Democratizing eye tracking

Introducing Web Gazer: an eye tracking library that uses common webcams to infer the eye-gaze locations in real time.

(2017) G. Federico, R. Santonicola, C. Santoriello

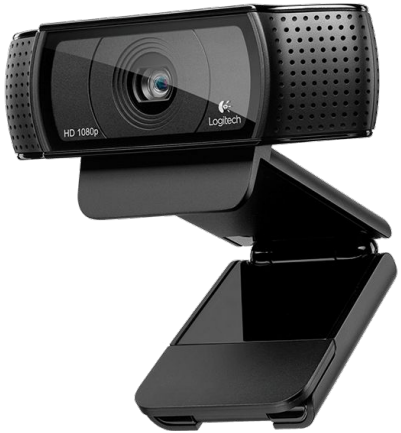
University of Campania “Luigi Vanvitelli”
Department of Psychology
www.psicologia.unina2.it

Eye Tracking: generalities

- Eye tracking technologies are conventionally used into several domains of knowledge (such as **experimental psychology**, **human-computer interaction** studies, **medical research**, etc.) to investigate the **eye-gaze** and **ocular behaviour**.
- A lot of interesting information can be "extracted" in this way. Just to name a few: **human attention**, **clinical patterns** and **interfaces usability** (in the HCI domain).
- A standard high-level Eye Tracker generally uses an **infrared camera** placed at a fixed distance from the subject and **requires an explicit calibration and setup**.
    - These kind of operations have to be performed within an **artificial context** (laboratory).
    - The use of a laboratory often imply **limitations in naturalistic studies**.
- Professional infrared eye trackers and related softwares cost **thousands of euros**.
- The start-up that produced the only **low-cost** ($99) research-grade (it means: SDK availability for research & development) infrared eye tracker in the market (The *EyeTribe* Eye Tracker) was recently acquired by Oculus. No more low-cost infrared eye tracking for now!

Eye Tracking using webcams

- A **webcam** is a video camera that streams its image in real time to a computer (traditionally).
- When the stream is “catched” by the computer it can be saved, viewed or managed/elaborated by softwares that operates **offline** (after stream acquisition), **online** (in real time) or in **both ways**.
- Every modern laptop, tablet, smartphone in the market includes an high-resolution webcam.
- If not, it can be purchased at **very low prices**.



So, an obvious question…

Is it possible to use a standard webcam as an eye tracking system?

Eye Tracking using webcams (cont.)

- Technological advancements have recently arrived (such as very high-resolution sensors for webcams and very powerful CPUs and GPUs).
- **Modern computers are enough fast to do eye tracking**. **But...**
- **Eye tracking using webcams** has been extensively studied in scientific literature with different approaches, mathematical models and distinct informatics algorithms. [Bomerleau and Baluja, 1993; Hansen & Pece, 2005; Sugano et al., 2010; Lu et al., 2011 and 2012; Alnajar et al., 2013]
- Considering the results of all these studies it can be unsurprisingly found that **all the approaches that uses standard webcams with "offline" software are less accurate than infrared eye trackers**.
- Common problems with these systems are related to the **high sensitivity to head movements**, to the **extensive calibration** required and to the **poor accuracy** due to diverse settings and human features.
- **Therefore: provide technologies that allows to reduce these limitations would seem to be the preferred way to make eye tracking with webcams a feasible option.**

Eye Tracking using webcams (cont.)

- In order to **reduce the impact of the aforementioned problems** several strategies that involves the use of very high-class algorithms have been developed. [Xu et al., 2015; Huang et al., 2016; Papoutsaki et al., 2016]
- Using advanced **face recognition models** it's now possible to acquire a stable visual pattern of the head not susceptible to small variations related to the movements. [Mathias, 2014; Tschirsich, 2012; Lundgren et al., 2014]
- When face detection is done, it's consequently possible to identify the eye regions; then, the **location of the pupil** can be analyzed making three assumptions: **(1)** the iris is darker than its surrounding area; **(2)** it is circular; **(3)** it's generally located at the center.
- In terms of informatics procedures, **pupil can be searched over all offsets and scales for the region with the highest contrast to its surrounding area** and some other advanced techniques.
- Extracted the position of the pupil (*contrast differences*), an appropriated **ridge regression model** [Hoerl and Kennard, 1970] can be used to **infer the eye-gaze location** of the user **matching eye pixels to gaze locations**. Technical details will be discussed later.

Eye Tracking using webcams (cont.)

- A **browser** is a software for retrieving and **presenting** information resources (generally, using the World Wide Web or Local Networks).
- Over **65% of web browsers** today support the functions for **accessing the webcam** of the users (*e.g.,* Google Chrome and Mozilla Firefox). [Deveria, 2015]
- Combining the potentialities of modern browsers and the possibility to access to the users webcams it would be possible - at the same time - to make **remote online eye tracking studies** (it means: very large population) and **in-lab studies** (smaller population but more experimental control) within a low cost scenario (easily accessible).
- Adopting **user interactions** (*e.g.,* user clicks or mouse movements) as a continuously self-calibrate reference through sophisticated mathematical models, problems related to poor accuracy can be partially fixed. [Papoutsaki et al., 2016]
- Using specified programming languages (such as PHP, Javascript, HTML5, etc.) **web applications that allows eye tracking studies** can be easily developed.

To sum up...

- Eye tracking using infrared trackers is **very expensive** and requires **specific settings (lab)**.
- At the state of the art webcams can be used for eye tracking applications where the **approximate location** of the gaze is sufficient (it means: no medical research for now).
- **Web 2.0** (and technologies present in recent browsers) permits to manage eye tracking paradigms with simplicity and scalability.
- Developing **cost-zero Web Applications** can be a way to make psychological research (especially if the main interest is related to *fixation points*) with a very large population.
- Analyzing the **user interaction** (click & mouse tracking) could be the key to solve problems related to the relative poor accuracy of these systems.
- In the next future, **machine learning** and **artificial intelligence** could become leading actors in the world of low-cost eye tracking.
- Further researches and mathematicals models are needed to improve algorithms actually used in this domain. But we are on the right way.

Introducing WebGazer

- **WebGazer** is a **javascript library** that implements a totally new approach for **scalable and self-calibrated eye tracking using common web browsers and webcams**.
- It was **presented in 2016 by Alexandra Papoutsaki and colleagues** from Brown University (USA) to the *25th International Joint Conference on Artificial Intelligence* (IJCAI-16).
- It aims to democratize eye tracking by making it accessible to the masses.
- To overcome the accuracy problems that webcams typically have, WebGazer adopts **user interactions** (*clicks*) to continuously self-calibrate the prediction model.
- Using **3 open-source face tracking libraries** (clmtrackr, js-objectdetect, and tracking.js) and a strong **regression model**, it permits to make eye tracking studies with a good accuracy.
- It's easily implementable in any kind of Web Application, since it's a Javascript module.
- Using this approach it's possible to provide a **natural experience to everyday users** that is **not restricted to laboratories** and highly controlled user studies.

Introducing WebGazer (cont.)

- **Real time gaze prediction** on most major browsers.
- No special hardware - it uses **common webcams**.
- **Self-calibration from clicks and cursor movements**.
- **Easy to integrate** with JavaScript.
- Swappable components for **eye detection**.
- **Multiple gaze prediction models**.
- It has been evaluated through a remote online study (N = 76) and a lab study (N = 4).
- A comparison with a commercial eye tracking device (Tobii EyeX) has revealed a **comparable mean errors** with an average visual angle of 4.17 deg.
- **Source code and documentation are publicly and freely available online**.
- The project is released under **GNU General Public Licence 3.0** (open-source).

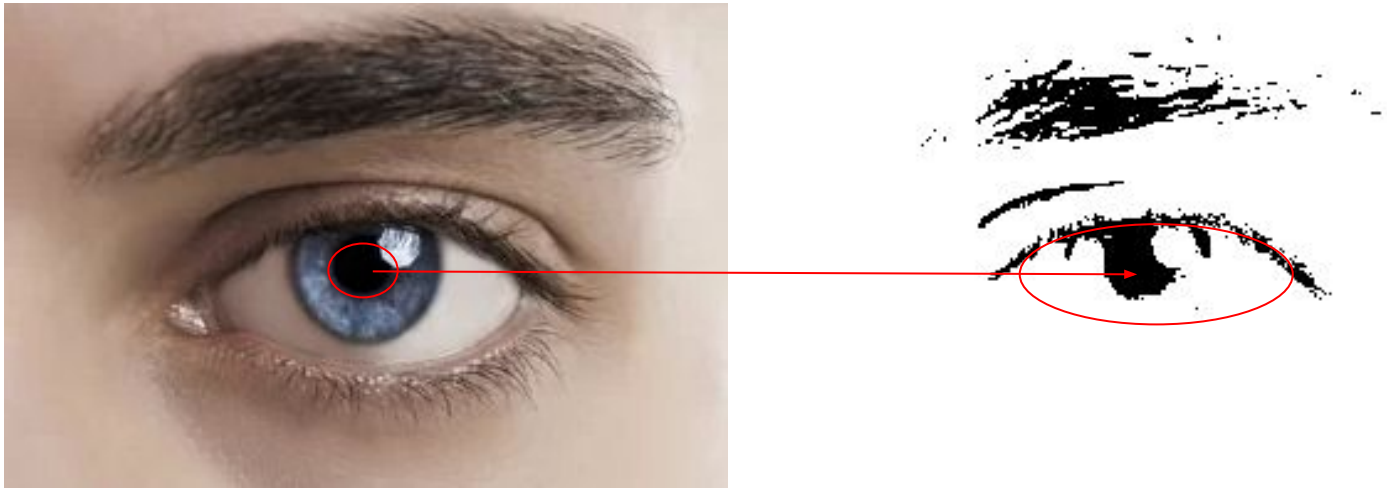
WebGazer: how it works

- **The library has two key components:**
  - **a pupil detector**, that can be combined with any **eye detection library**;
  - **a gaze estimator** using regression analysis informed by **user interactions**.
- **Questions addressed by the development of WebGazer are:**
  - How to perform pupil detection?
  - How to extract eye features?
  - How to match the pupil locations and eye features to screen coordinates?
  - How to combine these kind of data with user interaction (*clicks*)?

WebGazer: how it works (cont.)

- As you remember, the **location of the pupil** can be analyzed making three assumptions:
  - the iris is darker than its surrounding area;
  - it is circular;
  - pupil is generally located at the center.
- But this **2D representation** is not enough to catch the richness of the eye's appearance.
- WebGazer *learns* (machine learning) how to map from *pixels* to a *gaze location* **representing each eye as a 6 x 10 image patch** and using a **grayscaling and histogram equalization**, resulting in a **120D eye feature vector**.
- For the self-calibration, WebGazer assumes that when a user interaction takes place then **the gaze locations on the screen match the coordinates of that interaction**.
- So it assumes that **the gaze and mouse cursor align perfectly during clicks**.

WebGazer: how it works (cont.)



WebGazer: how it works (cont.)

- Considering *N pupil locations* (x) and their corresponding "*clicks*" *observations* (t) on the monitor:
  - $\mathbf{x} = (x_1, \dots, x_N)$
  - $\mathbf{t} = (D_{x1}, \dots, D_{xN})$
- Using a simple linear regression model, we obtain a function $\mathbf{f(v) \rightarrow D_x}$ [1] which given a *pupil feature vector* **v** predicts the location of the gaze on the screen along the x-axis.
- The function is $\mathbf{f(v) = \phi(x)^T w}$ [2] where $\mathbf{\phi(x)}$ is a basis function and **w** is a vector of weights that satisfy the following: $\mathbf{minimize_w} ||D_{xi} - f(x_i)||_2^2$.
- To match eye pixels to gaze locations, in order to map eye features, WebGazer uses a ridge regression model [Hoerl and Kennard, 1970] that links the *120D eye feature vector* to the display coordinates (D_x, D_y) for each *click*.
- Considering the ridge regression function for the x-coordinate prediction ([1]), the function [2] again depends on a vector of weights **w** which is estimated as: $\mathbf{minimize_w} \; ||D_{xi} - f(x_i)||_2^2 + \lambda||\mathbf{w}||_2^2$
- The weight vector (**w**) in matrix notation is $\mathbf{w} = (X^T X + \lambda I)^{-1} X^T Y$ where **X** is the matrix of eye features and **Y** is the response vector of display coordinates.

WebGazer: how it works (cont.)

Online demo of the **prediction model** and its **self-calibration** through *user interaction*.

A real experiment with WebGazer

- In 2017 **Adam Cellary** (a polish programmer) starts the development of an interesting Web Application that uses WebGazer to make commercial eye tracking studies on websites and commercial advertisements (Realeye.io).
- We asked Adam some software improvements to make RealEye.io **also usable for psychological research**, giving us the possibility to:
  - use a calibration task to avoid the use of the mouse to initialize WebGazer;
  - upload images;
  - randomize stimuli;
  - extract fixation points in terms of percentages and time (*ms*, means and std. devs.);
  - set a fixation-point before stimuli appears;
  - set a “retina-relax” stimuli (blank) between stimuli;
  - have a better experimental flow management.
- **Now the application fulfills our requests** (even if it’s actually in beta).

Demo session

- So, moving from a mathematical and algorithmical dissertation to a more practical session, we will do a *demo* experiment to show the possibilities offered by WebGazer, using the just born and developed RealEye.io web application.
- We need 1 participant.

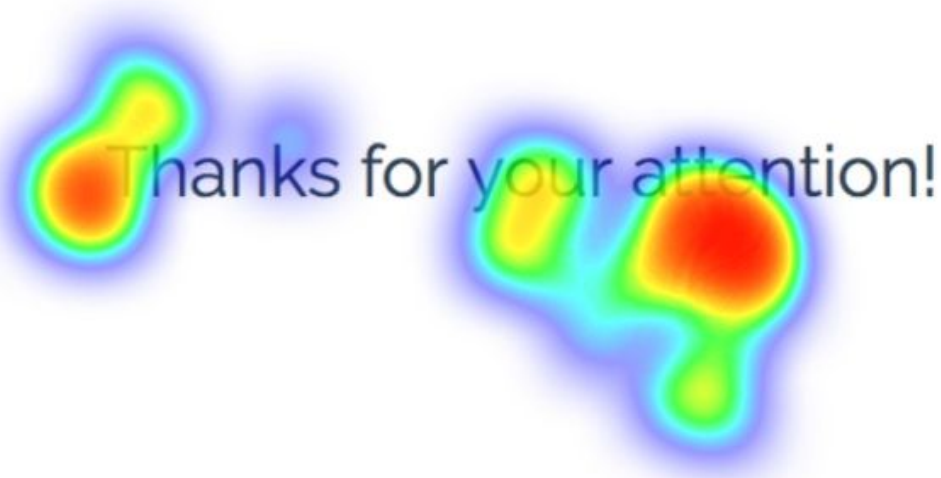
Let's start

Conclusions

- WebGazer is a self-calibrated eye tracking library that works with web browser and webcams.
- It's scalable, easy to use and well documented.
- As shown in RealEye.io implementation it can be easily programmed in any web application.
- Using 3 tracking libraries and a strong regression model, the mean error is **104 pixels**, so it's useful for applications where the approximate location of the gaze is sufficient.
- Further improvements are necessary to reduce the errors but results achieved until now are greats and comparable to some commercial eye tracker.
- Research activities often requires the use of very expensive equipment and very specific sensors. In recent years open software (and hardware) communities are working in the right direction, given a new impetus to the ability to make innovative research with limited resources. It's an "innovation boost" to the world of research and entertainment that can certainly have wide use in Cognitive Sciences.

References

- Papoutsaki, A., Daskalova, N., Sangkloy, P., Huang, J., Laskey, J., & Hays, J. (2016). WebGazer: scalable webcam eye tracking using user interactions. *International Joint Conference on Artificial Intelligence*.
- **WebGazer website**: https://webgazer.cs.brown.edu.


Thanks for your attention!