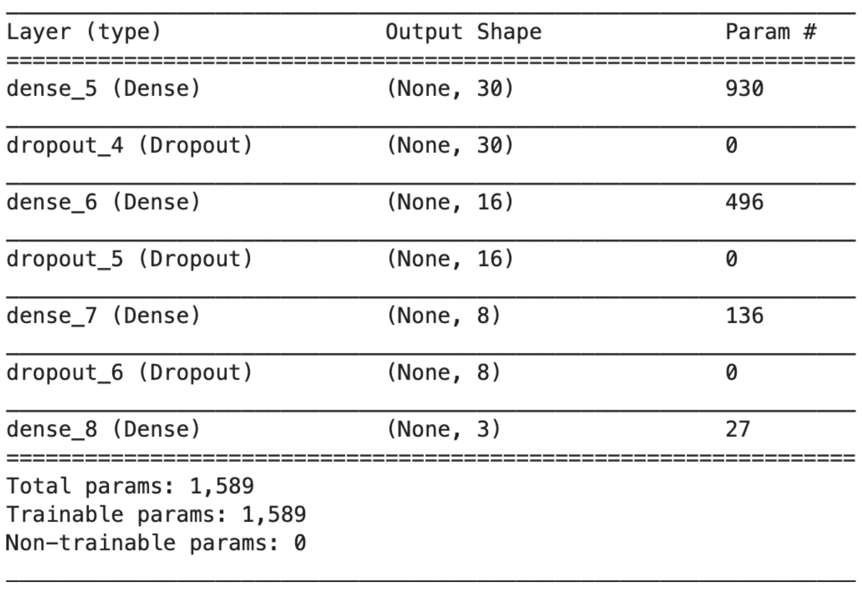

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_5 (Dense)              (None, 30)                930       
_________________________________________________________________
dropout_4 (Dropout)          (None, 30)                0         
_________________________________________________________________
dense_6 (Dense)              (None, 16)                496       
_________________________________________________________________
dropout_5 (Dropout)          (None, 16)                0         
_________________________________________________________________
dense_7 (Dense)              (None, 8)                 136       
_________________________________________________________________
dropout_6 (Dropout)          (None, 8)                 0         
_________________________________________________________________
dense_8 (Dense)              (None, 3)                 27        
=================================================================
Total params: 1,589
Trainable params: 1,589
Non-trainable params: 0
_________________________________________________________________
```

Figure 7: Model Summary

Module to RUN: **build_model()**

2.6 Model Training

- Step 1. We split the dataset into 2 parts train and test (800,200) and from the training set we use 70:30 ratio for validation and training.
- Step 2. We fit the model with train and validation data with 350 epochs where training and validation loss is converging.350 epochs are chosen after few trails.
- Step 3. Now the trained model is saved into a finalized_model.sav file. which can be used to test without training the model again.

Module to RUN: **Train_model()**

2.7 Model Evaluation

- Step 1. We run the code once and the trained model is saved into the file.
- Step 2. Now to evaluate the model we use the model from the finalized_model.sav file and give the test data to get the predicted output.

- Step 3. The predicted result is now de-normalized to give the exact coordinates of the gaze position.

Module to RUN: **model_evaluate()**

3 User guide for Analysts

The User guide to Analysts gives an overview of the project outcome.

- Project overview
- Dataset overview
- Model used
- Results

3.1 Project Overview

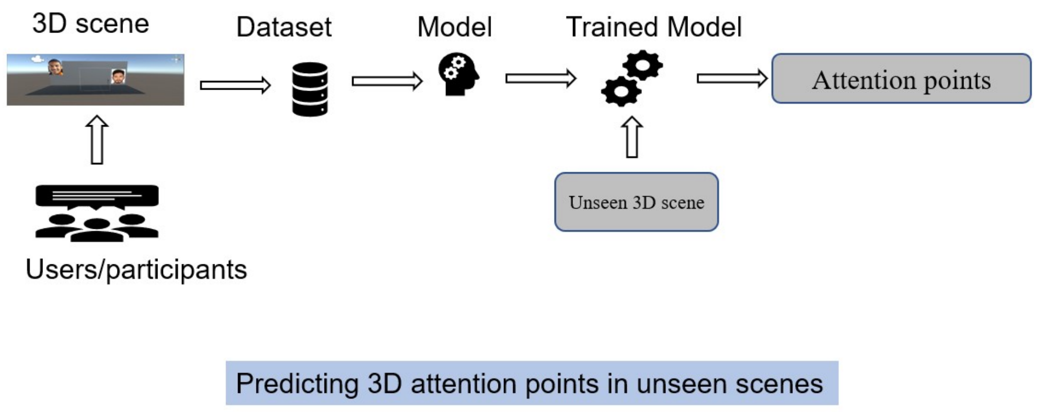


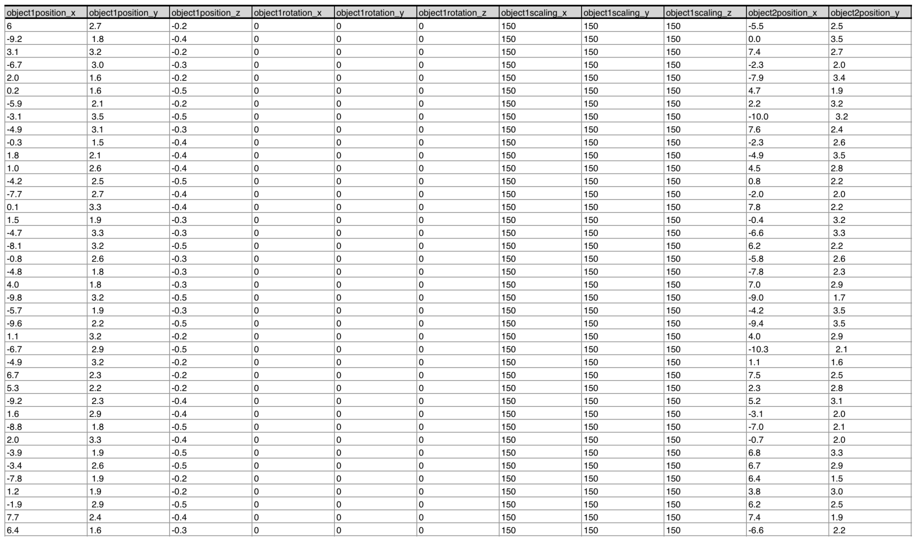
Figure 8: Project Overview

Figure 8 gives the overview of the project.

- We create a 3D Virtual Reality scene.
- Using eye tracker, gaze points of participants looking into scene are captured.
- These positions are used as training data to train a model.
- Trained model is given to an unknown scene to predict the attention points.

3.2 Dataset Overview

- The dataset consists of the [x,y,z] coordinates of the positions of each object in VR scene and gaze position of the participants.
- We need to have resources like eye tracker to create the dataset and we can get free softwares to use for the project(unity, tobii eye tracker).



object1position_x	object1position_y	object1position_z	object1rotation_x	object1rotation_y	object1rotation_z	object1scaling_x	object1scaling_y	object1scaling_z	object2position_x	object2position_y
6	2.7	-0.2	0	0	0	150	150	150	-5.5	2.5
-9.2	1.8	-0.4	0	0	0	150	150	150	0.0	3.5
3.1	3.2	-0.2	0	0	0	150	150	150	7.4	2.7
-6.7	3.0	-0.3	0	0	0	150	150	150	-2.3	2.0
2.0	1.6	-0.2	0	0	0	150	150	150	-7.9	3.4
0.2	1.6	-0.5	0	0	0	150	150	150	4.7	1.9
-5.9	2.1	-0.2	0	0	0	150	150	150	2.2	3.2
-3.1	3.5	-0.5	0	0	0	150	150	150	-10.0	3.2
-4.9	3.1	-0.3	0	0	0	150	150	150	7.6	2.4
-0.3	1.5	-0.4	0	0	0	150	150	150	-2.3	2.6
1.8	2.1	-0.4	0	0	0	150	150	150	-4.9	3.5
1.0	2.6	-0.4	0	0	0	150	150	150	4.5	2.8
-4.2	2.5	-0.5	0	0	0	150	150	150	0.8	2.2
-7.7	2.7	-0.4	0	0	0	150	150	150	-2.0	2.0
0.1	3.3	-0.4	0	0	0	150	150	150	7.8	2.2
1.5	1.9	-0.3	0	0	0	150	150	150	-0.4	3.2
-4.7	3.3	-0.3	0	0	0	150	150	150	-6.6	3.3
-8.1	3.2	-0.5	0	0	0	150	150	150	6.2	2.2
-0.8	2.6	-0.3	0	0	0	150	150	150	-5.8	2.6
-4.8	1.8	-0.3	0	0	0	150	150	150	-7.8	2.3
4.0	1.8	-0.3	0	0	0	150	150	150	7.0	2.9
-9.8	3.2	-0.5	0	0	0	150	150	150	-9.0	1.7
-5.7	1.9	-0.3	0	0	0	150	150	150	-4.2	3.5
-9.6	2.2	-0.5	0	0	0	150	150	150	-9.4	3.5
1.1	3.2	-0.2	0	0	0	150	150	150	4.0	2.9
-6.7	2.9	-0.5	0	0	0	150	150	150	-10.3	2.1
-4.9	3.2	-0.2	0	0	0	150	150	150	1.1	1.6
6.7	2.3	-0.2	0	0	0	150	150	150	7.5	2.5
5.3	2.2	-0.2	0	0	0	150	150	150	2.3	2.8
-9.2	2.3	-0.4	0	0	0	150	150	150	5.2	3.1
1.6	2.9	-0.4	0	0	0	150	150	150	-3.1	2.0
-8.8	1.8	-0.5	0	0	0	150	150	150	-7.0	2.1
2.0	3.3	-0.4	0	0	0	150	150	150	-0.7	2.0
-3.9	1.9	-0.5	0	0	0	150	150	150	6.8	3.3
-3.4	2.6	-0.5	0	0	0	150	150	150	6.7	2.9
-7.8	1.9	-0.2	0	0	0	150	150	150	6.4	1.5
1.2	1.9	-0.2	0	0	0	150	150	150	3.8	3.0
-1.9	2.9	-0.5	0	0	0	150	150	150	6.2	2.5
7.7	2.4	-0.4	0	0	0	150	150	150	7.4	1.9
6.4	1.6	-0.3	0	0	0	150	150	150	-6.6	2.2

Figure 9: Data Overview

3.3 Model Used

- We built a Multilayer perceptron for the specified problem and conditions to predict multitarget labels.
- This sequential model built has one input layer with 30 input neurons, 2 hidden layers with 16,8 neurons respectively and output layer with 3 neurons.
- We also added dropout layers between each layers to overcome overfitting.
- At each layer, activation function used is tanh.
- Optimization function used is Adam with parameter values learning_rate=0.0001, beta_1=0.88, beta_2=0.911, amsgrad=False and other default values.

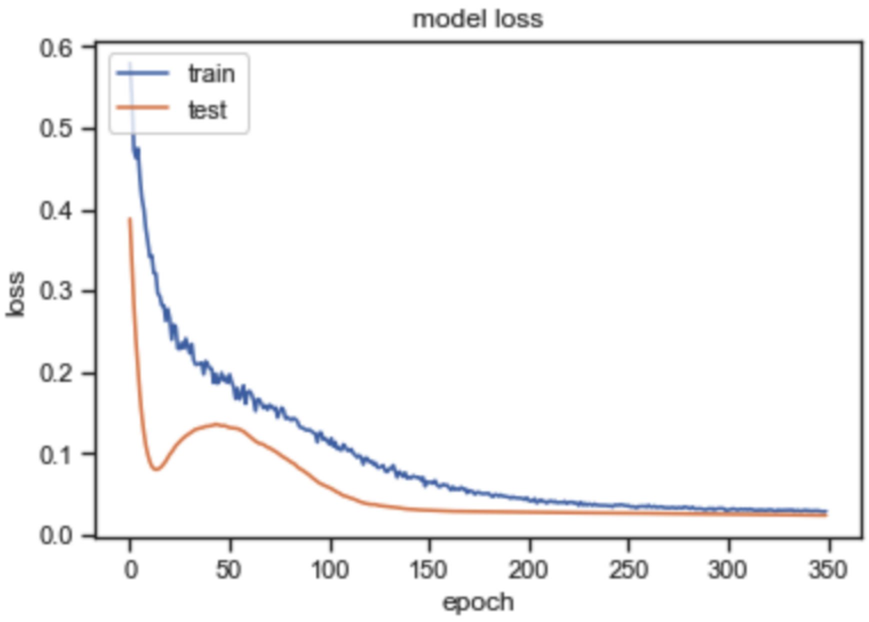


Figure 10: Model Loss

3.4 Results

- The trained model is saved into a file using **pickle** library .
- Using this model, we can load it and run the model on unseen data without training a model all time.
- Results are shown using metric mean square error[MSE] to show the loss/error.
- After running the model on training and test set, we got MSE =[0.02248380176602112 , 0.0376595427730712] respectively.
- However there is a little loss due to the positions we got from the eye tracker.