DGaze: CNN-Based Gaze Prediction in Dynamic Scenes

Zhiming Hu, Sheng Li*, Congyi Zhang, Kangrui Yi, Guoping Wang*, Dinesh Manocha

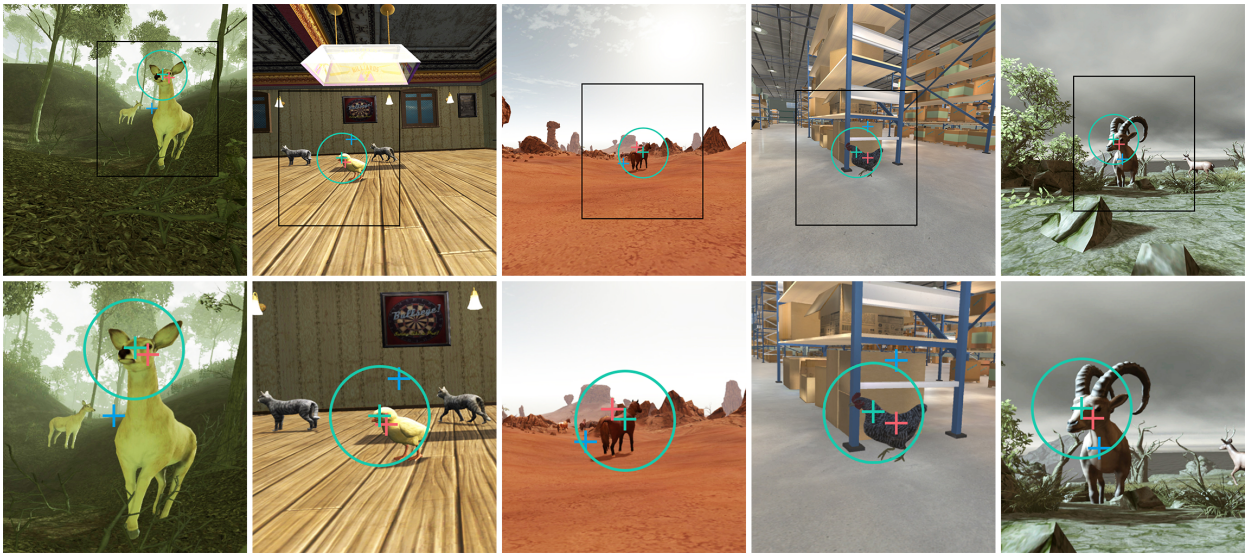

Fig. 1: Gaze prediction performance of our model (DGaze) in different scenes. The upper row shows some captured images from an HMD's screen and the lower row exhibits the corresponding enlarged view. The green cross denotes the ground truth of gaze position, the blue cross is generated using SGaze [21] algorithm, and the red cross represents the result of our novel algorithm, DGaze. The green circle illustrates the foveal region with radius 15°. In practice, our gaze prediction algorithm exhibits higher accuracy than prior method.

Abstract—We conduct novel analyses of users' gaze behaviors in dynamic virtual scenes and, based on our analyses, we present a novel CNN-based model called DGaze for gaze prediction in HMD-based applications. We first collect 43 users' eye tracking data in 5 dynamic scenes under free-viewing conditions. Next, we perform statistical analysis of our data and observe that dynamic object positions, head rotation velocities, and salient regions are correlated with users' gaze positions. Based on our analysis, we present a CNN-based model (DGaze) that combines object position sequence, head velocity sequence, and saliency features to predict users' gaze positions. Our model can be applied to predict not only realtime gaze positions but also gaze positions in the near future and can achieve better performance than prior method. In terms of realtime prediction, DGaze achieves a 22.0% improvement over prior method in dynamic scenes and obtains an improvement of 9.5% in static scenes, based on using the angular distance as the evaluation metric. We also propose a variant of our model called DGaze_ET that can be used to predict future gaze positions with higher precision by combining accurate past gaze data gathered using an eye tracker. We further analyze our CNN architecture and verify the effectiveness of each component in our model. We apply DGaze to gaze-contingent rendering and a game, and also present the evaluation results from a user study.

Index Terms—Gaze prediction, convolutional neural network, eye tracking, dynamic scene, gaze-contingent rendering, virtual reality

1 Introduction

With the development of virtual reality (VR) technology, users' gaze information in VR becomes increasingly important and can be used for different applications, including VR content design [39], VR content compression [39], eye movement-based interaction [14, 29, 34, 43], gaze behavior analysis [2, 4, 20, 21, 39], gaze-contingent rendering (or foveated rendering) [18, 33, 41, 42], etc. Currently, the most common solution for eye tracking is based on hardware-based eye trackers. Eye trackers are accurate and can be integrated with head mounted devices (HMDs). However, as an accessory equipment of an HMD, an eye tracker can be costly and lacks ease of use. Moreover, eye trackers are mainly used to measure the current gaze direction, and they cannot directly predict the future gaze position. Therefore, there is considerable interest in developing alternate methods for eye tracking and gaze prediction [21, 39, 45].

Human gaze behaviors in dynamic scenes are more intricate than that in static scenes because users' attention will be attracted not only by static stimuli but also by moving stimuli [1, 17]. In VR applications, dynamic scenes are much more common than static scenes. However, currently, many studies have focused on static scenes [21, 36, 39] while dynamic scenes have not been well explored. Therefore, it is very meaningful to analyze human gaze behaviors in dynamic scenes. One

- *Zhiming Hu, Sheng Li, Kangrui Yi, Guoping Wang are at Peking University, China. E-mail: {jimmyhu | lisheng | yikangrui | wgp}@pku.edu.cn.*
- *Congyi Zhang is at both The University of Hong Kong, China and Peking University, China. E-mail: cyzh@hku.hk.*
- *Dinesh Manocha is at University of Maryland, USA. E-mail: dm@cs.umd.edu.*
- *Sheng Li and Guoping Wang are the corresponding authors.*

- *Project URL: cranehzm.github.io/DGaze*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

of our goals is to analyze the gaze behavior for dynamic scene and use that behavior to design better prediction algorithms.

Our model is formulated based on the following procedures:

Gaze Data Collection: We record users' eye tracking data in dynamic scenes to analyze their gaze behaviors. Specifically, a total of 43 participants are asked to freely explore 5 dynamic virtual scenes including both indoor and outdoor scenes (See Fig. 2) and the participants are given no specific task. We collect users' data from the eye tracker to build a dataset. Our dataset includes 86 pieces of data from 43 users and each piece of data contains about 18,000 gaze positions, 18,000 object positions, 36,000 head velocities, and 10,800 frames of scene screenshots. The dataset has been released for public usage[1].

Gaze Behavior Analysis: We further analyze users' gaze behaviors based on our dataset. Specifically, we perform Spearman's rank correlation analysis [30] on the dataset and find that users' gaze positions are correlated with their head rotation velocities and the positions of dynamic objects. We also extract the saliency maps of the images viewed by observers and find that gaze positions correspond with salient regions. The above results reveal that head velocity, dynamic object position, and saliency feature facilitate the task of gaze prediction. The effectiveness of each component is further validated in our ablation study (Sect. 6.4). In addition, we analyze the distribution of users' gaze data and find that most of the gaze positions lie in the central region of the HMD's screen.

Novel CNN-Based Model: Based on our analyses, we propose a CNN-based model (DGaze) to predict gaze positions. Our model consists of three modules: sequence encoder module, saliency encoder module, and gaze prediction module. The sequence encoder module employs a 1D CNN layer to encode the object position sequence and the head velocity sequence. The saliency encoder module applies the state-of-the-art SAM-ResNet saliency predictor [13] to extract saliency features of the past images and utilizes a fully connected (FC) layer to encode the features. The gaze prediction module combines the outputs of the 2 modules and employs 3 FC layers to predict gaze positions. When an eye tracker is available, the sequence of past gaze data can also be used in the sequence encoder module to improve our model's performance. We refer to this revised model as DGaze_ET as it utilizes the data gathered using an eye tracker. We utilize 60% of our dataset (1045654 data points) to train the model and employ the remaining data (698304 data points) to test. The effectiveness of our CNN architecture is further validated (Sect. 6.4). The source code of our model and the pre-trained models are publicly available[1].

Evaluation of DGaze: We utilize the angular distance between the ground truth and the predicted gaze position as the evaluation metric (Sect. 6.1.2). DGaze achieves good realtime prediction performance on both dynamic (an improvement of 22.0% over prior method) and static datasets (9.5% improvement). DGaze can also be applied to predict gaze positions at any specified future time (0$\sim$1000 *ms*). When an eye tracker is provided, DGaze_ET can take advantage of accurate past gaze data to achieve higher performance. We apply DGaze to gaze-contingent rendering and conduct a two-alternative forced choice (2AFC) test to verify its effectiveness. Compared with prior method, DGaze is preferred by 68.9% of the users' responses and the result is statistically significant. We also apply DGaze to a game to evaluate its effectiveness and the result shows that DGaze outperforms prior method by 14.3% in terms of realtime prediction.

Overall, our contributions include:

- A novel CNN-based model (DGaze) for realtime and future gaze prediction in immersive HMD-based applications with high accuracy.
- Analyses of human gaze behaviors in dynamic virtual scenes that provide insights into formulating gaze prediction models.
- A dataset that contains 43 users' eye tracking data that is gathered using five dynamic scenes.

[1] cranehzm.github.io/DGaze

2 Related Work

In this section, we give a brief overview of prior works on gaze prediction, gaze behavior analysis, and the applications of eye tracking technology.

2.1 Gaze Prediction

In the area of vision research, gaze prediction or visual saliency prediction has been well-studied and many gaze prediction models have been proposed in the last three decades. Generally, most of the existing models are based on bottom-up approaches, top-down approaches, or hybrid approaches. Bottom-up approaches employ low-level image features such as intensity, color, and orientation to predict visual attention [10, 24] while top-down approaches take high-level features of the scene like specific tasks and context into consideration [8, 19]. Hybrid approaches combine low-level features and high-level features to obtain better performance [7, 32]. Recently, with advances in deep learning, many deep learning-based models have been proposed and have achieved good performances [13, 26].

However, there is limited work on gaze prediction in the area of virtual reality. Sitzmann et al. [39] and Rai et al. [36] both focused on saliency in 360° static images. They collected users' eye tracking data and predicted saliency maps of the scenes. Xu et al. [45] built a dataset that contains observers' gaze data in dynamic 360° videos and proposed a deep learning-based model for gaze displacement prediction. Koulieris et al. [25] predicted gaze object categories in a video game. Our approach is also related to the recent work on SGaze [21]. Hu et al. gathered the data of a large number of participants freely exploring static virtual scenes and proposed an eye-head coordination model called SGaze for predicting users' realtime gaze positions. In contrast with Hu et al.'s work, we focus on dynamic scenes and predict not only realtime gaze positions but also future gaze positions.

2.2 Gaze Behavior Analysis

There is some work on the analysis of human gaze behaviors. Itti [23] reported that human gaze behaviors are controlled by both a bottom-up mechanism and a top-down mechanism, which means visual attention is influenced not only by the content humans see but also by the tasks assigned to them. Pinto et al. [35] further revealed that the two mechanisms are independent. Baloh et al. [3] discovered many differences between horizontal and vertical eye movements and Rottach et al. [37] showed evidence for independent feedback control of horizontal and vertical saccades. Brockmann et al. [9] and Boccignone et al. [6] focused on gaze shifts and they modeled gaze shifts as a stochastic process. Franconeri et al. [17] revealed that moving stimuli capture visual attention and they further reported that looming stimuli capture attention while receding stimuli do not attract attention. The phenomenon that motion attracts attention was also observed in Abrams et al.'s work [1]. Yarbus [46] found that the eyes and the head move in coordination during gaze shifts and Einhäuser et al. [15] discovered human eye-head coordination in natural exploration. Nakashima et al. [31] proposed a model to improve the accuracy of saliency prediction by utilizing head direction.

The characteristics of human gaze behaviors in virtual reality have also been explored. Sitzmann et al. [39] revealed that there exists a latitudinal equator bias during users' exploration of 360° images. Xu et al. [45] explored human's gaze behaviors in dynamic 360° videos and found that users' gaze positions coincide with salient regions and moving objects. Hu et al. [21] revealed that, in static virtual scenes, users' gaze positions are correlated with their head rotation velocities and there exists a latency between eye movements and head movements. Based on the above-mentioned works, to explore huamn gaze behaviors in dynamic virtual scenes, we record the image sequences viewed by the observers, users' gaze positions, users' head rotation velocities, and the positions of dynamic objects for analysis.

2.3 Application of Eye Tracking

In VR systems, eye tracking technology has gained importance and it can be applied to many aspects. Tanriverdi et al. [43] utilized eye tracking technology as an interaction tool in virtual environments and

presented the benefits of eye movement-based interaction. In their recent work, Mardanbegi et al. also took advantage of eye tracking information to propose a novel interaction technique. Besides the works discussed in Sect. 2.2, eye tracking technology can also be used to analyze other aspects of gaze behaviors. For example, Berton et al. [4] collected users' eye tracking data for studying gaze behavior during collision avoidance with a virtual walker. Another important application of eye tracking technology is gaze-contingent rendering [18, 33, 41, 42], which improves rendering efficiency by decreasing the rendering quality in the peripheral region while maintaining high fidelity in the foveal region. In our evaluation process, we apply our model to gaze-contingent rendering to validate its effectiveness.

3 Data Collection

In this section, we present the details of our data collection process. 43 participants in total are asked to explore 5 dynamic virtual scenes under free-viewing conditions and we record the image sequences viewed by the observers, the positions of the dynamic objects, and the corresponding gaze positions and head rotation velocities of those observers. Our dataset can be utilized to analyze users' gaze behaviors in dynamic scenes (Sect. 4) and to train gaze prediction models (Sect. 5).

3.1 Stimuli

In the data collection process, participants are asked to freely explore 5 dynamic virtual scenes, as illustrated in Fig. 2. The test scenes include desert, forest, island, etc. environments, which are commonly used in VR applications. The original scenes are static. In each scene, we randomly place some animals like horses, deer, ibexes, cats, dogs, etc. and utilize them as dynamic objects. Each animal's movement is controlled by its own animation and its path is controlled by our own Unity script, which allows the animal to wander in the scene in a random fashion. During our experiments, we record the positions of the dynamic objects. An object's position refers to the position of the center of the object's bounding box and it contains three elements including the object's on-screen position and its distance from the observer.

Fig. 2: Five dynamic virtual scenes used for data collection, including both indoor and outdoor scenes. Some animals are placed in the scenes and are utilized as dynamic objects.

3.2 Participants

In total, 43 users (25 male, 18 female, ages $18-32$) took part in our experiments. Each user reported normal or corrected-to-normal vision and the eye tracker was calibrated for each participant before he/she started the experiment.

3.3 System Details

In our experiments, we employ an HTC Vive as our HMD to display the scenes and utilize a Vive controller for user interaction. Our HMD is equipped with a 7invensun VR eye tracker with a sampling frequency of 100 Hz and an accuracy of 0.5°. We record the head velocities at a sampling rate of 200 Hz using HTC Vive's Lighthouse tracking system. We utilize the Unity game engine to display all the scenes and record the dynamic object positions at a frequency of 100 Hz using our own Unity script. The image sequences viewed by the observers are recorded using a Bandicam screen-recorder at 60 fps. The CPU and GPU of our platform are an Intel(R) Core(TM) i7-8700 @ 3.20GHz and an NVIDIA GeForce RTX 2080 Ti, respectively. The snapshot of our experimental setup is demonstrated in the left of Fig. 3.

3.4 Procedure

For each dynamic scene, we set up 4 start positions beforehand and provide participants with a Vive controller to help them switch between the 4 preset positions. In addition, users can also utilize the Vive controller to teleport themselves to any position in their field of view. The above settings ensure that participants can fully explore the scenes. Before starting our experiments, each participant is given at least 3 minutes to get familiar with our experimental system. They are also provided with a pair of earplugs to prevent auditory disturbance. During the experiments, participants are asked to freely observe the scenes and they are given no specific task. Each participant explores 2 scenes (randomly chosen from our 5 scenes) and each exploration lasts for at least 3 minutes. We record the images viewed by the observers, their gaze positions (measured in visual angles), and their head rotation velocities. In addition, we also record the positions of the dynamic objects, i.e. centers of the objects' bounding box. Our dynamic objects are big mammals and this ensures there will not be too many of them appearing together in the users' field of view. Therefore, for simplicity, we only record the object positions of the nearest 3 objects.

3.5 Dataset

Our dataset includes 86 pieces of data. A piece of data corresponds to a user's exploration data in one scene. Each piece of data contains about 18,000 gaze positions (100 Hz sampling rate), 18,000 object positions (100 Hz), 36,000 head velocities (200 Hz), and 10,800 frames of scene screenshots (60 fps). The time stamps of these data are also recorded to help align them with each other. We name our dataset **DGaze-dataset**. Our dataset has been released for public usage[1].

4 Gaze Behaviors in Dynamic Scenes

In this section, we analyze users' gaze behaviors in dynamic scenes based on our dataset. Specifically, we analyze the characteristics of the gaze data, the correlation between gaze positions and object positions, the correlation between gaze positions and head velocities, and the correlation between gaze positions and salient regions. Our analyses not only reveal the characteristics of users' gaze behaviors but also provide meaningful insights for deriving gaze prediction models.

4.1 Gaze Analysis

We first analyze the distribution of users' gaze data. As illustrated in the right of Fig. 3, the central region of the HMD's screen contains most of the gaze data. To be more precise, 98.7% of the gaze data lies in the central 35° region. The distribution of the gaze data reveals that in dynamic scenes, users tend to look at the central region of the screen. The characteristic of the gaze data's distribution suggests that we should pay more attention to the information in the central region.

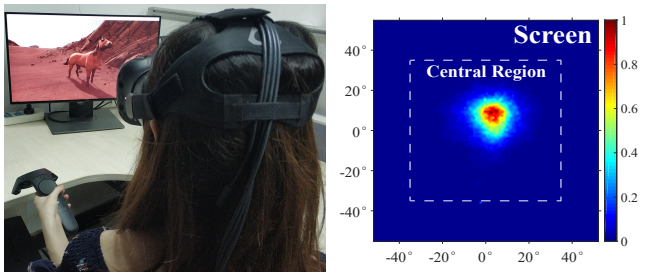

Fig. 3: Left: Our experimental setup. Right:The distribution of users' gaze positions on the HMD's screen. 98.7% of the gaze data lies in the central region of the screen. The central region is a square region that is confined to $[-35°, 35°] \times [-35°, 35°]$ in the domain of gaze position.

We also analyze the correlation between horizontal and vertical gaze positions. Specifically, we employ Spearman's rank correlation coefficient [30] to analyze correlation. Spearman's correlation assesses

[1] cranehzm.github.io/DGaze

monotonic relationships and it is robust to outliers. Spearman's correlation ranges from -1 (perfect monotone decreasing relationship) to 1 (perfect monotone increasing relationship). We calculate the correlation between horizontal and vertical gaze positions and obtain a value of 0.02 which indicates that horizontal and vertical gaze behaviors are different and independent to some extent. Therefore, we choose to analyze horizontal and vertical gaze behaviors independently in the following steps.

We further analyze the saccades that exist in our data. Saccades refer to fast eye movements and the occurrence of large saccades will make it quite difficult to predict users' gaze positions. We set the threshold velocity for the gaze speed to $75^\circ/s$ to extract the horizontal and vertical saccades [16]. We calculate the amplitudes of the saccades and find that the amplitudes of 90.6% of the horizontal saccades and 89.5% of the vertical saccades are very small ($\leq 5^\circ$). Furthermore, the total durations of horizontal and vertical saccades account for only 1.64% and 1.11% of the total gaze duration, respectively, which means saccades occur with low frequency. The characteristics of small amplitude and low frequency ensure that the saccades in our data have little impact on the task of gaze prediction.

4.2 Gaze-Object Analysis

We further analyze the correlations between gaze positions and object positions. Specifically, we calculate Spearman's correlations between gaze positions and object positions in situations with different numbers of valid objects. A valid object is an object that exists in a user's field of view. As illustrated in Fig. 4, the correlations in both horizontal and vertical directions are obvious and this suggests that dynamic object information can be employed to predict gaze positions. Fig. 4 also reveals that, in both horizontal and vertical directions, users' gaze behaviors are more likely to be influenced by objects that are closer. The correlations between gaze positions and Object 3 are not strong and farther objects may have even less influence on users' gaze behaviors. Moreover, as indicated in Table 1, the situation with 3 or more objects only accounts for 32.4% of all the situations. Therefore, we only consider the influences of the nearest 3 objects. In addition, Fig. 4 shows that dynamic objects seem to have a stronger influence on users' vertical gaze behaviors than horizontal gaze behaviors.

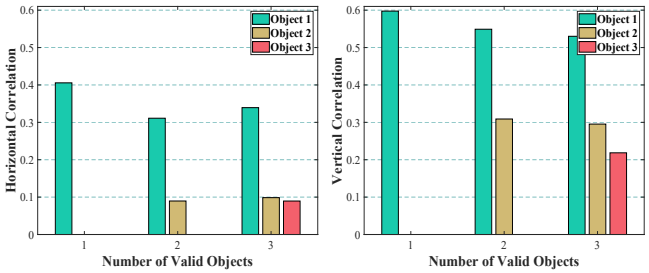

Fig. 4: The horizontal (left) and vertical (right) correlations between gaze positions and object positions in situations with different numbers of valid objects. Objects 1-3 are the recorded dynamic objects, ranked from nearest to farthest. Both the horizontal and vertical correlations are very strong, meaning that object information can facilitate the task of gaze prediction.

Object Number	0	1	2	≥3
Data Proportion	5.1%	37.1%	25.4%	32.4%

Table 1: The proportions of situations with different numbers of valid objects. Only 32.4% of all the situations contains 3 or more objects.

Generally, there exists a reaction time between the occurrence of a stimulus and a person's motor response to the stimulus [28, 44]. The reaction time may also exist in dynamic scenes, which means users' gaze positions may lag behind object positions. To validate this, at different time intervals, we calculate the correlations between gaze positions and the nearest object positions (in situations where the nearest object exists) because the nearest object has the most significant influence on gaze behaviors. From the graphs in Fig. 5, we can see that the horizontal and vertical correlations perform better than realtime situations (Time Interval $= 0$) when a suitable reaction time (Time Interval < 0) is considered. We consider the magnitude of time interval at the graph's peak point as the reaction time in dynamic scenes and get 230 *ms* in the horizontal direction and 280 *ms* in the vertical direction. The proposed reaction times correspond with prior works [28, 44], which reveal that reaction times are usually on the order of 200 *ms*.

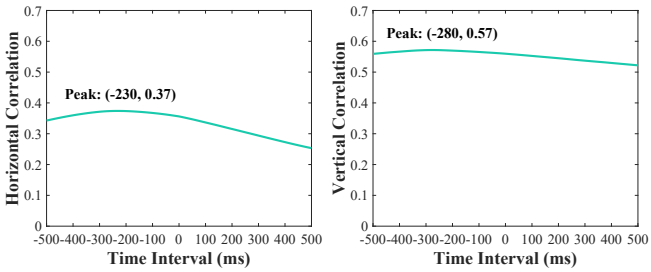

Fig. 5: The correlations between gaze positions and the nearest object positions at different time intervals in the horizontal (left) and vertical (right) directions. Time interval is the time difference between object data and gaze data. Each graph reaches its peak value at a negative time interval and this demonstrates the existence of a reaction time between the occurrence of dynamic objects and the response of gaze behaviors.

The above analysis reveals that not only realtime object positions but also past object positions are correlated with gaze positions and thus they can be both applied to the task of gaze prediction.

4.3 Gaze-Head Analysis

To explore the gaze-head correlation in dynamic scenes, we calculate Spearman's correlation between users' gaze positions and their head rotation velocities. We get 0.48 in the horizontal direction and 0.16 in the vertical direction. The result verifies that head velocities are correlated with gaze positions. In addition, we find that the gaze-head correlation performs better in the horizontal direction than in the vertical direction and this may be because users' vertical gaze behaviors are strongly influenced by dynamic objects (Sect. 4.2). We further calculate the correlations between gaze positions and head velocities at different time intervals (Fig. 6) and find that gaze positions also have correlations with head velocities in the past (Time Interval < 0). This reveals that head velocities in the past can also be employed to predict gaze positions.

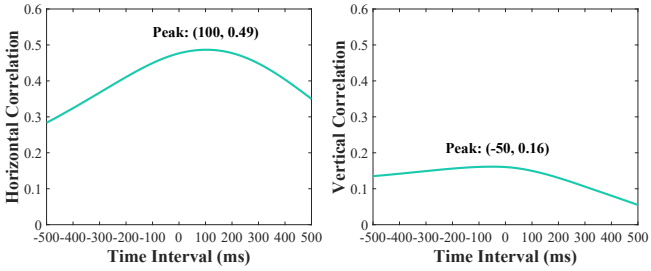

Fig. 6: The correlations between gaze positions and head velocities at different time intervals in the horizontal (left) and vertical (right) directions. Time interval is the time difference between head data and gaze data. In both the horizontal and vertical directions, gaze positions have correlations with head velocities in the past (Time Interval < 0).

4.4 Gaze-Saliency Analysis

In many prior works [10,12,24], salient regions of an image are assumed to attract more of the viewers' attention. To analyze the correlation between gaze positions and salient regions, we utilize the state-of-the-art SAM-ResNet saliency predictor to calculate the saliency maps of the realtime images viewed by the observers. Since there exists a reaction time between the occurrence of dynamic objects and the response of gaze behaviors (Sect. 4.2), we also calculate the saliency maps 250 *ms* before the gaze positions. We then evenly distribute all the pixels of a saliency map to 10 salient regions according to their saliency values where region 1 is the region with the most salient pixels. We further calculate the distribution of gaze positions on salient regions and the left of Fig. 7 illustrates that both in realtime and past saliency maps, most of the gaze positions lie in the pixels with the top 10% saliency. This result reveals that both realtime and past saliency features are correlated with gaze positions and thus they can both facilitate gaze prediction.

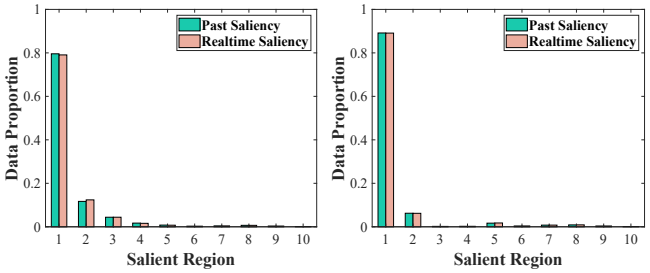

Fig. 7: The distributions of gaze positions on salient regions of the whole image (left) and the central image (right). Most of the gaze positions lie in the most salient region (region 1) in both realtime and past saliency maps. In addition, region 1 of the central image contains more gaze positions than region 1 of the whole image.

We also calculate the saliency maps of the central 35° region (the saliency values outside the central region are set to 0) and analyze the distribution of gaze positions (Fig. 7, right). We find that gaze positions correspond better with salient regions of the central image than the whole image. This indicates that saliency features of the central region are more useful than the features of the whole image when used to predict gaze positions.

5 DGaze Model

Our analyses in Sect. 4 reveal that dynamic object positions, head rotation velocities, and salient regions are correlated with users' gaze positions. Based on these analyses, we present the architecture of our model (Fig. 8). Our model is composed of three modules: sequence encoder module, saliency encoder module, and gaze prediction module. The sequence encoder module is utilized to encode the dynamic object position sequence and the head velocity sequence. The saliency encoder module is employed to extract and encode saliency features. The gaze prediction module combines the outputs of the 2 modules above to predict gaze positions. In this section, we first introduce the three modules and then show the details of our model's training process.

5.1 Sequence Encoder Module

Sect. 4.2 and Sect. 4.3 reveal that dynamic object positions and head rotation velocities can facilitate the task of gaze prediction. Therefore, in the sequence encoder module, we encode the sequence of dynamic object positions ($O_1, O_2, O_3, ..., O_t$, $O_i \in R^{3n}$, n is the number of recorded objects and $n = 3$ in our dataset) and head velocities ($H_1, H_2, H_3, ..., H_t$, $H_i \in R^2$) to predict gaze positions. In light of the good performance of 1D CNN for encoding sequence data [11, 27], we employ a 1D CNN layer with filters of size 2 and 128 output channels to encode the dynamic object position sequence and the head velocity sequence in the past 500 *ms* (sampled every 10 *ms*). The object position sequence contains the positions of the nearest 3 objects and invalid object data is padded with 0. We add a batch normalization layer [22] after the CNN layer to accelerate the training process and then utilize ReLU to activate the neurons. After the neurons are activated, we employ a max-pooling layer [38] with filters of size 2 to reduce the dimensions of the data and a dropout layer [40] with dropout rate 0.5 to avoid overfitting.

In general applications, DGaze is applied to predict users' gaze positions when eye trackers are not provided. However, even if an eye tracker is available, it can only provide us with realtime gaze positions and cannot offer us gaze positions in the future. In view of the eye tracker's limitation, when eye tracking data are available, we propose **DGaze_ET** to encode the sequence of gaze positions ($G_1, G_2, G_3, ..., G_t$, $G_i \in R^2$) in the past 500 *ms* (sampled every 10 *ms*) in the sequence encoder module to predict future gaze positions. The effectiveness of DGaze_ET is verified in Sect. 6.3.

5.2 Saliency Encoder Module

Sect. 4.4 reveals that gaze positions coincide with salient regions. Therefore, we extract the saliency features of the images and then encode the features for further gaze prediction. Since the calculation of saliency maps is usually time-consuming, to reduce time cost, we only compute the saliency maps of the central 35° region because they are more useful than the saliency maps of the whole image (Sect. 4.4). To further improve the computational efficiency, we sample the images every 250 *ms* ($\Delta t = 250ms$) as key frames and only calculate the saliency maps of the key frames. The above choices ensure that saliency maps can be obtained in real time. We utilize SAM-ResNet to extract the saliency maps in the past 500 *ms*, i.e. 2 saliency maps, downsample them to the resolution of 24×24, and flatten them to saliency features of size 1152 ($2 \times 24 \times 24$). We then apply a FC layer with 64 neurons to encode the saliency features. ReLU is used as the activation function for this layer. We add a batch normalization layer before activation and a dropout layer with dropout rate 0.5 after activation to prevent overfitting.

5.3 Gaze Prediction Module

The gaze prediction module incorporates the outputs of the sequence encoder module and the saliency encoder module to predict gaze positions. This module consists of 3 FC layers with 128, 128, and 2 neurons, respectively. The first two layers utilize ReLU as their activation function. A batch normalization layer is applied before they get activated and a dropout layer with dropout rate 0.5 is added after activation. The last layer is the output layer, which is used to predict gaze position (x_g, y_g). By setting the gaze positions at different time intervals as our targets, our model can be trained to not only predict realtime gaze positions but also to predict future gaze positions (Sect. 6.2.1 & Sect. 6.2.2).

5.4 Model Training Process

We employ the data from 3 scenes (randomly chosen from our 5 scenes) as our training data (60% of the total data, 1045654 data points) and utilize the data from the remaining 2 scenes (40% of the total data, 698304 data points, including an indoor scene and an outdoor scene) as our test data. We implement our model using the PyTorch framework and utilize L1 loss as our loss function due to its robustness to outliers. Many outliers may exist in our data because gaze shifts occur in a random way [5, 6, 9]. To train our model, we set mini-batch size to 64 and employ Adam as our optimizer with an initial learning rate of $1.0e^{-2}$. We reduce the learning rate to its $\frac{1}{10}$ after every 5 epochs and train our model for 30 epochs in total. In the evaluation process, different variants of our model are trained using the same approach. The source code of our method and the pre-trained models are available online[1].

6 Results

We evaluate our model's performance in this section. We first set some baselines and evaluation metrics for the task of gaze prediction in dynamic scenes. Then we demonstrate that DGaze's realtime prediction performance outperforms the baselines in both dynamic and static scenes. DGaze also performs best in predicting future gaze positions.

[1] cranehzm.github.io/DGaze

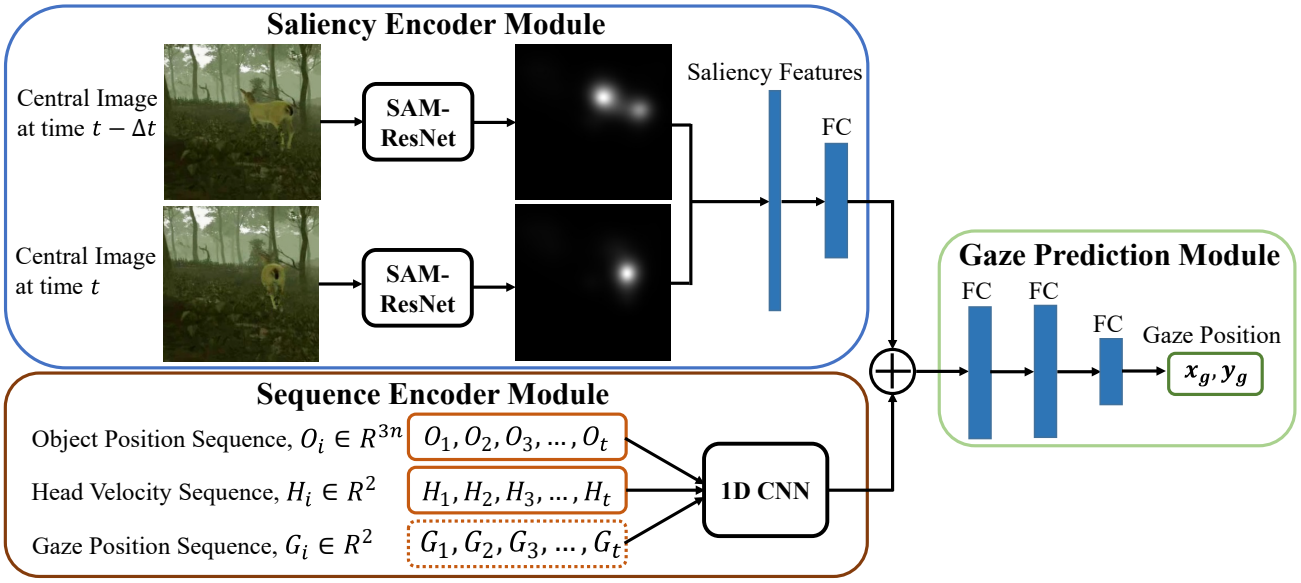

Fig. 8: Architecture of our proposed DGaze model. The sequence encoder module employs a 1D CNN layer to encode the object position sequence and the head velocity sequence. The saliency encoder module applies SAM-ResNet saliency predictor to extract saliency features of the past images and utilizes a FC layer to encode the features. The gaze prediction module combines the outputs of the 2 modules above and employs 3 FC layers to predict gaze positions. If the sequence of past gaze positions is available, it can also be encoded in the sequence encoder module to improve our model's performance.

When an eye tracker is available, DGaze_ET shows better performance in future gaze prediction than the baseline. We further analyze our CNN architecture and validate the effectiveness of each component in our model. We also apply DGaze to gaze-contingent rendering and conduct a user study to verify its effectiveness. Finally, we evaluate DGaze's performance in a game and the result shows that DGaze still performs best.

6.1 Baselines and Evaluation Metrics

6.1.1 Baselines

As described in Sect. 4.1, most of the gaze data lies in the central region of the screen. Therefore, we treat the screen center (0°, 0°) as one of our baselines (**Center Baseline**). To take advantage of the dataset, we also utilize the statistical mean of the gaze positions in the training data as another baseline (**Mean Baseline**), which is (2.62°, 5.64°). Since the gaze position is correlated with the position of the nearest object, we further set the position of the nearest object as our baseline (**Object Baseline**). The most recent method of predicting gaze position in virtual reality is SGaze [21], which is derived from static scenes. We retrain SGaze on our dataset and compare our model with it (**SGaze**).

6.1.2 Evaluation Metrics

To evaluate the performance of gaze position prediction, we calculate the angular distance between the ground truth and the predicted gaze position and utilize it as the prediction error. The smaller the angular distance, the better the performance. In addition, in some applications like gaze-contingent rendering, we aim to predict the gaze region rather than a single gaze position. In this case, we also utilize the recall rate, which is the proportion of the overlapped region at the ground truth region, as our evaluation metric. The higher the recall rate, the better the performance.

6.2 Model Evaluation

6.2.1 Realtime Prediction Performance

We first evaluate our model's realtime prediction performance. In this case, we train our model by setting the realtime gaze positions as our model's targets. To compare SGaze with our model, we also retrain SGaze on our dataset. We then apply our model and all the baselines, i.e. Center baseline, Mean baseline, Object baseline, and SGaze, to the test data and calculate their mean prediction errors (mean angular distances).We also calculate the standard error of the mean (SEM) to test whether the differences in prediction mean errors are statistically significant. The 95% confidence interval for a prediction mean M is $[M-1.96\times SEM, M+1.96\times SEM]$. If there is no overlap between the 95% confidence intervals of 2 means, this implies that the differences between the 2 means are statistically significant ($\alpha = 0.05$). Table 2 illustrates the results. Since the SEMs are very small and the confidence intervals for the means are very narrow, the differences between the means are statistically significant. In addition, our model achieves a 22.0% improvement over SGaze in dynamic scenes. Here the improvement is calculated by $(err_{sgaze} - err_{dgaze})/err_{sgaze}$. Fig. 1 highlights some of our prediction results. To further evaluate our model's generalization ability, we utilized "leave one scene out" cross-validation (CV) to calculate the realtime prediction performances of DGaze and SGaze in dynamic scenes. Specifically, we employed the same training parameters, as reported in Sect. 5.4, to train 5 separate DGaze models and then calculated the mean prediction performance of these models. The CV performance of DGaze is 7.57°, while the performance of SGaze is 9.33°. In terms of CV, DGaze demonstrates an improvement of 18.9% over SGaze in dynamic scenes and the result is close to our original result (DGaze outperforms SGaze by 22.0%). The performance of CV could be further improved by fine-tuning the training parameters, as part of future work.

Although our model is designed for dynamic scenes, we also test our model on a static dataset (Hu et al.'s dataset [21]). Since there are no moving objects in static scenes, we only utilize the head velocity sequence and saliency features as input to train our model and we do not use the Object baseline in this case. We test the newly trained model on the static dataset and the result shows that our model outperforms SGaze by 9.5% (Table 2). We further calculate the cumulative distribution function (CDF) of the prediction errors for performance evaluation. The higher the CDF curve, the better the performance. As illustrated in Fig. 9, our model achieves the best performance in terms of CDF on both the dynamic dataset and the static dataset. The above results verify that our model can be applied not only to dynamic scenes but also to static scenes.

		DGaze	SGaze	Mean	Center	Object
Ours	Mean	7.11°	9.11°	10.04°	12.46°	13.25°
	SEM	0.01°	0.01°	0.01°	0.01°	0.02°
Hu et al.'s	Mean	7.71°	8.52°	10.93°	11.16°	
	SEM	0.01°	0.01°	0.01°	0.01°	

Table 2: Our model and the baselines' realtime prediction performances on the dynamic dataset (our dataset) and the static dataset (Hu et al.'s dataset [21]). Our model achieves an improvement of 22.0% over SGaze on the dynamic dataset and an improvement of 9.5% on the static dataset.

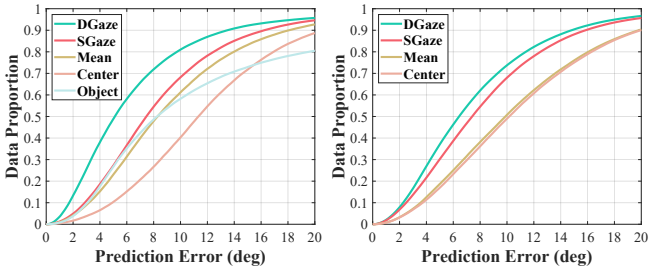

Fig. 9: Cumulative distribution function of the prediction errors on the dynamic dataset (left) and the static dataset (right). Our model (DGaze) outperforms the baselines in both dynamic and static scenes.

6.2.2 Future Prediction Performance

Our model can also be applied to predict gaze positions in the future. By separately setting gaze positions in the future 100 *ms*, 200 *ms*, ..., 1000 *ms* as our targets, we retrain 10 new models and test their future prediction performances. SGaze is also retrained for comparison. We test our model's future prediction performance on the dynamic dataset. The results are illustrated in Fig. 10. Center baseline and Mean baseline retain the same performance as realtime prediction when used to predict future gaze positions because they are constant. With the increase of prediction time, the performances of SGaze and DGaze both get worse, but our model always outperforms SGaze. At the prediction time of 1000 *ms*, SGaze performs almost the same as Mean baseline and this means SGaze is no longer effective. Although our model also deteriorates at 1000 *ms*, it still outperforms SGaze by 12.2%. The above results verify that our model can be applied to predict gaze positions within 1000 *ms*. Since our model only achieves a 12.5% improvement over Mean baseline at 1000 *ms*, we do not recommend applying our model to predict gaze positions after 1000 *ms* because its performance will be close to Mean baseline.

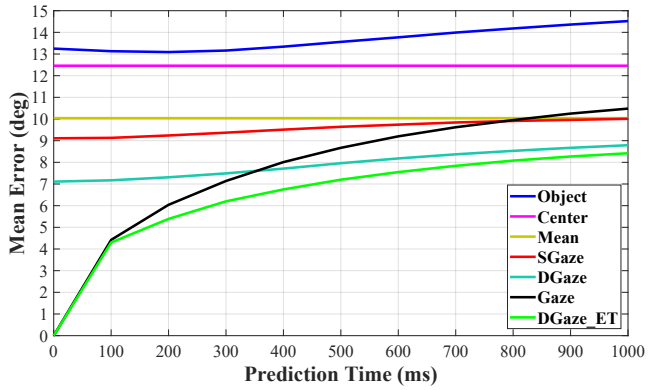

Fig. 10: Our model and the baselines' future prediction performances in dynamic scenes. DGaze always outperforms SGaze [21] in different prediction times. When eye tracking data are available, DGaze_ET performs better than Gaze baseline.

In the paragraph above, we retrain our model to predict future gaze positions at only 10 discrete time intervals. But how can we predict gaze positions at any given time within the range of 1000 *ms*? There exist many choices. First, we can directly apply an already trained model, whose prediction time is earlier (**Earlier Model**) or later (**Later Model**) than the given time, to perform the prediction. For example, if we want to predict gaze positions at future 250 *ms*, we can directly utilize the model that is trained for 200 *ms* or 300 *ms*. Second, we can retrain our model to predict gaze positions at the given time (**Retrained Model**). Finally, we can employ an ensemble method to combine the existing models together (**Ensemble Model**):

$$H(x) = \sum_{i=1}^{N} w_i h_i(x), \tag{1}$$

where $H(x)$ is the Ensemble Model; x is the input; N is the number of existing models; $h_i(x)$ is one of the models and w_i is its corresponding weight with $\sum_{i=1}^{N} w_i = 1$ and $w_i \geq 0$. For simplicity, we perform linear interpolation between the Earlier Model and the Later Model to derive the Ensemble Model:

$$H(x) = \frac{t_l - t}{t_l - t_e} h_e(x) + \frac{t - t_e}{t_l - t_e} h_l(x), \tag{2}$$

where t is the prediction time of $H(x)$; $h_e(x)$ is the Earlier Model and t_e is its prediction time; $h_l(x)$ is the Later Model and t_l is the corresponding prediction time. We test the performances of the above models at future 50 *ms*, 150 *ms*, 250 *ms*, 350 *ms*, and 450 *ms*, and indicate the results in Table 3. To test whether the differences in prediction mean errors are statistically significant, we perform two-tailed independent two-sample t-tests to compare the Ensemble Model with other models, i.e. Ensemble Model vs. Earlier Model, Ensemble Model vs. Later Model, and Ensemble Model vs. Retrained Model. The p-values are highlighted in Table 3. The test data contains 698304 data points and the degree of freedom for each t-test is 1396606 ($698304 \times 2 - 2$). We find that at the 5% significance level, the Ensemble Model significantly outperforms other models in most situations. This result validates that by training some separate models, which predict gaze positions at different future times, and employing an ensemble method, our model can be applied to predict gaze positions at any specified time within the range of 1000 *ms*.

		50*ms*	150*ms*	250*ms*	350*ms*	450*ms*
EA	Mean	7.15°	7.23°	7.39°	7.58°	7.81°
	p-value	2e-5	3e-2	1e-2	1e-1	6e-1
LA	Mean	7.13°	7.25°	7.41°	7.63°	7.88°
	p-value	3e-3	2e-4	2e-4	1e-6	2e-8
RE	Mean	7.13°	7.23°	7.35°	7.58°	7.85°
	p-value	3e-3	3e-2	6e-1	1e-1	2e-5
EN	Mean	7.10°	7.20°	7.36°	7.56°	7.80°

Table 3: Future prediction performances of the Earlier Model (EA), Later Model (LA), Retrained Model (RE), and Ensemble Model (EN). In most situations, the Ensemble Model performs significantly better than other models.

6.2.3 Runtime Performance

The most time-consuming part in our model is the calculation of saliency maps. Fortunately, however, only the saliency maps of the central region of the sampled images are computed, and this ensures that saliency maps can be obtained in real time (Sect. 5.2). Our model is implemented on an NVIDIA TITAN Xp GPU platform with an Inter(R) Xeon(R) E5-2620 v4 2.10 GHz CPU. We calculate DGaze's average prediction time for a single gaze position. The average prediction time is 0.07 *ms* on GPU and 0.16 *ms* on CPU. The result verifies that our model is fast enough for realtime usage.

6.3 Performance of DGaze_ET

To evaluate the performance of DGaze_ET, we add the sequence of gaze positions in the past 500 *ms* into the sequence encoder module to train the model. Since gaze data is available, we utilize the current gaze position as our baseline (**Gaze Baseline**). As illustrated in Fig. 10, DGaze_ET outperforms Gaze baseline in different prediction times and this verifies the effectiveness of our model. In addition, we find that Gaze baseline deteriorates significantly with the increase of prediction time. After 400 *ms*, it even performs worse than DGaze, which does not utilize past gaze data. This indicates that users' gaze positions in dynamic scenes change frequently and it is therefore difficult to perform long-term gaze prediction. At 1000 *ms*, DGaze_ET only achieves a small improvement over DGaze and thus we do not recommend utilizing DGaze_ET to predict gaze positions after 1000 *ms*.

6.4 Analysis of our CNN Architecture

We perform an ablation study to analyze our CNN architecture. Specifically, we remove the saliency features, head velocity sequence, and dynamic object position sequence separately and retrain the ablated models. We test the ablated models' realtime prediction performances and indicate them in Table 4. The result indicates that each component in our model helps improve our model's accuracy. In addition, from Fig. 10, we can see that DGaze_ET always outperforms DGaze and this verifies that the gaze position sequence contributes to DGaze_ET's performance. The above results validate the effectiveness of our CNN architecture.

	DGaze	w/o Saliency	w/o Head	w/o Object
Mean Error	7.11°	7.32°	7.52°	7.69°
SEM	0.01°	0.01°	0.01°	0.01°

Table 4: Realtime prediction performances of the ablated models. DGaze outperforms the ablated models, meaning that each component contributes to DGaze's performance.

In our training process (Sect. 5.4), L1 loss is utilized due to its robustness to outliers. In this section, we also apply L2 loss to retrain DGaze and test its realtime prediction performance. The mean prediction error of L2 loss is 7.32° and the SEM is 0.01°, which performs worse than L1 loss (Mean Error = 7.11°, SEM = 0.01°). This result indicates that L1 loss is more suitable for our model.

6.5 User Study

6.5.1 Gaze-Contingent Rendering

We also evaluate our model's effectiveness in gaze-contingent rendering [18, 33]. Gaze-contingent rendering requires a gaze region (foveal region) rather than a single gaze position. In this case, we calculate the recall rates of our model and the baselines. The higher the recall rate, the better the performance. In our calculation, the ground truth region is centered at the ground truth gaze position and its radius is set to 15°, as is used in gaze-contingent rendering [33], while the predicted region of the methods (DGaze and the baselines) is centered at the predicted gaze position and its radius is set to 20° to obtain better performance [21]. As illustrated in Table 5, our model outperforms the baselines and it reaches a recall rate of 90.1%, which is high enough for practical applications.

	DGaze	SGaze	Mean	Center	Object
Mean Recall Rate	90.1%	85.0%	82.0%	73.8%	74.6%

Table 5: Recall rates of our model and the baselines in dynamic scenes. Our model performs best, reaching a recall rate of 90.1%.

We further conducted a user study to compare the performances of DGaze and SGaze. Specifically, DGaze and SGaze were used to determine the gaze region (foveal region) with radius 20°. We utilized the 5 dynamic scenes that are presented in Sect. 3.1 as our stimuli and rendered the scenes using gaze-contingent rendering algorithm [18, 33]. Fig. 11 exhibits the implementation of our gaze-contingent rendering. We set the radius of the foveal region to 20° and the width of the blending border to 60 pixels. 15 users (11 male, 4 female, ages 18-28) with normal or corrected-to-normal vision participated in our experiments and the eye tracker was calibrated for each user. There is no overlap between the users and the participants in the data collection process (Sect. 3). The same system as described in Sect. 3.3 was employed.

Fig. 11: Implementation of our gaze-contingent rendering. The inner circle denotes the foveal region, which is rendered with high fidelity; the blending border refers to the transitional region; and the outer region is the peripheral region rendered with low quality.

We asked each participant to freely explore 3 scenes (randomly chosen from the 5 scenes) and ran a two-alternative forced choice (2AFC) test to collect users' responses. Specifically, each scene was rendered using DGaze and SGaze one after another in random order and each model (DGaze or SGaze) was applied for about 1 minute. In each test, participants were required to indicate which model had higher quality. If DGaze is chosen, DGaze will get 1 score and SGaze will get 0 score, and vice versa. There were 45 tests (15 users $\times$ 3 scenes) in total and we collected users' responses in these tests for analysis. We find that DGaze is preferred in 68.9% of the total responses. We further performed a two-tailed dependent (paired-sample) t-test on the responses and obtained $t(44) = 2.71, p = 0.0096$. We also performed a two-tailed dependent Wilcoxon signed-rank test and obtained $p = 0.0113$. At the 5% significance level, both the results of t-test and Wilcoxon signed-rank test are statistically significant. The above results validate our model's effectiveness in gaze-contingent rendering.

6.5.2 Performance in a Game

To further evaluate DGaze's availability, we apply it to a task-oriented game and conduct a user study. We created a game scene and randomly placed some dynamic objects (deer, ibexes, etc.), which wander in the scene in a random way as described in Sect. 3.1, and static objects (chests, boxes, etc.) in it (Fig. 12). The dynamic and static objects are utilized as the targets in the game. A total of 12 players (9 male, 3 female, ages 18-28) took part in our game and these players were not involved in the data gathering process described in Sect. 3. Each player reported normal or corrected-to-normal vision and the eye tracker was calibrated for each player. The same system as Sect. 3.3 was utilized.

Players could utilize a Vive controller to teleport themselves to any position in their field of view and they were given a sword, which was controlled by the Vive controller, to hit the dynamic and static objects in the game. Once an object was hit, it disappeared and the player got 1 score (1 hit). This game lasted for 2 minutes and the objects were sufficient, ensuring that players could not finish hitting all the objects in such a short time. During the game, we collected users' data for later analysis in the same way as described in Sect. 3.4. We evaluate the realtime prediction performances of our model and the baselines in the game scene. Specifically, we train DGaze and SGaze on our dynamic dataset and evaluate them on the newly collected data. As

Fig. 12: Our game scene. Some dynamic objects (deer, ibexes, etc.) and static objects (chests, boxes, etc.) are randomly placed in the scene and players are given a sword to hit these objects. The more objects they hit, the higher their scores.

illustrated in Table 6, our model outperforms the baselines, achieving an improvement of 14.3% over SGaze. In terms of CDF curve, our model also performs better than the baselines (Fig. 13). The above results indicate that our model can also be applied to game scenes. In addition, we find that our model performs worse in a task-oriented game than in free-viewing situations (Table 2) and this may be because players' behaviors in a game are highly correlated with the present state of the game [25]. In our scene, the game state variables like the game time, the game score may have influences on players' gaze behaviors. Exploring task-oriented situations such as games is an interesting avenue for future work.

	DGaze	SGaze	Mean	Center	Object
Mean Error	8.30°	9.69°	10.08°	12.27°	14.45°
SEM	0.02°	0.02°	0.02°	0.02°	0.04°

Table 6: Our model and the baselines' realtime prediction performances in a game. Our model performs best and it achieves an improvement of 14.3% over SGaze.

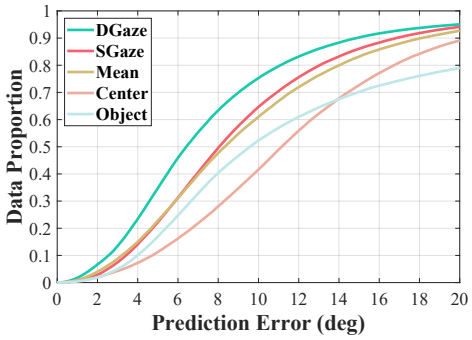

Fig. 13: Cumulative distribution function of the prediction errors in the game scene. Our model performs best in terms of CDF curve.

7 Conclusion, Limitations, and Future Work

In this paper, we conduct thorough analyses of users' gaze behaviors in dynamic scenes and present a CNN-based model (DGaze) for gaze prediction. We first build a dataset that contains users' eye tracking data in dynamic scenes. Then we analyze our dataset in detail and find that dynamic object positions, head rotation velocities, and saliency features can facilitate the task of gaze prediction. Based on our analyses, we propose a CNN-based model that takes the sequence of object positions, the sequence of head velocities, and saliency features as input to predict gaze positions. Our model not only works well in dynamic scenes but also outperforms the baselines in static scenes. DGaze can be applied to predict not only realtime gaze positions but also future gaze positions. We also propose DGaze_ET to predict future gaze positions, when accurate eye tracking data is available. We have evaluated our model in gaze-contingent rendering and the preliminary results are promising. Our model also outperforms the baselines when applied to a game.

Our model has some limitations. First, our dataset is restricted to free-viewing conditions and no specific task was assigned to the users. Therefore, our analyses and our model do not necessarily handle task-oriented situations. In addition, during the data collection process, we only utilize moving animals as our dynamic objects, and thus our results might have a bias within the recorded dataset. Human gaze behaviors in virtual scenes with other kind of dynamic objects (vehicles, pedestrians, etc.) still remain to be explored. Furthermore, our scenes are silent, and the influence of sound is not considered in our model. Combining the influence of sound into our model has the potential to improve our model's performance.

Besides overcoming the above limitations, there are many avenues for future work. First, there is still some room to improve our model's performance. For example, to improve time efficiency, we only utilize the saliency maps of sampled images. Taking more saliency maps into consideration may further boost the performance. In addition, fine-tuning the training parameters may further improve our model's cross-validation performance. Second, the architecture of our model has good extensibility. In the sequence encoder module, we only encode the object position sequence of the nearest 3 objects. However, if there are more objects appearing in users' field of view, these objects can also be encoded in the sequence encoder module. If other information related to users' gaze behaviors (e.g., users' gestures) are provided, we can develop a separate module to encode the information and then transmit the encoded features to gaze prediction module for predicting gaze positions. Furthermore, we are also looking to convert our model to other systems like augmented reality, mixed reality, and mobile virtual systems.

Acknowledgments

We would like to thank anonymous reviewers for their valuable comments. We would also thank 7invensun for their eye-tracking resource. This project was supported by the National Key R&D Program of China (No.2017YFB1002700) and National Natural Science Foundation of China (No.61632003, No.61661146002, No.61631001).

References

[1] R. A. Abrams and S. E. Christ. Motion onset captures attention. *Psychological Science*, 14(5):427–432, 2003.

[2] R. Alghofaili, M. Solah, H. Huang, Y. Sawahata, M. Pomplun, and L.-F. Yu. Optimizing visual element placement via visual attention analysis. In *The 26th IEEE Conference on Virtual Reality and 3D User Interfaces*, 2019.

[3] R. Baloh, L. Richman, R. Yee, and V. Honrubia. The dynamics of vertical eye movements in normal human subjects. *Aviation, space, and environmental medicine*, 54(1):32–38, 1983.

[4] F. Berton, A.-H. Olivier, J. Bruneau, L. Hoyet, and J. Pettré. Studying gaze behaviour during collision avoidance with a virtual walker: Influence of the virtual reality setup. In *The 26th IEEE Conference on Virtual Reality and 3D User Interfaces*, 2019.

[5] G. Boccignone and M. Ferraro. Modelling gaze shift as a constrained random walk. *Physica A: Statistical Mechanics and its Applications*, 331(1-2):207–218, 2004.

[6] G. Boccignone and M. Ferraro. Ecological sampling of gaze shifts. *IEEE transactions on cybernetics*, 44(2):266–279, 2013.

[7] A. Borji. Boosting bottom-up and top-down visual features for saliency estimation. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 438–445. IEEE, 2012.

[8] A. Borji, D. N. Sihite, and L. Itti. Probabilistic learning of task-specific visual attention. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 470–477. IEEE, 2012.

[9] D. Brockmann and T. Geisel. The ecology of gaze shifts. *Neurocomputing*, 32:643–650, 2000.

[10] M.-M. Cheng, N. J. Mitra, X. Huang, P. H. Torr, and S.-M. Hu. Global contrast based salient region detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):569–582, 2015.

[11] H. Cho and S. Yoon. Divide and conquer-based 1d cnn human activity recognition using test data sharpening. *Sensors*, 18(4):1055, 2018.

[12] R. Cong, J. Lei, C. Zhang, Q. Huang, X. Cao, and C. Hou. Saliency detection for stereoscopic images based on depth confidence analysis and multiple cues fusion. *IEEE Signal Processing Letters*, 23(6):819–823, 2016.

[13] M. Cornia, L. Baraldi, G. Serra, and R. Cucchiara. Predicting Human Eye Fixations via an LSTM-based Saliency Attentive Model. *IEEE Transactions on Image Processing*, 2018.

[14] A. T. Duchowski. Gaze-based interaction: A 30 year retrospective. *Computers & Graphics*, 73:59–69, 2018.

[15] W. Einhäuser, F. Schumann, S. Bardins, K. Bartl, G. Böning, E. Schneider, and P. König. Human eye-head co-ordination in natural exploration. *Network: Computation in Neural Systems*, 18(3):267–297, 2007.

[16] Y. Fang, R. Nakashima, K. Matsumiya, I. Kuriki, and S. Shioiri. Eye-head coordination for visual cognitive processing. *PloS one*, 10(3):e0121035, 2015.

[17] S. L. Franconeri and D. J. Simons. Moving and looming stimuli capture attention. *Perception & psychophysics*, 65(7):999–1010, 2003.

[18] B. Guenter, M. Finch, S. Drucker, D. Tan, and J. Snyder. Foveated 3d graphics. *ACM Trans. Graph.*, 31(6):164:1–164:10, Nov. 2012.

[19] J. Harel, C. Koch, and P. Perona. Graph-based visual saliency. In *Advances in neural information processing systems*, pp. 545–552, 2007.

[20] Z. Hu, S. Li, and M. Gai. Temporal continuity of visual attention for future gaze prediction in immersive virtual reality. *Virtual Reality & Intelligent Hardware*, 2020.

[21] Z. Hu, C. Zhang, S. Li, G. Wang, and D. Manocha. Sgaze: A data-driven eye-head coordination model for realtime gaze prediction. *IEEE transactions on visualization and computer graphics*, 25(5):2002–2010, 2019.

[22] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[23] L. Itti. *Models of bottom-up and top-down visual attention*. PhD thesis, California Institute of Technology, 2000.

[24] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 20(11):1254–1259, 1998.

[25] G. A. Koulieris, G. Drettakis, D. Cunningham, and K. Mania. Gaze prediction using machine learning for dynamic stereo manipulation in games. In *2016 IEEE Virtual Reality (VR)*, pp. 113–120. IEEE, 2016.

[26] M. Kümmerer, T. S. Wallis, L. A. Gatys, and M. Bethge. Understanding low-and high-level contributions to fixation prediction. In *2017 IEEE International Conference on Computer Vision*, pp. 4799–4808, 2017.

[27] D. Li, J. Zhang, Q. Zhang, and X. Wei. Classification of ecg signals based on 1d convolution neural network. In *2017 IEEE 19th International Conference on e-Health Networking, Applications and Services (Healthcom)*, pp. 1–6. IEEE, 2017.

[28] G. D. Logan, W. B. Cowan, and K. A. Davis. On the ability to inhibit simple and choice reaction time responses: a model and a method. *Journal of Experimental Psychology: Human Perception and Performance*, 10(2):276, 1984.

[29] D. Mardanbegi, K. Pfeuffer, A. Perzl, B. Mayer, S. Jalaliniya, and H. Gellersen. Eyeseethrough: Unifying tool selection and application in virtual environments. In *The 26th IEEE Conference on Virtual Reality and 3D User Interfaces*, 2019.

[30] J. L. Myers, A. D. Well, and R. F. Lorch Jr. *Research design and statistical analysis*. Routledge, 2013.

[31] R. Nakashima, Y. Fang, Y. Hatori, A. Hiratani, K. Matsumiya, I. Kuriki, and S. Shioiri. Saliency-based gaze prediction based on head direction. *Vision research*, 117:59–66, 2015.

[32] V. Navalpakkam and L. Itti. An integrated model of top-down and bottom-up attention for optimizing detection speed. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2, pp. 2049–2056. ieee, 2006.

[33] A. Patney, M. Salvi, J. Kim, A. Kaplanyan, C. Wyman, N. Benty, D. Luebke, and A. Lefohn. Towards foveated rendering for gaze-tracked virtual reality. *ACM Trans. Graph.*, 35(6):179:1–179:12, Nov. 2016.

[34] T. Pfeiffer, M. E. Latoschik, and I. Wachsmuth. Evaluation of binocular eye trackers and algorithms for 3d gaze interaction in virtual reality environments. *JVRB-Journal of Virtual Reality and Broadcasting*, 5(16), 2008.

[35] Y. Pinto, A. R. van der Leij, I. G. Sligte, V. A. Lamme, and H. S. Scholte. Bottom-up and top-down attention are independent. *Journal of vision*, 13(3):16–16, 2013.

[36] Y. Rai, J. Gutiérrez, and P. Le Callet. A dataset of head and eye movements for 360 degree images. In *Proceedings of the 8th ACM on Multimedia Systems Conference*, pp. 205–210. ACM, 2017.

[37] K. G. Rottach, R. D. Von Maydell, V. E. Das, A. Z. Zivotofsky, A. O. Discenna, J. L. Gordon, D. M. Landis, and R. J. Leigh. Evidence for independent feedback control of horizontal and vertical saccades from niemann-pick type c disease. *Vision research*, 37(24):3627–3638, 1997.

[38] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[39] V. Sitzmann, A. Serrano, A. Pavel, M. Agrawala, D. Gutierrez, B. Masia, and G. Wetzstein. Saliency in vr: How do people explore virtual environments? *IEEE Transactions on Visualization and Computer Graphics (IEEE VR 2018)*, 24(4):1633–1642, April 2018.

[40] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[41] N. T. Swafford, D. Cosker, and K. Mitchell. Latency aware foveated rendering in unreal engine 4. In *Proceedings of the 12th European Conference on Visual Media Production*, p. 17. ACM, 2015.

[42] N. T. Swafford, J. A. Iglesias-Guitian, C. Koniaris, B. Moon, D. Cosker, and K. Mitchell. User, metric, and computational evaluation of foveated rendering methods. In *Proceedings of the ACM Symposium on Applied Perception*, pp. 7–14. ACM, 2016.

[43] V. Tanriverdi and R. J. K. Jacob. Interacting with eye movements in virtual environments. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '00, pp. 265–272. ACM, New York, NY, USA, 2000.

[44] R. S. Woodworth and H. Schlosberg. *Experimental psychology*. Oxford and IBH Publishing, 1954.

[45] Y. Xu, Y. Dong, J. Wu, Z. Sun, Z. Shi, J. Yu, and S. Gao. Gaze prediction in dynamic 360 immersive videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5333–5342, 2018.

[46] A. Yarbus. Eye movements and vision. 1967. *New York*, 1967.