EyeKnowYou: A DIY Toolkit to Support Monitoring Cognitive Load and Actual Screen Time using a Head-Mounted Webcam

Tharindu Kaluarachchi
tharindu@ahlab.com
Augmented Human Lab, Auckland
Bioengineering Institute, The
University of Auckland
New Zealand

Shardul Sapkota
Augmented Human Lab, Auckland
Bioengineering Institute, The
University of Auckland
New Zealand
Yale-NUS College
Singapore

Jules Taradel
Augmented Human Lab, Auckland
Bioengineering Institute, The
University of Auckland
New Zealand

Aristée Thevenon
Augmented Human Lab, Auckland
Bioengineering Institute, The
University of Auckland
New Zealand

Denys J.C. Matthies
Augmented Human Lab, Auckland
Bioengineering Institute, The
University of Auckland
New Zealand
Technical University of Applied
Sciences Lübeck
Germany

Suranga Nanayakkara
Augmented Human Lab, Auckland
Bioengineering Institute, The
University of Auckland
New Zealand

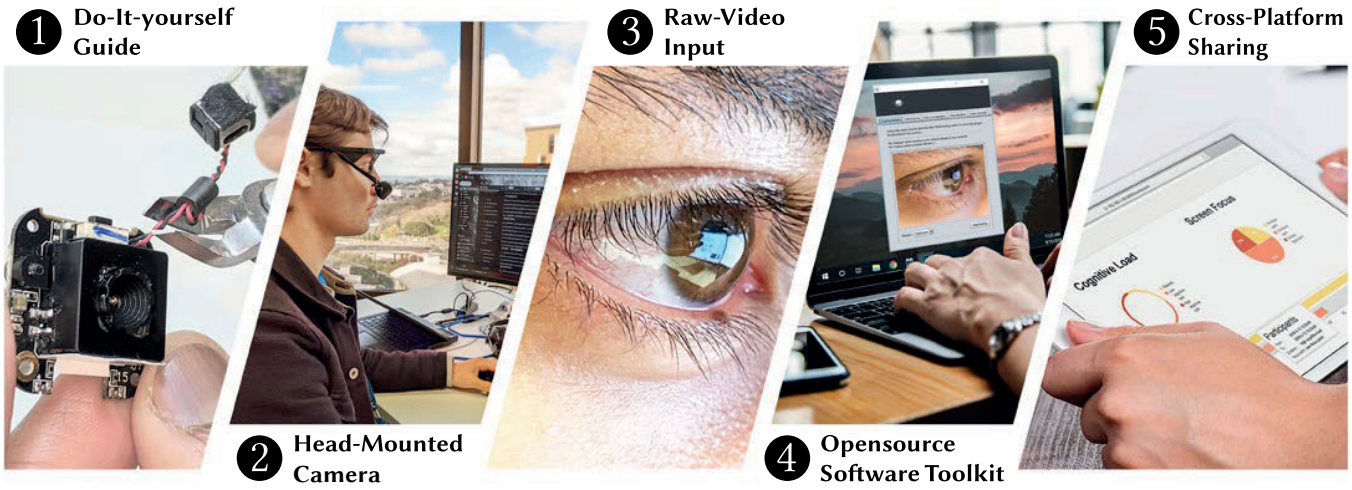

Figure 1: This paper contributes a toolkit for researchers interested in detecting an increased cognitive load and actual screen time. We present (1) a Do-It-Yourself guide to easily build (2) a Head-mounted Eye Recorder based on a regular webcam. We use the (3) camera's raw data as an input for our (4) Opensource Software Toolkit, which utilizes a pre-trained machine-learning model we developed based on a 3D Convolutional Neural Network. The (5) results can then be shared cross-platform via a web interface.

MobileHCI '21, Sept 27 – Oct 1, 2021, Toulouse, France & virtually

ACM ISBN 978-1-4503-8329-5/21/09…$15.00
https://doi.org/10.1145/3447527.3474850

ABSTRACT

Studies show that frequent screen exposure and increased cognitive load can cause mental-health issues. Although expensive systems capable of detecting cognitive load and timers counting on-screen time exist, literature has yet to demonstrate measuring both factors across devices. To address this, we propose an inexpensive DIY-approach using a single head-mounted webcam capturing the user's eye. By classifying camera feed using a 3D Convolutional Neural Network, we can determine increased cognitive load and actual screen time. This works because the camera feed contains corneal surface reflection, as well as physiological parameters that

contain information on cognitive load. Even with a small data set, we were able to develop generalised models showing 70% accuracy. To increase the models' accuracy, we seek the community's help by contributing more raw data. Therefore, we provide an opensource software and a DIY-guide to make our toolkit accessible to human factors researchers without an engineering background.

CCS CONCEPTS

• **Computing methodologies** → *Activity recognition and understanding*; • **Human-centered computing** → **User interface toolkits**.

KEYWORDS

software toolkits; neural networks; screen time; cognitive load

ACM Reference Format:
Tharindu Kaluarachchi, Shardul Sapkota, Jules Taradel, Aristée Thevenon, Denys J.C. Matthies, and Suranga Nanayakkara. 2021. EyeKnowYou: A DIY Toolkit to Support Monitoring Cognitive Load and Actual Screen Time using a Head-Mounted Webcam. In *MobileHCI '21: The ACM International Conference on Mobile Human-Computer Interaction, Sept 27 – Oct 1, 2021, Toulouse, France & virtually*. ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3447527.3474850

1 INTRODUCTION

Recent research has shown that long exposure to screens can lead to screen dependency disorder [28] and Campbell [3] has shown that high screen exposure among children results in higher probabilities of depression diagnosis, compared to low screen users. To determine screen time, many applications offer counters and app timers, such as those implemented in smartphones. However, these are estimations because it is unknown whether the user indeed looked at the screen or not. Along with prolonged screen time, frequently occurring high cognitive loads can worsen the aforementioned effects [25]. Thus, tracking both screen time and changes in cognitive load is vitally important. To determine cognitive load, research proposes the use of eye-related measurements such as pupil diameter [23], blinks [22], and pupil oscillations [8].In contrast to prior work, in EyeKnowYou, we measure both cognitive load and actual screen time using a 3D Convolutional Neural Network.

Following a Human-Centered Machine Learning [14] approach, we interviewed prospective users to understand the user needs. We used a single webcam, mounted on an eye-tracker frame to the head, which records the user's eye. This way, by leveraging spatio-temporal features, we trained a deep learning model to detect an increased cognitive load and track whether the user is actually looking at a screen. Based on the user needs, we developed a software toolkit to interact with our model. We collected data from 17 participants with diverse ethnic backgrounds and eye colors, and we conducted our data recording in different environmental settings and under varying lighting conditions [24]. Although developing a robust and generalized model under such conditions was challenging, we achieved 70% accuracy with a comparably small training set. Figure 1 shows an overview of the paper. In summary, we contribute with:

- A straightforward DIY opensource software toolkit and evaluation of its usability.
- All the source codes we developed and trained parameter values of the neural network for the research community to build upon and towards a new methodology for human factors researchers to measure cognitive load simultaneously with actual screen time.

2 RELATED WORK

For many decades, researchers exploring social implications of technology have looked into how people track different aspects of their lives [6, 26]. More recently, the increase in multi-device environments have sparked a renewed interest in how people spend time using their devices [18]. Studies in this area have often been conducted using software installations on the users' target devices, whereas some studies were done without any user interface [7, 15]. Also, there has been released screen time tracking apps through different app stores, which provide a separate interface to understand the tracking process [4].

A meta analysis [5] has highlighted that 16 eye-related measurements have shown correlations with cognitive load. Meanwhile, eye-tracking [32] is a common method to estimate cognitive load where some researchers [9] have used eye-tracking measures like pupil dilation and spontaneous blink rate while using eye-tracking to identify focus point on the screen. Yang et al. [30] have used eye gaze to measure cognitive load in real-world driving scenarios while Fridman et al. [10] have used a camera to measure cognitive load using the video feed of the eye by running a machine learning approach, particularly Deep Learning algorithms. This can be identified as the most related work as it utilizes spatio-temporal features from video recordings of the eye to determine the level of cognitive load.

Researchers have also attempted to identify gaze and recognize objects from such reflections [11], although the capability was limited. A few researchers have looked into using corneal surface reflection to determine the context of a scene [1, 20, 21]. Previous work have explored the use of a single camera capturing the spherical corneal reflections for lifelogging [16, 17] and reconstructing scene which the user is observing. [19, 29]. Although not explicitly shown yet, we can infer on the user's actual screen time.

3 EYE KNOW YOU

3.1 Motivation

Two major limitations in prior works are the incapability of detecting whether the user is actually looking at the screen and the inability to sum up the overall screen time across multiple devices (personal and non-personal devices). Looking at the corneal surface reflection [19] of the eye at any time could reveal such information, overcoming both issues. To achieve this, we decided to train a 3D Convolutional Neural Network [13], given the rapid advancement and demonstrated superior capabilities of this technology. We conducted three semi-structured interviews with experts. The interviewees were practitioners in the field of User Experience and either held a Master's or a PhD degree. All of our interviewees use EDA measurement as the state-of-the-art method to determine cognitive load. Not surprisingly, P1 stated difficulties with processing EDA data. *"This [data processing] is actually software engineering work."* P2 mentioned that she was looking for a solution to automatically trigger cognitive load sensing just in the moment the user is

looking at the screen. After we described our envisioned system capable of sensing cognitive load and context, such as screen time, all three interviewees sounded interested.

3.1.1 System Requirements. Based on prior experience and expert interviews, we derived these system requirements:

- Simple data recording with or without labels along with timestamps plus save and export function,
- Both post-processing and real-time classification
- Easy installation process with step-by-step guidelines
- Resource considerate, as end-user hardware has limited processing power to handle all calculations
- Visualisation of input and output data enabling efficient understandings and sharing of results
- Non-expert friendly – coding or programming expertise is not required

3.2 Toolkit

We designed EyeKnowYou, a self-contained DIY toolkit utilizing a simple webcam mounted to the head focusing on the user's eye. This way, we can determine an increased cognitive load using physiological properties and simultaneously identify whether the user is actually looking at a screen through reflections of the eye's corneal surface. Initially released as a research apparatus, we envision a future where EyeKnowYou becomes a wearable life logger or a classroom tool for teachers to better understand student engagement and cognitive load. Using a single inward looking camera generates lower power consumption compared to using an additional world camera to get the context. Moreover, privacy concerns are also not an issue [31]. Suitable for our device, we provide an opensource software toolkit with a graphical user interface enabling easy-use for researchers without technical background. In the Appendix, we explain how to replicate our device. We opensource all the source codes, learned weights of the neural network model, and our dataset for tech-savvy researchers enabling for a further development.

3.2.1 Installing and Connecting to the Toolkit: Our Java-based software toolkit is available through our website[1]. It requires having the Java RunTime Environment[2] (JRE) installed. A user can check whether JRE is installed by simply running the *"PythonModuleInstaller.jar"* file. The core of our toolkit utilizes Keras[3], which runs on top of the Tensorflow[4] backend to run the neural network, which requires a 64-bit processor. A "readme.pdf" file in the installation folder contains further guidelines including how to check whether your processor and operating system architecture match the requirements. Executing *"PythonModuleInstaller.jar"* file will install all the dependencies including Keras and Tensorflow.

After both packages are successfully installed, the machine is set to run the EyeKnowYou software toolkit. Now, the camera can be plugged in to the machine. Most Windows operating systems do not require a manual driver installation. Once the camera is successfully installed, disabling the camera's autofocus and manually setting the focus is required. We provide a video[5] demonstrating how to carry out this using the default *"Camera"* application of Windows 10. This step is only required once when the camera is plugged. For users using Linux-based environments, disabling autofocus is unnecessary, given our source code handles this itself. Executing the *"EyeKnowYou.jar"* will run our software.

3.2.2 Handling the Toolkit (Walk-through): The EyeKnowYou toolkit enables the user to record data, classify, and export results. To do this, the researcher is required to set some parameters, which this walk-through will describe. EyeKnowYou consists of four independent tabs and each tab carries out a different task. The following subsections explain what each tab does and how to execute it. All the tabs are shown in Figure 2.

Test Camera: When trying the app for the first time or connecting a new camera, the *"Test Hardware"* tab is useful to check whether the plugged camera is fully working. The user will be able to select the camera (Internal or External Camera) from the dropdown menu. Clicking "Start testing" after selection, will visualize the video.

Record Data: This tab (*see Figure 2–a*) helps recording data and saving in HDF5 format. The first field is "Camera", to which the researcher should select the DIY webcam that is pointing towards the eye. In the "File Directory" area, the storage location of the recorded data needs to be specified. The last two optional parameters – Scene and Cognitive Load – can be used for labelling your data if the intention is to collect data to contribute to the dataset. A terminal window will pop up after the "Start collecting" button is pressed where the user needs to follow the instructions displayed.

Classify Recorded Data: Using this tab (*see Figure 2–b*), the previously recorded data files can be processed. The data files are processed in 64-frame batches and classified by our pre-trained model (context – looking at a screen: yes/no, and cognitive load: high/low). This results in a resolution of about 4 seconds. Note that the calculation requires a considerable amount of processing power. Closing all or some other applications is recommended depending on the computational power available. The results will be stored in the user-chosen file location in human readable *.csv format, which can be opened in any spreadsheet application.

Visualize Data: In this tab (*see Figure 2–c*), the operator can replay video frames from a file previously recorded (in HDF5 format). This option is useful to visually inspect the recorded data. To visualize and share the classified results we developed a Webapp which shows the cognitive load and screen time over the duration of the recording.

3.3 Usability Test

To gain an initial impression on our toolkit's usability, we ran a pilot user study with three young HCI researchers aged 23, 25, and 29 – all males with some experience in running studies. The overall average SUS score across these participants was 56% (*SD*=16.5). The major problem all participants mentioned was the image cropping procedure at the beginning of the data recording process. Several software bugs our tool had at the time provoked the low SUS score. However, all participants were able to achieve the set goals. P3 stated: *"The guide was very easy to follow, as many videos illustrated the procedure [...] The interface was also intuitive."*

Following the outcomes of the pilot study, we identified the weak points of the application and fixed the usability issues. Then we recruited 8 new participants (age *M*=26.75, *SD*=3.6) who were researchers in the HCI field. They were familiar with the concept of

[1] https://sites.google.com/view/eye-know-you/home
[2] https://www.oracle.com/technetwork/java/javase/downloads/
[3] https://keras.io
[4] https://www.tensorflow.org
[5] https://vimeo.com/360496718

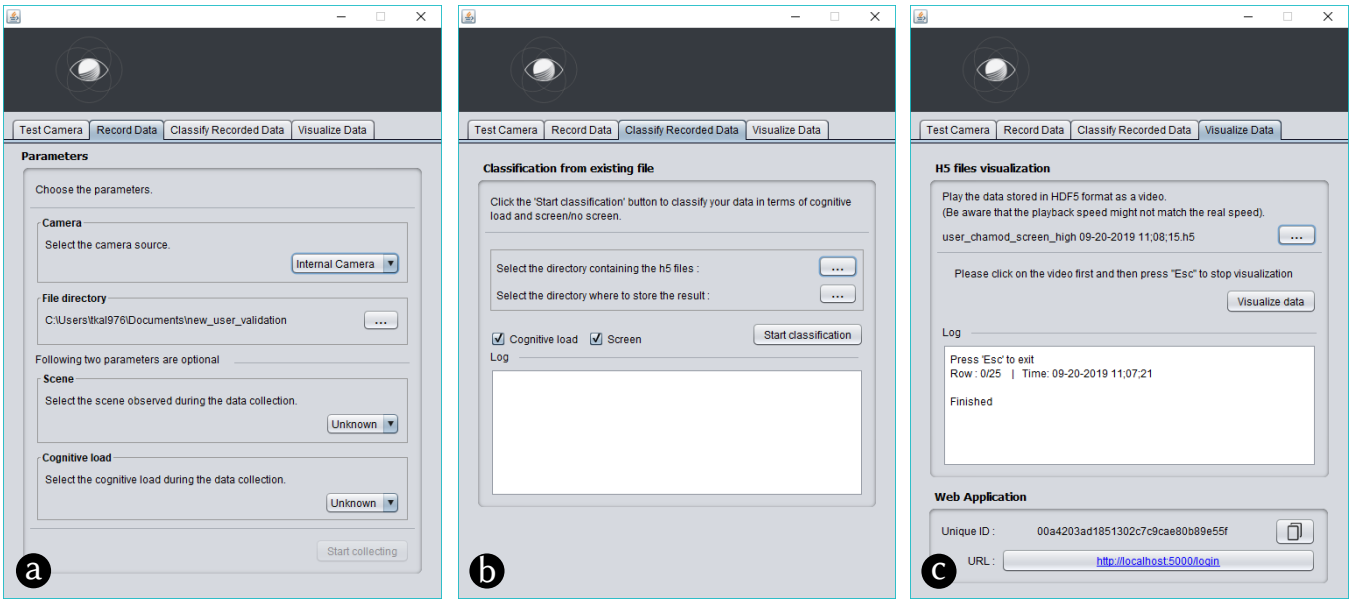

Figure 2: Showing the following tabs: a) Record Data, b) Classify Recorded Data, and c) Visualize Data.

cognitive load and running a NASA Task Load Index. The participants were given an oral explanation on our toolkit's functionality and the upcoming task. We also printed out the walk-through guide, which we read aloud to the participant after our oral explanation. The task consisted of six parts: (1) Installation of the software toolkit at the given workstation, (2) Plugging and testing the DIY head-mounted device, (3) Disabling the camera's autofocus following the video guide, (4) Recording data from the given user under two conditions (a: playing a 1-min youtube video, b: making the user read the first page of a printed research paper), (5) Visualizing the collected raw data, (6) Classifying the data, and (7) visualizing the classification results. Towards the end of the study, we asked the participant to fill out a Standard Usability Scale Questionnaire (SUS) [2] and to discuss his experience. The SUS is a well-established method in HCI calculating a usability score by deploying 10 questions that are rated on a 5pnt Likert scale. The entire study took about 30 minutes.

3.3.1 Results. The overall average score across 8 participants is 87.81% (*SD*=8.96) which is a 31.81 points improvement of what we saw during the initial pilot. This score is well above the industry standard 'A grade' threshold, which is 80.3%. All the participants successfully completed the tasks. P4 stated *"I would even love to have automated neural network training inside the application"*, going towards towards our future vision of the application. P7 gave us suggestions to further improve the visualization of the recorded data by providing the image information. This feedback was useful to constantly develop our software toolkit further to enable greater usability.

4 DEVELOPMENT AND EVALUATION

This section provides details on how the back-end of the toolkit is developed. The process consisted of several stages; Data collection for the dataset, model development, and model evaluation.

We collected data using the device from a set of 17 users (12 male and 5 female) aged between 18 to 38 years (*M*= 26.9). To create a robust model, we did not control characteristics given the many different eye-colours and shades, different eye-sizes, and the 12 different ethnicities (German, Sri Lankan, Chinese, Indian, Australian, French, Spanish, Colombian, Singaporean, Iranian, Malaysian, and Indonesian). The data collection was conducted in various lighting conditions. Each participant was asked to carry out 4 tasks, with each task lasting about 4 minutes. Between each task, the participant was asked to remove the device and then re-wear it after a small rest to avoid overfitting to the position. We collected EDA data from the participant during the entire session. After each task, participants self-reported cognitive load via printed NASA-TLX [12] forms. We used *n*-back tasks to induce cognitive load. *n*-back secondary tasks are shown to induce a high cognitive load [10].

4.0.1 Tasks. Working in front of screens also involves documents. Therefore, we selected four tasks involving screens and documents and one task from each type was loaded with a secondary number based 1,2-back task to emulate high cognitive loads. Following are the labelled tasks the user carried out:

- Writing down numbers from 1 to 50 on a sheet of paper - *No screen, low cognitive load*
- Completing 3 mazes within 4 minutes while carrying out a secondary 1-back task - *No screen, high cognitive load*
- Watching a relaxation video of the participant's choice - *screen, low cognitive load*
- Watching a relaxation video while carrying out a secondary 2-back task - *screen, high cognitive load*

We considered the tasks without n-back tasks as low cognitive load and tasks with n-back tasks as high cognitive load. However, to have secondary measures of this, we used the self reporting from NASA-TLX (mental demand isolated; 50% or more was considered as high cognitive load) and EDA data. We only included the data which had all the three measures (n-back presence, self-reported

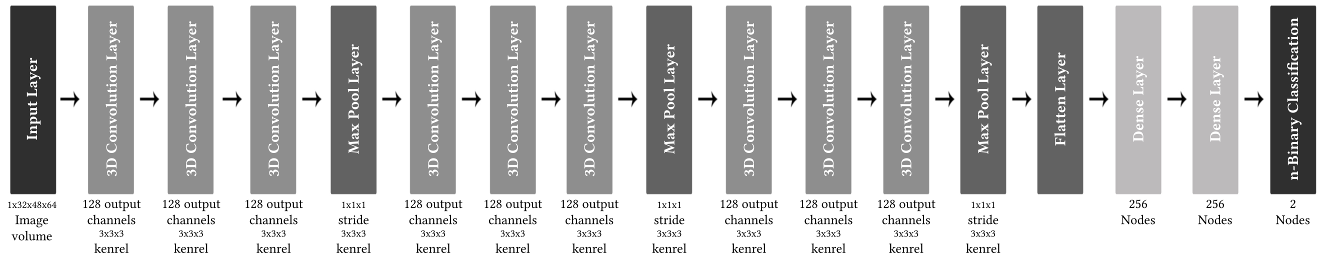

Figure 3: In our 3D CNN architecture a NN extracts features from video input as a set of frames at consecutive time samples. Our network has 1152 3D convolution kernels that go through spatial and time axes to extract features.

mental demand and number of SCR peaks in EDA) reporting the same cognitive level label. We stored all the files in Hierarchical Data Format 5 [6] (HDF5) for ease of management. The entire dataset is available from our website (see section 3.2.1) for research purposes and all source codes are published to read, write, modify, and concatenate HDF5 data.

We used a custom 3DCNN [13] as our neural network (see figure 3). During our first few testing trials, we found that cutting off data related to the first and last 60 seconds of each task from the dataset, improved the model's accuracy. A possible reason for such an improvement is that the user requires adaption to the task before a fairly steady cognitive level is reached, and upon reaching task completion, efforts are reduced. In total we had 3128 instances to train our model and 800 instances to test our model. We evaluated two different methods for classification. The first approach was to use two independent models to detect screens and determine cognitive load, while the other involved using a single network to classify both. By using a subset of the same participants performing a mix of same and different tasks in different sessions, we achieved a 70% and 72% accuracy respectively for screen and cognitive load classifying two classes. The single network did show consistent accuracy and scored 50% classifying four classes similar to a cascaded dual class model ($70\% \times 72\% = 50.4\%$), where the random accuracy is 25%. We observed that false-positives and false-negatives to be of similar amount. Based on these results we derive several findings along with methods to improve the accuracy. One consequence is having more data, which is likely to increase model robustness. We envision that other researchers are going to contribute with their recorded datasets to further elevate the accuracy.

5 CHALLENGES, LIMITATIONS, & OPPORTUNITIES

Building a generalized model working across users stemming from different ethnic backgrounds seems to be a great challenge. Previous work that employ a similar technique to detect objects on the corneal surface reflection [11] have selected homogeneous eye types. In contrast, we did not have this selection criteria. From the interviews and follow up studies, it is clear that EyeKnowYou could be a very useful tool. The main limitation is the accuracy of our model. We acknowledge that a greater number of users is likely to increase the model's accuracy. Since our data set along with the source codes to record and modify the data set is opensource, collaboratively increasing the size of the data set is an option to further improve the accuracy [27] and robustness of the underlying model. To broaden the applicability for other areas, identifying other objects than screens (e.g., people, activities), may also be particularly important for research purposes. In order to do so, our work is opensource, enabling advanced users to modify a few layers of the Neural Network and start with our pre-trained weights classifying a greater variety of classes.

6 CONCLUSION & FUTURE WORK

With this paper, we contributed a new concept to sense increased cognitive load simultaneously with actual screen time. We presented a straightforward DIY software toolkit for human factors researchers. This toolkit can be used to collect labeled data and process them using our neural network. Also, advanced users can access all the source codes and our dataset to further develop the tool and model, such as to apply it in different contexts. They can also use new models to be used with our software toolkit.

For future work, we envision our toolkit to grow with a large amount of raw data generated by the community. This will help us to build a more robust and generalizable model contributing to a reasonable accuracy. To further boost accuracy, we suggest training separate models for specific ethnic groups or recording a large volume of users, such as 92, which Fridman et al. [10] demonstrated.

REFERENCES

[1] Michael Backes, Tongbo Chen, Markus Duermuth, Hendrik PA Lensch, and Martin Welk. 2009. Tempest in a teapot: Compromising reflections revisited. In *2009 30th IEEE Symposium on Security and Privacy*. IEEE, 315–327.
[2] John Brooke et al. 1996. SUS-A quick and dirty usability scale. *Usability evaluation in industry* 189, 194 (1996), 4–7.
[3] W Keith Campbell. 2018. Associations between screen time and lower psychological well-being among children and adolescents: Evidence from a population-based study. *Prev Med Rep.* 12 (2018), 271–283.
[4] Karen Church, Denzil Ferreira, Nikola Banovic, and Kent Lyons. 2015. Understanding the challenges of mobile phone usage data. In *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services*. ACM, 504–514.
[5] Melissa Patricia Coral. 2016. "Analyzing Cognitive Workload through Eye-Related Measurements: A Meta-Analysis. *Browse all Theses and Dissertations* 1507 (2016).
[6] Kate Crawford, Jessa Lingel, and Tero Karppi. 2015. Our metrics, ourselves: A hundred years of self-tracking from the weight scale to the wrist wearable device. *European Journal of Cultural Studies* 18, 4-5 (2015), 479–496.
[7] Trinh Minh Tri Do, Jan Blom, and Daniel Gatica-Perez. 2011. Smartphone usage in the wild: a large-scale analysis of applications and context. In *Proceedings of the 13th international conference on multimodal interfaces*. ACM, 353–360.
[8] Andrew T. Duchowski, Krzysztof Krejtz, Izabela Krejtz, Cezary Biele, Anna Niedzielska, Peter Kiefer, Martin Raubal, and Ioannis Giannopoulos. 2018. The Index of Pupillary Activity: Measuring Cognitive Load Vis-à-vis Task Difficulty with Pupil Oscillation. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. ACM, New York, NY, USA, Article 282, 13 pages. https://doi.org/10.1145/3173574.3173856

[6]https://www.hdfgroup.org/solutions/hdf5/

[9] Maria K. Eckstein, BelÃ©n Guerra-Carrillo, Alison T. Miller Singley, and Silvia A. Bunge. 2017. Beyond eye gaze: What else can eyetracking reveal about cognition and cognitive development? *Developmental Cognitive Neuroscience* 25 (2017), 69 – 91. https://doi.org/10.1016/j.dcn.2016.11.001 Sensitive periods across development.

[10] Lex Fridman, Bryan Reimer, Bruce Mehler, and William T. Freeman. 2018. Cognitive Load Estimation in the Wild. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. ACM, New York, NY, USA, Article 652, 9 pages. https://doi.org/10.1145/3173574.3174226

[11] L. E. Hafi, M. Ding, J. Takamatsu, and T. Ogasawara. 2017. Gaze Tracking and Object Recognition from Eye Images. In *2017 First IEEE International Conference on Robotic Computing (IRC)*. 310–315. https://doi.org/10.1109/IRC.2017.44

[12] Sandra G Hart and Lowell E Staveland. 1988. Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research. In *Advances in psychology*. Vol. 52. Elsevier, 139–183.

[13] S. Ji, W. Xu, M. Yang, and K. Yu. 2013. 3D Convolutional Neural Networks for Human Action Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 1 (Jan 2013), 221–231. https://doi.org/10.1109/TPAMI.2012.59

[14] Tharindu Kaluarachchi, Andrew Reis, and Suranga Nanayakkara. 2021. A Review of Recent Deep Learning Approaches in Human-Centered Machine Learning. *Sensors* 21, 7 (2021), 2514.

[15] Amy K Karlson, Brian R Meyers, Andy Jacobs, Paul Johns, and Shaun K Kane. 2009. Working overtime: Patterns of smartphone and PC usage in the day of an information worker. In *International Conference on Pervasive Computing*. Springer, 398–405.

[16] Christian Lander and Antonio Krüger. 2018. EyeSense: Towards information extraction on corneal images. In *Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers*. 980–987.

[17] Christian Lander, Antonio Krüger, and Markus Löchtefeld. 2016. " The story of life is quicker than the blink of an eye" using corneal imaging for life logging. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct*. 1686–1695.

[18] Laura Lascau, Priscilla NY Wong, Duncan P Brumby, and Anna L Cox. 2019. Why Are Cross-Device Interactions Important When It Comes To Digital Wellbeing? (2019).

[19] Atsushi Nakazawa, Christian Nitschke, and Toyoaki Nishida. 2016. Registration of eye reflection and scene images using an aspherical eye model. *JOSA A* 33, 11 (2016), 2264–2276.

[20] Ko Nishino and Shree K Nayar. 2006. Corneal imaging system: Environment from eyes. *International Journal of Computer Vision* 70, 1 (2006), 23–40.

[21] Christian Nitschke, Atsushi Nakazawa, and Haruo Takemura. 2013. Corneal imaging revisited: An overview of corneal reflection analysis and applications. *IPSJ Transactions on Computer Vision and Applications* 5 (2013), 1–18.

[22] Nargess Nourbakhsh, Yang Wang, and Fang Chen. 2013. GSR and Blink Features for Cognitive Load Classification. In *Human-Computer Interaction – INTERACT 2013*, Paula Kotzé, Gary Marsden, Gitte Lindgaard, Janet Wesson, and Marco Winckler (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 159–166.

[23] Vsevolod Peysakhovich, Frédéric Dehais, and Mickaël Causse. 2015. Pupil diameter as a measure of cognitive load during auditory-visual interference in a simple piloting task. *Procedia Manufacturing* 3, 5199–5205. https://doi.org/10.1016/j.promfg.2015.07.583

[24] Bastian Pfleging, Drea K. Fekety, Albrecht Schmidt, and Andrew L. Kun. 2016. A Model Relating Pupil Diameter to Mental Workload and Lighting Conditions. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (San Jose, California, USA) *(CHI '16)*. ACM, New York, NY, USA, 5776–5788. https://doi.org/10.1145/2858036.2858117

[25] Karen Pugliesi. 1999. The consequences of emotional labor: Effects on work stress, job satisfaction, and well-being. *Motivation and emotion* 23, 2 (1999), 125–154.

[26] Jill W Rettberg. 2014. *Seeing ourselves through technology: How we use selfies, blogs and wearable devices to see and shape ourselves*. Springer.

[27] David Rolnick, Andreas Veit, Serge Belongie, and Nir Shavit. 2017. Deep Learning is Robust to Massive Label Noise. (05 2017).

[28] Aric Sigman. 2017. Screen Dependency Disorders: a new challenge for child neurology. *Journal of the Int. Child Neurology Association* (2017).

[29] Peter Sturm, Srikumar Ramalingam, Jean-Philippe Tardif, Simone Gasparini, Joao Barreto, et al. 2011. Camera models and fundamental concepts used in geometric computer vision. *Foundations and Trends® in Computer Graphics and Vision* 6, 1–2 (2011), 1–183.

[30] Shiyan Yang, Jonny Kuo, and Michael G Lenné. 2018. Analysis of gaze behavior to measure cognitive distraction in real-world driving. 62, 1 (2018), 1944–1948. https://doi.org/10.1177/1541931218621441 arXiv:https://doi.org/10.1177/1541931218621441

[31] Roberto Yus, Primal Pappachan, Prajit Das, Eduardo Mena, Anupam Joshi, and Tim Finin. 2014. FaceBlock: Privacy-Aware Pictures for Google Glass. https://doi.org/10.1145/2594368.2601473

[32] Johannes Zagermann, Ulrike Pfeil, and Harald Reiterer. 2016. Measuring Cognitive Load Using Eye Tracking Technology in Visual Computing. In *Proceedings of the Sixth Workshop on Beyond Time and Errors on Novel Evaluation Methods for Visualization* (Baltimore, MD, USA) *(BELIV '16)*. ACM, New York, NY, USA, 78–85. https://doi.org/10.1145/2993901.2993908

A APPENDIX

A.1 Step 1: Hardware Preparation

For the camera, we used the Microsoft HD6000 life camera[7]. As illustrated in Figure 4, we removed the plastic enclosure and separated both PCBs. Keeping them plugged in together is also possible.

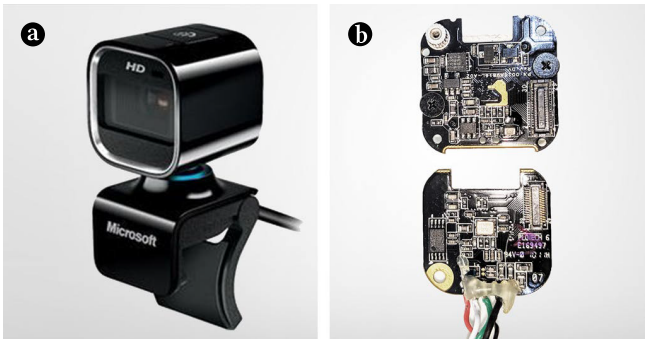

Figure 4: Before (a) and after (b) removing the enclosure of the camera. Both PCBs have been detached from each other.

Since we only utilize visual information, we can remove or de-solder the microphone to reduce visual obstruction, as shown in Figure 5. However, omitting this step does not affect the functionality.

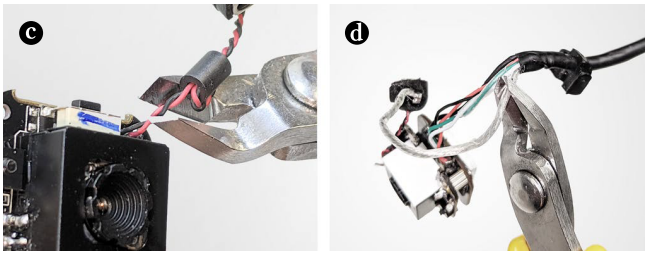

Figure 5: Cutting off the microphone at both ends; at the PCB (c) and before it enters the cable duct (d).

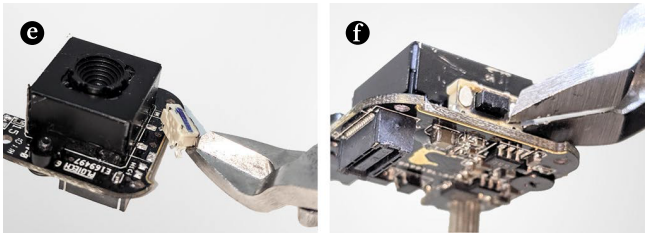

Figure 6: Mechanically pulling off the push button – captured from top (e) and side (f) view points.

We also carefully removed the white push button resting next to the black focusing lens box (*see Figure 6*). Again, de-soldering is an option. Removal makes it easier to fit the camera into the mounting frame, although not mandatory for operation.

Next, we removed both blue indicator LEDs, given its creation of unwanted reflections on the eye ball, distracting the user (*see Figure 7*). We de-soldered them, however, covering them with opaque tape/glue or mechanically destroying them is also possible.

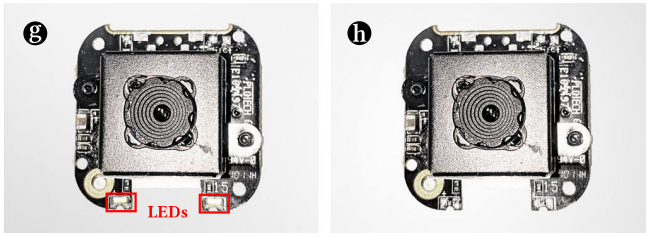

Figure 7: De-soldering indicator LEDs; before (g), after (h).

After careful execution of all the previous steps, we plugged both PCBs back together as shown in Figure 8. A screw tightens both PCBs.

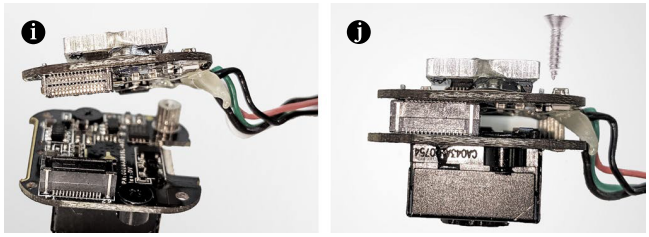

Figure 8: Before (i) and after (j) plugging the two PCBs

We created a video[8] showing how to carry out step 1 a-j.

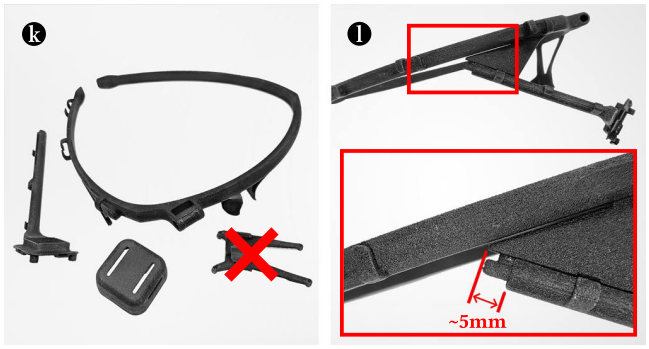

Figure 9: 3d printed parts (k) of the head-mounted frame. The socket to mount the real world camera is not required. The other camera socket is to be attached to the sliding rod, which goes into the head mount (l). We slide the rod all the way in until a gap of ~5mm.

[7] https://amzn.to/2Hpqqrt

[8] https://vimeo.com/361288458

A.2 Step 2: Final Hardware Assembly

We used the eye tracker frame Pupil Labs designed as the head mounting frame, which can be purchased at the Shapeways web store[9]. The 3d printed frame comes in four parts as depicted in Figure 9. Mounting the real world camera is not needed as crossed out in Figure 9–k. Fixing the camera socket with the sliding rod just about 5 mm before the end point (*see Figure 9–l*) provides optimal results for capturing the entire eye of people with different physiological characteristics (ethnicities, eye-sizes, face sizes).

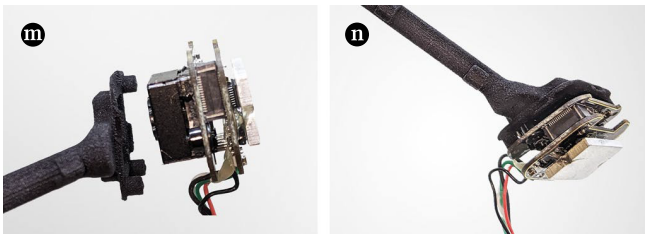

Figure 10: A moment before (m) and after (n) fitting the camera in to the camera socket.

Figure 10 shows how the camera should be fitted upright into the socket. The colored wires from the PCBs pointing towards the bottom indicate the corrected position.

Then, as shown in the Figure 11–o, the camera can be enclosed by the lid. Thereafter, the USB cable should be pushed to the hooks following the frame path (*see Figure 11–p*). A similar DIY guide

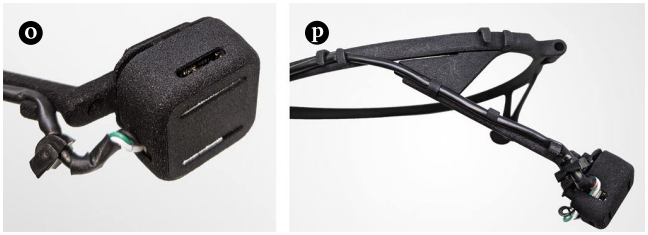

Figure 11: Closing the lid (o) and attaching the cable to the frame (p).

for the eye-tracker is also available on the Pupil Labs' website[10]. However, since we do not use IT, this setup is simpler and we provide more details, making it easier for non-experts.

[9]https://www.shapeways.com/product/LQJJK2CHQ
[10]https://docs.pupil-labs.com/#diy