Prediction of Head Movements in a Virtual Reality Setting

Hüseyin Camalan

Abstract

Virtual Reality (VR) is a rapidly trending development that allows users to experience visual stimuli in a 360° setting. It may constitute a more realistic way of studying visual saliency due to free movement of the head. Conversely, saliency research is not only valuable in itself as a means to understand cognitive agents, but also for practical reasons such as improving the quality of experience in VR settings. This lab rotation project involved setting up an experiment, which combined the use of VR glasses and eye trackers to simultaneously record head and eye movements while participants watched 360° videos of various themes. The data were then preprocessed and modeled using some basic regression methods in order to predict head motion. These results suggest that head motions are predictable to an extent given simple information, i.e. head poses of other users given the same video.

Introduction

Figure 1: A VR headset.

Virtual Reality (VR), the concept of immersing the user in a different "reality", has concretely been around since 1960s, with the invention of Sensorama, which attempted to give its users a realistic experience of biking in Brooklyn (Heilig, 1962). Though the Sensorama attempted to utilize multiple senses at once, such as tactile, smell, sound and vision; more modern VR solutions are mostly centered around the visual sense. Current VR technology involves the use of omnidirectional cameras for recording spherical videos or computer software for creating virtual rooms. Then these rooms are displayed using head-mounted displays which have spatial sensors for recognizing the user's orientation. The sensor readings from the displays are used to show the corresponding field from the 360° video to the user, thus creating the perception of being "submerged" in an alternate reality.

VR technology is becoming ever-more accessible to the public as the prices of head mounted displays (also referred to as VR glasses or VR goggles) gradually decrease. This is stimulating the multimedia and gaming industries since these are the most straightforward enhancements VR could bring to the public experience. However, more relevant to this paper is the use of VR in scientific contexts, and in this specific case, saliency research. Saliency detection is the act of assigning importance to elements or objects in the environment (Itti, Koch, & Niebur, 1998). As a simple example: in an empty dark room with a source of flickering light, the flickering light source would have a higher salience compared to the rest of the scenery, as it stands out with its strikingly different visual properties. Most if not all animals do salience detection successfully to an extent, even though it can be a rather complex computational task. Therefore, creating successful artificial models of salience and successfully predicting natural salience patterns can bring forth a substantial leap to understand cognition as well as practical applications.

Saliency modeling is typically kept only in the visual domain because it is the simplest and most straightforward to study (as opposed to e.g. smell). Typically, once a model of saliency is created, it is then tested on real agents (for an example, see Hou & Zhang, 2007). This involves predicting areas of fixation in visual scenes and then comparing these predictions to where the real agents actually fixated. The latter part happens in a vision laboratory with an eye tracker attached to a monitor, and the subjects fix their head through the use of a head rest while they are watching visual stimuli. It could be argued that this setting is rather reductivist, since agents are not enforced to interact with the world through a computer monitor in their natural environments. Thus, use of VR might ultimately create more realistic fixation patterns by allowing the free movement of the head. There exists evidence confirming a relationship between head and eye movements (Zangemeister & Stark, 1982).

From a reversed (and more practical) perspective, it could also be argued that saliency research could benefit VR technology as well. Material that is presented with VR is more resource-intensive than a conventional display, due to the increased surface area associated with a sphere. However, the human field of view is rather narrow, i.e. around 114° for binocular field (Howard & Rogers, 1995). Therefore, a lot of this information is not accessible to the retina. Additionally, given that the visual system of humans highly prioritizes foveal vision compared to peripheral vision, areas not directly fixated upon are not processed sharply (for a review, see Strasburger, Rentschler, & Jüttner, 2011). Thus, with a perfect saliency model, the amount of information to be downloaded would be orders of magnitude smaller; and with a modestly successful model, substantial reductions would still be achieved. As a modest example, successfully predicting that a user will look inside one hemisphere and then reducing the pixels in the other hemisphere by half, would result in a 25% reduction in total data load.

This lab rotation project was part of a greater effort to study saliency using a VR setting. More specifically, this project consisted of conducting an experiment, in which participants watched 360° videos using a VR headset as their head and eye movements were recorded. This was made possible by using VR glasses in conjunction with new generation eye tracking hardware that is extremely compact and can be fitted inside glasses. The data was then preprocessed to be suitable for analysis and some preliminary analysis was made on the dataset.

Materials and Methods

Participants

Ten adult (μ_{age} = 27.2 years, SD_{age} = 1.77) participants were recruited through social media for the experiment. Requirements for joining the experiment were intact corrected or uncorrected vision as well as proficiency in English or German. Participants were allowed to wear contact lenses and glasses, but were asked to avoid wearing mascara before the experiment. Each participant was paid €9 per hour. Two experimenters also participated in the study.

Materials

Hardware

HTC Vive[1](pictured in fig. 1) VR glasses were used in conjunction with compact eye trackers from *Pupil Labs* (Kassner, Patera, & Bulling, 2014). The eye trackers were fastened to the insides of the VR headset, at the bottom medial position for each eye.

Software

Multiple software solutions were combined to conduct the experiment. Firstly, the gaming platform *Steam*[2] along with its VR package (*Steam VR*) was installed to interact with the VR headset. Installation of *Steam* allowed other additional pieces of software that introduced new functionalities. Among these were (i) *OpenVR*, with which the sensor readings of the VR headset were broadcasted to the host computer; and (ii) *Whirligig Media Player*[3], which played the 360° videos on the VR headset and simultaneously broadcasted relevant information on the video being played. The *Pupil Labs* eye tracker had its own proprietary software, *Pupil Capture*, that allowed access to the eye trackers along with other functionalities like setting parameters for recording eye movements, calibrating the algorithm to correctly assign gaze positions and broadcasting eye gaze positions to the computer. The broadcasts from *Pupil Capture*, *OpenVR* and *Whirligig Media Layer* were captured, synchronized and saved in the data storage using *Lab Streaming Layer*, or *LSL* (Korte, 2012).

For the programming and analysis of the experiment, the *Python* programming language[4] was used, along with the additional external libraries: *NumPy* (Oliphant, 2006) for numerical computing, *Pandas* (McKinney, 2010) for database manipulation, *Scikit-learn* (Pedregosa et al., 2011) for the machine learning algorithms as well as *Matplotlib* (Hunter, 2007) for plotting the figures. For analysis with the quaternions, external python libraries *Pyquaternion* (Wynn, 2015) and *Quaternion* (Boyle, 2018) were utilized.

Stimuli

A large number of 360° videos of various categories were collected from public sources on the internet, namely *YouTube*[5] and *Vimeo*[6]. These categories can be roughly categorized as short documentaries, artistic renditions, movie trailers or short movies, extreme sports and nature sightings.

[1] http://www.vive.com [2] http://www.valvesoftware.com [3] http://www.whirligig.xyz
[4] http://www.python.org [5] http://www.youtube.com [6] http://www.vimeo.com

Among these videos, those in which the camera perspective moved very fast, as well as those which contained violent content, were removed. Of the remaining videos, 50 were picked to ensure a variety of different video categories and to provide interesting content for the participants.

Three of the videos were trailers of upcoming horror movies. Since they were not violent, they were kept in the experiment. However, the erratic nature of the content still proved to be difficult to watch for one subject. This subject was thus exempted from watching these videos and these videos were later removed from the analysis. Additionally, another video was removed because of difficulties with regards to its codec settings.

Procedure

Subjects were introduced to the laboratory and informed about the experiment upon their arrival. Any questions they posed were answered prior to handing the consent form. Upon signing the consent form, the experiment was initiated.

The VR headset had to be taken off (due to user fatigue or overheating) and put back on each session. Therefore, the eye tracker had to be calibrated each time for correct assignment of the eye gaze. A custom calibration procedure (and validation thereof) was programmed and communicated with *Pupil Capture* through its API.

The calibration procedure involved looking at some flashing points (gaze locations) on the screen, whose locations were predetermined. These locations were then passed to *Pupil Capture* for it to correct its calculations. The calibration procedure was performed with the subjects seated, their elbows on a table and their hands supporting their head in order to minimize head movement. Subject head movements were monitored to restrict head orientation changes to at most 1-2° in each plane.

The validation procedure was exactly the same as the calibration, but using a different background brightness (causing changes in the pupil diameter of the subjects) and with different gaze locations. Once the subjects looked at this second set of points, their gaze points (given the corrected calculations) were plotted in conjunction with the fixation points. Given the results, the experimenter decided whether to start the session or repeat the calibration procedure once more.

For the first two subjects (the experimenters), the chair of the subjects were also rotated by 100° during the validation procedure. This was done to ensure that visual scale changes in VR; namely, the magnification of a scenery as a subject moves in a direction, didn't affect the calibration. Later on, when a functionality was discovered to counteract the magnification, this practice was discontinued.

Eyelashes often presented a challenge for the experiment, as the eye tracker algorithm mistook the eyelashes for the pupils, especially on subjects with darker complexion/darker hair color. When this problem persisted, subjects were asked to apply liquid chalk on their eyelashes with a mascara brush (at the consent of the participants). The purpose of this was to reduce the visual contrast between the eyelashes and the skin so that the contrast between the iris and the white of the eye was relatively stronger for the eye tracking algorithm.

Analysis

Preprocessing

As mentioned earlier, the stitching together of multiple broadcasts from various hardware was achieved by *LSL*, which also synchronized them using the system time.

Since system time tends to be quite precise, the dataset was sparse, because while one broadcast gave a value, the other systems were usually silent in that small time window. Thus, it was a necessity to approximate these values. This was done separately for each broadcast given its properties. For example, for video time, the values were interpolated, since we could assume that there was a linear relationship between system time and video time. However, as saccadic eye movement is not linear, the eye gaze data were padded (empty values were replaced by the last known value).

Corrupted data were removed from the dataset. The cause for data corruptions was a disconnection with a broadcast, which could arise from many different sources (e.g. a cable unfastening, overheating of the VR sensors or software failure).

At the end of the preprocessing, a dataframe with the following columns was obtained: system time, subject ID, session ID, eye coordinates (x and y), confidence of assignment of eye gaze, spatial coordinates of the VR headset (x,y,z), four-dimensional quaternion representation of the head pose, video ID and video time. The choice of quaternions over Euler angles or rotation matrices is due to the robustness and the low dimensionality of the former.

A final adjustment was the downsampling of the dataset. This was due to a multitude of reasons. Firstly, the division of data points into inputs and outputs for the prediction procedure required that chunks of the database had to (i) have the same array shape and (ii) have their rows correspond to the same times of the video. Secondly, considering that the output from the eye tracker was noisy, it was thought that averaging these values would be more reliable to give the location of salient locations in the visual scenery. Thus, data binning was applied, with a bin interval of 1 second and inter-bin interval of 0.5 seconds. Inside the bins, the gaze positions as well as the head poses were averaged.

To take the averages of head poses inside a bin, Spherical Linear Interpolation, or SLERP (Shoemake, 1985) was applied to quaternions. Although SLERP can average only two quaternions at once, it can do so in a weighted fashion. Therefore, the average of multiple quaternions was done by iterating the SLERP algorithm serially over the quaternions while gradually shifting the weights in favor of the quaternion that contains more of the initial quaternions within itself.

Prediction

Various regression models were used to predict head orientations of users watching 360° videos. Regression was done by feeding into the models where the other users looked at the same time in that video. In this sense, it could be said that each model tried to learn the relation of how the tested subject compared to other subjects, i.e. finding a given subject's systematic viewing preferences. The models were trained with all videos except the one being tested, thus learning how a subject's viewing behavior compares to that of others. Once training was finished, the model was given other users' gaze locations and head poses for the video being tested. The model then returned predictions for an array of head positions for given subject-video pairing (for a graphical description, see fig. 2). The head gaze predictions are supposed to be unit quaternions, thus each prediction

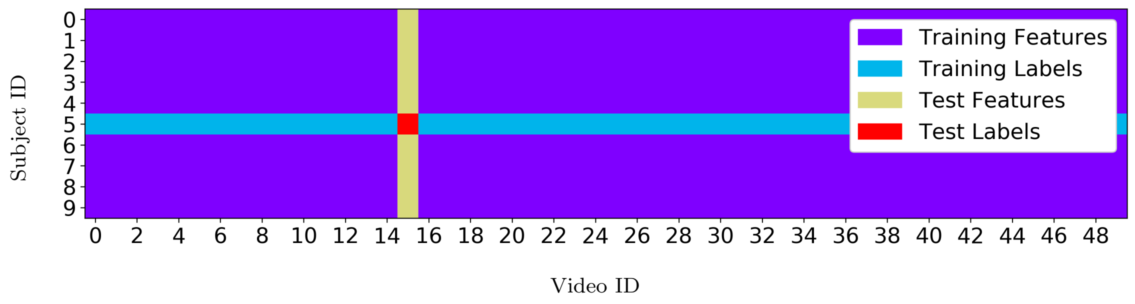

Figure 2: Example partitioning of the dataset for supervised learning. This procedure is repeated ($\#subjects \times \#videos$) times, with each subject-video pairing set as the test label once.

quaternion was divided by its norm.

Head poses (4-d) as well as eye gaze locations (2-d) from all subjects except the tested one were passed as input to the models. The output was an array containing the head pose predictions of the tested subject. Thus, these models contained 54 input dimensions (9 subjects$\times$6 dimensions), and 4 output dimensions.

Similar to a leave-one-out crossvalidation method, the regression procedure was repeated separately for each subject-video pairing.

Four types of models were separately developed and tested in the aforementioned manner: linear regression (Bishop, 2006), linear support vector regression (Smola & Schölkopf, 2004), multilayer perceptron regression (Haykin, 1994) as well as a very basic k-nearest neighbors regression (Altman, 1992) that is equivalent to simple averaging. The parameters and specifications of the models were as follows:

- Multilayer perceptron regression (MLPR): one hidden layer with 100 units, ReLu activation function for hidden layer units, no activation function for the output, constant learning rate, 200 iterations.

- Linear regression (LR): ordinary least squares, axes not normalized individually (to keep the quaternions intact), intercept is calculated.

- Linear support vector regression (LSVR): epsilon set to 0, crossing penalty parameter C set to 1, epsilon-insensitive loss function, intercept was calculated. Since SVR does not allow multi-output regression, each label axis was predicted separately.

- k-nearest neighbors regression (kNNR): k=9, with each subject not being predicted counted as a neighbor. No optimization for optimal k was made. Averaging (not weighted) was done across the quaternions corresponding to the neighbors. For averaging quaternions, the SLERP algorithm as described above was utilized. No gaze data were relayed to this model.

The predictions from these models were compared to the experimental data obtained from participants.

Results

Figure 3 shows the averaged geodesic distance (on a unit sphere) of the linear regression analysis for each video. The vertical axis shows the average geodesic distance

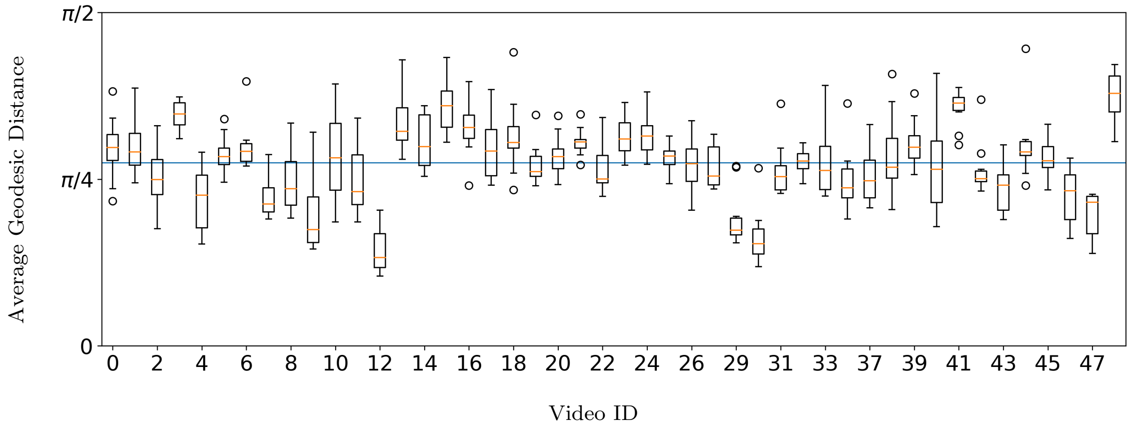

Figure 3: Boxplot of prediction errors for videos (averaged across subjects) for the LR model. Blue line shows the grand average.

between the predicted and real head poses. Red horizontal lines indicate the group median for a given video, the boxes indicate the quartiles neighboring the median, ends of the whiskers indicate the furthest non-outlier data points (1.5 inter-quartile ranges away from median) and the circles indicate outliers. The grand average indicated by the blue line corresponds to roughly $\pi/4$ distance. In this sense, the normalized error of the LR model lies between 20%-40%, with the grand average being 27%.

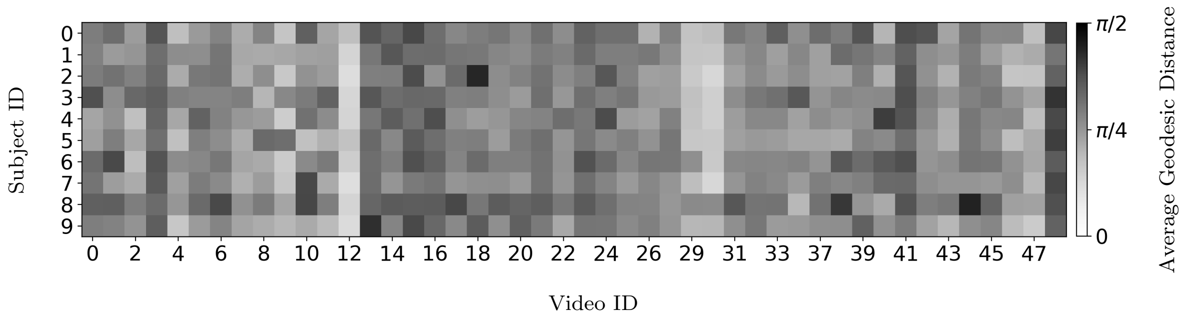

Figure 4: Heatmap of error estimates for each subject and video for the LR model.

Figure 4 shows a heatmap of the same information; this time with prediction performance of individual subject-video pairings. Colors of the heatmap correspond to the prediction performance. The vertical axis shows the subjects and the horizontal axis shows the videos as before. On visual inspection, it can be seen that prediction performance tends to be the similar across subjects, but not across videos. This is confirmed by averaged standard deviation of 0.12 across subjects vs. 0.19 across videos.

Figure 5 shows a comparison of various models as well as the baseline predictions from averaging. It can be seen that the averages, LR and LSVR have performed very similarly (27-28%), whereas MLPR seems to perform slightly worse (32%). With that being said, all models perform above chance levels.

Figure 6 shows a comparison of various models similar to 5, but for the prediction of a single subject-video pairing. Upon visual inspection, it can be seen that all models predict very similarly to each other; however, there are some marked differences between the model predictions and real data. In the case of LR and LSVR, the predictions are almost identical. However, all the predictions differ somewhat from the experimental

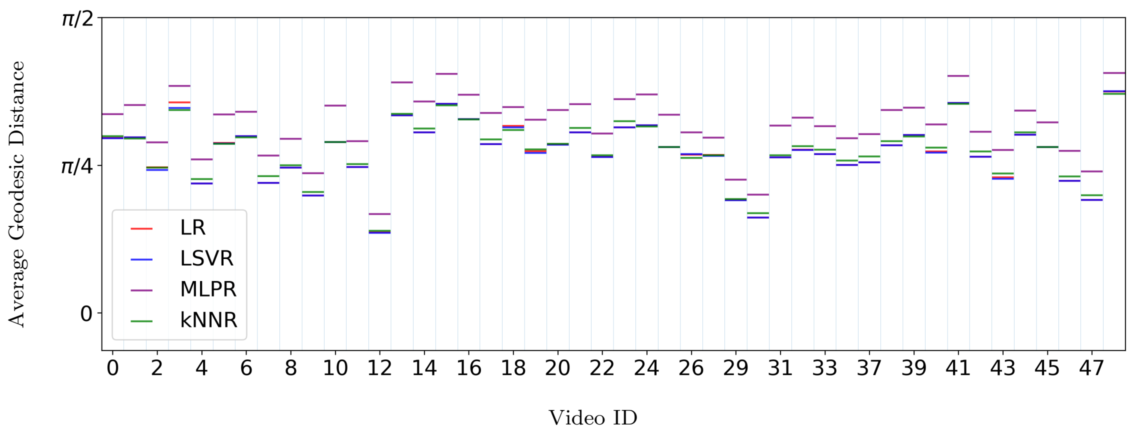

Figure 5: Comparison of models given their prediction performance across videos (averaged across subjects). LR: Linear Regression. LSVR: Linear Support Vector Regression. MLPR: Multilayer Perceptron Regression. kNNR: k-nearest neighbor regression.

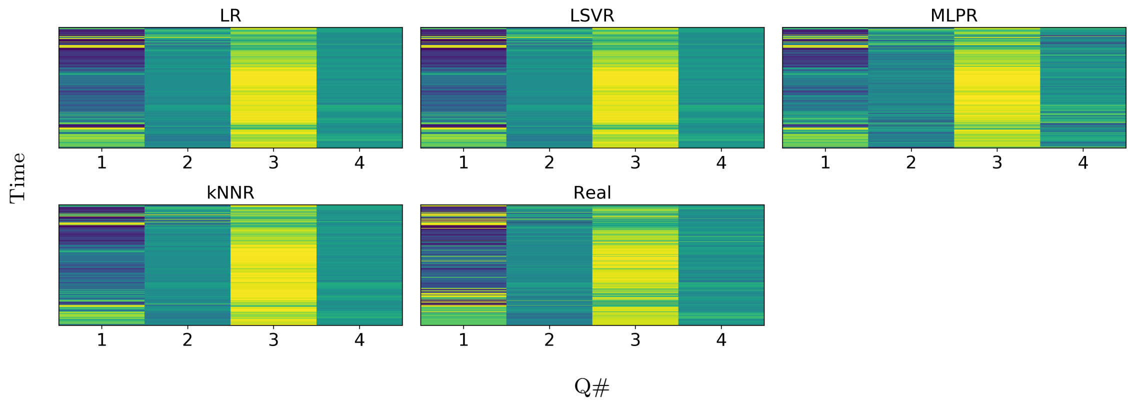

Figure 6: Comparison of model performances for a given subject-video pairing. Vertical axis shows video time (ascending), while each column on the horizontal axis indicates the individual values of the quaternion vectors. LR, LSVR, MLPR, kNNR: same as above. Real: Experimental data.

data.

Discussion

This lab rotation project involved an experiment in which head poses of participants viewing 360° videos were attempted to be predicted with simple machine learning algorithms.

Results of all analyses indicate that prediction rate is variable, but above the chance level for all videos. If we consider head orientations as locations on a unit sphere, chance level would indicate a $\pi/2$ distance between the predicted and true head orientations. This is derived from half of the maximum distance two locations can have on the sphere, which is π. Given infinite pairs of orientations picked randomly from a uniform distribution, the average distance amounts to half of the sum of the maximal and minimal distances, which is $(\pi + 0)/2 = \pi/2$.

With regards to comparisons between different model types, LR and LSVR performed almost identically, while the MLPR model performed slightly worse. The reason for the worse performance of the neural network may be due to not having enough iterations to converge. The resemblance between LR, LSVR and kNNR is peculiar. One possible explanation for this phenomenon is that all models tend to converge towards averaging other users' head poses for a given video. If true, this would imply that all models learn to do their own version of quaternion averaging. The similarity of the model predictions to the subject averages (kNNR) in fig. 6 seems to confirm this suspicion, because the kNNR model was given no eye gaze data.

From the above conclusion, it follows that including the eye gaze information as input to the models doesn't result in a meaningful contribution to the prediction of head movements. Thus, the current approach of this project fails to reflect the interactions between head and eye movements. It is possible that any relevant information disappeared (i) inside the noise of the eye gaze data or (ii) during the eye gaze averaging of the binning procedure.

As a final point, some of the videos appear to be predicted better than others. This might be due to artifacts regarding to the content of the video. Qualitatively speaking, some 360° videos are produced in a way that most of the relevant information is still at the center orientation. However, this is only speculation and further analysis may be necessary to confirm this suspicion. A more optimistic explanation would be that, in the well-predicted videos, viewers' head poses are more consistent, such that the model has an easier time predicting unknown head poses.

In conclusion, predicting saliency patterns of a viewer from the usage information of other viewers appears possible to a certain extent. Although it is unknown what exactly the models learn, it appears that model predictions are made by averaging the head poses from other subjects.

Future Directions

As mentioned above, although prediction performance was somewhat successful, there is still a large room for improvement. Nonlinear models tend to be more powerful than linear models and give better performance when appropriate models and model parameters are chosen. In this sense, a nonlinear model could result in a better prediction rate. Although a simplistic feedforward neural network model was used in this analysis, there are recurrent neural network variants that are arguably more suited for time series prediction, such as long short-term memory networks (Hochreiter & Schmidhuber, 1997).

Another potential argument to be made is there is more possible information to be fed into the classifier. As stated earlier, only the viewing information of users (where they looked at what time) is evaluated in this analysis. Typically, visual saliency is computed through the features inside the content, which this study did not utilize. Since the time information of each video is available in our dataset, it is possible to extract frames from videos and create saliency maps. However, since videos are being dealt with instead of still photographs, the computational load of such an analysis would be substantially higher. With that being said, it is likely that a successful model will rely on an ensemble of information sources rather than just one source.

Once the head movement can be better predicted, the logical progression is the prediction of the exact point at which users are looking. The ultimate practical application in this case would be on-line prediction of user head orientations and gaze.

References

Altman, N. S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, *46*(3), 175–185.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning.* Springer.

Boyle, M. (2018). *Quaternion.* Retrieved from `https://github.com/moble/quaternion/blob/master/LICENSE` (Online, accessed on 05.10.2018)

Haykin, S. (1994). *Neural Networks: a Comprehensive Foundation.* Prentice Hall PTR.

Heilig, M. L. (1962, August 28). *Sensorama Simulator.* Google Patents. (US Patent 3,050,870)

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.

Hou, X., & Zhang, L. (2007). Saliency detection: A spectral residual approach. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on* (pp. 1–8).

Howard, I. P., & Rogers, B. J. (1995). Binocular vision and stereopsis. In (p. 32). Oxford University Press, USA.

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing In Science & Engineering*, *9*(3), 90–95. doi: 10.1109/MCSE.2007.55

Itti, L., Koch, C., & Niebur, E. (1998). A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Kassner, M., Patera, W., & Bulling, A. (2014). Pupil: An open source platform for pervasive eye tracking and mobile gaze-based interaction. In *Adjunct Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing* (pp. 1151–1160). New York, NY, USA: ACM. doi: 10.1145/2638728.2641695

Korte, C. (2012). *Lab Streaming Layer.* Retrieved from `https://github.com/sccn/labstreaminglayer` (Online; accessed 28.09.2018)

McKinney, W. (2010). Data structures for statistical computing in Python. In S. van der Walt & J. Millman (Eds.), *Proceedings of the 9th Python in Science Conference* (p. 51 - 56).

Oliphant, T. E. (2006). *A guide to NumPy.* USA: Trelgol Publishing. Retrieved from `http://www.numpy.org/` (Online; accessed 28.09.2018)

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Shoemake, K. (1985). Animating rotation with quaternion curves. In *ACM SIGGRAPH Computer Graphics* (Vol. 19, pp. 245–254).

Smola, A. J., & Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and Computing*, *14*(3), 199–222.

Strasburger, H., Rentschler, I., & Jüttner, M. (2011). Peripheral vision and pattern recognition: A review. *Journal of Vision*, *11*(5), 13–13.

Wynn, K. (2015). *Pyquaternion.* Retrieved from `https://github.com/KieranWynn/pyquaternion/blob/master/LICENSE.txt` (Online, accessed on 05.10.2018)

Zangemeister, W., & Stark, L. (1982). Types of gaze movement: variable interactions of eye and head movements. *Experimental Neurology*, *77*(3), 563–577.