IIB Project Log Book

Junxiao Shen[a,b,c]

[a] *Trinity College*
[b] *Department of Engineering, University of Cambridge, UK*
[c] *js2283@cam.ac.uk*

Keyword: Human-Computer Interaction, Deep Learning, Machine Learning, Gesture Text Input, Augmented Reality

Abstract: This logbook will be a description of our work on a weekly basis. The link for the GitHub repository for the fourth year project is `https://github.com/shawnshenjx/FourthYearProject`

October

10/8

I read many pieces of literature on Human-Computer Interaction, especially in Augmented Reality and text entry methods. There are many problems to be addressed in the Augmented Reality control from a high level, such as an efficient and intuitive user interface, to a low level like image segmentation and registering. The machine learning method is a powerful and convincing way to boost the performance of the control in AR because numerous data is being transmitted between the environment and the hardware. Different machine learning methods can tackle various problems. Text input is one of the most effective ways to interact with computer systems so far, and different methods are used in different devices. For example, QWERTY keyboard has been one of the most popular text input keyboards in personal computers, whereas T9 predictive text was initially invented for users in the mobile phones considering the small user interface. After smartphones came onto the market, and the screen got bigger, the QWERTY keyboard started to become a very effective method for text input in smart mobile phones.

10/15

I discussed the project with my supervisor, and we made a project plan. In AR, the method for text input will be different since it will be in 3D space. Due to the inaccuracy for fingers to 'touch', the swipe method is now considered to be a more effective and more intuitive way to input text in AR. However, in QWERTY keyboard, due to the offset between the actual movement track and the sensed movement track of the fingers, there will be a difference between the distribution of the likelihood of the word given the gesture and the actual word we intended to input.

Therefore, I came up with two ideas to tackle this problem. One is to use T9 keyboard so to decrease the offset, and the other is to use a machine learning method to minimise the offset in the QWERTY keyboard. The advantage to having a T9 keyboard is that it only has 9 blocks so the offset might be small enough compared to the actual scale of the gesture tracking. However, the difficulty is that there is not a good gesture model for the T9 keyboard to have a right prediction on the word based on the minimal gestures since there are only nine blocks. Furthermore, it is not a standard text keyboard for users now, and it is against the intuition of users. The advantage of using the QWERTY keyboard is that it has an established mature gesture model together with the language model to give a good posterior distribution over the word. Therefore, considering the pros and cons of two different designs of the text board, we decided to use the QWERTY keyboard.

Due to the hardware inability in the HoloLens to accurately sense the trace of the figure, there is a big displacement between the trace of the figure tips we see in the user interface in HoloLens and the actual data input of the trace. Therefore, a recognition model to map the trace to word or sentences should be invariant of the starting position of the keyboard. Therefore, Deep learning can be used to train this recognition model.

Speech is a 1D temporal data and the trace is a temporal and spatial data. Therefore, the network models and state-of-art algorithms for speech recognition can be used to innovate us to design and train the neural networks for trace recognition.

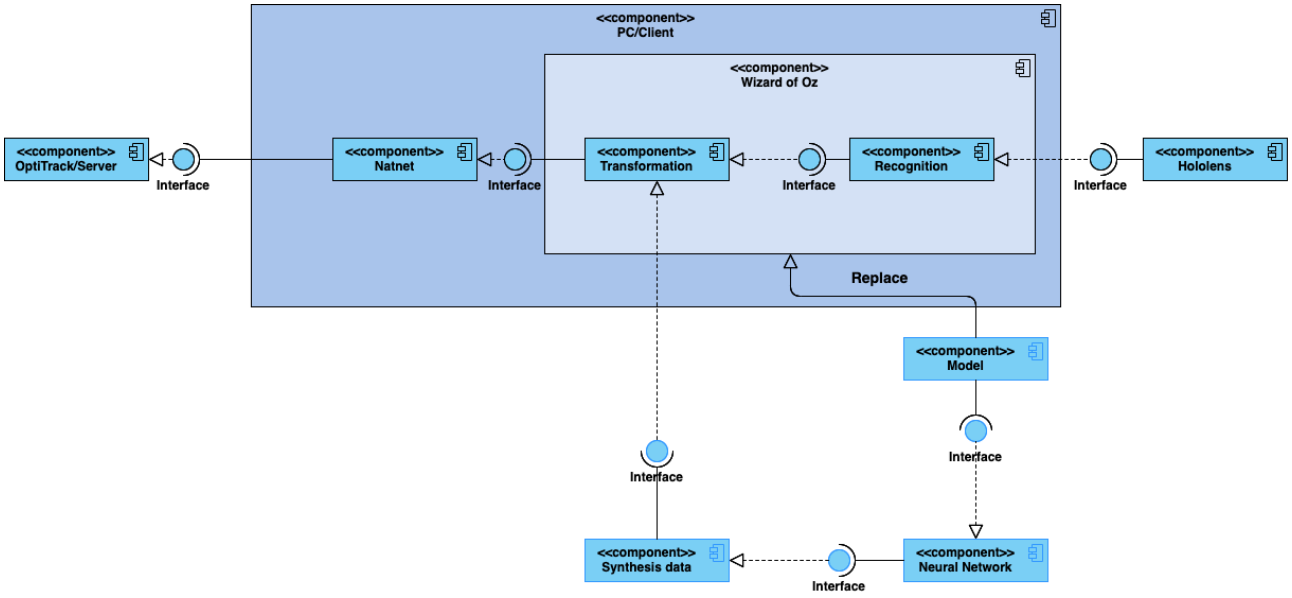

Figure 1: Component diagram for the development

A large amount of data should be used to train the deep neural network. However, it is not possible to collect this amount of gestures. Therefore, we need to synthesis artificial data. Furthermore, to have a deeper understanding of the gestures, an experiment to study the user habit for gesture input should be used to investigate how the user will input the gesture traces. Figure 1 shows the overall plan. It is a component diagram, and it is used to visualise and help us to have a better idea of the working process.

10/22

After we decided the general frame of the project, we began to start the project. The first thing is to learn about the transformation and know how to transform the position and orientation of the fingertip to the frame within the HoloLens. Therefore, extensive maths is required to compute the relative position and rotation.

10/29

After learning about the quaternion, which is an essential concept in computer vision, I began to do the actual math to compute the rotational matrix and translation matrix.

There are two different coordinate systems in the process. One is the OptiTrack system, and the other is the HoloLens coordinates. The OptiTrack is in the right-hand coordinates, and the Hololens is the left-hand coordinate. Furthermore, the Eular angle order is different in the two systems as well. In the OptiTreack Euler convention, it is pitch-yaw-roll whereas in the Hololens convention it is pitch-row-yall. The process of the transformation is in Figure 2. OptiTrack data includes the position and quaternion orientation of the fingertip and the markers placed on the HoloLens; HoloLens data includes the keyboard's position and Euler angles, the camera position and quaternion rotations. The camera is the Hololens which is how we see the virtual world in HoloLens. Therefore, there is an offset between the HoloLens marker and the Hololens. Details of the transformation process can be seen in the appendices.

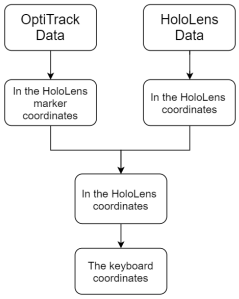

Figure 2: Workflow of transformation between coordinates

There are two systems, OptiTrack and HoloLens. In OptiTrack, the finger position and the rigid body positioned above HMD are recorded as fingerot_x,_y,_z and HMDOT_x,_y,_z with the quaternion being HMDOT_x,_y,_z,_w.

First Step

Transform finger position to rigid body position in OptiTrack. Transformed finger position is rotation-matrix*translation-matrix*finger-position.

Where rotation-matrix is the inverse (or transpose) of the rotation-matrix obtained from the quaternion through MATLAB toolbox (quat2mat), and this toolbox is right hand.

Translation matrix here is

$$\begin{bmatrix} 1 & 0 & 0 & -x \\ 0 & 1 & 0 & -y \\ 0 & 0 & 1 & -z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

where x,y,z is the position of the rigid body position.

Second Step

Transform finger position and keyboard position to the camera (HMD) coordinate in HoloLens. First of all, I transformed the position vectors all into right-hand coordinates by swapping the x and z positions. I then transformed finger position and keyboard position to camera coordinate using rotation-matrix*translation-matrix*finger-position. Rotation-matrix here is obtained from making camera quaternion into right-hand coordinates by using (qx,qy,-qz,-qw) and use the same method above to transform the quaternion to a rotation matrix. The translation matrix is similar to the above. Assuming the camera and rigid body match each other. Then now everything is in the HMD frame.

Third step

Transform everything (finger position from OptiTrack and from HoloLens) to keyboard frame. Same transformation applied using rotation-matrix*translation-matrix*finger-position. Now the rotation-matrix is using the keyboard quaternion and again, using (qw,-qz,-qy,-qx)

November

11/5

The rotational matrix and translational matrix were implemented using MATLAB, and the transformation was verified by using some collected raw data from OptiTrack and HoloLens.

11/12

OptiTrack system tracks the fingertip and the HoloLens by tracking the reflective balls attached to them. Firstly, The OptiTrack system, which is a motion capture system that can accurately track the coordinates of the markers, including HMD (Head Mounted Display), fingertips. The reason why we used OptiTrack was that the HoloLens inbuilt sensor is not good due to the large latency and offset, whereas OptiTrack is one of the world's most accurate and wide-area VR trackers. This leads to an ultra-low latency and better smooth tracking for HMD and fingertips. Then the data is transferred to PC by NatNet, which is an SDK for streaming motion tracking data across networks. The data received on the PC is transformed to the keyboard frame since the original data is in the world frame. The gesture trace is compared with the previous setpoints, once it approaches the region of the points, the PC will send a query with a world to the HoloLens, and it will output the word for the user.

There is a displacement between the balls and the HoloLens. How to minimise the displacement is another technical issue.

11/19

We successfully solved the displacement issue and implemented and verified all the maths within the OptiTrack system using MATLAB and its own software package called NatNet.

11/26

We now began to design the virtual keyboard system. The virtual keyboard system was designed using Unity and visual studio. We designed how to enable the user to interact with the keyboard and how to design the simulator.

We used the OptiTrack to track the position and orientation of the keyboard, figure tips and the HMD in the world frame. NatNet is used to get the data from OptiTrack, which is the server to send the data to the client. The client is the PC to run the script. A Wizard of Oz experiment will be designed so to mimic the perfect software for gesture input. Two data kinds will be used in the future to train the neural network. They are the trace of the figure tip and the trace of the gaze both in the keyboard frame. Firstly, a transformation software was written to transform the coordinates of the HMD and fingertips from world frame to keyboard frame. Secondly, a recognition script was written to recognise whether the figure tip has hit the region of the previous setpoints if they are within the region in the right order, then a word will be sent to the HoloLens. We decided to use 3D data of the fingertips instead of the 2D data that is projected on the keyboard.

December

12/3

We proposed several simulators and chose the one with the relaxed region. Then we implemented this simulated recogniser into the virtual keyboard system.

This is to recognise the position of the trace of the fingertips so that the Wizard of Oz system can tell whether this trace has successfuly matched the intended word and output the word in the HoloLens. There are two ways to recognise the trace, the first one is to put relax region around each of the intended word characters and if the trace has passed through the indented region, then the word will be output. The second way is to use a mathematical expression for the similarity between the experimental trace and the 'perfect' trace previously input; this measure of divergence can be auto-correlation. The first one is simple and easy to run and the second one method can also be used in the second stage quality control so this leads to more consistency. However, the second one will add more complexity, and thus the first one was used.

12/10

We began to combine the whole system together to make it a functional system.

There are two ways to construct the whole software, the first one is to use C all in one and the other one is to integrate MATLAB into C. They will have different advantages and disadvantages. The advantage of the first one is that it is a common practice that a project should be written in the same language. It will run faster as well if it is all in C. However, C does not have a strong community for scientific and deep learning support. Therefore, in the early stage of the project, it will be harder to gain support on those perspectives and it can potentially lead to a delay of the project, such as spending numerous time trying to find the right package and to debug. Integrating MATLAB to C, on the other hand, can be very efficient to develop for prototyping and there are many different ways to integrate them. Integration of the .NET Framework with MATLAB. There are three possible methods. Firstly, low-level C API; secondly, DDE; Thirdly, COM. C API is a low level and very hard solution but it is very efficient. This uses the magnet .NET library as a method to access MATLAB functionalities from .NET applications in a very efficient way. This is because MATLAB exposes itself as a COM Automation Server, so the interpretability of .NET COM can be used for its functions. However, the solution can be slow due to the COM type exchange. The second method is to use the dynamic data exchange even though this method is quite old, but it is still very powerful since it can let applications communicate and exchange data. The problem with the DDE exchange is the data types used to exchange information because it uses the Windows clipboard data types. The third methods are to use direct access to the MATLAB C API. This is the most powerful solution out the whole three. We chosed to use the COM approach.

- COM approach: This solution is quite slow because of the COM type exchange
- Dynamic Data Exchange: The Dynamic Data Exchange is a quite old but powerful service of Windows that lets applications communicate and exchange data
- C API: The direct access to the MATLAB C API is the best solution in terms of performance and features, just let use P/Invoke and some unsafe pointer operations.

12/17

We did the first user test and collected some raw data to preprocess.

12/24-1/7

Christmas holiday and working on other coursework/PhD application

January

1/7

We evaluated the raw data and the user experiment we did as a test. Then we redesigned the user experiment.

1/14

Based on the redesign of the user experiment, changes to the virtual keyboard system were also made.

1/21

Several bugs in the system were resolved to make the system have low latency and high smoothness, making the system user friendly and robust. This is to ensure the data collected is representative of the true gesture data.

1/28

We began to make slides so that we can present to the user on how to use the system and how the user experiment will be carried out. We also submitted the ethic application form to ask for the permission of the user experiments.

February

2/4

Phrases that we would like to collect was chosen from a combination of two phrase data set. A script was written to automatically select the phrases that meet my requirements.

2/11

We began to do user experiments. We collected data from 20 users.

2/18

Initial data analysis was done. We mainly work on the preprocess of the data to make the data easier to analyse in the future.

2/25

We proposed several micro metrics to measure the quality of the collected data and to analyse user behaviours.

March

3/4

The micro metrics were computed using python and they were analysed by drawing out the plots.

3/11

The project was completed and we began to work further. We began to work on the construction of the neural networks to generate synthetic gesture trace.

We thus used the data we collected from the user experiments to synthesis the data together with the data for the gaze collected from the experiment since we also believe that the trace of the gaze can be relevant to the final output as well. The synthesised data will be used to train the neural network, and the trained model will thus replace the recognition component in Figure 1. The reason to synthesis the data is that unlike speech recognition, the recorded labelled data can be accessed easily nowadays since people can pay others to read and label. Whereas in our project, the dataset is limited due to the lack of funding to hire a lot of people to generate the traces and due to the short time frame considering it is an one-year project.

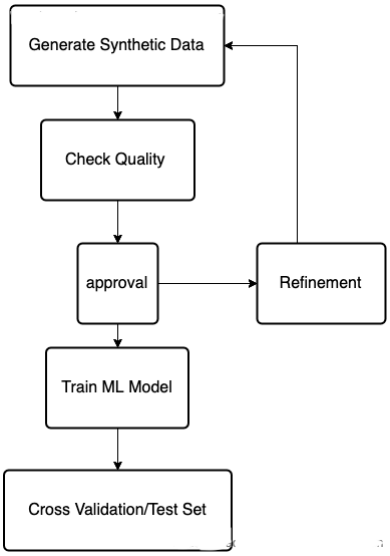

Figure 3: Work flow for data synthesis and neural network training

3/18

I was reading a lot of literature to think which structure is best for the generation of the gesture trace.

We generated data based on the data we obtained from the user experiments. There were many methods that I could use to generate the data. There are many existing generative models, one of the most popular ones is Generative Adversarial Networks[1] which use a generator produces training samples from the vector of random noise and use discriminator to classify whether a training sample is real or produced by the generator. However, this model is mainly used in computer vision, but this can still be of usefulness to our project. More relevant generative models in Text to Speech can be very important as well, such as WaveNet[2]. As can be seen from the workflow diagram in Figure 3, a quality check needs to be conducted to see whether the synthetic data is similar to real data. Different methods can be used such as calculating the KL divergence like TSNE (T-Distributed stochastic neighbour embedding) which learn low-dimensional embedding of feature vector by minimising KL between similarities in both spaces[3]. Classifier approach can also be used so that if it fails to classify the real data and synthetic data, then it is a good indicator.

Different neural networks can be used. I started from Recurrent Neural Networks (RNN) first since the trace of fingertips is a 3D spatial and temporal signal, and this can be in analogy to speech recognition. Furthermore, we also tried to combine CNN and RNN to see whether it would lead to better results. Transfer learning can also be used since the fine-tuning train machine learning model with synthetic data in first iterations and the with real data in further iterations. We should also be careful that we shouldn't use synthetic data in a test set since it can have a corresponding pair of real data which may lead to a dangerous result. Later, I decided to use an RNN with three layers of LSTM and one attention mechanism.

3/25

I constructed the RNN and began to train this deep neural network.

April

4/1

Different training techniques were added to improve performance.

4/8-5/8

Began to revise for the exam and apply for PhD scholarships

May

5/8

Began to write the final report

Appendix A. Conversions

$$\begin{bmatrix}\phi\\ \theta\\ \psi\end{bmatrix} = \begin{bmatrix}\arctan\frac{2(q_0q_1+q_2q_3)}{1-2(q_1^2+q_2^2)}\\ \arcsin(2(q_0q_2-q_3q_1))\\ \arctan\frac{2(q_0q_3+q_1q_2)}{1-2(q_2^2+q_3^2)}\end{bmatrix} \tag{A.1}$$

Appendix B. Rotation matrix

$$\begin{aligned}\begin{bmatrix}x\\ y\\ z\end{bmatrix} &= R_z(\psi)R_y(\theta)R_x(\phi)\begin{bmatrix}X\\ Y\\ Z\end{bmatrix}\\ &= \begin{bmatrix}\cos\psi & -\sin\psi & 0\\ \sin\psi & \cos\psi & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\cos\theta & 0 & \sin\theta\\ 0 & 1 & 0\\ -\sin\theta & 0 & \cos\theta\end{bmatrix}\begin{bmatrix}1 & 0 & 0\\ 0 & \cos\phi & -\sin\phi\\ 0 & \sin\phi & \cos\phi\end{bmatrix}\begin{bmatrix}X\\ Y\\ Z\end{bmatrix}\\ &= \begin{bmatrix}\cos\theta\cos\psi & -\cos\phi\sin\psi+\sin\phi\sin\theta\cos\psi & \sin\phi\sin\psi+\cos\phi\sin\theta\cos\psi\\ \cos\theta\sin\psi & \cos\phi\cos\psi+\sin\phi\sin\theta\sin\psi & -\sin\phi\cos\psi+\cos\phi\sin\theta\sin\psi\\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta\end{bmatrix}\begin{bmatrix}X\\ Y\\ Z\end{bmatrix}\end{aligned} \tag{B.1}$$

References

[1] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative Adversarial Networks, arXiv e-prints (2014) arXiv:1406.2661arXiv: 1406. 2661.

[2] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, K. Kavukcuoglu, WaveNet: A Generative Model for Raw Audio, arXiv e-prints (2016) arXiv:1609.03499arXiv: 1609. 03499.

[3] L. van der Maaten, G. Hinton, Visualizing data using t-SNE, Journal of Machine Learning Research 9 (2008) 2579–2605.