Supplement to "EyeFormer: Predicting Personalized Scanpaths with Transformer-Guided Reinforcement Learning"

Yue Jiang[*]
Zixin Guo[*]
yue.jiang@aalto.fi
zixin.guo@aalto.fi
Aalto University
Finland

Hamed R. Tavakoli
hamed.rezazadegan_tavakoli@nokia.com
Nokia Technologies
Finland

Luis A. Leiva
name.surname@uni.lu
University of Luxembourg
Luxembourg

Antti Oulasvirta
antti.oulasvirta@aalto.fi
Aalto University
Finland

ABSTRACT

The document provides additional details on the implementation of the dataset, algorithm, and training process reported upon in "EyeFormer: Predicting Personalized Scanpaths with Transformer-Guided Reinforcement Learning." In addition, it presents a more comprehensive set of experimental results and analysis of population-level and individual-level scanpath predictions.

1 IMPLEMENTATION DETAILS

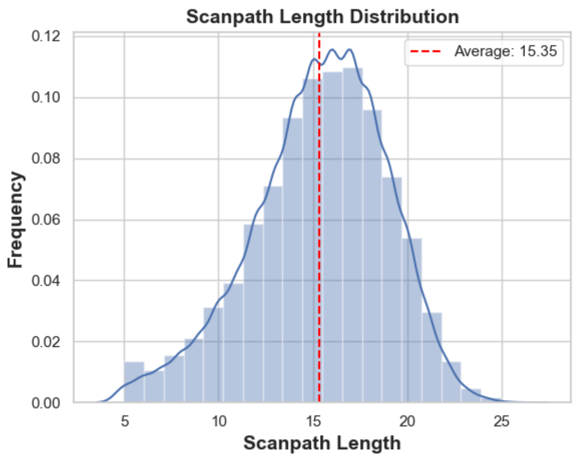


Figure 1: The distribution of scanpath lengths for the eye-tracking-based dataset UEyes. With an average scanpath length of 15.3 points, the visualization highlights that most scanpaths are clustered around length 15.

1.1 Dataset

GUIs and information graphics. We utilized the **UEyes** dataset [7], which comprises 1,980 images, from four common types of GUI or informational graphics (namely, posters, desktop GUIs, mobile GUIs, and webpages), with eye-tracking data collected from 62 participants. This dataset was collected via a high-fidelity eye tracker in a laboratory setting, to guarantee precise coordinates in the X−Y plane, and the coordinate values were subjected to participant-specific calibration accounting for factors such as eye–display distance. Throughout the data-collection process, the screen angle for each participant was checked and adjusted, for emulation of the individual-specific typical viewing scenario. Each participant, seated roughly 50–65 centimeters from the screen, experienced the same visual angle for all four types of interface. Uniformity of visual angle was enforced for ease of comparison across interface types: all data collection and subsequent analysis adhered to this method consistently. By guaranteeing that the limits to the tracking technology's accuracy cannot unduly influence the assessment of the mobile-device user interfaces, the uniform presentation strategy maintains fairness in the evaluation process. Fixation duration was ascertained directly from the timings recorded by the eye tracker. A participant free-viewed each image for seven seconds.

We used the same training/test image split that Jiang et al. [7] did, dividing the GUI image set such that 1,872 images were allocated to the training set and 108 to the test set while making sure that the four interface types were distributed evenly in each set. In addition, to establish a training/test split for individual-level prediction, we employed a further procedure: randomly assigning 53 users (85% of the sample) to the training set and the remaining nine (15%) to the test set. Our model was trained on the gaze data collected from when the training users looked at the interfaces represented by the set of training images. The viewers' average scanpath length was 15.3 points, with most scanpaths being clustered around length 15. Figure 1 presents the distribution of scanpath lengths from the UEyes dataset.

1.2 Parametric Distributions

The method's step i employs a multi-layer Transformer decoder to generate the distribution for the fixation point $p_i = (x_i, y_i, t_i)$. Meanwhile, fixation duration t_i is modeled as a Gaussian distribution, with the mean μ_{t_i} and variance σ_{t_i} generated by the policy. Similarly, a point's coordinates x_i and y_i can be modeled as Gaussian distributions, with means μ_{x_i,y_i} and variances σ_{x_i,y_i} dictated

[*]Both authors contributed equally to this research.

UIST '24, October 13–16, 2024, Pittsburgh, PA, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0628-8/24/10.
https://doi.org/10.1145/3654777.3676436

Algorithm 1 A comprehensive description of the RL approach proposed for scanpath prediction – an algorithm to optimize the scanpath-prediction process iteratively, with feedback from salient-value and DTWD rewards employed to enhance performance

```
1: Initialize policy model $\pi_\theta$ parameterized by $\theta$
2: Initialize reward function $R_{DTW}$, $R_{sal}$
3: For each training step, do
4:     Initialize empty list of sampled points $\hat{p}$, empty set of predicted points $p^{pred}$, and empty list of salient-value rewards $\hat{r}_{sal}$, $r_{sal}^{pred}$
5:     For each step $i$, do
6:         Sample images $\mathcal{I}$, ground truth $p$, and saliency maps $M_{sal}$
7:         If Gaussian distribution for coordinates, do
8:             Generate the mean and variance of the points' Gaussian distribution: $\mu_{x_i,y_i}, \sigma_{x_i,y_i}, \mu_{t_i}, \sigma_{t_i} \longleftarrow \text{forward}(\mathcal{I}, \hat{p}_{:i-1}, \theta)$
9:             Sample predicted points from the Gaussian distribution: $\hat{p}_i \longleftarrow \text{sample}(\mu_{x_i,y_i}, \sigma_{x_i,y_i}, \mu_{t_i}, \sigma_{t_i})$
10:        If mixed Gaussian distribution for coordinates, do
11:            Generate parameters of the points' Gaussian distribution: $\lambda_i, \mu_i, \sigma_i, \rho_i, \mu_{t_i}, \sigma_{t_i} \longleftarrow \text{forward}(\mathcal{I}, \hat{p}_{:i-1}, \theta)$
12:            Sample predicted points from the Gaussian distribution: $\hat{p}_i \longleftarrow \text{sample}(\lambda_i, \mu_i, \sigma_i, \rho_i, \mu_{t_i}, \sigma_{t_i})$
13:        Generate the mean of the Gaussian distribution for the points: $\mu_i^{pred} \longleftarrow \text{StopGradient}(\text{forward}(\mathcal{I}, p_{:i-1}^{pred}, \theta))$
14:        Compute salient-value rewards for each sampled point: $\hat{r}_{sal,i} \longleftarrow R_{sal}(\hat{p}_i, M_{sal})$
15:        Compute salient-value rewards for each predicted point: $r_{sal,i}^{pred} \longleftarrow R_{sal}(p_i^{pred}, M_{sal})$
16:        Add $\hat{p}_i$ to the set $\hat{p}$; and add $\hat{r}_{sal,i}$ to the set $\hat{r}_{sal}$
17:        Add $\mu_i^{pred}$ to the set $p^{pred}$; and add $r_{sal,i}^{pred}$ to the set $r_{sal}^{pred}$
18:    end for
19:    Compute DTW rewards for all sampled and predicted coordinates: $\hat{r}_{DTW} \longleftarrow -R_{DTW}(\hat{p}, p)$; $r_{DTW}^{pred} \longleftarrow -R_{DTW}(p^{pred}, p)$
20:    Compute DTW rewards for all sampled and predicted durations: $\hat{r}_{DTW,D} \longleftarrow -R_{DTW,D}(\hat{p}, p)$; $r_{DTW,D}^{pred} \longleftarrow -R_{DTW,D}(p^{pred}, p)$
21:    Compute total rewards for sampled and predicted coordinates: $\hat{r} \longleftarrow \hat{r}_{DTW} + \text{discount}(\hat{r}_{sal})$; $r^{pred} \longleftarrow r_{DTW}^{pred} + \text{discount}(r_{sal}^{pred})$
22:    Compute advantages for coordinates and durations: $A_{coord} \longleftarrow \hat{r} - r^{pred}$; $A_{dur} \longleftarrow \hat{r}_{DTW,D} - r_{DTW,D}^{pred}$
23:    Compute the loss gradient with respect to the model's parameters: $\nabla_\theta \longleftarrow \text{compute\_gradient}(\pi_\theta(\hat{p}), A_{coord}, A_{dur})$
24:    Update the parameters by using gradient ascent: $\theta \longleftarrow \theta + \alpha \nabla_\theta$
25: end for
```

by the policy:

$$x_i, y_i \sim \mathcal{N}(\mu_{x_i,y_i}, \sigma_{x_i,y_i}) t_i \sim \mathcal{N}(\mu_{t_i}, \sigma_{t_i}). \tag{1}$$

Furthermore, coordinates x_i and y_i can be represented by a mixed Gaussian distribution with K components, determined by the policy-generated parameters: component weight λ_{ik}, mean μ_{ik}, variance σ_{ik}, and correlation ρ_{ik}. This distribution is expressed as

$$x_i, y_i \sim \sum_{k=1}^{K} \lambda_{ik} \mathcal{N}(\mu_{ik}, \Sigma_{ik}).\ t_i \sim \mathcal{N}(\mu_{t_i}, \sigma_{t_i}), \tag{2}$$

Here, the covariance matrix (Σ_{ik}) is defined in terms of variance σ_{ik} and correlation ρ_{ik}.

1.3 Algorithm

The pseudocode presented for Algorithm 1 thoroughly describes the RL-based approach we propose for training the scanpath-prediction model. The algorithm is designed to optimize the prediction process iteratively by taking advantage of feedback from salient-value and DTWD rewards to enhance performance. The algorithm progresses through the following six stages:

(1) **Generating the policy model's output**: In each iteration, the algorithm generates the output from the model, comprising the parameters of a Gaussian distribution. This distribution informs the sampling of points, and its mean values function as the predicted points.
(2) **Computing the salient-value reward**: The algorithm computes a salience reward for each iteration's sampled and predicted points. This reward reflects the importance of the points with reference to the overall saliency map.
(3) **Constructing scanpaths**: The sampled and predicted points from each iteration get added to two empty sets, forming the sampled and predicted scanpath as the algorithm continues its work.
(4) **Calculating the DTWD reward**: The algorithm calculates the DTWD reward for the sampled and predicted scanpath, conducting comparison with the ground-truth scanpath. As a metric for similarity between two sequences, DTW aids in assessing how well the predicted and real-world scanpath match.
(5) **Computing the total reward**: Total-reward values are obtained from combining the salient-value rewards and DTWD rewards. These rewards are usually discounted, to give greater weight to recent predictions.
(6) **Updating the policy**: Finally, the algorithm updates the policy derived via the reward. This step involves adjusting the model's parameters or the strategy behind sampling and prediction of points. The goal is to optimize the algorithm's performance for accurate scanpath predictions.

1.4 Training Process

Natural scenes. Following the approach of Chen et al. [2], we employed ResNet-50 [5] as the vision encoder. With an input image in 240×320 resolution, a 30×40 grid of features gets generated as the encoder's output. In our technique, a four-layer Transformer with eight heads and a hidden layer size of 256 then operates as the fixation decoder, generating a sequence of $T = 10$ fixation points. A mixed Gaussian distribution, utilizing $K = 10$ components, is employed to model fixation coordinates. At each step, the fixation decoder generates the component weight, mean, variance, and correlation, in the manner represented by Equation 2.

GUI images. Our decision to implement the vision encoder by means of the ViT-Base [4] architecture, with the aforementioned 12-layer version of the model, was driven by its demonstrated ability to handle diverse visual patterns and its potential to outperform traditional convolutional neural networks in various task conditions. By means of self-attention and Transformer-based mechanisms, the ViT-Base model can learn intricate relationships among image elements without requiring handcrafted architecture-level modifications for specific problem domains. Before feeding an image to the vision encoder, we resized it to a square with dimensions of $w_{\text{inp}} = h_{\text{inp}} = 256$, divided that square into 256 patches each of size 16×16, and embedded these with a dimensionality of $d_{\mathcal{I}} = 768$. For the fixation decoder, we employed a three-layer Transformer, utilizing the first three BERT-Base [3] layers. The policy model is optimized via policy gradients to generate a sequence of $T = 15$ points as the final output. This setting for T accounts for the fact that many methods [1, 6–9] rely on a fixed number of generated points, with metrics' results depending considerably on the lengths. A value of 15 is consistent with the concentration of scanpath lengths around said value as captured in Figure 1. The Gaussian distribution is employed to model the fixation coordinates. At each step, the fixation decoder generates the mean and variance in line with Equation 1.

Two-stage training. To mitigate the policy-gradient-associated training variance, we took a two-part approach to the training. The process initializes the decoder-generated parameters of Gaussian distribution $\pi_\theta(\hat{p}i)$ for fixation points with the negative log-likelihood loss. This initialization process entails 30 training epochs with a batch size of 64 and a learning rate of 0.0001. The second step employs policy-gradient techniques to train the policy further within an RL framework, utilizing DTWD and salient-value rewards. This step applies 40 training epochs for natural scenes and 20 for GUIs, with a batch size of 64 and a learning rate of 0.00001.

2 EXPERIMENT RESULTS

Below, we present additional results and in-depth analysis of 's population-level and individual-level scanpath prediction.

2.1 Ablation Studies

For further assessment of the proposed RL framework's effectiveness and versatility, we carried out ablation studies with various vision-encoder architectures and with alternative approaches to Gaussian modeling utilized in the fixation decoder. The first ablation study, of architectures for vision encoders that apply Gaussian distributions for the UEyes data, demonstrated RL's advantages over non-RL approaches for the dataset. This is evidenced by its superior performance across most evaluation metrics for both ResNet and ViT techniques, as detailed in Table 1. The findings indicate that utilizing RL leads to more accurate fixations relative to not applying RL, irrespective of the architecture employed.

With another ablation study, exploiting a ResNet vision encoder, we examined the RL framework with alternative approaches to Gaussian modeling. This involved generating fixation positions by means of both Gaussian distributions and mixed Gaussian distributions. The results illustrate that RL's incorporation consistently enhances model performance across most metrics, whichever Gaussian-based approach is implemented.

2.2 Population-Level Scanpath Prediction

Next, we supplement the evaluation presented in the paper for population-level scanpath prediction with more result sets and deeper analysis.

2.2.1 Qualitative evaluation. Figure 4 and Figure 5 present performance for population-level prediction with, respectively, natural scenes and GUI images. The figures showcase the performance of our model well. Comprehensive comparison of our model with prior models for both of these stimulus classes is provided by Figure 6 and Figure 3.

2.2.2 Scanpath properties. Analyzing scanpath properties facilitates comparing scanpath-prediction models at another level. Accordingly, we address two key facets of scanpaths below: 1) saccades' angle and amplitude distribution and 2) visited- and revisited-element ratios. All results shown here were evaluated by means of the test data. These results were computed from the scanpath predictions made with the GUI data.

Saccade angle and amplitude distributions. Figure 8 characterizes the saccade-angle and amplitude-distribution aspect of our comparison. Visualizations of saccade bias assist in examining the direction and distance traveled between successive fixation points. Researchers have observed that the ground-truth gaze directions predominantly lie toward the right and the bottom portion(s) of GUIs, with the distances being greater toward the right-hand side. This is evidence that users tend to move their gaze left–right (rather than the opposite) across large distances and from top to bottom. Our model excels at capturing and replicating said characteristic, and it outperforms preexisting models in this regard. The pre-trained PathGAN model and DeepGaze III's distributions manifest clustering, caused primarily by the presence of point clusters in the predicted scanpaths. Both PathGAN models trained on UEyes and PathGAN++ suffer from erroneous center bias in their distributions. This limitation is evident also with the ScanGAN and ScanDMM models, as their distributions' middle-centricity attest. Furthermore, the Chen et al. model exhibits a tendency to place consecutive fixations in close mutual proximity. The Itti–Koch-based one and DeepGaze++, by incorporating IOR, prevent short distances between successive fixation points. Consequently, both output saccade amplitudes larger than those visible in the ground-truth data, thereby manifesting a weakness in their approach to capturing natural gaze behavior. SaltiNet and UMSS are similar in

Model	DTW ↓	TDE ↓	Eyenalysis ↓	DTWD ↓	MultiMatch ↑					
					Shape	Direction	Length	Position	Duration	Mean
Different Vision Encoders on UEyes with Gaussian										
ViT w/o RLs	4.304 ± 1.309	0.143 ± 0.041	0.049 ± 0.024	5.299 ± 1.235	**0.946**	0.709	0.925	0.820	0.736	0.827
ViT	**4.069 ± 1.089**	**0.122 ± 0.029**	**0.036 ± 0.018**	**5.043 ± 1.052**	0.942	**0.748**	**0.940**	**0.825**	**0.750**	**0.841**
ResNet w/o RLs	4.146 ± 1.344	0.136 ± 0.037	0.045 ± 0.025	5.176 ± 1.282	**0.945**	0.702	0.923	0.825	0.723	0.824
ResNet	**3.843 ± 1.103**	**0.116 ± 0.030**	**0.035 ± 0.017**	**4.864 ± 1.084**	0.944	**0.749**	**0.937**	**0.837**	**0.747**	**0.843**
Different Gaussian Modelling on UEyes with ResNet										
Gaussian w/o RLs	4.146 ± 1.344	0.136 ± 0.037	0.045 ± 0.025	5.176 ± 1.282	**0.945**	0.702	0.923	0.825	0.723	0.824
Gaussian	**3.843 ± 1.103**	**0.116 ± 0.030**	**0.035 ± 0.017**	**4.864 ± 1.084**	0.944	**0.749**	**0.937**	**0.837**	**0.747**	**0.843**
Mixed Gaussian w/o RLs	4.721 ± 1.543	0.127 ± 0.040	0.042 ± 0.028	5.887 ± 1.441	**0.940**	0.748	0.935	0.801	0.723	0.830
Mixed Gaussian	**3.938 ± 1.139**	**0.115 ± 0.031**	**0.035 ± 0.019**	**4.936 ± 1.087**	**0.940**	**0.758**	**0.937**	**0.836**	**0.753**	**0.845**
Different Gaussian Modelling on OSIE with ResNet										
Gaussian w/o RLs	2.555 ± 0.840	0.123 ± 0.036	0.047 ± 0.024	2.898 ± 0.794	0.932	0.687	0.918	0.818	0.696	0.810
Gaussian	**2.169 ± 0.759**	**0.114 ± 0.036**	**0.046 ± 0.026**	**2.525 ± 0.698**	**0.944**	**0.694**	**0.930**	**0.846**	**0.707**	**0.824**
Mixed Gaussian w/o RLs	2.492 ± 0.886	0.120 ± 0.042	0.045 ± 0.027	2.863 ± 0.825	0.935	0.675	0.922	0.833	0.700	0.813
Mixed Gaussian	**2.193 ± 0.831**	**0.115 ± 0.042**	**0.044 ± 0.026**	**2.562 ± 0.756**	**0.944**	**0.679**	**0.932**	**0.850**	**0.706**	**0.822**

Table 1: Results from an ablation study examining the performance of various vision encoders and forms of Gaussian modeling in population-level scanpath prediction with the UEyes and OSIE datasets.

this respect, with the shortcoming being evident from the extent of the movement distance. Though employing inhibition of return, our model still generates some short vertical-movement distances.

Ratios for the elements visited and returned to. To enable comparing the visited- and re-visited-element ratios from the various models' scanpaths for people looking at GUIs, we performed segmentation and classified the interface elements into three distinct categories: images, text, and faces. Then, we calculated the number of elements in each category that were visited (fixated on) and re-visited (fixated upon at least once more), where an element already visited is considered re-visited if encompassing at least three fixation points after a fixation on a different element. Figure 9 presents the results of our comparison.

Of the models evaluated, ours produces the estimation nearest the ground truth for both sets of ratios. The other models' performance varies greatly, with some scanpaths representing underestimation of these ratios and others leading to overestimation. For instance, both the visit and the return-visit ratios from the pre-trained PathGAN and DeepGaze III models' output are unrealistically low. While SaltiNet and UMSS perform well at producing accurate element-visit and re-visit ratios for images, the ratios are still too low for text and faces. The Itti–Koch-based model captures both ratios well for text in addition to images, but its performance is worse for faces, with excessively low re-visit ratios. Conversely, DeepGaze++'s predictions match the ground-truth visit ratios excellently while the re-visit ratios are too high. Both ScanGAN and ScanDMM yield overly high return-visit ratios for textual elements, with ScanDMM's predictions simultaneously generating underestimates for return visits to faces (these results are consistent with our observation in the qualitative comparison that ScanDMM is more text-focused). Finally, the model by Chen et al. generates predictions that can be characterized as underestimating the re-visited-element ratio for both text and faces while overestimating the visited-element ratio for face elements.

2.3 Individual-Level Scanpath Prediction

We verified our model's ability to generate personalized scanpaths by proceeding from a few scanpath samples from the individual, thus confirming that the model can effectively capture each user's viewing preferences/behaviors and reflect them in its output. Figure 10 presents a few sample individual-level scanpath predictions in qualitative terms. These and the additional example scanpath predictions that Figure 11 offers, for several distinct viewers, demonstrate that the personalized scanpaths generated for each training user lie reasonably close to the ground truth. On the quantitative side, Figure 2 depicts the relationship between sample quantity and the numerically judged performance of individual-level scanpath prediction. We set n_{path} to 50, to strike a balance between accuracy and data volume. The choice to use 50 samples is supported by the performance enhancement evident upon increasing the quantity from 30 to 50 samples, coupled with the consistent performance plateau evident at 50–70 samples.

3 APPLICATION FOR PERSONALIZED VISUAL FLOWS

Figure 3 shows the results from the implementation for personalized visual flows, which is outlined in the paper. The designer supplies a GUI layout and the visit order desired for the most important elements: the three or more to be fixated upon first. From this input, outputs both population-optimized and individual-optimized layouts. The model optimizes the individual-specific ones on the basis of the personalized scanpath-prediction results. Specifically, given a viewer with n_{path} scanpath samples (in our experiments, $n_{\text{path}} = 50$), it generates a corresponding personalized layout optimized from the predicted scanpaths at the individual level for this specific viewer.

Tests of three source designs for 62 viewers produced the following results:

(1) With "Design 1," 56 viewers would follow the desired viewing order for the designer-specified elements with the population-optimized layout, and the associated average total fixation duration is 1.29 seconds. All 62 would adhere to the desired order when shown the corresponding personalized layout. The average duration sum, 1.86 seconds, is 44.19% greater than the figure for the population-optimized layout.

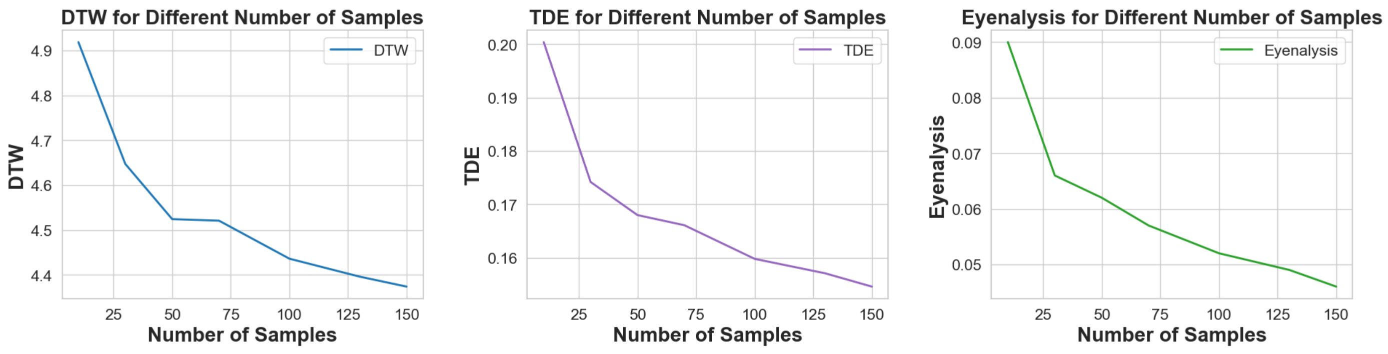
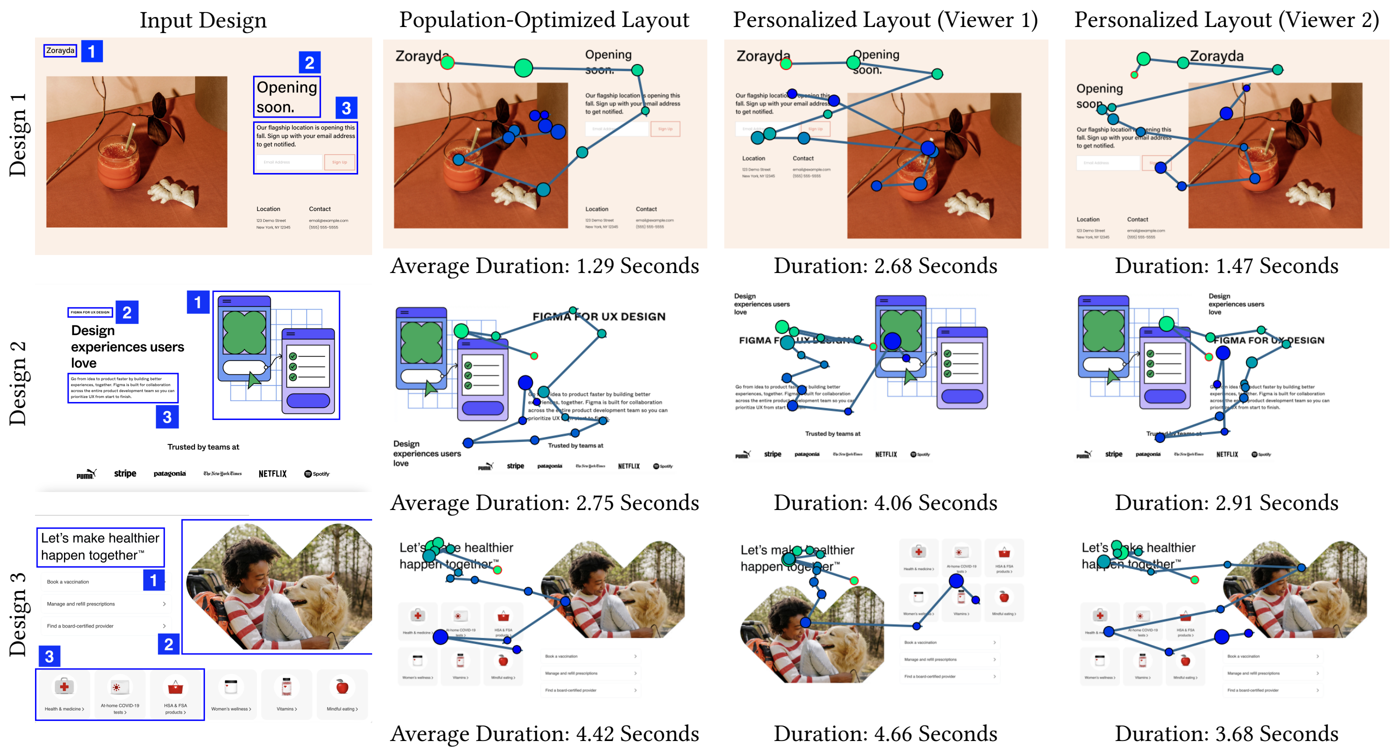


Figure 2: Performance for individual-level scanpath prediction plotted against sample quantity. As the sample count rises, the quantitative outcome (as measured by DTW, TDE, and Eyenalysis) improves. A noteworthy pattern is detectable for the span between as few as 10 samples and about 50, with all three panes likewise showing a clear deceleration in advancement as the sample count climbs to 70 or more (this is particularly evident with the DTW and TDE metrics).



Input Design
Population-Optimized Layout
Personalized Layout (Viewer 1)
Personalized Layout (Viewer 2)
Design 1
Design 2
Design 3
Average Duration: 1.29 Seconds
Duration: 2.68 Seconds
Duration: 1.47 Seconds
Average Duration: 2.75 Seconds
Duration: 4.06 Seconds
Duration: 2.91 Seconds
Average Duration: 4.42 Seconds
Duration: 4.66 Seconds
Duration: 3.68 Seconds

Figure 3: When given a starting GUI design accompanied by the order for the three elements deemed most important by the designer, the system generates both the population-optimized layout and a layout personalized for each individual viewer. The figure shows the average total fixation time for the specified elements across the tested viewers with the population-optimized layout and the total duration of fixations on these elements for each personalized layout shown, within the 7 s viewing window. The data show that the personalized layouts can attract more of the respective viewer's attention to the target elements than the population-optimized layout does.

(2) In the conditions of "Design 2," 46 viewers would follow the desired viewing order for the chosen elements when presented with the population-optimized layout, and the average duration is 2.75 seconds. All viewers but one would attend to the elements in the desired order with the corresponding personalized layout, and the average duration, at 3.19 seconds, is 16.00% higher than the equivalent sum for the population-optimized layout.

(3) Finally, with "Design 3," 31 viewers would follow the desired order for viewing the elements in the population-optimized setting. The average duration here is 3.78 seconds. Shown the individual-specific layout, 58 viewers would follow the specified order, with an average duration sum of 4.45 seconds. This is a visit time 17.72% longer than that with the population-optimized layout.

These results show that personalized layouts can steer users toward the desired viewing order and that, thereby, the specified elements can attract more attention from the target viewer relative to a population-optimized layout. Note that for Viewer 2, the population-optimized layout is the same as the personalized one.

One limitation of the current optimizer is that it cannot handle groupings of GUI elements. To address this, future work could entail adding grouping constraints so as to let optimization take advantage of element-grouping.

REFERENCES

[1] Marc Assens, Xavier Giro i Nieto, Kevin McGuinness, and Noel E. O'Connor. 2018. PathGAN: Visual Scanpath Prediction with Generative Adversarial Networks. *ECCV Workshop on Egocentric Perception, Interaction and Computing (EPIC).*

[2] Xianyu Chen, Ming Jiang, and Qi Zhao. 2021. Predicting human scanpaths in visual question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 10876–10885.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers).* 4171–4186.

[4] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2020. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *CoRR* abs/2010.11929 (2020). arXiv:2010.11929 https://arxiv.org/abs/2010.11929

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition.* 770–778.

[6] Laurent Itti, Christof Koch, and Ernst Niebur. 1998. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on pattern analysis and machine intelligence* 20, 11 (1998), 1254–1259.

[7] Yue Jiang, Luis A. Leiva, Paul R. B. Houssel, Hamed R. Tavakoli, Julia Kylmälä, and Antti Oulasvirta. 2023. UEyes: Understanding Visual Saliency across User Interface Types. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) *(CHI '23).* Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3544548.3581096

[8] Matthias Kümmerer, Matthias Bethge, and Thomas SA Wallis. 2022. DeepGaze III: Modeling free-viewing human scanpaths with deep learning. *Journal of Vision* 22, 5 (2022).

[9] Yao Wang, Andreas Bulling, et al. 2023. Scanpath prediction on information visualisations. *IEEE Transactions on Visualization and Computer Graphics* (2023).

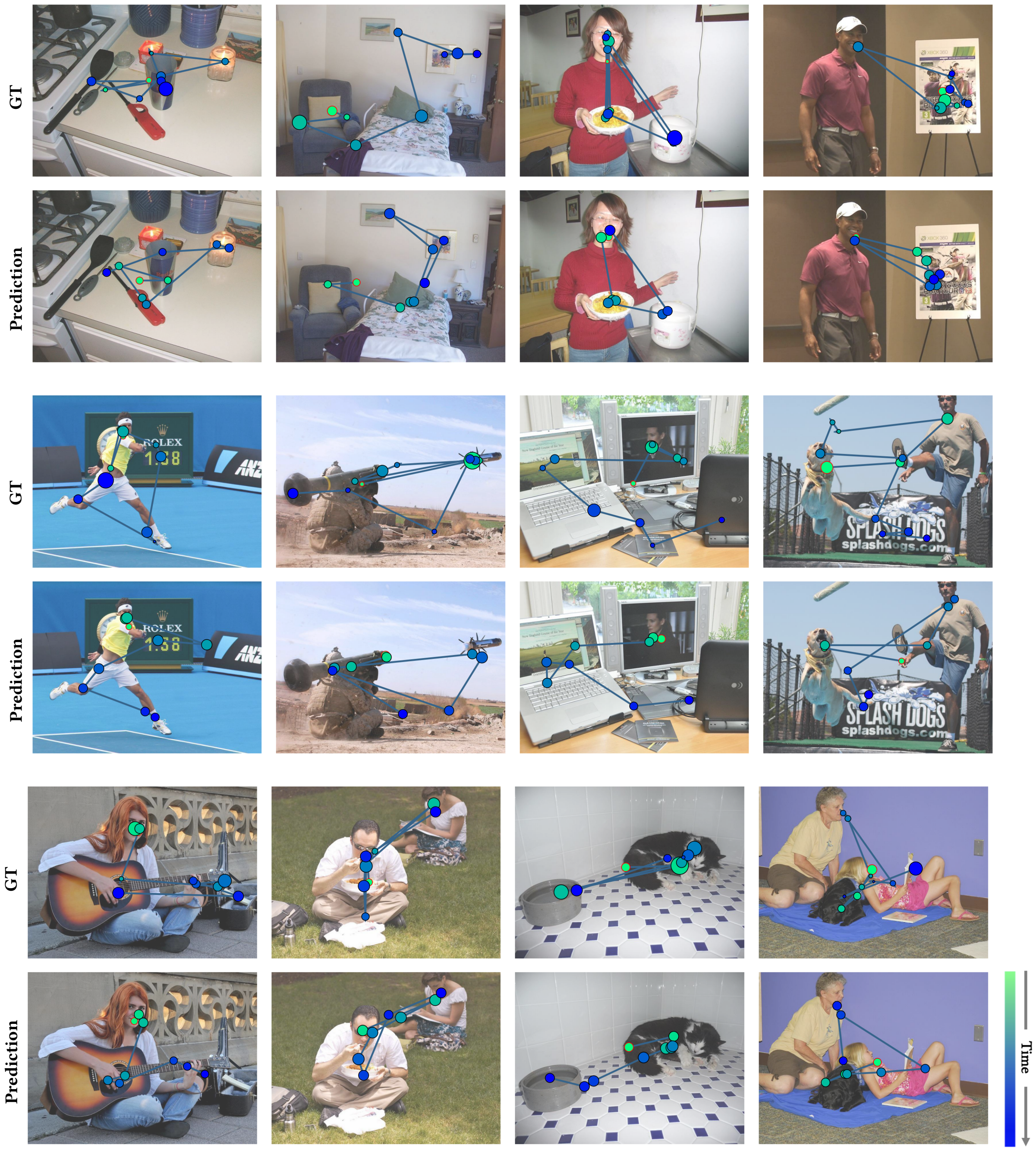


Figure 4: Results of our population-level scanpath predictions with natural scenes.

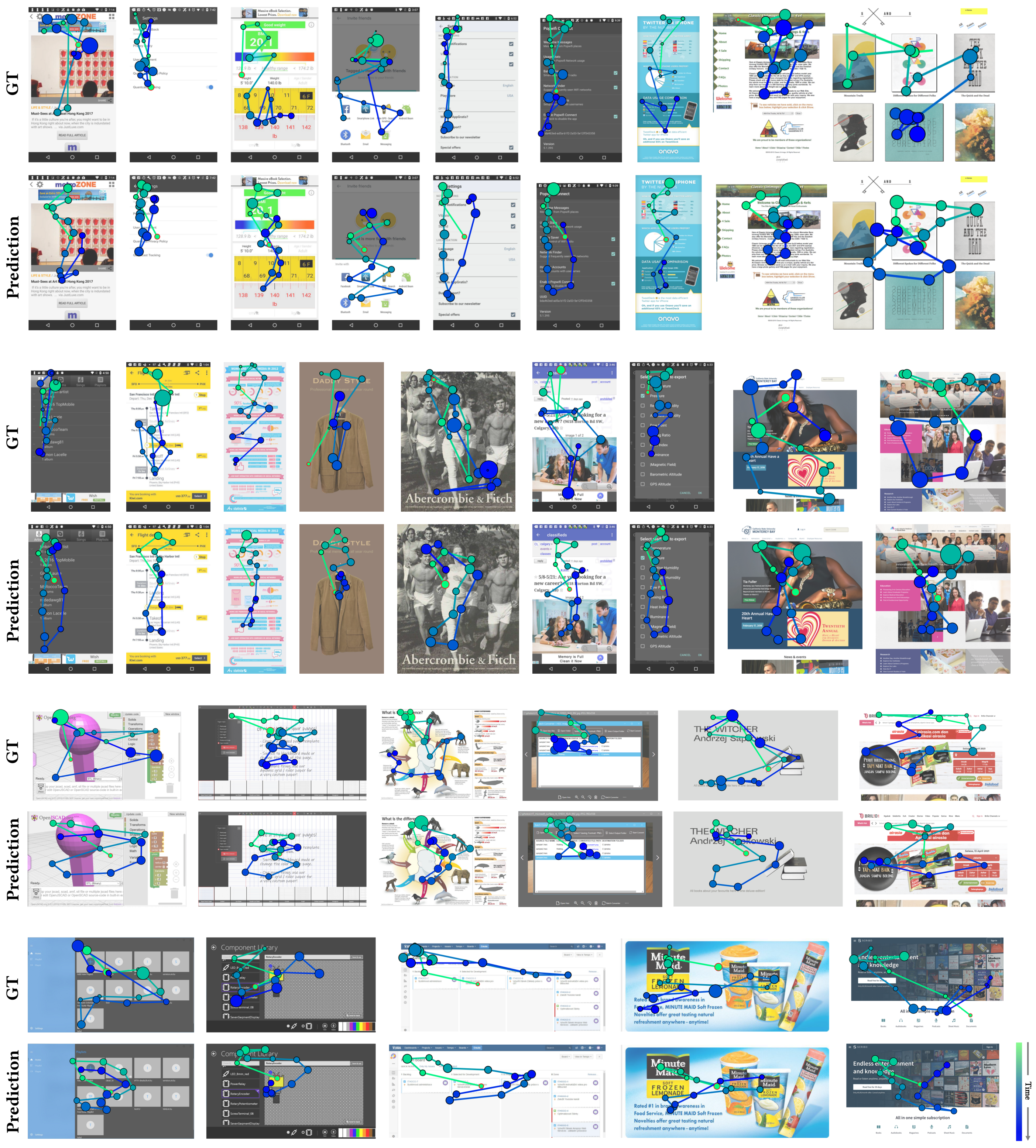


Figure 5: The population-level scanpath prediction results with GUIs.

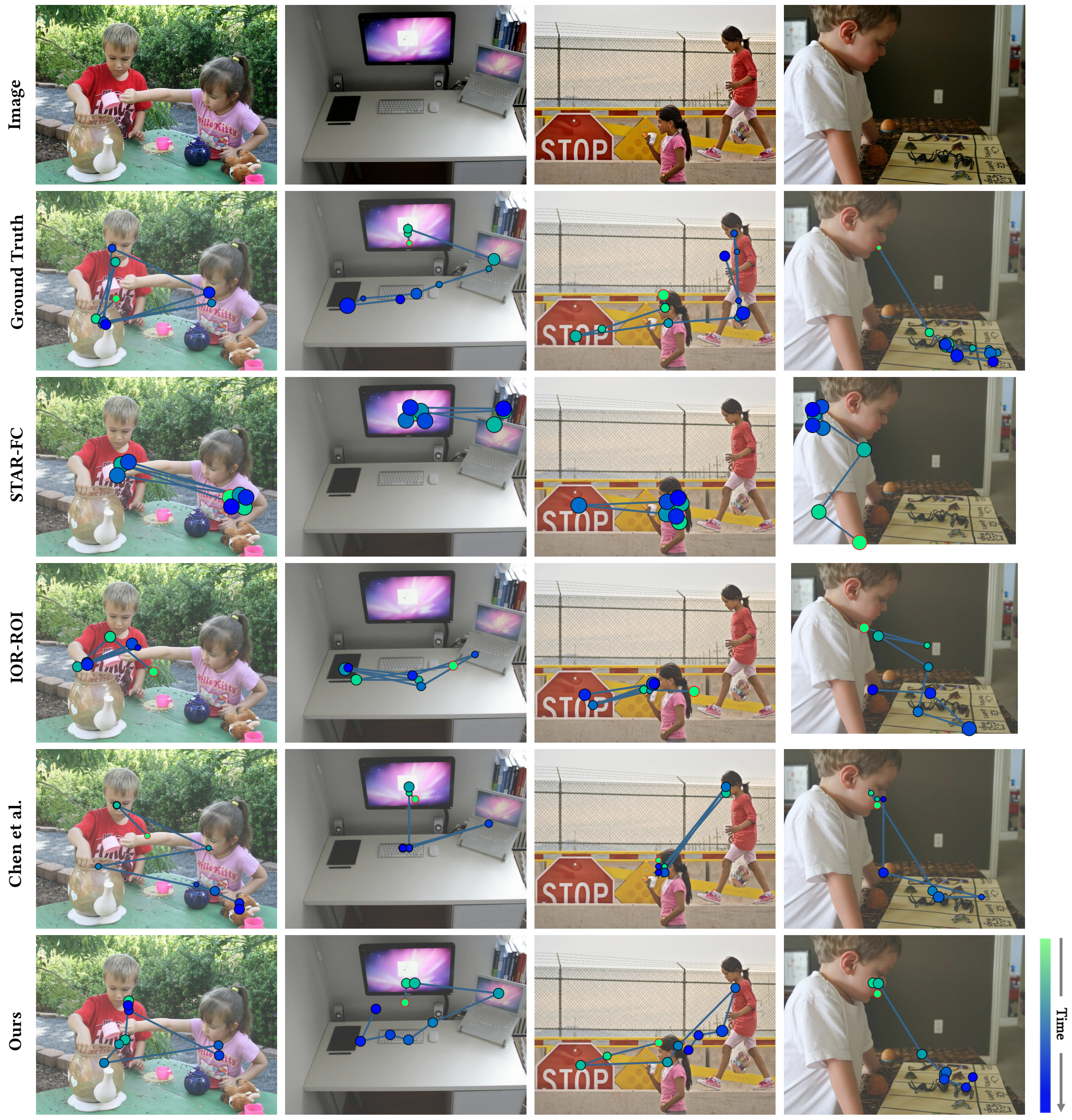
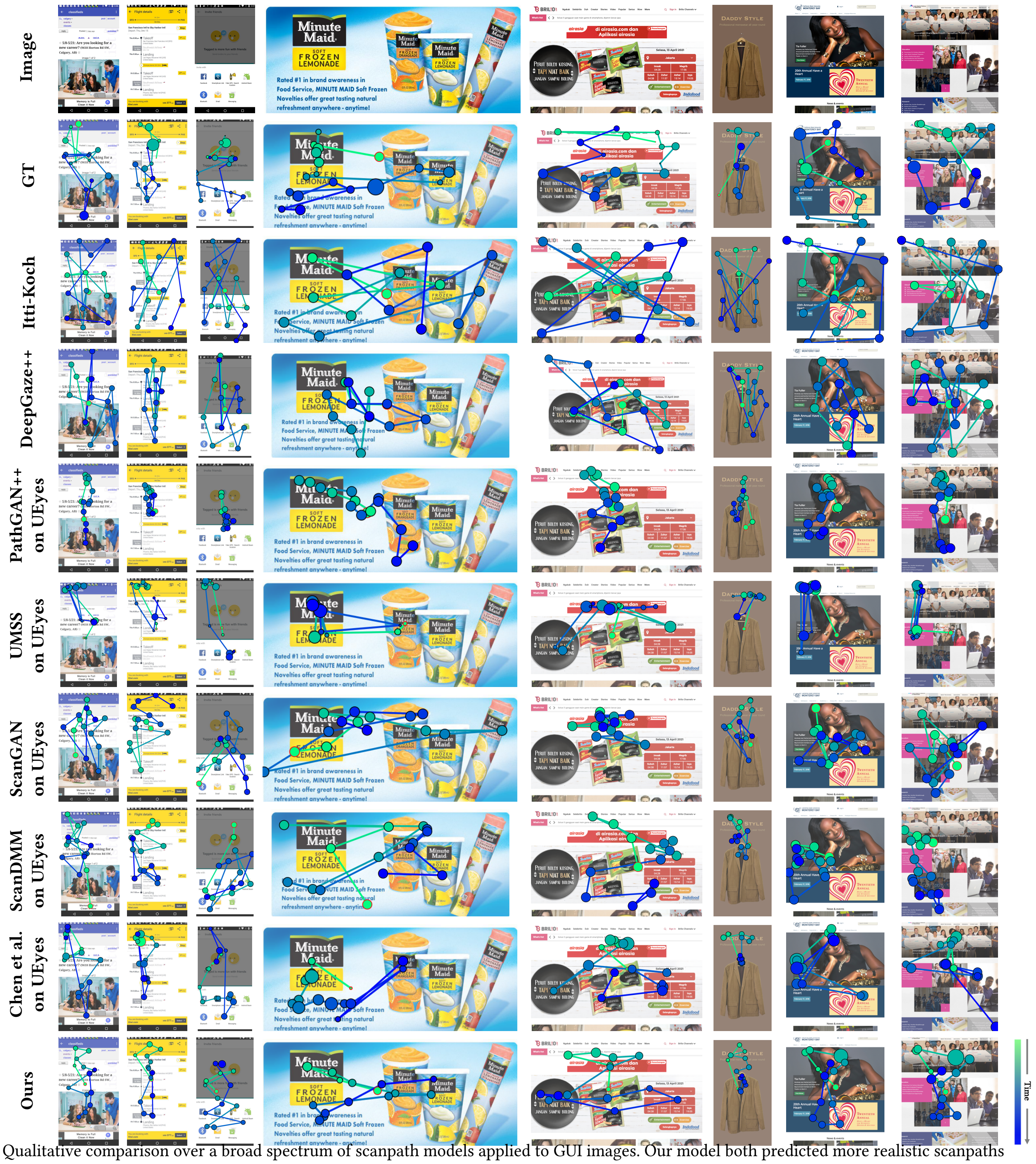


Figure 6: Qualitative comparison over a broad spectrum of scanpath models, applied to natural scenes. Our model predicts more realistic scanpaths than others did. Also, it exhibits greater accuracy for individual fixation points. For a fair comparison, all models were trained on OSIE.



Qualitative comparison over a broad spectrum of scanpath models applied to GUI images. Our model both predicted more realistic scanpaths than others and was more accurate with regard to individual fixation points. For the best possible comparison, all models apart from the Itti–Koch one (not deep-learning-based) and DeepGaze++ were trained on UEyes.

Qualitative comparison over a broad spectrum of scanpath models applied to GUI images. Our model both predicted more realistic scanpaths than others and was more accurate with regard to individual fixation points. For the best possible comparison, all models apart from the Itti–Koch one (not deep-learning-based) and DeepGaze++ were trained on UEyes.

Figure 7:
Qualitative comparison over a broad spectrum of scanpath models applied to GUI images. Our model both predicted

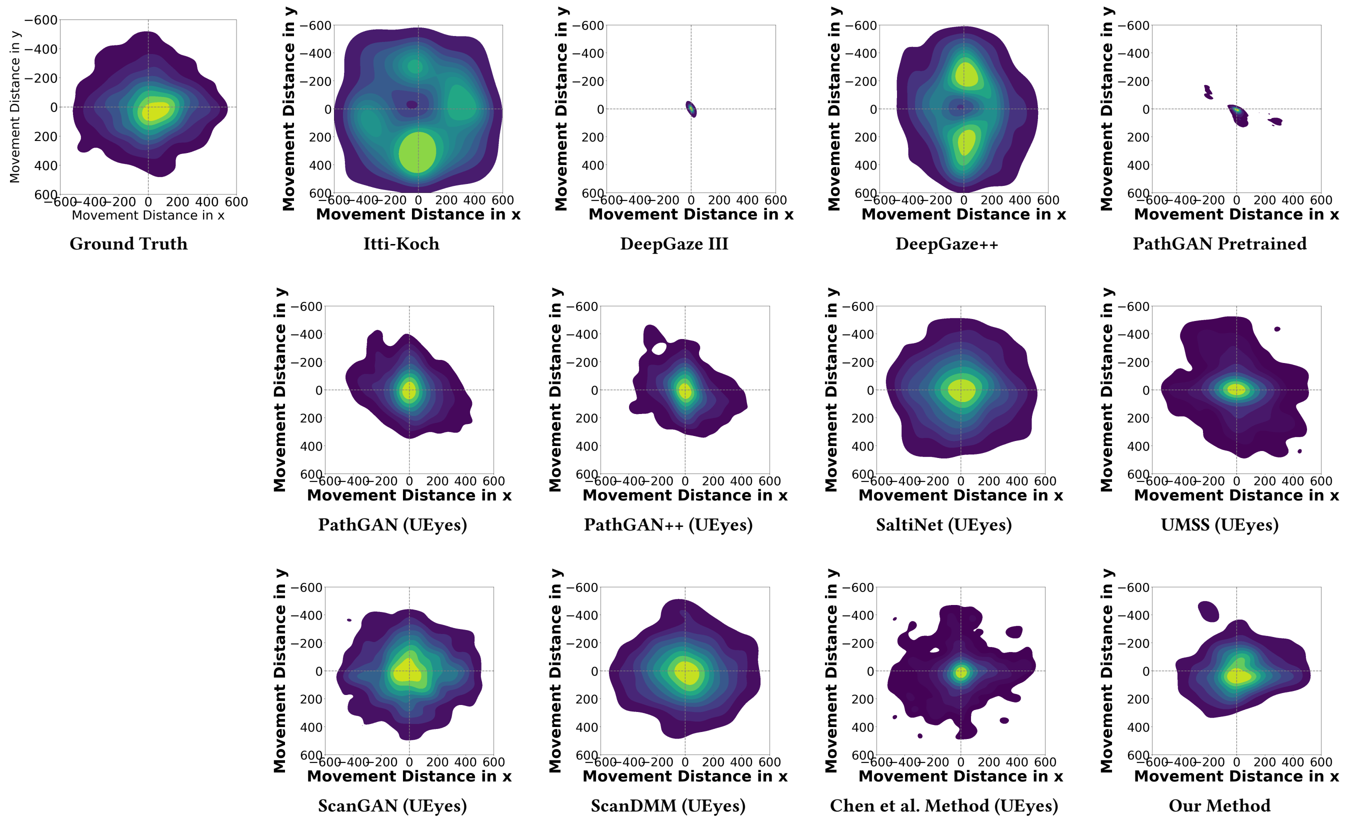


Figure 8: Plots of ground-truth and predicted gaze directions. Analyzing saccade bias yields insight related to directions and distances between consecutive fixation points. For instance, real-world gaze directions' convergence toward the right and bottom of the images, with greater distances observed particularly near the right, suggests a preference for directing the gaze from left to right (with fairly large motions) and from top to bottom. Our model captures this property better than others do.

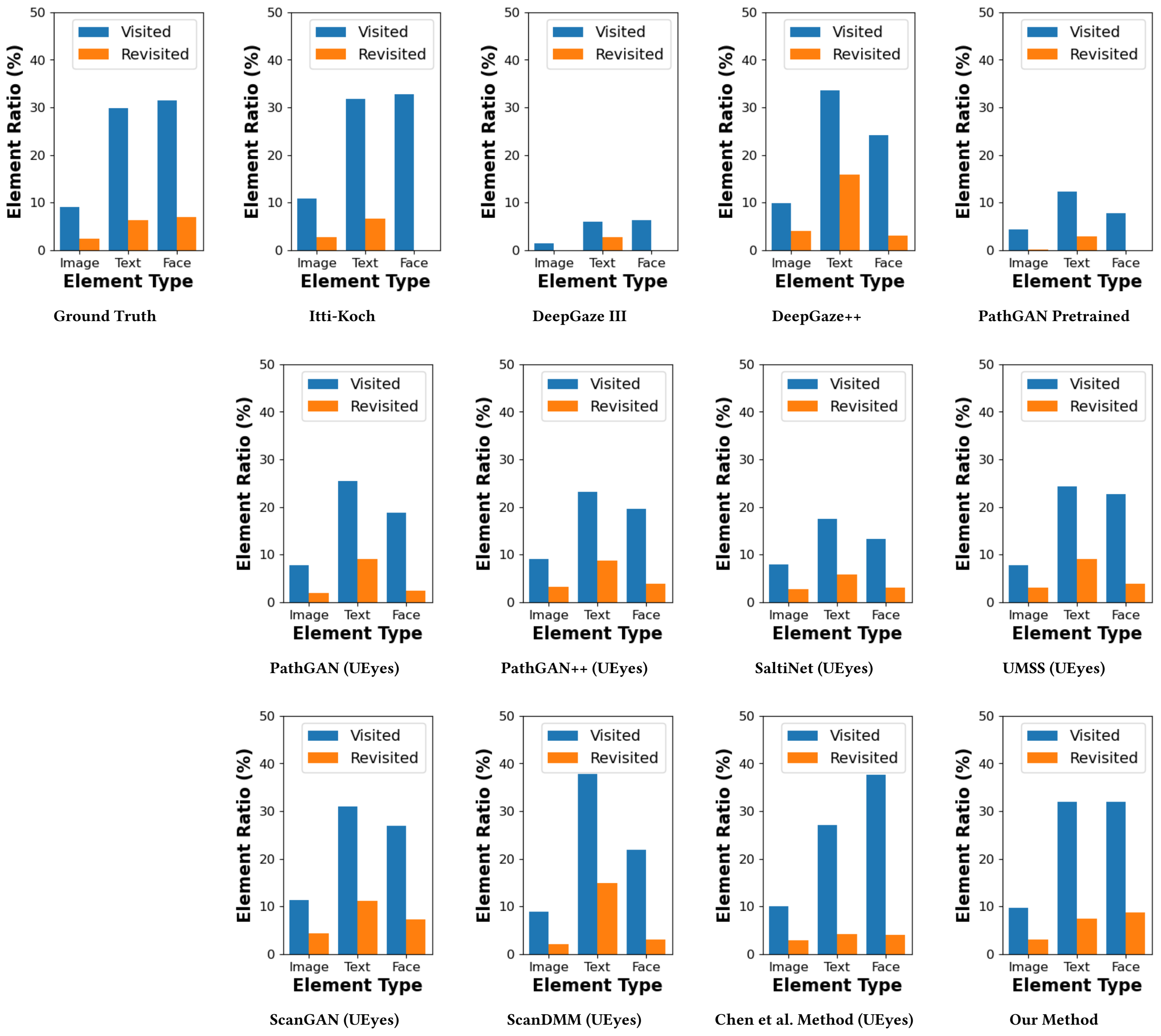


Figure 9: For each scanpath-prediction model, the percentage of the elements of each type visited and the same for elements returned to. Of the models evaluated, ours produced the estimation closest to the ground truth. The "Visited" figure is calculated from dividing the count of visited elements by the number of elements belonging to the element-type class in question, across all GUI images. We consider an element re-visited when it is subject to two or more non-consecutive fixations in the sequence.

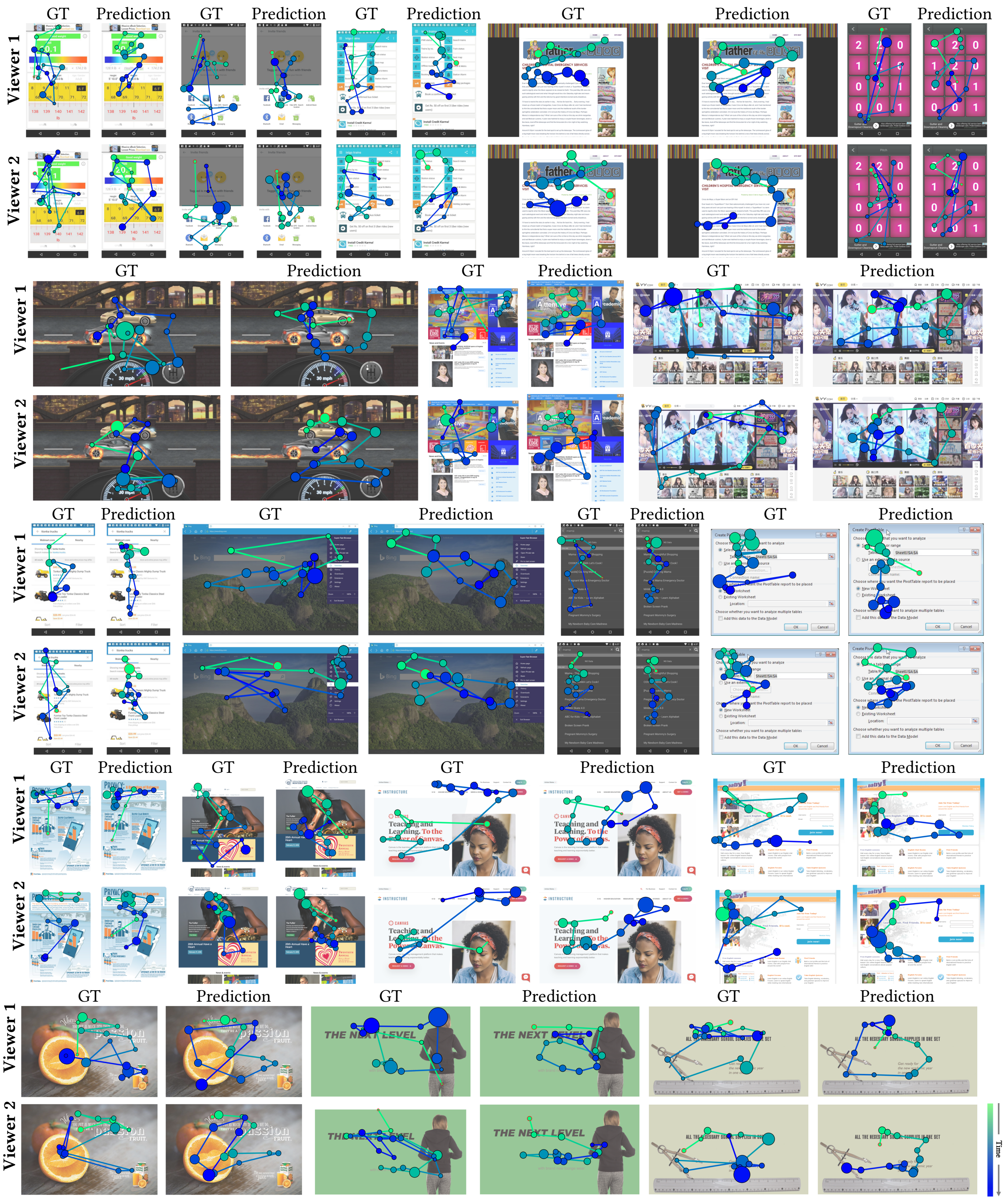


Figure 10: Scanpaths personalized for two viewers, illustrating our model's ability to generate these by means of only a few scanpath samples from each viewer (note that "Viewer 1" and "Viewer 2" are generic terms; the viewers are not the same across all examples).

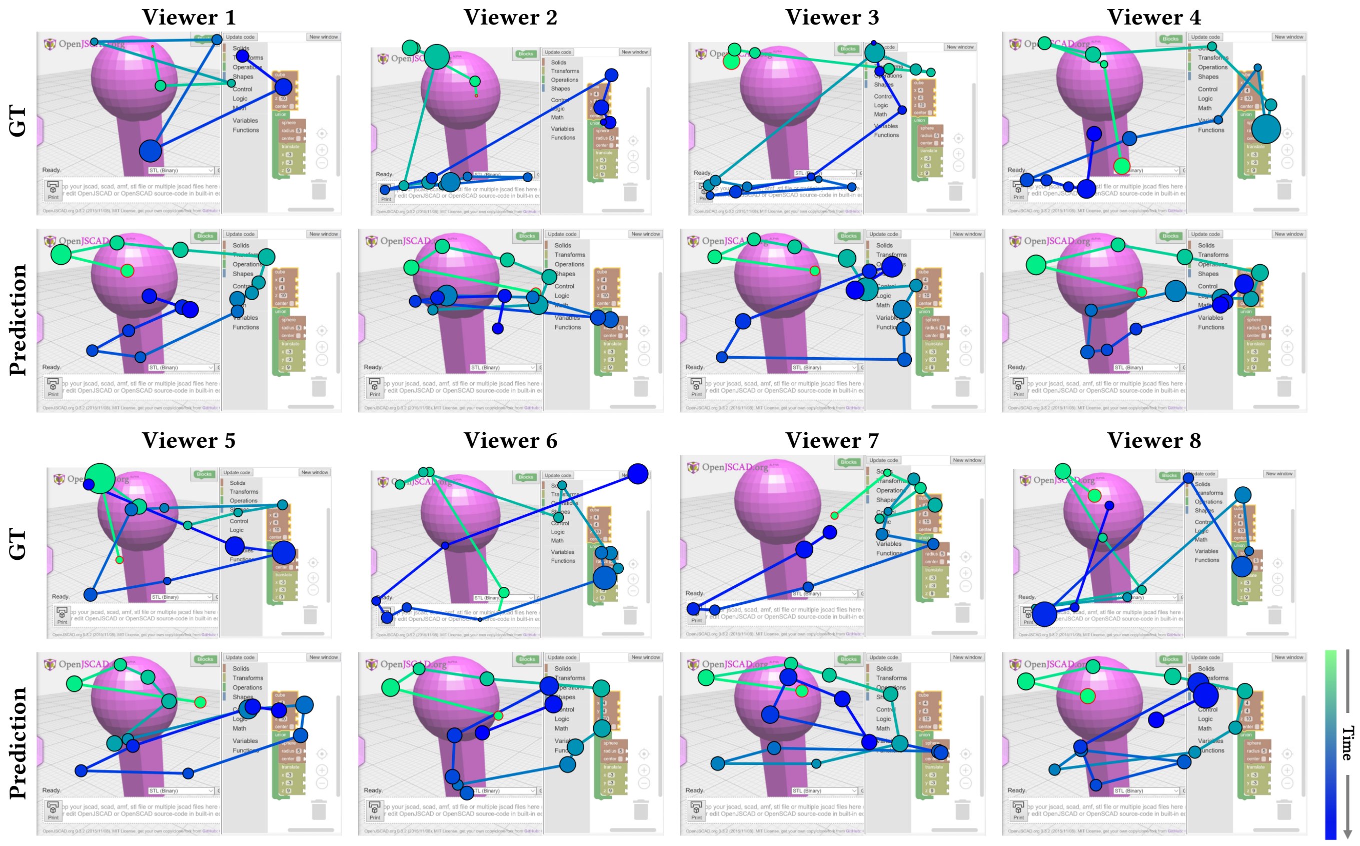


Figure 11: Example scanpath predictions for several distinct viewers.