U2Eyes: a binocular dataset for eye tracking and gaze estimation

Sonia Porta
Public University of Navarre
Pamplona, Spain
sporta@unavarra.es

Benoît Bossavit
Trinity College Dublin
Dublin, Ireland
bossavib@scss.tcd.ie

Rafael Cabeza
Public University of Navarre
Pamplona, Spain
rcabeza@unavarra.es

Andoni Larumbe-Bergera
Public University of Navarre
Pamplona, Spain
andoni.larumbe@unavarra.es

Gonzalo Garde
Public University of Navarre
Pamplona, Spain
gonzalogarde3@gmail.com

Arantxa Villanueva
Public University of Navarre
Pamplona, Spain
avilla@unavarra.es

Abstract

Theory shows that huge amount of labelled data are needed in order to achieve reliable classification/regression methods when using deep/machine learning techniques. However, in the eye tracking field, manual annotation is not a feasible option due to the wide variability to be covered. Hence, techniques devoted to synthesizing images show up as an opportunity to provide vast amounts of annotated data. Considering that the well-known UnityEyes tool provides a framework to generate single eye images and taking into account that both eyes information can contribute to improve gaze estimation accuracy we present U2Eyes dataset, that is publicly available. It comprehends about 6 million of synthetic images containing binocular data. Furthermore, the physiology of the eye model employed is improved, simplified dynamics of binocular vision are incorporated and more detailed 2D and 3D labelled data are provided. Additionally, an example of application of the dataset is shown as work in progress. Employing U2Eyes as training framework Supervised Descent Method (SDM) is used for eyelids segmentation. The model obtained as result of the training process is then applied on real images from GI4E dataset showing promising results.

1. Introduction

Some years ago researchers realized that the users community of infrared eye tracking systems was reduced due to the technical and hardware constraints of the technology. In order to broaden the applications of eye tracking systems the technology should be cheaper and more "plug & play", e.g. webcams or mobile gadgets cameras. Previously, feature based image processing techniques and gaze estimation methods were used. Today, one of the main research topics of the groups working in eye tracking and gaze estimation technology for low-resolution systems involves developing, among others, landmark detection and gaze estimation techniques applying machine learning and deep learning algorithms [8] [7].

Among the obstacles that eye tracking community encounters when facing the challenge of low resolution gaze estimation, the lack of large scale labelled datasets to be used for these purposes is remarkable. Ideally, datasets including images of the eye/face area are required where not only face but also eye area landmarks (eyelids, iris, pupil) are included. Moreover, images should be annotated with gaze information and, preferably, head pose should be also labelled.

Many efforts have been made by researchers in order to generate large datasets containing the corresponding labels [14] [5]. However, although deep learning techniques proved to be successful in most areas of research, the accuracies obtained using these datasets in terms of gaze estimation are insufficient. One of the hypothesis is that the models do not learn to generalize because datasets employed for training purposes lack of enough variability. In order to enlarge these datasets size and trying to avoid the burdensome manual labelling option other possibilities have been proposed, such as image augmentation techniques or synthesizing images [9] [2] [10] [1].

UnityEyes is one of the main contributions to the field of eye tracking devoted to creating artificial eye images [13]. This open source software provides the possibility of creating monocular images in which the head shape, skin and iris texture, head pose, gaze direction and camera parameters can be controlled. The eyeball model employed by UnityEyes is a simplified eye model that resembles most of the characteristics required by videoculograhpy theory [3] [11]. However, this model has two cons: first, the *kappa*

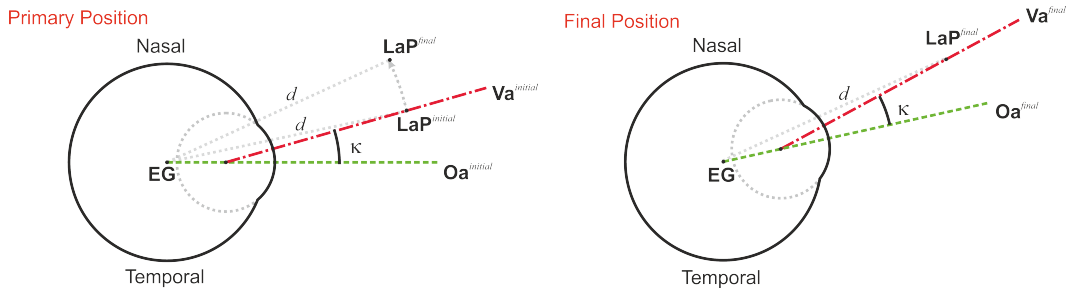

Figure 1: In the upper part of the figure the primary position is shown together with the **LaP** in the final position. An imaginary axis is calculated in both, initial and final positions connecting **EG** to $\mathbf{LaP}^{initial}$ and $\mathbf{LaP}^{final}$ respectively. After estimating the rotation between the imaginary axes it is applied to the whole eyeball. In the lower part, both optical and visual axes have been rotated accordingly to the final position. Now, the visual axis points to the pursued $\mathbf{LaP}^{final}$.

angle, κ, which is one of the key aspects of eyeball physiology and was demonstrated to be essential in order to develop feasible gaze estimation models, is always 0. Second, UnityEyes does not permit to create binocular images, preventing methods from improving the accuracy on low resolution systems by combining the information of both eyes.

In this paper we present U2Eyes dataset. It contains binocular images created using UnityEyes as starting point in which essential eyeball physiology elements are included and binocular vision dynamics are modelled. The images are annotated with head pose and gaze direction information. In addition, 3D and 2D landmarks are provided as part of the annotated data. In the following section, the basics about eye modelling required for videoculography are reviewed. In sections 3 and 4 details about the database and its construction are provided. Finally, an example of a possible application of the database using real images is shown.

2. Binocular vision dynamics

Eyeball geometry plays an important role in a database like the one described in this paper. Videoculography theory, more specifically those works devoted to gaze estimation using geometrical models, have established the minimum eyeball characteristics that should be modelled in order to obtain a reliable gaze estimation system [3]. Images and files provided in our dataset were produced according to this minimum model introducing the corresponding improvements with respect to UnityEyes.

First, if we consider the eyeball as a static 3D object, the following aspects should be modelled in the Unity framework: the cornea is approximated as a single surface sphere. Corneal refraction is critical and should be carefully modelled. The eyeball center is a fixed point around which the eye rotates. The Line of Sight (LoS) can be approximated by the visual axis, connecting the fovea with the focused point. This imaginary line is modelled as the line presenting individual's specific angle, κ, with respect to the eyeball symmetry axis and crossing the center of the cornea. Angle κ presents horizontal (3-7°) and vertical (2-3°) components. Although most of models consider the vertical component as negligible we included it in our eye model for completeness. The symmetry axis of the eye is named as optical axis, and it is assumed to contain the eyeball, cornea and pupil centers.

Second, if we now think about eyes as moving 3D objects, rotation dynamics should be studied. Several gaze estimation papers model the minimum eyeball rotation features to be included by any eye tracking method based on geometry [11]. In theory, when moving from a primary to a tertiary position the eyeball rotates about an axis perpendicular to a plane containing visual axes in both positions. Eyeball dynamics are nicely explained using Donder's Law and more specifically Listing's Law [4]. Our proposal, implemented in Unity, is to rotate the eyeball using as reference an imaginary axis connecting the eyeball globe center (WCS origin), **EG**, and the look-at-point in the final position $\mathbf{LaP}^{final}$, i.e $\mathbf{EG} - \mathbf{LaP}^{final}$, where $\|\mathbf{EG} - \mathbf{LaP}^{final}\| = d$ (see figure 1). In the primary position the visual axis **Va** can be easily calculated by knowing both, the vertical and horizontal angular offsets between optical, i.e. $\mathbf{Oa} = (0, 0, 1)^T$, and visual axes. The $\mathbf{LaP}^{initial}$ is calculated as that fictional point in the visual axis in the primary position at a distance d from **EG**. Then, the imaginary axis in the initial position is calculated as, $\mathbf{EG} - \mathbf{LaP}^{initial}$. Eyeball rotation is calculated as the rotation between imaginary axes in initial and final positions. In figure 1 a summary of the implementation is shown.

Since our dataset provides two eyes images gazing at the same point binocular vision mechanisms were implemented. Once the visual axis of the second eye is defined, the rest of the eye elements, i.e. optical and imaginary axes, are calculated and incorporated into the framework. Thus, the same procedure is applied in order to estimate the pose and rotation of the second eye.

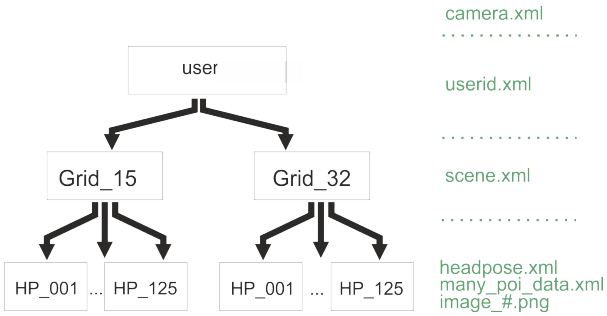

Figure 2: Folders and files arrangement of U2Eyes.

3. U2Eyes DATASET

U2Eyes database includes 1000 users, at a rate of 5,875 images (in png format) and 503 files (in xml format) per user, arranged in independent folders in a three levels structure as shown in figure 2. In total, about 6 million images are contained in the dataset.

A user's complete folder requires about 2.5 GB and its whole generation takes three and a half hours when running over a 6^{th} Generation Intel Core i5 of 3.2 GHz and 20 GB RAM. Due to space limitations a reduced version of the dataset is publicly available (link). Each user is identified by a different face shape, according to a PCA model offered by UnityEyes, whereas the 20 skin-textures and 5 eye-textures available in UnityEyes are regularly distributed over the dataset (individual data saved in the corresponding userid.xml file). The scene.xml file, that is also user's specific, contains information about light color/direction/intensity and exposure/rotation of the scene. This file has not changed from the original UnityEyes [13], so that the same ranges for lighting parameters and identical 18 available regarded scenes are involved in the dataset generation. Data such as camera focal length (3.67 mm) and images resolution (4K=3840x2160 pixels) are stored in the camera.xml file, which is shared by all the users.

4. Design of the dataset

U2Eyes is an extension of UnityEyes [13] that calculates the entire user's eyes area. For this purpose, the mesh provided by UnityEyes is first duplicated, then transformed by inverting the scale on the Y-axis in order to be symmetric, and finally translated to match the left side of the face. All these operations are calculated and drawn in real-time. This provides a generic application where a new mesh created by UnityEyes could also be used by U2Eyes without manual editing.

Design decisions regarding the number of head poses and look-at-points for the dataset were taken trying to emulate real situations and reproducing scenarios contained in existing datasets with real subjects. A completely random framework, although computationally possible, could result in non feasible head pose and look-at-points combinations in reality. Therefore, carefully studied random components were introduced in the dataset in order to increase the variability and avoid overfitting when the database is used in learning algorithms, as it is detailed next.

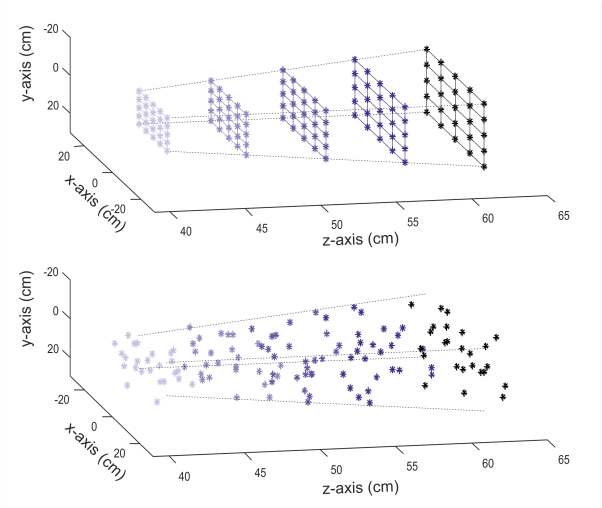

Figure 3: In the upper part, the 125 head positions following a frustum distribution and at five different distances from the camera are shown. In the lower part, the 125 head positions once the random component is introduced are shown.

For each user 125 head poses combining head center position and face orientation were modelled and saved in the corresponding headpose.xml file. The head center positions were initially distributed over five planes at different depths, to provide images where the user's face is in between 30 and 50 cm from the camera. Plane size grows with the distance, always containing 5x5 points. An example of frustum distribution is shown in figure 3 (up).

These 125 points are then randomized adding some uniform noise in the range $[-\Delta, \Delta]$ where, for each spatial direction and each plane, Δ equals a fifth of the frustum size, yielding a distribution as that shown in figure 3 (down). Dimensions were carefully selected to provide realistic images, with no eye area clipping and able to resemble those available in physical datasets. In order to avoid baleful look images, a facing algorithm was designed to ensure that the face is directed towards the gazed area. Once the face is oriented, some additional random noise is added to the rotation components (roll, yaw and pitch), uniformly distributed in the range of $[-5°, 5°]$ for the closest plane and progressively increased up to $[-8°, 8°]$ in the farthest one.

At each head pose two gazing grids are observed (see figure 2), respectively containing 15 points (for training

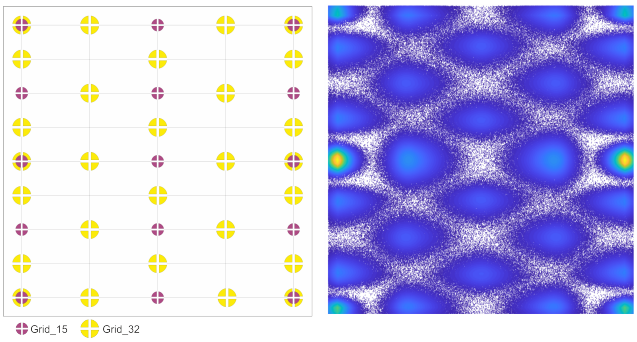

Figure 4: Left: each user gazes 15 and 32 grids points from every 125 head poses. Right: representation of random distributions of look-at-points in the screen.

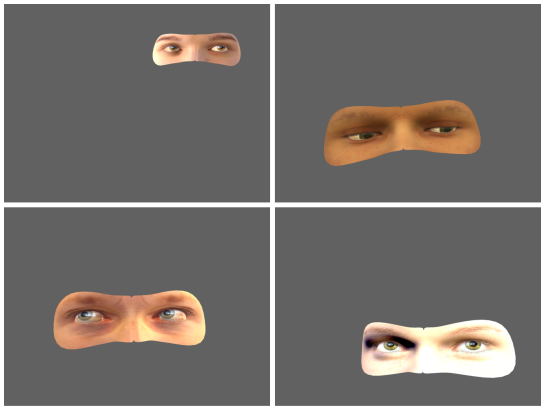

Figure 5: Images from U2Eyes dataset.

purposes) and 32 points (conceived for test) not uniformly spread in a 220x220 mm squared surface (see figure 4, left). In this manner, for each user a total amount of 125x(15+32) images are generated. While the 15 points grid is kept the same for all users and head poses, over the 32 points grid some Gaussian noise, designed independently for each one of the four quadrants per point, is added. Random distributions calculated for each point after 5000 iterations are shown in figure 4 (right) having into account the screen limits and the non-uniform distribution of points.

The camera is assumed to be located in the center of the gazed area. In this way a symmetrical framework is selected and we can simulate both physical scenarios, i.e. having the camera in the upper or in the lower part of the gazed screen. In figure 5 examples extracted from the dataset are provided.

Regarding eye features, U2Eyes dataset increases the information provided by UnityEyes. An output xml file stored in each head pose folder provides information about 2D/3D landmarks of eyelids, iris and pupil contour points and centers, caruncle and eye corners. In figure 6 an example of an image from U2Eyes dataset is shown, in which 2D eye landmarks are plotted.

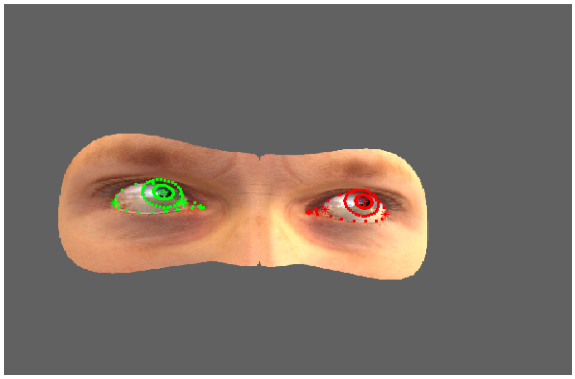

Figure 6: U2Eyes image in which the 2D landmarks can be observed.

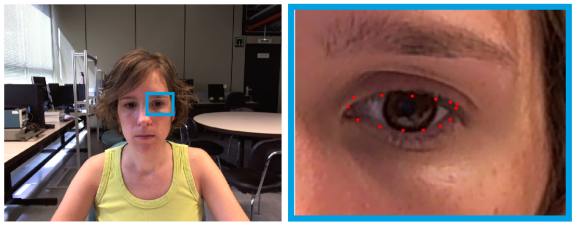

Figure 7: In the upper part the original GI4E image is shown highlighting Region Of Interest (ROI) in blue. In the lower part the cropped ROI is represented in which the points resulting from the segmentation are plotted in red.

5. Example of application

As mentioned before, this work wants to contribute to the challenge of obtaining large scale labelled datasets for eye tracking purposes. We would like to show briefly one of the potential applications in which we are working at the moment. The objective is to use Supervised Descent Method (SDM) as eyelids detection technique [6]. The algorithm is trained using images and labels from U2Eyes and applied to real scenarios. This is a "work in progress" project for which promising results are obtained, that is mentioned here to show the possibilities offered by U2Eyes. In figure 7 the algorithm is applied to one of the GI4E images [12]. In the lower part the result of the segmentation using the SDM model is provided. As it can be observed the model achieves to detect eyelids points nicely. Results for the left eye are shown but information from the right eye was also used in the SDM model.

6. Conclusions

In this work U2Eyes dataset is presented. U2Eyes is a binocular database of synthesised images reproducing real gaze tracking scenarios and including corresponding 2D/3D labels. It contributes to the area of large scale datasets to be used in machine/deep learning field for eye tracking. A potential eyelids segmentation tool is shown as example of application.

7. Acknowledgment

We would like to acknowledge the Spanish Ministry of Science, Innovation and Universities for their support under Contract TIN2017- 84388-R.

We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan X Pascal GPU used for this research.

We also thank Erroll Wood for his useful comments.

References

[1] Shaharam Eivazi, Thiago Santini, Alireza Keshavarzi, Thomas Kübler, and Andrea Mazzei. Improving real-time CNN-based pupil detection through domain-specific data augmentation. In *Proceedings of the 11th ACM Symposium on Eye Tracking Research and Applications*, ETRA'19, pages 40:1–40:6, New York, NY, USA, 2019. ACM.

[2] Chao Gou, Y. Wu, Kang Wang, Fei-Yue Wang, and Q. Ji. Learning-by-synthesis for accurate eye detection. In *23rd International Conference on Pattern Recognition (ICPR'16)*, pages 3362–3367, Dec 2016.

[3] Elias Daniel Guestrin and Moshe Eizenman. General theory of remote gaze estimation using the pupil center and corneal reflections. *IEEE Trans. on Biomedical Engineering*, 53(6):1124–1133, 2006.

[4] Amir A. Handzel and Tamar Flash. The geometry of eye rotations and Listing's law. In *Proceedings of the 8th International Conference on Neural Information Processing Systems*, NIPS'95, pages 117–123, Cambridge, MA, USA, 1995. MIT Press.

[5] Kyle Krafka, Aditya Khosla, Petr Kellnhofer, Harini Kannan, Suchendra Bhandarkar, Wojciech Matusik, and Antonio Torralba. Eye tracking for everyone. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'16)*, 2016.

[6] Andoni Larumbe-Bergera, Sonia Porta, Rafael Cabeza, and Arantxa Villanueva. SeTA: Semiautomatic tool for annotation of eye tracking images. In *Proceedings of the 11th ACM Symposium on Eye Tracking Research and Applications*, ETRA'19, pages 45:1–45:5, New York, NY, USA, 2019. ACM.

[7] J. Lemley, A. Kar, A. Drimbarean, and P. Corcoran. Convolutional neural network implementation for eye-gaze estimation on low-quality consumer imaging systems. *IEEE Transactions on Consumer Electronics*, 65(2):179–187, May 2019.

[8] Seonwook Park, Xucong Zhang, Andreas Bulling, and Otmar Hilliges. Learning to find eye region landmarks for remote gaze estimation in unconstrained settings. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, (ETRA'18), pages 1–10, Warsaw, Poland, 2018. ACM.

[9] Yusuke Sugano, Yasuyuki Matsushita, and Yoichi Sato. Learning-by-synthesis for appearance-based 3D gaze estimation. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'14)*, pages 1821–1828. IEEE Computer Society, 2014.

[10] Lech Świrski and Neil A. Dodgson. Rendering synthetic ground truth images for eye tracker evaluation. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, ETRA'14, pages 219–222, New York, NY, USA, 2014. ACM.

[11] Arantxa Villanueva and Rafael Cabeza. A novel gaze estimation system with one calibration point. *Trans. on Systems, Man and Cybernetics, Part B*, 38(4):1123–1138, Aug. 2008.

[12] Arantxa Villanueva, Victoria Ponz, Laura Sesma-Sanchez, Mikel Ariz, Sonia Porta, and Rafael Cabeza. Hybrid method based on topography for robust detection of iris center and eye corners. *ACM Trans. on Multimedia Computing, Communications and Applications*, 9(4):25:1–25:20, Aug. 2013.

[13] Erroll Wood, Tadas Baltrusaitis, Louis-Philippe Morency, Peter Robinson, and Andreas Bulling. Learning an appearance-based gaze estimator from one million synthesised images. In Pernilla Qvarfordt and Dan Witzner Hansen, editors, *Proceedings of the Symposium on Eye Tracking Research and Applications*, ETRA'16, pages 131–138. ACM, 2016.

[14] Xucong Zhang, Yusuke Sugano, Mario Fritz, and Andreas Bulling. MPIIGaze: Real-world dataset and deep appearance-based gaze estimation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Early Access, 2018.