Construction of an Inexpensive Eye Tracker for Social Inclusion and Education

Otthar A.N. Krohn[1], Vako Varankian[1], Pedro G. Lind[1], and Gustavo B. Moreno e Mello[2]

[1] Department of Computer Science, OsloMet Oslo Metropolitan University,
[2] Department of Mechanical, Electronic and Chemical Engineering, OsloMet Oslo Metropolitan University,
P.O. Box 4 St. Olavs plass, N-0130 Oslo, Norway

Abstract. In this paper we described how to build an inexpensive eye-tracker and to apply it social inclusion and educational activities in schools. The building of the device includes both the construction of the eye-tracking headset as well as the implementation of the code for translating eye trajectories into output data which can afterwards be analyzed and modeled. The procedure is cheap and can be easily implemented in high-schools and first-years undergraduate courses, to teach specific matters in computer sciences, physical sciences and mathematics. Moreover, we also discuss up to which extension such a cheap device can substitute commercial solutions for promoting social inclusion, particularly to develop empathy in communities by showing the difficulties behind eye-movement languages used by non-verbal paralyzed individuals.

Keywords: eye tracker · augmented communication · accessibility · social inclusion · education.

1 Introduction

Presently, eye-trackers are produced by different companies with particular purposes in the context of social equality. In particular, some commercial solutions allow paralyzed people, who are unable to use their hands, to use their eyes to control a computer through specialized human-computer interfaces (HCI). This kind of HCI maps eye-tracking records, as illustrated in Figure. 1, into computer commands according to an established eye-movement-based code. These eye-triggered commands can be used by non-verbal individuals with associated paralysis to control specialized software that enables them to type texts and convert it to speech. Thus, eye-tracking not only enables accessibility, but also augmented communication (AC) technologies, which might be the only possible mean of communication for some.

There are however two main drawbacks in AC application. First, although the technology enables communication and improvements have been done [1], it is yet far from supporting a comfortable and easy conversation [2–4]. The word

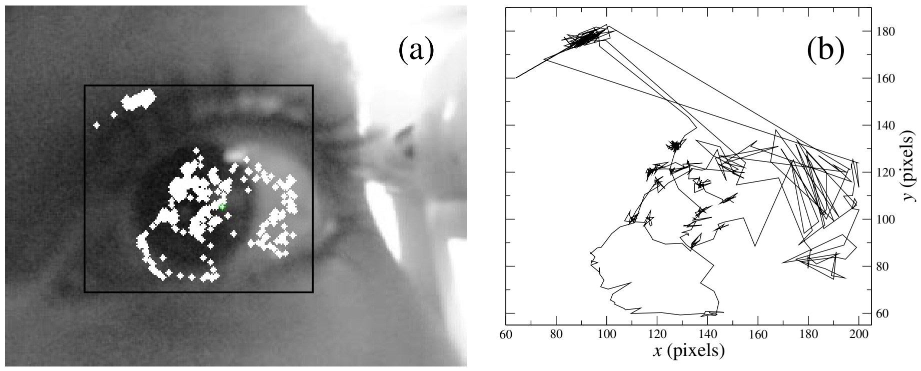


Fig. 1: **(a)** Illustration of one frame of eye-motion video (photo) superposed with the full set of locations of the pupil center during the same video recording, determined with our software. The region in the square is zoomed in in **(b)** showing the only the full eye trajectory. A full description of the construction of such an eye-tracker as well as the algorithm for locating and keeping track of the pupil are presented in Section 2.

selection process tends to be slow and prone to frequent mistakes. The speech synthesizer voice is often robotic and lacks emotional context, which makes substantial part of verbal communication. Finally, learning how to communicate through such a HCI is a challenging endeavour to the user, usually requiring special assistance and supervised training. These factors tend to decrease the chance for integration of non-verbal AC technology users, since the speaking community is typically not aware of the difficulties inherent in these limitations of the technology.

Second, the commercial eye trackers are typically expensive. While they can be used for many purposes, they are not affordable in several important situations, namely in high-school classes. Professors or teachers may want to use eye-trackers in their lectures as a mean to promote the engagement of the students. Nonetheless, they are dissuaded by the cost because, usually, the education budget cannot cover such investments in equipment. By providing a cheaper eye-tracker, one is able to open a new panoply of different tools and pedagogical approaches for teaching specific subject matters.

In this paper, we describe in detail how to build such an eye-tracker and explain how it can be used in classes of different participants, namely computer sciences, physics, mathematics. The total cost is approximately 100 NOK, i.e. 10 Euros, which corresponds to $\sim 0.1\%$ of the average price of a commercial device nowadays. As it will become clear, the eye-tracker here presented is easy and cheap to build. For comparison, one should be aware of some of the most typical commercial solutions namely PCEye Plus which costs \$1700 approximately[3], Gazepoint GP3 HD[4] for \$2000, or even Tobii Pro X3-120[5] for more than \$16000. Although inexpensive the eye-tracker we build operates at high sampling frequency, being able to record up to 180 frames per second.

Moreover, we also discuss its use in teaching activities at high-schools and describe a possible framework for developing social awareness and positive attitudes towards non-verbal or paralyzed people. In particular, we propose an experiment with university students to promote awareness about the communication issues faced by people that depend on eye-tracking solutions to communicate.

We start in section 2 describing how to construct the hardware and implement the necessary software. In section 3 we explain how such device can be used in high-school lectures and during the first years of university courses in natural sciences, information technologies and engineering. In section 4 the applicability of such cheap solution is discussed beyond the scope of the education sector, aiming at substituting commercial solutions for communication purposes, establishing an experimental design for promoting social empathy in a community of students. Section 5 concludes the paper.

2 How to build a cheap head-mounted eye-tracker?

In this section we describe how to build the hardware, how to implement the software, and describe its graphical user interface (GUI). We also communicate how the solution at hand is tested and validated, comparing its recorded eye trajectories with those obtained with a standard software, namely *EyeRecToo* [5]. Figure 2 shows the eye-tracker mounted on a subject at OsloMet.

2.1 Hardware

The hardware of the head-mounted eye-tracker is illustrated in Figure 3(a) and includes:

- One camera (figure 3(a1)) such as PlayStation 3-eye, which can film in up to a resolution of 640x480 pixels with a sampling frequency of 60 Hz, up to a resolution of 320x240 pixels with 120 Hz.
- Accessory material for the headset:
  - Frames (figure 3(a2)) which form the skeleton of the headset.
  - A supporting arm (figure 3(a3)) for connecting the camera to the frames. This arm is composed by three parts, namely a camera holder, an L-shape arm and an adjuster.

[3] According to `https://www.tobiidynavox.com/en-us/devices/eye-gaze-devices/pceye-plus-access-windows-control/`.

[4] According to `https://www.gazept.com/product/gp3hd/`.

[5] Recently purchased by Faculty of Technology, Arts and Design of the Oslo Metropolitan University. Informations available at `https://www.tobiipro.com/product-listing/tobii-pro-x3-120/`.

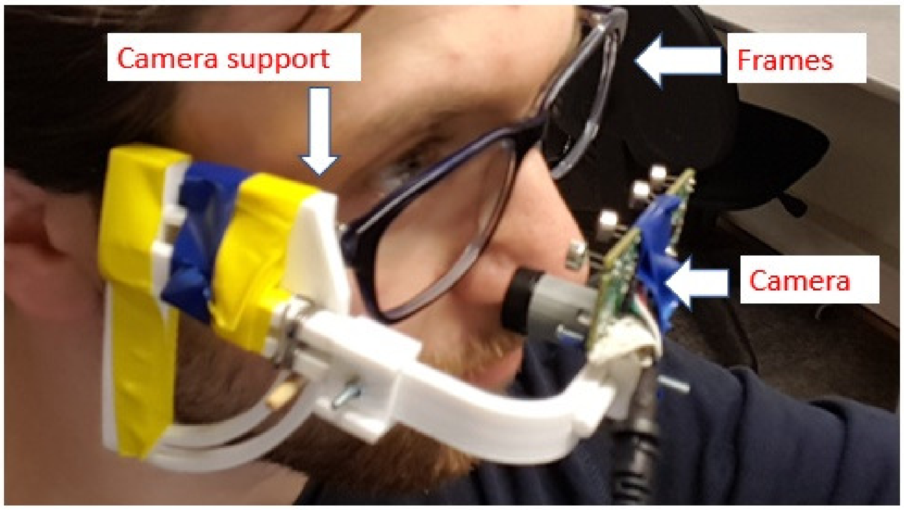


Fig. 2: Photo of the eye-tracker built at the AI Lab (OsloMet, Norway), showing its three main components, namely frames, a supporting arm and a camera. The cost of the camera is of the order of $10 and the cost of the other two components is negligible.

– One PC station, with standard specifications such as Acer Aspire V3-572G-50K9 15.6" 2015, Processor i5-5200U 2.2 GHz, Memory/RAM 8 GB 1600 MHz, a storage capacity of 1 TB or similar and a graphics card NVIDIA GT840M. We also install the PS3 eye drivers for *Windows*.

There are three important remarks regarding the choice or the printing of the components of the supporting arm. First, due to reasons that will become clear in the subsection about the software, it is important to avoid parts with dark coloring. Black frames and black components for the supporting arm interfere with the image, making it difficult to perfectly focus on the eye. Second, the focus length of the PlayStation 3 eye camera is exceptionally wide and do not provide good focus at short distance. This implicates that at the focus distance, the image capture areas of the face surrounding the eyes, which are irrelevant for eye-tracking purposes. To overcome this limitation we replaced the lens and used a custom 3D printed lens holder. This enabled to correct the focal length and the field of view, thus sharpening the image and maximizing area in the image occupied by the eye.

Third, we keep the number of different parts of the supporting arm to a minimum to simplify its assembly and minimize its cost. Apart the PC, commonly available in schools and institutions in general, the total cost of the camera is around $10 and the cost of the different parts of the supporting arm is negligible. The camera is mounted on the frame of normal glasses. One can recycle the frame of old glasses or shades, being enough to remove the lenses. Alternatively, cheap plastic frames can be found in any toy store or extracted from

from old glasses. As for the different parts of the supporting arm, they should be printed in a 3D printer, but the supporting arm can be replaced by a 5mm flexible copper wire, which is strong enough to hold the camera in place, but malleable to position adjustments. In fact, it is possible to even reduce the three components to the L-shaped arm solely, which is the fundamental component connecting camera and frames.

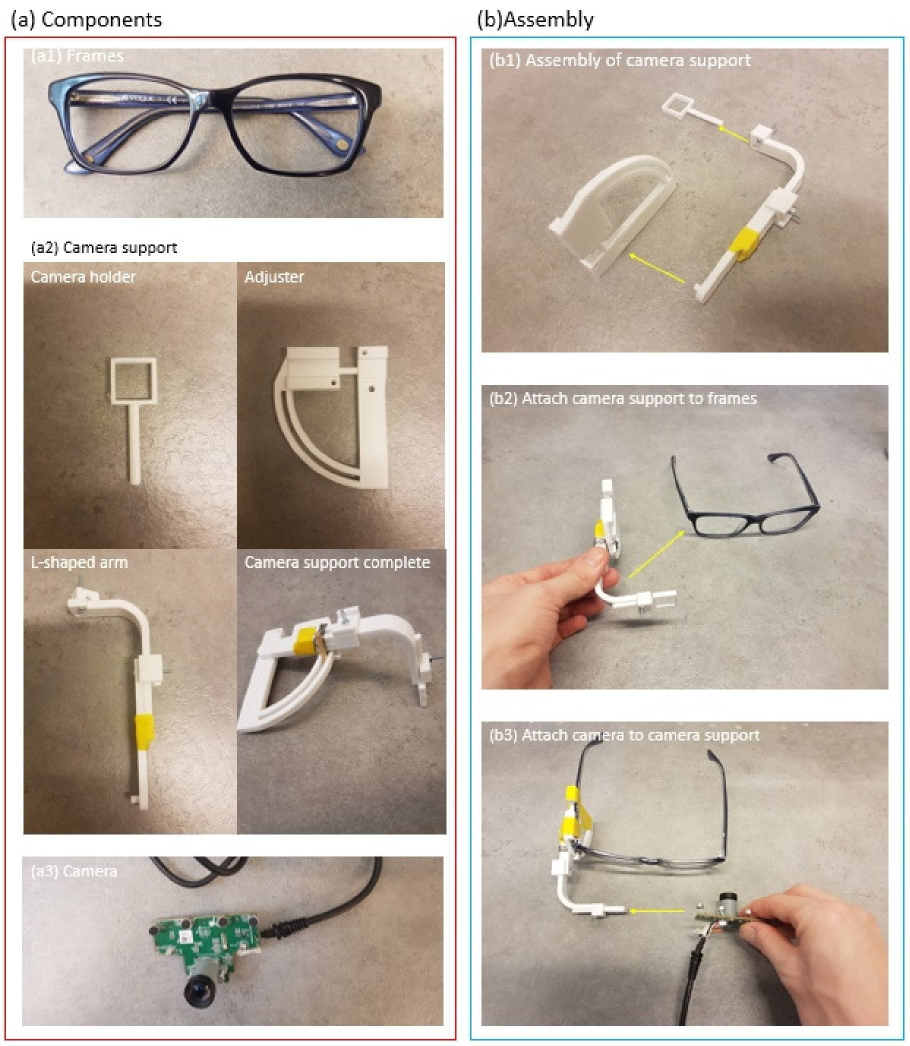


Fig. 3: **(a)** Illustration of the three components of the eye-tracker: (a1) frames, (a2) the supporting arm, including camera holder, adjuster and an L-shaped arm, and (a3) the camera. **(b)** Main steps for mounting the eye-tracker: (b1) assembly the supporting arm, (b2) attach the arm to the frames, and (b3) attach the camera to the arm.

The full mounting procedure is sketched in Figure 3(b). One first assembles the three parts of the supporting arm, (figure 3(b1)). Next, one assembles the supporting arm to the frames (figure 3(b2)). Both these assemblies can be done using simple glue tape. The mounting is then finished by assembling the camera to the camera holder (figure 3(b3)).

Notice that, it is necessary to strip the camera from its unnecessary plastic to make it lighter, and only then attach it to the arm. In addition to reducing the weight of the camera, stripping away the plastic shell exposes a flat side that can be used to attach the headgear.

2.2 Back-end software for the segmentation procedure

The full software is an open source code in Python (version 3.7.3) available at

`https://github.com/OttharKrohn/Eye-tracking`

which uses some of its standard packages and libraries from Anaconda distribution, including the usual libraries for this kind of software, namely *NumPy* (version 1.16.4) for scientific calculus, *CSV* for reading and writing csv-files, and *OS* and *Shutil* for handling files and folders. For the translation of sequences of images into eye trajectories we use the Python library *Open-CV 4.1.0*. This package is necessary for the fundamental parts of the algorithm, frame capturing, reading, writing and processing as well as performing these operations on videos. Additionally, several of its functions are used in the segmentation algorithm (see below).

The software applies an image-region segmentation procedure [6–8] to a set of frames composing the video of the motion of the eye. To better understand the segmentation procedure, it may be helpful to look at each frame of the video as a graph, or a matrix, instead of a picture. In our case the resolution of the pictures is $320\times$ 240 pixels, so they are two-dimensional data matrices with 320 columns (x-axis) and 240 rows (y-axis). The entries of this matrix are values, quantifying the color of the corresponding pixel. We use the *RGB* color-value system.

The segmentation procedure is illustrated in Figure 4 and is divided into the following stages:

(a) The recording stage, when the set of frames are collected as set of matrices of color values (figure 4(a)). Using the *RGB* color-value system, each entry of a matrix is composed by a 3-tuple, $\boldsymbol{C} = (R, G, B)$, given the values associated to each color of the *RGB* system, respectively red (R), green (G) and blue (B).

(b) The grayscale stage, when the algorithm converts the original color scale into a grayscale (figure 4(b)). Here one uses the function *grayscale* in *open-CV*, which stores only the intensity of light in each pixel, removing possible disturbing colors. More specifically, it converts the 3-tuple $\boldsymbol{C}$ value into a scalar, so-called the linear luminosity, Y_ℓ, which is given by a linear combination of the three values, R, G and B, and then redefines a new color-value of the pixel as the gray color $\boldsymbol{C}_{\text{new}} = (Y_\ell, Y_\ell, Y_\ell)$.

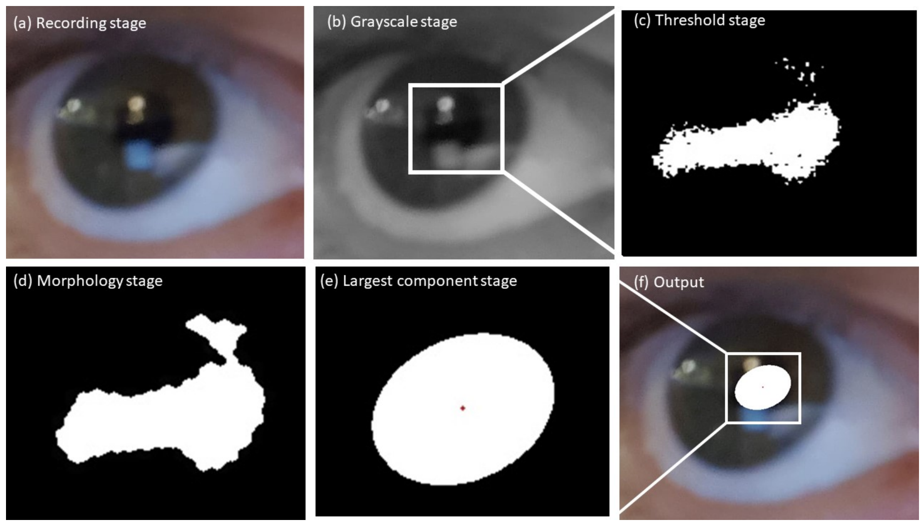


Fig. 4: The segmentation procedure: **(a)** In the first picture, you can see an image, a frame from a video of eye motion. **(b)** The first step is to convert the picture to gray scale, to make it easier to process. **(c)** The darkest parts of the picture is turned white, the rest is black. **(d)** Next, some smoothening is done with morphological transformations. **(e)** The largest connected component is selected and fitted to an ellipse. Center is marked with a red point. **(f)** The result is the superposition of the frame and the ellipse flagging the eye pupil.

(c) The threshold stage, when the algorithm maps the gray image into a black-white set of pixels (figure 4(c)). The white pixels indicate the region where the pupil most likely is located. To that end the algorithm uses the light-intensity of the pixels and imposes a threshold $Y_\ell^{(th)}$, to color each pixel either black, $\boldsymbol{C} = (0,0,0)$, if $Y_\ell < Y_\ell^{(th)}$ or white, $\boldsymbol{C} = (255,255,255)$, if otherwise.
According to our empirical tests, the threshold should be chosen in the range between black and the dark gray around $\boldsymbol{C} = (50,50,50)$. This range corresponds to the typically observed light-luminosity of the pupil. We fixed the threshold as $Y_\ell^{(th)} = 20$, which showed to be a reasonable trade-off between accuracy in localizing the pupil region in each frame and avoiding spurious regions with similar light luminosity. This stage is executed using the function *inRange* in *open-CV*. The threshold should be adjusted in each segmentation procedure, since different videos have different levels of light and shadow.

(d) The morphology stage starts with the set of black and white pixels and determines more accurately the region in the image where the pupil is located, coloring it in white (figure 4(d)). In the previous step one typically observes white points in the area covering the pupil, but without forming a connected region. To derive a connected region of white pixels, one uses morphological transformations, which evaluates if each pixel keeps its color or changes to the other one, depending on the surrounding pixels and according to a prescribed criterion. The criterion is implemented by defining a kernel, in our case a circle of pixels, and then the picture is scanned with this kernel. When the anchor point of the kernel (center of the circle) is white, the remaining points in the kernel are also white. This procedure is called dilation, since it leads to the increase of the number of white pixels. A complementary procedure leading to the decrease of white pixels, is erosion, which consists in coloring black all pixels inside the kernels whose center is black. Here, we choose the composition of one dilation with one erosion, both with a circle kernel with a radius of 7 pixels. In this stage, one uses the function *close* in *open-CV* and retrieves a connected white component to the next step.

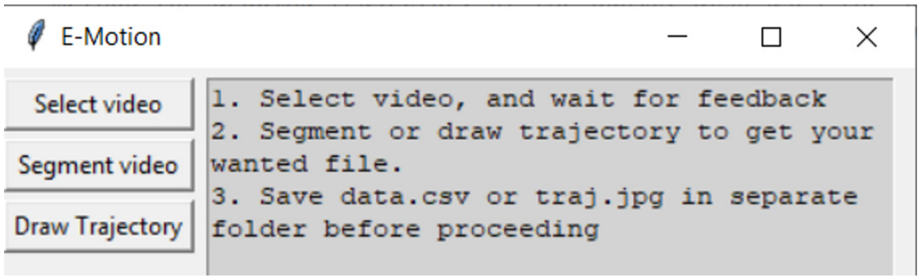
(e) The <u>largest component stage</u>, when the connected white component is fitted to a "pupil-shaped" curve, namely an ellipse (figure 4(e)). This stage is important, since some frames or videos may have shadows or other dark areas, such as eyelids, yielding several connected components disconnected from each other. If the previous stages were properly implemented, the largest connected component should be the one with the highest probability of covering the region where the pupil is located. Moreover, the image of the pupil will often show some reflection, which induces a spurious variation in the luminosity resulting in a "deformed" image of the pupil. Therefore, the fitting of the ellipse is helpful, since it enables a more accurate representation of the pupil. The different connected components in one image can be determined using function *connectedComponentsWithStats* function in *open-CV*, which then selects the one with the largest number of pixels.

    The fit of the largest component to an ellipse is done in two steps. In step one, using function *contours* in *open-CV*, one derives the set of points defining the boarder of the largest connected component. In step two, using the function *fitEllipse*, one derives the ellipse that best matches that boarder.

(f) The <u>output</u> of the segmentation procedure yields the superposition of the ellipse matching the pupil of the eye and the original image (figure 4(f)).

The center of the pupil is assumed to be given by the center of the ellipse, colored in red in Figure 4(e) and 4(f). It is this pixel which is tracked in time, and the series of its coordinates define the so-called eye trajectory, as illustrated in Figure 1. The segmentation procedure is repeated for every frame composing the video of eye-motion, writing out the coordinates for the pupil center in each frame, identified with the corresponding original image. By default, the code generates an output CVS-file with three columns: the frame number is stored in the first column, the x-coordinate of the center of the pupil is stored in the second column, and the y-coordinate in the third.

To summarize, the back-end part of our software segments a video of eye-motion, providing for each frame an estimate of the location of the pupil center, which is recorded as an output file. The software has a simple graphical user interface, which is described in the next section.



```
1. Select video, and wait for feedback
2. Segment or draw trajectory to get your
wanted file.
3. Save data.csv or traj.jpg in separate
folder before proceeding
```

Fig. 5: Illustration of the Graphical User Interface as the front-end part of our software. The main functionalities are described in Section 2.3.

2.3 Front-end software: the graphical user interface

At the front-end, the software opens a GUI window, shown in Figure 5. The GUI window is made using Python's *Tkinter* (version 8.6), which uses an item-hierarchy to make the desired GUI. The interface shows a text box describing the main steps for performing the segmentation of a video, and three functionalities on the left:

– "Select video": This functionality allows one to select a video for segmentation, which is then read and stored as frames in a dedicated folder. Figure 4(a) shows a typical example of the frame of a video stored with this functionality. We also use the *Tkinter* functions to open a pop-up window which allows one to click and select the video. The video is then split up into frames by the *videoreader* class and the frames are saved as jpeg-files, in a dedicated folder, called *frames* by default.
  Notice that, the program deletes and creates a new folder every time it reads a video, so as to avoid processing frames from two separate videos at the same time. The *videoreader* function also provides feedback if an error occurs, or when the storage of the video frames is finished.
– "Segment video": This functionality starts the segmentation of the chosen video, as described in Section 2.2 and retrieves the output CSV-file with the eye trajectory.
– "Draw trajectory": This functionality enables the possibility of drawing the eye trajectory, as the one shown in Figure 4. For the user choosing this functionality a video of the trajectory is also shown. By default, points are drawn on the coordinates of the position of the center of the pupil (figure 1(a)) and lines are drawn between the points to illustrate the trajectory of the center of the pupil (figure 1(a)). Here, the algorithm reads the picture of the frame using function *imread* in *open-CV* and simultaneously reads the black slate generated during the segmentation procedure, indicating the area covering the pupil in white. It also reads the trajectory picture and shows the superposition using the *NumPy* "stitching function" called *horizontal stack* and also the function *imshow*.

By default, the black-white slate is defined with size of 320x240 pixels. This size should be adjusted depending on the resolution of the input video.

2.4 Testing the cheap eye-tracker

For analyzing the accuracy of the eye-trajectories derived with our software, we compare it with the ones generated with the open-source software *EyeRecToo* [5]. This software is also designed for head-mounted eye-tracking devices.

The comparative analysis is done for examples of eye-motion videos recorded at 60 Hz and with frames of 320×240 pixels, as follows. For each specific eye-motion video, we extract the eye trajectory $\boldsymbol{r}_{\mathrm{new}}(n) = (x_{\mathrm{new}}(n), y_{\mathrm{new}}(n))$ with our software and another eye trajectory $\boldsymbol{r}_{\mathrm{std}}(n) = (x_{\mathrm{std}}(n), y_{\mathrm{std}}(n))$ with software *EyeRecToo* (ER2). Here n labels the frames composing the video.

Then, assuming trajectory $\boldsymbol{r}_{\mathrm{std}}(n)$ to be the "correct one", we measure the cumulative deviations of $\boldsymbol{r}_{\mathrm{new}}(n)$ from it during a time-span covering T frames. The left plots of Figures 6(a) and 6(b) show two illustrative examples, signaling $\boldsymbol{r}_{\mathrm{std}}(n)$ and $\boldsymbol{r}_{\mathrm{new}}(n)$ with different colors.

The cumulative deviations between $\boldsymbol{r}_{\mathrm{std}}(n)$ and $\boldsymbol{r}_{\mathrm{new}}(n)$ are accounted by a quantity D, which we define as

$$D^2(T) = \sum_{n=1}^{T} \left|\boldsymbol{r}_{\mathrm{new}}(n) - \boldsymbol{r}_{\mathrm{std}}(n)\right|^2 , \tag{1}$$

with $|\cdot|$ representing the vector norm. A perfect matching between both trajectories occurs when $D = 0$. Since all terms in the sum in equation (1) are positive, D is an increasing monotonic function of T. This monotonic behavior results from the increasing length of the eye-trajectory itself, since the total length L of a T-steps trajectory is given by

$$L(T) = \sum_{n=1}^{T} |\Delta\boldsymbol{r}_{\mathrm{new}}(n)| = \sum_{n=1}^{T} \left|\boldsymbol{r}_{\mathrm{new}}(n) - \boldsymbol{r}_{\mathrm{new}}(n-1)\right| , \tag{2}$$

and also increases monotonically with T. In equation 2, one uses the initial condition $\boldsymbol{r}_{\mathrm{new}}(0) = \mathbf{0}$. The middle panels of Figures 6(a) and 6(b) illustrate the monotonic behavior of both quantities, D and L.

To properly account for a *relative* deviations, we consider a normalized quantity, which converges towards a constant (typical) value, for a large number T of frames. This normalized quantity is defined as

$$I(T) = \frac{D(T)}{L(T)} . \tag{3}$$

The right panels in Figure 6 show this converge, and typically we find deviations below 10%, which shows a reasonable accuracy of our software.

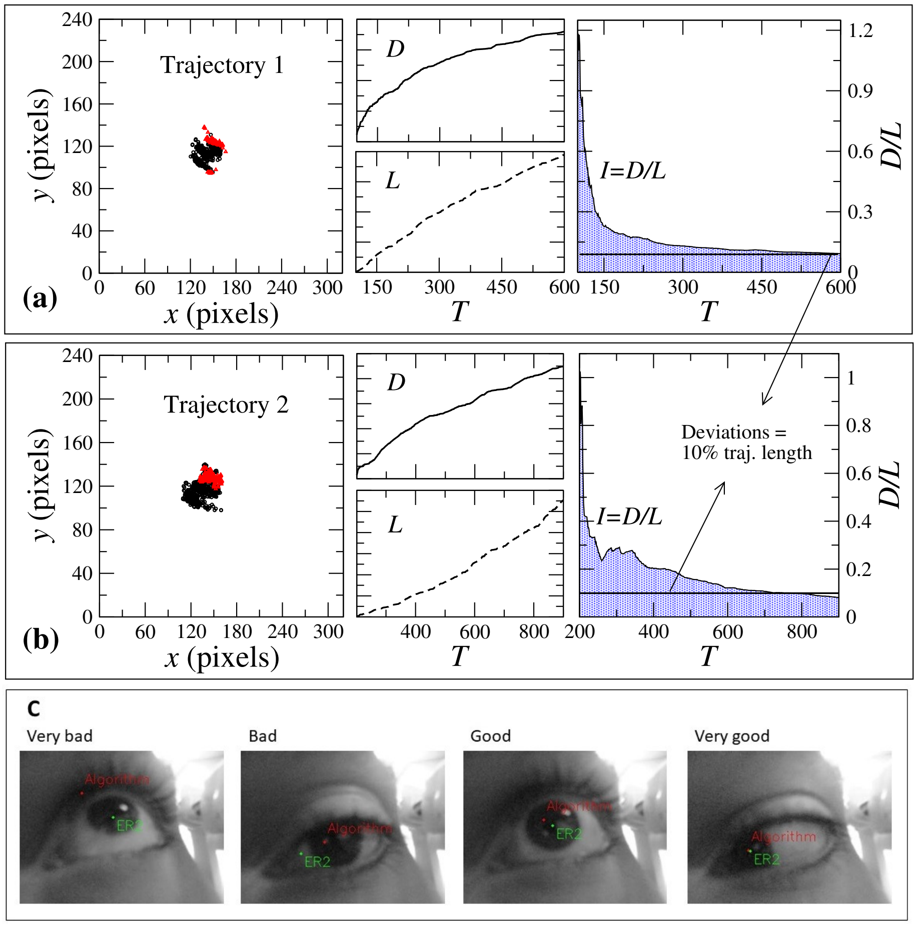


Fig. 6: Two illustrative examples, **(a)** and **(b)**, of eye-trajectories. **(Left)** The two-dimensional trajectories, colored in different colors, in the entire region of 320×240 pixels, to show the qualitative similarities. **(Middle)** The corresponding deviations, D, and total length, L, of the trajectories, as defined in equations (1) and (2) respectively. **(Right)** The index which measures the relative deviations between both trajectories, as defined in equation (3), indicating the level of 10% deviation between the trajectory measured with our software and the one extracted with *EyeRecToo* (ER2). In **(c)** four examples of pupil's location illustrate the limitations of both ER2 and our algorithm.

Neither our algorithm nor the ER2 software are exempt from errors and misdetections. As shown in Figure 6(c), while for most of the frames in the movie sequences (around 70-90%) show a good or very good match between the center of the pupil and the pixel selected by ER2 or our algorithm, there are a few cases where either one or the other detection procedure fails an accurate detection.

These few cases were filtered out in the trajectories shown in Figure 6(a) and 6(b).

3 The eye-tracker as a tool for teaching

A cheap eye-tracker as the one described in previous sections is within reach of almost any consumer, in particular school communities that can use it for educational purposes. In this section we suggest some of the possible situations in high-schools and at the universities where such a device can be of use. The single input needed if the dataset of eye-positions $\boldsymbol{r}(t)$ as a function of time t. While such dataset could be generated synthetically in any computer, the previous data collection through the usage of the eye-tracker promotes student engagement and motivation in the introduction of the topics afterwards. Furthermore, it also enables to bridge between scientific computing approaches for lectures in physics and mathematics and programming software in computer sciences.

A lecture on computer programming, such as Python or R, is becoming more and more ubiquitous at high-schools since the last years. Such a lecture could also profit from an eye-track device in two different ways.

First, at the programming level, the manipulation of the open-source code we provide in Github (see section 2.2) could be used for basics of Python programming, namely working with variables and data types, reading and writing I/O data, introducing lists and data matrices, learning if-statements and while-loops, and defining and plotting functions.

Second, at the scientific computing level, the contents described in the scope of physics and mathematics could be imported also for the programming course. The specific matters that can be taught in a class of mathematics and physics are described next separately.

3.1 Physics

Since eye trajectories are typically trajectories in space, they can be used as real examples of trajectories from the physical world, where fundamental concepts in physics can be tested. Starting from the concept of trajectory itself, typically represented as a vector $\boldsymbol{r}(t)$ whose coordinates are functions of time and correspond to the measurements collected by the eye-tracker itself, all basics of kinematics can follow:

- The computation of the velocity vector, which can easily be approximated as
$$\boldsymbol{v}(t) = \frac{\boldsymbol{r}(t) - \boldsymbol{r}(t - \Delta t)}{\Delta t} \, . \tag{4}$$
Such approximation is acceptable for sufficiently small Δt. The camera was used with a sampling frequency of 60 Hz which implies $\Delta t \sim 17$ ms. A better approximation would be the implicit scheme
$$\boldsymbol{v}(t) = \frac{\boldsymbol{r}(t + \Delta t) - \boldsymbol{r}(t - \Delta t)}{2\Delta t} \, . \tag{5}$$

Notice that while such a numerical scheme is implicit, it is easily to be computed from the set of values defining the recorded trajectory. Only the first and last trajectory position have no velocity defined.

– The computation of the acceleration vector, which could be done using a similar equation to equation (5), also known as Verlet scheme, which yields an acceleration approximated as

$$\boldsymbol{a}(t) = \frac{\boldsymbol{r}(t+\Delta t) + \boldsymbol{r}(t-\Delta t) - 2\boldsymbol{r}(t)}{(\Delta t)^2}\,. \tag{6}$$

– The computation of the curvature κ at each point of the trajectory, which can be parameterized by time, yielding

$$\kappa = \frac{\left|\frac{d^2y}{dx^2}\right|}{\left(1+\left(\frac{dy}{dx}\right)^2\right)^{3/2}}\,, \tag{7}$$

where x and y are the two coordinates of the eye position $\boldsymbol{r}(t) = (x(t), y(t))$. As explained below, both the first and second derivatives of coordinate y with respect to the coordinate x can be computed directly from finite differences of the time derivatives of x and y.

– The computation of the trajectory's center of mass $\boldsymbol{r}_{\mathrm{CM}}$, which also follows from direct inspection of the dataset, namely

$$\boldsymbol{r}_{\mathrm{CM}}(T) = \frac{1}{T}\sum_{t=1}^{T}\boldsymbol{r}(t)\,. \tag{8}$$

Notice that since the eye trajectories are typically not closed, its center of mass is not static and has its own dynamics.

3.2 Mathematics

In a mathematics lecture, most of the basics in numerical analysis and statistics could also be approached with the output data from experiments with eye-trackers.

In numerical analysis, after introducing the basics of numerical differential calculus, with the discrete version of first and second derivatives with respect to an independent variable, such as time (see equations (5) and (6)), more complicated derivatives can be introduced. For instance, with the motivation of computing the curvature along an eye trajectory one needs to compute numerically the derivative of one coordinate with respect to the other (see equation (7)). This is a different kind of derivative since we want to derive one dependent variable, y, with respect to *another* dependent variable, x.

However we can express $\frac{dy}{dx}$ and $\frac{d^2y}{dx^2}$ in equation (7) as expressions with the first and second *time* derivatives of x and y. For the first derivate one has

$$\frac{dy}{dx} = \frac{dy}{dt}\frac{1}{\frac{dx}{dt}}\,, \tag{9}$$

and after substituting the discretization of the time derivatives, which are similar to equation (5), yields

$$\frac{dy}{dx} \simeq \frac{y(t+\Delta t) - y(t-\Delta t)}{x(t+\Delta t) - x(t-\Delta t)} \,. \tag{10}$$

Notice that equation (10) has singularities only for $x(t+\Delta t) = x(t)$, i.e. when eye pupils are exactly static which in practice never occurs.

For the second derivate one applies the chain rule twice, yielding

$$\begin{aligned} \frac{d^2y}{dx^2} &= \frac{dt}{dx}\frac{d}{dt}\left(\frac{dy}{dt}\frac{1}{\frac{dx}{dt}}\right) \\ &= \frac{1}{\left(\frac{dx}{dt}\right)^2}\left(\frac{d^2y}{dt^2} - \frac{dy}{dt}\frac{\frac{d^2x}{dt^2}}{\frac{dx}{dt}}\right) \,, \end{aligned} \tag{11}$$

and substituting first and second time derivatives (see equations (5) and (6)) for x and y yields

$$\begin{aligned} \frac{d^2y}{dx^2} \simeq\, & 4\frac{y(t+\Delta t) + y(t-\Delta t) - 2y(t)}{\left(x(t+\Delta t) - x(t-\Delta t)\right)^2} \\ & - 4\frac{\left(y(t+\Delta t) - y(t-\Delta t)\right)\left(x(t+\Delta t) + x(t-\Delta t) - 2x(t)\right)}{\left(x(t+\Delta t) - x(t-\Delta t)\right)^3} \,. \end{aligned} \tag{12}$$

The choice of Δt in all calculations above has a lower bound given by the inverse of the sampling frequency of the sequence of the camera. However no upper boundary exists. Using different time increments, Δt, $2\Delta t$, $3\Delta t$, etc, the precision of the numerical calculus can be investigated and demonstrated how the global precision decreases with increasing time increment.

Finally, since the eye trajectory is typically a stochastic trajectory, it is characterized by specific *statistical* features. Consequently, basic statistical tools, such as the (two-dimensional) histogram of the most visited locations in space can be assessed.

4 Using eye-tracker for promoting social inclusion

For severely physically disabled non-speaking people, eye tracking technologies in conjunction with letter boards [9] offer the possibility for communication. This augmented mean of augmented communication (AC) has yet many drawbacks that have not been surpassed by engineering. For instance, the slow rate of communication, the energy expenditure and the cognitive load during the transmission of messages using AC systems impacts the quality of the interactions between augmented communicators and others [10, 11]. It has been observed that this issue can be mitigated by increasing the familiarity and ability of verbal speakers to communicate with a particular augmentative communication user [12, 13]. Although a similar facilitation-by-familiarity also happens between

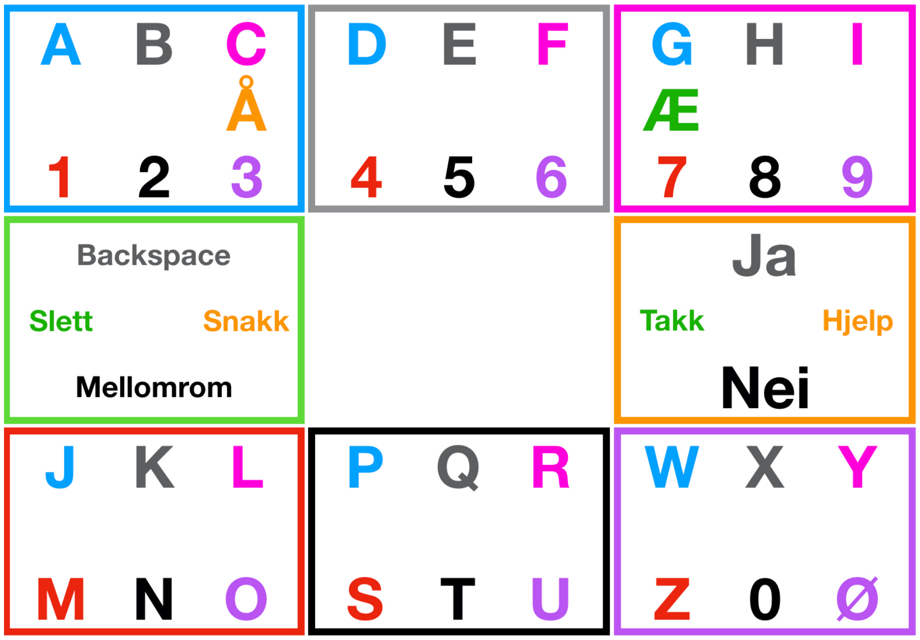


Fig. 7: E-Tran letter board interface in Norwegian: Each color identifies a direction towards which the eye should move. The first move selects a group of characters and the second move selects a character within the group.

two verbal speakers, part of the facilitation in the communication with the user of augmentative communication might be enabled by the understanding of the constraints imposed by the technology.

Here we propose that an eye-tracker can also be employed in educational activities to promote social inclusion of physically impaired and/or non-verbal people. By showing how the AC technology works, and how AC users communicate, we might increase the familiarization of students in a classroom regarding the communications issues. We can further allow verbal students to use the technology themselves in games, as if they had the impairment themselves, hence introducing the opportunity to empathize with AC users in a ludic and positive way. And we finally can follow up the motivation and awareness provided by the technology and the games to introduce information regarding the different health conditions that might lead to the need of AC technologies. These activities done in a class of students that is about to receive an AC user as a colleague, might severely diminish the alienation and the difficulties that AC users experience. What follows is a description of the interface, and the intervention protocols.

4.1 The interfaces

We propose a digital version of E-TRAN lettering board [14] as AC interface, as sketched Figure 7. E-tran boards are divided in nine segments, one central,

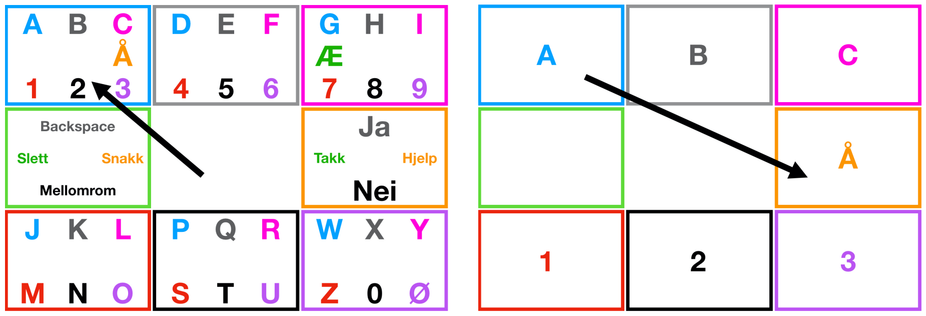


Fig. 8: Two-movements character selection process using an E-Tran letter board: Once the group of characters is selected, the board changes replacing the groups by the characters of the group in their respective positions for selection. **(Left)** 1st eye movement: Group of characters selection. **(Right)** 2nd eye movement: Character selection.

considered neutral, and eight peripheral sections. Each of these eight peripheral sections can be identified by direction or color (e.g., top-left : blue, top-center: white, top-right: yellow, middle-right: red, etc.). Within each one of these colored sections, groups of 6 to 8 characters are arranged by the same position and color rule of the eight initial sections of board. The user can communicate through two sequential eye-movements to key directions. First, the user moves the eyes towards the direction of the group of letters, and then to the direction that corresponds to the elected character.

Relative to other more sophisticated interfaces such as MinSpeak [9], the E-Tran interface has several advantages: it is relatively easy to learn, demands little effort because any letter can be chosen with two movements, and because only the direction of the eye is relevant for choice, it works well with eye-trackers with poor precision and accuracy. Additionally, because it does not require precise mapping of the eye-movement to the position on the screen, E-tran boards can be presented using any digital display. Thus, increasing its applicability into the educational setting. Figure 8 sketches the two-movements character selection.

Special functions, sentences and characters were introduced into the interface to account for the Norwegian language and to enable text editing capabilities (e.g., backspace, restart, space, etc.). Once the sentence is written, the user selects the speak command and the interface displays the entire sentence. Also, the interface speaks the typed sentence out loud using an open-source text-to-speech library, such as gTTS (Google Text-to-Speech) [15].

Additionally, other more limited interface can be implemented to explore the conditions of those with multiple limitations like associated blindness. In this second interface, there is no board. Letters are spoken to the user through a voice synthesizer. The user can chose the letter by looking down, go back the previous letter by looking left, start over by looking up, and say the message by looking right. The proprioceptive feedback from the eyes is enough for the user to know where they are looking to.

4.2 The protocol

The eye-tracker and the E-tran board might be introduced to a classroom in the following way. Firstly, by explaining that there are people with special health conditions that prevent them from using their voices or their limbs to communicate. Secondly, the instructor shows the equipment and explains how it is used as means of communication. Thirdly, the instructor will challenge the students to communicate using just the eyes. This can be done through quiz games, where the students have to answer without mistake, or who can copy a sentence the fastest. Finally, following up with a discussion regarding the experience of using the AC device, their reflections about the technology and how they would feel if they needed to use AC technologies to communicate.

The instructor must pay special attention to insights regarding failures of the equipment and limitations of the interface. These points are important because they create awareness about the difficulties of the user and the need to assist them when the equipment is failing. Additionally, the instructor might ask what the students would change in the interface, or what other people should do to help someone that is using AC systems. In this way, the students are employing creative thinking towards an inclusive mindset, that might help in future difficult situations. This dynamic can take place few days before the introduction of a class member that needs to use AC devices.

5 Discussions and conclusions

In this paper we describe how to build an eye-tracker using low-cost material solely, and provide the necessary software for localizing the eye-pupil in a series of frames composing a video. While we chose a head-mounted eye-tracker, there are other possible alternatives. The most common among them is the table-top mounted tracker, like the commercially available Tobii eye-tracker. Unlike the table-top mounted eye-tracker, the head-mount allows for freedom of movement and it is also more portable. Thus, we have made a design choice to develop around head-mounted setup because it offers more flexibility of use in the educational setting. A trade off between camera resolution/speed and its price and weight is an important factor. While there are cameras that are much lighter, faster and produce images with much higher resolution, these cameras are also very expensive. In our case, we re-purposed an old camera used for video-game applications. Because it is out-dated, it is exceptionally affordable, and it might be that the same strategy can be used in the future for cameras that are currently very expensive. This means that quality of image may increase using the same strategy without increasing the cost substantially. The integration of lighter and more capable cameras might support further developments, such as integrate it into virtual reality masks. This would now only allow to control the lighting around the eye, which may drastically improve accuracy and precision, but also integrate with all sorts of digital interfaces.

As important applications of the inexpensive eye-tracker, we described in some detail how it can be used for addressing specific contents at high-school classes or starting semesters of university studies in natural science. In particular, in physics the eye-tracker can be useful for teaching general kinematics and in mathematics it may increment engagement from the students when addressing basics in numerical analysis and statistics. In addition, the open-source code provided as the auxiliary software to process the eye-tracking recorded by the camera, can be manipulated by the students to implement additional functions discussed during the lecture.

Finally, we also proposed how such an eye-tracker can be adapted for performing specific experiments within universities. These experiments enable any student to access AC tools and in that way develop compassionate attitude towards AC users. The value of this approach in promoting empathy is yet to be assessed. But if it is proven valid, it will establish the eye-tracker as an important teaching tool for social inclusion.

References

1. M.Caligari, M.Godi, S.Guglielmetti, F.Franchignoni and A.Nardone: Eye tracking communication devices in amyotrophic lateral sclerosis: Impact on disability and quality of life. Amyotrophic Lateral Sclerosis and Frontotemporal Degeneration **14**, 546-552 (2013)
2. J.J.van Middendorp, F.Watkins, C.Park and H.Landymore: Eye-tracking computer systems for inpatients with tetraplegia: findings from a feasibility study. Spinal Cord **53**, 221-225 (2015)
3. H.A.Caltenco, B.Breidegard, B.Jönsson and L.N.S.A.Struijk: Understanding Computer Users With Tetraplegia: Survey of Assistive Technology Users. International Journal of Human-Computer Interaction **28**, 258-268 (2012)
4. L.Trojano, P.Moretta and A.Estraneo: Communicating using the eyes without remembering it: Cognitive rehabilitation in a severely brain-injured patient with amnesia, tetraplegia and anarthria. Journal of Rehabilitation Medicine **41**, 393-396 (2009)
5. T. Santini, W. Fuhl, D. Geisler and E. Kasneci: E. EyeRecToo: Open-source Software for Real-time Pervasive Head-mounted Eye Tracking. InVISIGRAPP (6: VISAPP) 2017, pp. 96-101 (2017)
6. S.J.Garbin, Y.Shen, I.Schuetz, R.Cavin, G.Hughes and S.S.Talathi: Garbin SJ, Shen Y, Schuetz I, Cavin R, Hughes G, Talathi SS. Openeds: Open eye dataset. arXiv preprint arXiv:1905.03702. Apr 30 (2019)
7. M.Tonsen, X.Zhang, Y.Sugano and A. Bulling: Labelled pupils in the wild: a dataset for studying pupil detection in unconstrained environments. In: Ninth Biennial ACM Symposium on Eye Tracking Research & Applications 2016 Mar 14 pp. 139-142 (2016)
8. B.Luo, J.Shen, Y.Wang and M.Pantic: The iBUG Eye Segmentation Dataset. In: Editor. Imperial College Computing Student Workshop 2018, Imperial College Computing Student Workshop (ICCSW 2018), Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, pp. 7:1 - 7:9 (2019)
9. E.Trefler and D.Crislip: No aid, an Etran, a Minspeak: A comparison of efficiency and effectiveness during structured use. Augmentative and Alternative Communication **1** 151–155 (1985)

10. D. Harris: Communicative interaction processes involving nonvocal physically handicapped children. Topics in Language Disorders, **2(2)**, 2137 (1982). https://doi.org/10.1097/00011363-198203000-00005
11. G.C.Vanderheiden: Non-conversational communication technology needs of individuals with handicaps. Rehabilitation World, **7** n2 p8-12 Sum (1983)
12. D.Beukelman and K.Yorkston: Non-vocal communication-performance evaluation. Archives of Physical Medicne and Rehabilitation **61**, 272-275 (1980)
13. A. Kraat: Communication interaction between aided and natural speakers: A state of the art report. Toronto: Canadian Rehabilitation Council for the Disabled (1985)
14. L.Lloyd, D.R.Fuller and H.H.Arvidson(Eds.): Augmentative and Alternative Communication: A handbook of principles and practices. Boston: Allyn & Bacon (1997)
15. gTTS - Google Text-to-Speech Python Wrapper. https://pypi.org/project/gTTS/. Last accessed 20 Jan 2020.