IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received January 28, 2021, accepted February 17, 2021, date of publication March 22, 2021, date of current version March 30, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3068055

A Reinforcement Learning Assisted Eye-Driven Computer Game Employing a Decision Tree-Based Approach and CNN Classification

JOÃO PERDIZ[1,3], (Graduate Student Member, IEEE), LUÍS GARROTE[1,3], (Student Member, IEEE), GABRIEL PIRES[1,2], (Member, IEEE), AND URBANO J. NUNES[1,3], (Senior Member, IEEE)

[1]Institute of Systems and Robotics, University of Coimbra, 3000-079 Coimbra, Portugal
[2]Engineering Department, Polytechnic Institute of Tomar, 2300-313 Tomar, Portugal
[3]Department of Electrical and Computer Engineering, University of Coimbra, 3004-531 Coimbra, Portugal

Corresponding author: João Perdiz (joao.perdiz@isr.uc.pt)

This work was supported in part by the Project B-RELIABLE financed by Fundo Europeu de Desenvolvimento Regional (FEDER) and Orçamento de Estado (OE) through Programs CENTRO2020 and Portuguese Foundation for Science and Technology (FCT) under Grant PTDC/EEI/-AUT/30935/2017, and in part by Institute of Systems and Robotics of the University of Coimbra through Portuguese Foundation for Science and Technology (FCT) under Grant UIDB/00048/2020. The work of João Perdiz was supported by the Portuguese Foundation for Science and Technology (FCT) through the Ph.D. Grant SFRH/BD/104985/2014.

ABSTRACT Human-Machine Interfaces employing biosignal-based inputs are hard to translate to real-life applications, in part because of the difficulty of developing generalized models to classify physiological events representing a user's actions. In the proposed framework, an Electrooculography (EOG)-based game is operated through a pipeline of decision methods. These include a user-independent classification model of eye movements using a Convolutional Neural Network (CNN), which is fed with images created from signal windows, and an Ensemble of Utility Decision Networks (EUDN), which moderates the impact of oftentimes conflicting ocular events while enabling a more natural level of control over the interface. The CNN and the EUDN replace the normally used feature-based ocular event detection methods for EOG. Finally, a Reinforcement Learning-based game actuation approach simultaneously updates multiple (*State*, *Action*) pairs for each rewarded outcome, intervenes to mitigate the consequences of wrongful game Commands, and can be used as part of a ''shared-control'' paradigm based on EOG. Results show a positive impact of Reinforcement Learning both in improving participants' game performance as well as in reducing some of their subjective workload indicators.

INDEX TERMS CNN, decision tree, electrooculography, reinforcement learning.

I. INTRODUCTION

Serious games are usually developed in connection to rehabilitation scenarios, or for players with some form of cognitive or motor impairment that prevents normal human-computer interaction modes. These specific needs have made biosignals natural candidates for alternative input modes in a game, and depending on the source, they can detect either voluntary or involuntary physiological events from the player. There are, however, several issues preventing quick adaptation of players to biosignal-based games, of which the most important might be the system calibration and familiarity with the gameplay.

The associate editor coordinating the review of this manuscript and approving it for publication was Wentao Fan.

Serious games have already been implemented using different types of biosignals as their input mode, such as Electroencephalographic signals [1], [2] or Electromyographic signals [3]. In this work we focus on Electrooculographic (EOG) signals, which are generated by electrical potentials measurable around the eye, given that each ocular globe works as a dipole. Almost all types of ocular movements can be detected in this way [4], which includes vertical and horizontal saccades. Eyelid movements such as blinks can also be detected using EOG. Typically, physical EOG setups include two pairs of bipolar electrodes and a grounding electrode, which can be self-standing or integrated on a goggle-like device; they are primarily used to classify eye movements into four classes – up, down, left and right.

This type of classification is done in [5], as the player controls a dance game by issuing directional commands using

EOG. In [6], a goggle-like device is used to acquire EOG signals to classify eye movements so that players can perform increasingly difficult sequences of eye movements. In [7], EOG is used to position gaze so that an agent throws a baseball in the chosen direction, with eight directional targets available.

Given the relative scarcity of EOG-based game literature, it is also important to refer to EOG-based Human-Machine Interfaces (HMI) aimed at improving the quality of life of motor-impaired people. Saccade detection and eye-tracking can be used to either enable wheelchair navigation [8], remotely control a television [9], or type phrases [10], [11]. In [12], character recognition is obtained from the continuous tracking of eye movements through EOG. A speller using monopolar EOG acquisition is demonstrated in [13] to detect blinks synchronized with flashes on a virtual keyboard.

Limitations of EOG include the discreteness of ocular events and the unreliability of signal-derived features. This means EOG-based interfaces are very different from conventional game interfaces, making initial unfamiliarity with it a major point of concern. The impact of these issues may be minimized by adapting some of the game's features to the player's general expectations, improving game flow for people controlling EOG-driven games for the first time and who might be disappointed by the very specific limitations posed by this type of input.

The use of Reinforcement Learning (RL) is one of the ways in which these limitations can be mitigated. RL is a group of machine learning methods in which every action a machine takes is registered and assigned a reward, impacting its future actions. In an RL machine learning approach, future actions from an agent are then modified with the goal of maximizing the machine's predetermined metric of global return [14]. In HMI trials, the relatively small amount of recorded data compared to other implementations of machine learning means that a direct RL approach is generally favored, instead of a data-driven deep learning-based approach. Examples of this approach can be seen in the game-like robotic rehabilitation HMIs implemented in [3], [15], in which RL allows for in-game adaptations of the physical system's response so the user remains motivated and engaged.

EOG classification is usually accomplished using either time or frequency domain features [4], [16]. Among the various classification methods available, Neural Networks (NNs) have the particularity of bypassing the step of feature selection, as the most relevant are automatically selected when training the NN [17]. NNs have been used in both raw and filtered EOG signal classification [18], [19], and have also been used to classify selected features [20]. Convolutional Neural Networks (CNNs), a subset of NNs, have been introduced and remain especially popular in the field of computer vision [21]. A CNN can extract features directly from raw images, yielding very good performances in dataset-based image recognition [22], [23]. The application of CNNs towards classification of EOG signals is rare, although examples can be found in [24], to detect the state of drowsiness using filtered EOG signals as inputs, or for eye movement recognition by using the EOG's FFT as the CNN's input [25].

A. GOALS AND CONTRIBUTIONS

In this paper we strive to mitigate the user adaptation problems often raised by the need for calibration and unfamiliarity to an interface, which prevent normal engagement with an EOG-based HMI. These problems were found in [26], where we first introduced an EOG-based videogame whose performance was excessively dependent on the user's EOG calibration and on the detection of the player's centering saccades, limiting the playability of the game.

The methodological contributions of this paper, which as a whole form a new EOG event decision and classification pipeline, are threefold:

1) A user-independent CNN model, acquired using EOG, is used for ocular event classification. The CNN input is based on RGB images obtained from EOG windows, which to the best of our knowledge is the first time such an approach is used for classifying ocular events. A sliding window allows the player to issue Commands asynchronously.
2) An Ensemble of Utility Decision Networks (EUDN) that allows to prevent conflicting classification decisions from the CNN's sequential decisions, which may result from the overlapping windows.
3) Two RL approaches are proposed, with two modified Q-Learning methods with new update and reward methods. Both approaches are continuously updated during gameplay, and their models are used to help the user avoid collisions. The first approach only does so if the user has issued prejudicial game Commands, while the second intervenes even if the user has issued a ''Null'' Command. This configures a ''shared-control'' paradigm. In their reward policies, for negatively-rewarded outcomes, three (*State*, *Action*) pairs are revalued instead of one, so as to discourage repetition of similar Event sequences in the future.

B. PAPER ORGANIZATION

The paper is organized as follows. In Section II, the system architecture and the methods used in each version of the game are described. In Section III we present the gameplay results of three different modes of processing the EOG command signals, including game performance for the three different versions of the game incorporating RL. In Section IV we expand this presentation into a discussion on the results of the online game sessions. We reserve Section V for a concluding note about the possible impact of this work and its implications in developing Neural Networks and Reinforcement Learning mechanisms for the recognition and classification of human activities using biosignals as their sole input.

II. MATERIALS AND METHODS

A. ACQUISITION SETUP AND PARTICIPANTS

EOG signals are acquired from electrodes, placed in the periphery of the eye according to montage in Fig. 1. EOG

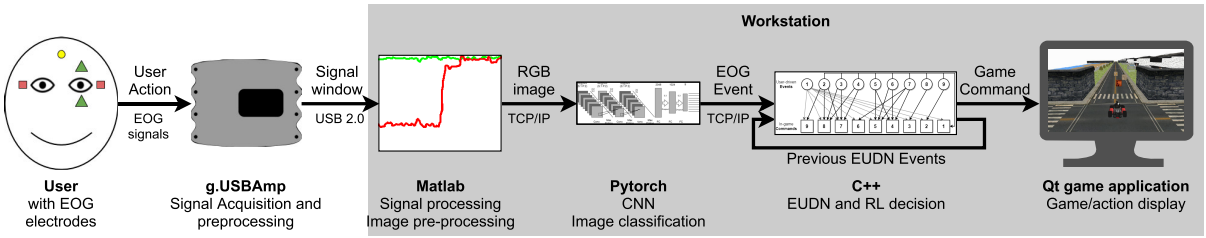


FIGURE 1. Diagram of the overall system architecture. All operations related to signal processing, classification, decision and game display are executed in the same workstation. User commands are issued through ocular movements, which are acquired by two bipolar EOG channels placed on the face. Squares denote the horizontal EOG electrodes, triangles point the vertical EOG electrodes. The ground/reference electrode is placed on the user's forehead (circle).

signals are acquired using a g.tec gUSBamp amplifier with a sampling rate of 256 Hz, filtered with a 50 Hz notch filter, and transmitted to a real-time Simulink environment. Eleven participants took part in the trials (8 male), with an average age of 27.7 years ($\sigma = 7.4$). For the trials, participants were seated about 55cm away from the 24$''$ computer game screen.

The study was conducted complying with the code of Ethics of the Declaration of Helsinki. Informed consent was obtained from all participants, explaining the aims of the study, their role as participants (e.g., voluntary participation) and the ethical commitments of the research team (e.g., data anonymization, guarantee of confidentiality). The EOG electrodes mounted on the face are non-invasive and without health risks, and the experiments were conducted only with healthy participants.

B. OVERALL SYSTEM ARCHITECTURE

After acquisition and pre-processing, EOG signals are transformed into intensity maps of the vertical and horizontal EOG signals, which are encoded into an RGB matrix to form an image in PNG format. This image is sent via TCP/IP into the application running the CNN, which receives one new image every 0.25 seconds. Successive decisions outputted by the CNN are fed into our Ensemble of Utility Decision Networks (EUDN), whose rule-based output takes into account the CNN's last eight classification decisions. The resulting game Command is displayed to the user in the game screen. The overall pipeline of the data processing system is shown in Fig. 1. Each module is detailed below.

C. EOG SIGNAL, USER EVENTS AND GAME COMMANDS

To provide the player with a reasonable degree of control over the vehicle, we have designed our kart game with nine different EOG Event classes, reflecting several different user actions. Six of these classes are related to saccades of different amplitudes, with three classes for rightward saccades and three for leftward saccades. We do not account for vertical saccades because only lateral movement of the kart is needed in the game. There are two classes of blinks considered, single blinks and double blinks. User action that does not fall into one of these eight classes is classified as a "Null" Event, originating no game Command. The nine Event classes from User Actions are described in Table 1.

TABLE 1. Discrete encoding of Event classes to their corresponding User Action.

Event class	Originated by
1	Null Event
2	"Strong" leftward saccade
3	"Medium" leftward saccade
4	"Weak" leftward saccade
5	"Weak" rightward saccade
6	"Medium" rightward saccade
7	"Strong" rightward saccade
8	Single Blink
9	Double Blink

TABLE 2. Codes of game Commands.

Command	Effect
1	Stay on course, do not change speed
2	Stay on course, increase linear speed by 20%
3	Stay on course, decrease linear speed by 20%
4	Lateral swipe to right, at 80% of normal lateral speed
5	Lateral swipe to right, at 100% of normal lateral speed
6	Lateral swipe to right, at 120% of normal lateral speed
7	Lateral swipe to left, at 80% of normal lateral speed
8	Lateral swipe to left, at 100% of normal lateral speed
9	Lateral swipe to left, at 120% of normal lateral speed

Game Commands and go-kart actions are indifferently defined, as no go-kart dynamics are considered. Game Commands have an indirect connection to eye movements, and hence to Events. Game Commands are described in Table 2. The mapping between Events and Commands will be described further ahead, in connection with the description of the classification and decision methods.

D. GAME OPERATION AND GRAPHICAL INTERFACE

The game's graphical interface follows the same layout proposed in [26], with a go-kart (controllable agent) going down a straight road and capable of avoiding obstacles by moving laterally (see Fig. 2). Obstacles appear on the road in random lateral positions, but are generated at a fixed rate. A collision with the go-kart is detected using an axis-aligned bounding box chain between the obstacle and the agent. Each collision increments a penalizing score, which can be compared at any time with the total number of obstacles presented to provide a measure of a success rate for the player. The scenario is a 3D infinite scrolling model in which the agent can only move over the road. The game was developed using C++/Qt and

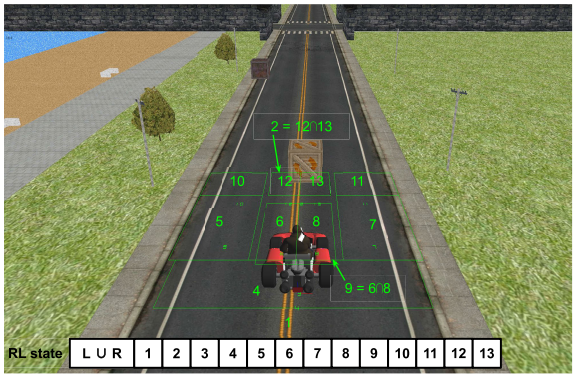


FIGURE 2. A snapshot showing the gaming environment, the kart and two obstacles. Obstacles appear randomly on the road at fixed intervals, and the user must perform saccades to deviate from them, or blinks in order to change the kart's speed. The kart can only move over the road and the sidewalk. Superimposed over the game environment are also the bounding boxes and the numbering of the states used to determine RL State information, which is given by a 14-bit value shown in the bottom. After the bit value for whether the go-kart is in the left or right-hand sides of the road, there are 13 other non-mutually-exclusive bits that inform on the position of an obstacle relative to the go-kart. Bit 1 is set when obstacles are outside of any of the bounding boxes. Bit 9 is set when both Bits 6 and 8 are set as well; together with Bit 3, these four are collision bits, pointing to situations in which the bounding boxes of the kart and an obstacle overlap. Bits 5, 7, 10 and 11 are set when a lateral collision might occur if the next command is issued in the wrong direction.

uses OpenGL to draw its visual elements at a frame rate of $\approx 24Hz$. Game parameters, such as the player's point of view over the scenario, the initial speed of the kart or the frequency with which obstacles appear, are easily adjustable, but were kept fixed across all participants and trials.

From the user's perspective, the go-kart can move to its left and right sides, to which it must be steered to avoid obstacles along the way. This is done by executing leftward and rightward saccades depending on the direction to which the user wants to steer the kart to. Saccades are completely asynchronous, being executed at the player's discretion and not dependent on any visual cue from the scenario other than the boxes to avoid; sometimes more than one saccade may be necessary in order to completely avoid a collision. The participant must look at the center of the screen and define a strategy to avoid boxes by making saccades. Any returning saccade (a natural movement to the center) will be discarded by the system, i.e. will not be considered as a command.

Besides registering the player' saccades, the decision module also allows for detection of eye blinks, as they are an inescapable part of eye movements and can become more frequent in cases of ocular fatigue. Here blinks are used for speeding up or slowing down the kart, allowing the user to regulate the game's pace to some extent. A double blink increases the go-kart's speed along the road by a fixed percentage relative to the base speed. To decrease the go-kart's speed, a player must perform a single blink, clearly detached from other ocular events in time.

E. CLASSIFICATION AND LEARNING ARCHITECTURE

In the present work we replace the saccade detection algorithm in [26], which was based on saccadic amplitudes and fixed thresholds based on a calibration session, with a decision provided by a CNN which is included in the information pipeline shown in Fig. 3.

The CNN takes as its input an EOG signal window that has been scaled and transformed into an RGB image, working in an analogous way to that in image recognition [22]. The CNN's output is a code specifying which Event class was found. This class is appended to an Event list together with the previous seven Events outputted by the EUDN,[1] forming a sequence that is inputted to the EUDN, which generates a shortlist of possible game Commands. This shortlist is then passed to the Q-Learning algorithm, which chooses a suitable Command according to the Event found and its specific Q-Table values for the relevant State and possible Commands (as detailed in Section II-E3). The resulting game Command will likely generate a change of the State of the kart and a reward that modifies one or more Q-values. Given enough user actions, the rewards will modify the Q-Table of this Event towards choosing Commands less detrimental to the player's performance within the game, helping to correct his/her errors when issuing game Commands via saccades and blinks.

1) CNN-BASED EYE MOVEMENT CLASSIFICATION

The User Action classification is based on the operation of a CNN and a decision system, whose modules are described in the following subsections.

a: RGB IMAGE GENERATION

Acquired EOG signals are converted into images, which are inputted into the CNN. Given that there are two EOG channels being recorded, we convert horizontal signals into the Red values and the vertical channel into the Green values of an RGB image. The Blue channel of the image is not used and is thus left as zero.[2] A sliding window with 256 samples is taken from each EOG channel and its maximum and minimum are taken. The range of variation of each channel within the window is then calculated, with both signals being scaled to the highest range between them. This allows the image to retain information on the differences in variation between horizontal and vertical signals, enabling the network to distinguish between different User Actions. After this scaling, signal intensity values are scaled again to fit the image's fixed height, with each image being 256×200 pixels. An example of EOG signals converted into RGB images can be seen in the top half of Fig. 4.

With EOG signal amplitudes correctly scaled, we employ the Bresenham rasterization algorithm [27] for filling in the pixels. To achieve this, we find the pixel nearest to each intensity datapoint and connect it to the next datapoint (the column of pixels to the right, i.e. intensity in the following

[1] If there are not enough Events in the EUDN's buffer memory, it is filled in with Null Events.

[2] The Blue channel is not fed to the CNN in order to reduce the size of input data, but is kept in the recorded images for convenience, i.e. for visual inspection of training images.

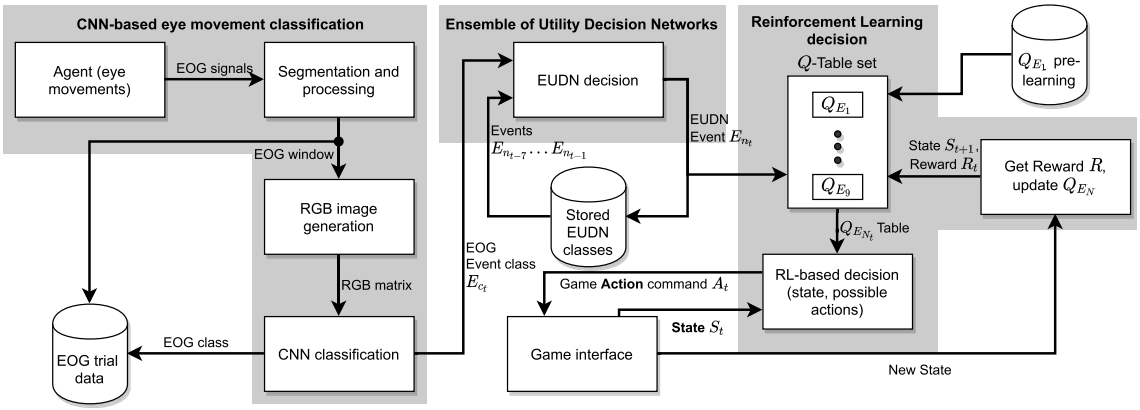


FIGURE 3. Diagram of the game's information pipeline on each step, with a duration of 0.25s, showing each of the processing modules involved. The algorithm runs every 0.25s and starts with the scaled RGB images that are fed into the CNN. The class from the CNN is fed into the EUDN, along with the last seven classes decided by previous iterations of the EUDN. The selected Event E_N will determine which of the nine Q-Tables (one per Event) will be used in the subsequent RL algorithm. Values in the relevant Q_N table are employed to determine a game Command, an outcome, and eventually a Reward. The algorithm's cycle is completed upon updating the Q_N table if RL is activated, otherwise it ends upon selection of a game Command according to the classified Event at the EUDN. This algorithm is the same for passive and "shared-control" RL. However, in the former the RL model (table Q_1) for the "Null" Event has not been trained beforehand, while in the latter that table has been updated through a training session without human intervention.

sample). The pixels are linked in each samples to each other, creating a continuous line for each channel and producing an image where only 0s or 1s exist in the Red and Green channels of the image data, as the path between points is absolute.

b: TRAINING A USER-INDEPENDENT CNN

The standardized RGB images are used to train a static CNN and bypass a calibration phase, thereby having the participants play the game as soon as the physical setup is ready. This is an important advantage in improving the experience for biosignal-naïve players. The CNN was implemented in PyTorch with three convolutional layers and three fully-connected layers. Each layer is followed by a Rectified Linear Unit (ReLU). The network is trained with a cross-entropy loss function and uses the Adam optimizer [28] with a learning rate of 0.0001. The CNN architecture is shown in the bottom half of Fig. 4.

We trained the CNN using participants data obtained from preliminary trials during this work, and from participants' data from other EOG experiments ran in our lab with similar EOG training paradigms, such as [26] and [29]. The collected data were automatically assigned to one of the classes listed in Table 1. Training windows for double-blinks had to be synthetically created from recorded blink windows because they did not explicitly exist in the recorded data of any previously used EOG paradigm. To label saccades according to their amplitude, an average range of amplitude variation was calculated for each different loaded trial; saccades are labeled according to their variation on the final image. 5237 RGB images were extracted and labeled, with each label being reviewed by an expert after assignment. The horizontal and vertical ranges of EOG signal variation that were used on the online game trials were taken from a grand average of all signal windows' amplitude variations, from all the sources used to train the CNN. These scaling ranges are automatically adjusted during each game session so that EOG signal traces do not overshoot the image windows being created.

Null-class EOG Events are not explicitly generated by participants, so they were extracted from data using two different processes. In the first, we searched data segments without any detectable User Actions in them, originating signal images resembling flat lines. The second process looked for windows where an overlap of detectable user actions occur, rendering them invalid. These will influence the CNN's training, helping us to avoid non-Null classification of signal windows whose interpretation could be described as ambiguous, e.g., in which a saccade and a blink could be seen in close proximity.

2) ENSEMBLE OF UTILITY DECISION NETWORKS

In our algorithm, an RGB image obtained from an EOG window is fed to the CNN every 250*ms* (see Fig. 3). This means an intentional action by a user, and its corresponding signal, will appear in multiple windows, shifted by 25% of the window's width (each window has 75% overlap with the previous window). This makes it necessary to establish a mechanism to prevent multiple classification outcomes that might result in unnatural game behavior, which would become a very limitative issue. Thus, we developed a decision method, called EUDN, which is shown in Fig. 5. The EUDN selects the most relevant Event within a sequence of classification outcomes, after which a game Command is generated. Utility trees evaluate the utility of assigning an Event with class c from the CNN to class n (where $c, n \in \{1, \dots, N\}, N = 9$) based on its own history of previous decisions and on the most recent CNN classification decision.

We take the Event decisions from the last eight EOG signal windows – epochs – and their decision output classes, and feed them to a set of utility trees. Each epoch's decision is weighted according to its "age", with more recent decisions weighting more. From this inputted sequence of eight Event

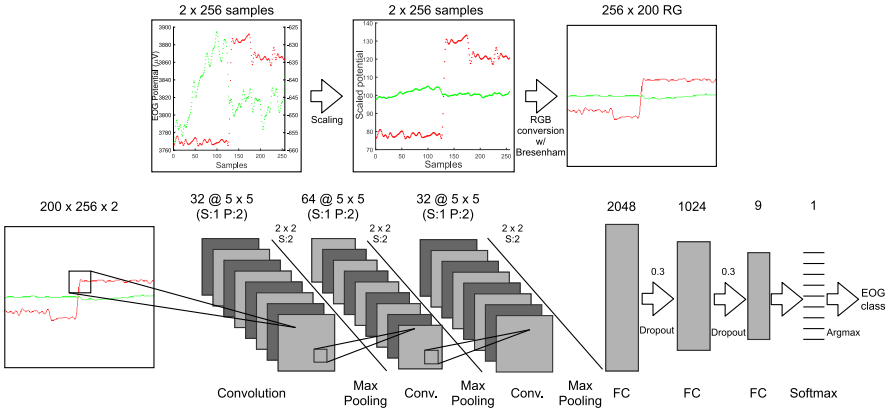


FIGURE 4. On top, the EOG signal processing path, from acquisition to RGB image. From left to right: raw horizontal (red) and vertical (green) EOG signal windows (respectively traced to left and right y-axis); the scaled windows; converted RGB matrix created using the Bresenham algorithm, used as input to the CNN whose architecture is shown on bottom. It is comprised of three convolutional layers and three fully connected layers, with softmax and argmax used to output an Event class.

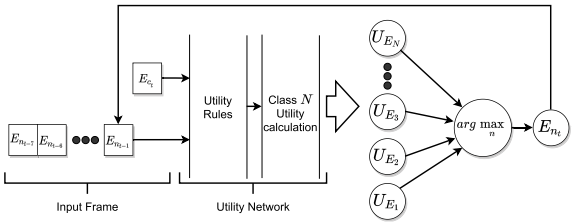


FIGURE 5. EUDN block diagram. Each Event E_n has, associated with it, a set of rules with which the class of that Event's utility (U_{E_n}) is determined, and which takes as input the current CNN-based input (E_{c_t}) as well as the last seven EUDN decisions ($E_{n_{t-1}}, \ldots, E_{n_{t-7}}$). The class with the highest utility is the decision of the EUDN and corresponds to the outputted class E_{n_t}, appearing as $E_{n_{t-1}}$ in the following decision cycle.

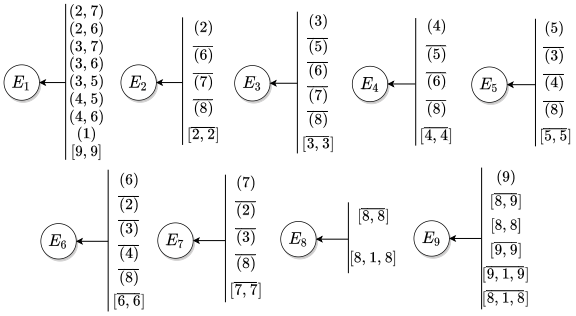


FIGURE 6. The utility rules used for valuing each Event class. For each class, a set of rules describes either a sequence that benefits the decision toward that class, or the affirmation or denial of an Event class. Tuples (x_1, x_2) are rules increasing or decreasing the utility of a class in the case of prior decisions x_1 and x_2 being present, in no particular order. The presence of isolated decisions (x) is also considered in some rules, as are sequences of two or three prior decisions [x_1, x_2, x_3].

decisions, seven are previous decisions from the EUDN, so only the most recent Event class is the current CNN classification decision.

When the set of trees is presented with a decision sequence, each tree will calculate the score of its class according to the rules shown in Fig. 6, with all scores being compared between them. Since every tree is associated with a specific class, the class value whose tree utility score is maximized by the eight-Event sequence is inputted in the next module of the pipeline, and stored as the EUDN's Event decision for that epoch (E_{n_t}).

a: EUDN RULES

The trees associated to the classes are shown in Fig. 6 and follow a set of intuitive rules governing the detection of successive user actions in different classes:

- The system must be able to distinguish between deviation saccades and back-to-center saccades;
- If a leftward or rightward saccade is two or more decision windows away, a following saccade in the opposite direction is treated as a back-to-center saccade, producing a "Null" EUDN Event class;
- If two conflicting saccades (left vs. right) are found in sequence, a "Null" Event class is issued instead;
- A blink can be detected in a single window, but not in consecutive windows within the same EUDN frame. This means a valid double blink is allowed to be either a real double blink or a timely succession of valid blinks at least two decision windows apart;
- Saccades take precedence over blinks, and blinks take precedence over "Null" Events.

As an example referring to Fig. 6, the utility of declaring the Event in class 7 (a strong saccade to the right) is incremented when there are no Events in classes 2 or 3 (leftward saccades) in the Event sequence – $\overline{(2)}$ and $\overline{(3)}$, but decremented if two consecutive code classes of its own ("7"s) appear in the sequence – $\overline{[7, 7]}$. Utility for class 7 is also incremented if there are no blinks (Event 8) in the sequence – $\overline{(8)}$. This is meant to avoid duplicated Event classes in the sequence, as it is not possible for two strong saccades to be made consecutively, by the user, in the same direction.

3) REINFORCEMENT LEARNING-BASED DECISION

In the context of a computer game, RL cannot be the sole driver of the game agent's actions. Rather, it must act as an

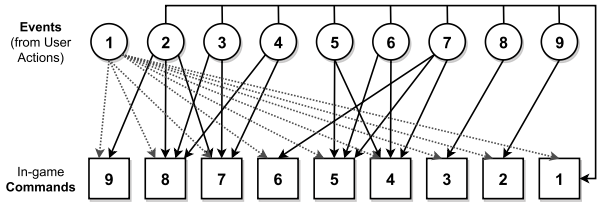


FIGURE 7. Mapping of Events (circles) to possible game Commands (boxes) when Reinforcement Learning is employed. Afterwards, RL is responsible for selecting one of the Commands for the appropriate user action. When RL is not employed a game Command is selected outright from this mapping. Note that Command #1 is always selectable as it involves no change in direction or speed on the go-kart's part.

assistant that mitigates both user and classification mistakes by intermediating between possible user actions and game Commands, trying to adapt its response to the former in order to maximize the outcomes of the latter. This means, for example, preventing the kart from issuing a rightward movement when the user's intention was to command it to the left. However, Events classified by the CNN, which are coded in Table 1, are not univocally mapped to a single outcome (game Command) in the game's environment. Instead, each Event is mapped to up to four different game Commands. The Event-to-Command mapping employed by the RL algorithm is illustrated in Fig. 7, and the Events and Commands are described in Tables 1 and 2, respectively.

There are two distinct ways in which we have applied Reinforcement Learning to our game – we call them the RL_1 and RL_2 modes –, but first we describe their common core, the modified Q-Learning method. The standard Q-Learning algorithm [14] was modified to better suit our goals. We employ an array of modified Q-Learning matrices – Q-Tables –, in which Actions are chosen through a greedy choice paradigm applied upon Q-matrices of (*State*, *Action*) pairs valued differently. In the standard Q-Learning method, the Q-matrix would be updated every time a game Command (the RL's action) resulted in a Reward being attributed to the agent. Here, as detailed below, a Q-Table can be retroactively updated if a negative outcome occurs in a small number of decision cycles taking place immediately after the (*State*, *Action*) pair has been recorded. The Action selection process is not changed, as it still depends on choosing the (*State*, *Action*) pair that maximizes the given State's recorded Reward values.

a: GAME STATES

In order to work in the context of Reinforcement Learning, the aforementioned game Commands must be translated into Actions functioning within a (*State*, *Action*) pair fed into a RL algorithm, which means States must be defined. We have chosen to represent States as 14-bit words comprised of several non-mutually-exclusive bits that point to the position and proximity of an obstacle relative to the kart, or the occurrence of a collision. As shown in Fig. 2, there are 13 collision state bits that can be set, and a road-side bit indicating whether the go-kart is on the left or right side of the road. Some state bits' triggers are not directly observable in that Figure, such as for bit 1 – set when obstacles are away from the go-kart, in any of the areas not occupied by numbered boxes; and bit 9, set when go-kart is superimposed on an obstacle on both of its left and right sides. Multiple states can be occupied at once depending on the obstacle's position relative to the go-kart.

b: MODIFIED Q-LEARNING REWARDS AND UPDATES

In the process pipeline, shown in Fig. 3, we initialize an empty Q-Table for each Event class, which is to be filled with addresses representing different game States. Each position will have as many Q-values as the possible game Commands (RL Actions) that the Event allows. For training the RL model, we define one preferred game Command for each Event class. When updating State pointers in that Event's Q-Table the values pertaining to the Event's preferred Command suffer greater modifications than values related to other Commands, and when compared to that Command's values in other Event's Q-Tables.

For online playing, a particular State's Q-values within an Event's Q-Table are evaluated to select an appropriate RL Action (a game Command), and updated with the appropriate Reward depending on the outcome. A Q-Table can only be updated by obtaining, in the decision process, the Event that the Q-Table represents; the update process is shown in Algorithm 1. After taking an Action (a game Command) derived from that Event's Q-Table and observing a Reward, we form a window composed by (*Event*, *State*, *Action*, *Reward*) tuples containing information about the three previous Actions. If there is more than one non-Null Action in this window the Reward is set to (-1) to avoid assigning delayed rewards stemming from an Action that was not the last classified Action. The current Action is appended to this Action window. If for any of these three Actions a negative Reward was issued, all of the previous Actions' Q-Tables are penalized with this negative Reward, in order to prevent the assignment of positive Q-values to a similar chain of (*Event*, *State*, *Action*, *Reward*) instances in the future. This behavior is expressed in the *if* statement starting at line 15 of Algorithm 1. Otherwise, the Q-value of the last (*Event*, *State*, *Action*, *Reward*) is updated with its Reward, found earlier.

This method was adapted from the Q-Learning algorithm in [14]. Its α and γ values, for step-size and discount-rate respectively, were chosen to be close to unity so that the Q-Tables can quickly adapt to an individual user's behavior pattern, a desirable outcome because each user only inputs a relatively small number of actions.

The reward should depend on the distance d between the go-kart and its nearest obstacle after the Action is taken. We have chosen to model the reward given to each Action as follows:

$$R(d) = \frac{1}{1 + e^{k_1(d-k_4)}} + e^{-(d-k_4)^2/k_2} + k_3 \qquad (1)$$

In the experimental validation of the go-kart game the following parameter values, obtained empirically, were applied:

Algorithm 1 Q-Learning RL Algorithm as Implemented in Our Game, Adapted From [14]

Initialize $Q(s,a) = Q_0(s,a), \ldots, Q_E(s,a)$
$R \in [-1,1]$, $\alpha = 0.8$, $\gamma = 0.9$
for each EOG Event (e) **do**
  Choose Action A from state S using policy derived from Q_e
  $A \leftarrow \underset{a \in \mathcal{A}_e}{\text{argmax}}(Q_e(S,a))$
  Take action A
  if training **then**
    Observe actual reward $R = R(d)$, where d is the distance to the nearest obstacle
    Compute new state S'
    Window of previously performed actions $(D = (e_0, S_0, A_0, R_0), \ldots, (e_M, S_M, A_M, R_M))$
    if $|D|_{A \neq 0} > 1$ **then**
      $R \leftarrow -1$
    end if
    $D \leftarrow D \cup (e, S, A, R)$
    Update $Q_e(S,A)$:
    if $\min_R D < 0$ **then**
      for each $(e_i, S_i, A_i, R_i) \in D$ **do**
        $Q_{e_i}(S_i,A_i) \leftarrow Q_{e_i}(S_i,A_i) + \alpha[\min_R D + \gamma \max_{a \in \mathcal{A}_{e_i}} Q_{e_i}(S',a) - Q_{e_i}(S_i,A_i)]$
      end for
      $D \leftarrow \emptyset$
    else
      $Q_{e_M}(S_M,A_M) \leftarrow Q_{e_M}(S_M,A_M) + \alpha[R_M + \gamma \max_{a \in \mathcal{A}_{e_M}} Q_{e_M}(S',a) - Q_{e_M}(S_M,A_M)]$
      $D \leftarrow D \setminus (e_M, S_M, A_M, R_M)$
    end if
    $S \leftarrow S'$
  end if
end for

$k_1 = -10$, $k_2 = 0.22$, $k_3 = 0.75$ and $k_4 = 1$. The function was designed to encourage moderation in the changes effected upon the Q-Tables' entries by establishing a point beyond which a large deviation of the go-kart is discouraged. This effectively moderates, on the RL model, the trend towards making large go-kart deviations when they may not be necessary to safely avoid a collision.

c: RL MODES

Two different RL modes (RL_1 and RL_2) were implemented, and compared to a baseline mode (without actuation from RL, just policy training) which we call RL_0. RL_1 and RL_2 modes differ from each other in that the first uses only the trained Q-Tables for Non-Null Events, and does not act when a collision is imminent but the user generates a "Null" Event. This difference in operation modes must be evaluated if we want to test the generalization of the proposed decision pipeline for future control of mobile platforms in the real world, such as indoors wheelchair navigation [30]. In mode RL_2, learning stored in the Q-Tables actuates in the control loop of the go-kart even when the user does not issue a Command. This is a kind of shared-control mode, in which the machine agent momentarily takes control of the go-kart's steering regardless of the user input. This is used to prevent harmful behavior and reduce collisions at the expense of full user control of the go-kart. Next we give an abridged description of how each RL mode is implemented.

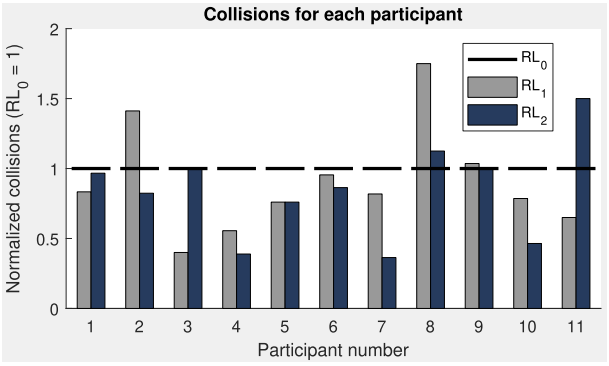


FIGURE 8. Collisions for each participant, normalized to the number of collisions each participant experienced in RL_0 (thick baseline).

- RL_0 EUDN-based decisions are directly converted to game Commands, without intervention by the RL module. RL is, however, active in the sense that it is updating the values of Q-Tables that will be used in the participant's subsequent sessions.
- RL_1 A mode in which RL is used. Events are moderated by RL according to the Event-to-Command mapping from Fig. 7 and each Event's Q-Table is continuously updated. The Q-Table for the case of the "Null" Event is not directly employed, although it is still being trained. This means a direct action to avoid a collision can be taken by the RL model only if the user-based Command is non-Null; this will cause RL to sometimes ignore the mapping in Fig. 7.
- RL_2 A mode in which RL is active and the pre-trained Q-Table for "Null" is used in an attempt to eliminate Null Event-based collisions, using the same Event-to-Command mapping as RL_1. The machine agent is thus proactive in avoiding a collision in case a user does not produce a Non-Null Command to prevent it, i.e. the EUDN's output is "Null". As in this mode the machine agent can temporarily take action without player intervention, in practice it works similarly to shared control.

III. RESULTS

Here the results are presented for the overall classification pipeline integrating the three different experimental modes: RL_0, RL_1 and RL_2, tested in this order for each participant. Each mode's session ends when the kart has passed by 150 boxes, which takes approximately 12m30s – small variations are caused by the use of blinks to accelerate and

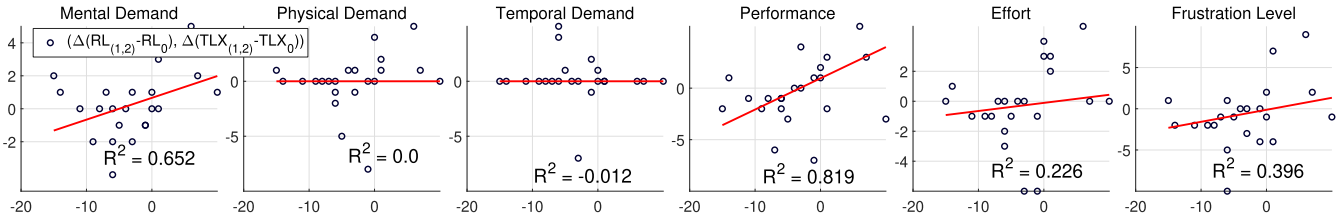


FIGURE 9. Variation of the answers in each TLX item, across all participants, and its relation to the variation of total collisions of each RL mode in relation to RL_0. For each item we did a repeated median estimator regression and calculated its R^2 value, showing low correlation across the board except for the Mental Demand and Performance survey items.

decelerate the kart throughout each session. After a session for each mode, the participant's score and its variation to the previous experimental mode are registered, and the participant is handed a questionnaire with an assessment of the physical system's usability characteristics and the perceived workload for each of the modes played. This workload assessment was carried out using the NASA-Task Load Index (TLX) survey [31]. We also detail the validation procedures followed when training the Convolutional Neural Network.

A. CNN TRAINING AND VALIDATION

For training the CNN, the available dataset was divided into 67% training data and 33% testing data, with samples randomly selected but with a balanced class representation which ensured that the most frequent class could only have three times as many samples as the least frequent. This restriction means 2270 images from the full dataset were used to train the CNN. Higher global training accuracies were attained using different balancing multipliers, but we observed that using a balancing multiplier of 3 allowed for higher precision and recall values of groups of classes that aggregate similar eye movements, which enhances playability in the online sessions. Data, comprised of 9 classes, were subdivided in batches with size 55 and the network was optimized over the course of 50 epochs. The network was trained on a workstation with an Intel 4770K processor, 24GB of memory and a Nvidia 1070Ti graphics unit. This training procedure attained a global accuracy of 87, 73%. A confusion matrix for the cross-validation of the training underwent by the CNN is shown in Table 3.

B. MODAL INFLUENCE ON PERFORMANCE

A means of evaluating the performance of the three modes (RL_0, RL_1 and RL_2) consisted of recording the number of collisions on three sequential sessions. These results are presented in Fig. 8 in normalized form, since the absolute number of collisions varied greatly between participants. Global statistics on collisions and elapsed time (their averages, μ, and their standard deviations, σ) are shown in Table 4. It can be seen that both RL modes result in reductions in the number of collisions, which are more pronounced for the "shared-control" variant. The high standard deviations for collisions can be attributed to variation of the familiarity to the paradigm among participants. The RL methods were, in general, helpful

TABLE 3. Cross-validation results of CNN training with the parameters mentioned in the text (2270 sample images, split 67%-33% between training and testing sets, with a balancing multiplier of 3). Event classes are the same as those described in Table 1. Accuracy results are presented in percentages for each class. Global accuracy is 87, 73%.

		Expected Class								
		1	2	3	4	5	6	7	8	9
Detected Class	1	87.25	1.49	1.49	8.22	11.54	0.88	0.00	7.09	0.00
	2	0.00	92.54	3.73	0.00	0.00	0.00	0.00	0.00	0.00
	3	1.96	4.48	89.55	6.85	0.00	0.88	0.00	0.00	0.00
	4	0.98	0.00	2.99	79.75	0.00	0.00	0.00	0.00	0.00
	5	3.92	0.00	0.00	0.00	71.15	3.54	2.82	0.79	0.00
	6	0.98	0.00	0.00	0.00	9.62	89.38	8.45	0.00	0.00
	7	0.98	0.00	0.00	0.00	0.00	0.88	78.87	0.00	0.00
	8	2.94	0.00	0.75	1.37	0.00	0.00	1.41	77.17	4.26
	9	0.98	0.00	0.00	0.00	0.00	0.00	0.00	9.45	78.72

TABLE 4. Overall statistics on the number of collisions and game duration for each session.

	Collisions per mode			Time elapsed per mode (s)		
Mode	RL_0	RL_1	RL_2	RL_0	RL_1	RL_2
μ	21.18	18.27	17.36	751.73	754.36	753.18
σ	6.54	6.93	8.45	5.71	5.55	3.71

enough to reduce the number of collisions occurring in each session, although there is some discrepancy among participants. Notably, participant 8 experienced far more collisions in modes where RL was active, probably because the low number of collisions was not enough to train a useful RL model for both RL_1 and RL_2. Overall, the increase in the number of collisions when in mode RL_1 can be partially attributed to the same cause, as well as the fact that in this mode the decision process is inherently more conservative and tends to favor small lateral movements instead of the large movements that are frequently needed to avoid a collision.

C. WORKLOAD ASSESSMENT

Results of the workload assessment conducted using the TLX survey – using the six item, 0 to 20 scoring evaluation but without item weighing by the participants – have yielded a reduction of the Mental Demand and Performance indicators when going from session in mode RL_0 to either RL_1 or RL_2. This effect is mixed across the participants sample, but becomes more visible when variations in perceived workload are compared variations in collision numbers, which is done in Fig. 9.

When testing for the correlation between the variation in number of collisions and the variation of demand indicators

using Siegel's repeated median estimator [32], we found a correlation in the Mental Demand and the Performance indicators, which points to more successful steering of the kart during RL-aided session modes. There were no TLX items where a significant negative correlation manifested, with perceived Temporal Demand showing no influence from the type of experimental mode.

IV. DISCUSSION

A. RELEVANCE OF THE CNN IN EOG SIGNAL PROCESSING

Deployment of the CNN for the classification of images of EOG signal windows presents a significant advancement over the methods proposed in [26]. Conversion of EOG signals to RGB images is a new approach that was empirically concluded to be successful in classifying a relatively high number of classes (nine) in a CNN with a generalized training model. This ocular Event classification pipeline is, to the best of our knowledge, the first of its kind implemented, and it is further strengthened by the use of a user-independent model for online classification. This in turn allows us to introduce users to the game without having to subject them to a calibration phase, an important asset for the general deployment of Human-Computer Interfaces. It must be noted, however, that because CNN performance is not being measured during an online trial, the CNN must be carefully trained beforehand to prevent both over-fitting and low global accuracy rates, as the remainder of the implemented decision pipeline relies on a high accuracy rate being achieved by its first classification stage. The EUDN and RL can both moderate the impact of wrongful CNN decisions, but their primary goal is to moderate the impact of correct, but possibly conflicting, CNN decisions that could lead to negative in-game outcomes.

B. REINFORCEMENT LEARNING: USEFUL IN EITHER CASE?

The reduction in the number of collisions on modes RL_1 and RL_2 relative to RL_0 is inconsistent across participants, albeit in most there is a reduction in the number of collisions when going from RL_0 to either RL_1 or RL_2. As there is a high number of possible (*State*, *Action*) pairs that can be present in the RL models, not all will be adequately trained during each game session due to the randomness in obstacle placement, and a first run of 150 obstacles may not be enough for the model to adequately evaluate a significant proportion of the possible states and their outcomes for each participant. This effect, however, does not explain the inconsistent variations in collision numbers between RL_1 and RL_2 for each participant, which may point to the need to fuse the models obtained from several participants to obtain more complete descriptions of (*State*, *Action*) pairs across the Q-Tables.

The influence of RL on perceived effort is considered positive, as the data presented in Fig. 9 shows that the use of an RL model during a game session has a significant correlation to an improvement in the Mental Demand and Performance indicators of the TLX survey. The constancy of the Temporal and Physical Demand indicators can probably be attributed to the trials of all three modes having roughly the same pace.

V. CONCLUSION

The proposed EOG-game classification pipeline has shown empirical evidence of delivering the goals set out for this type of Human-Machine Interface, namely an improved, more playable game interface (based on user's assessment of the system), a user-independent biosignal classification approach that provides a low-frustration interface, and a modifiable, user-based machine assistance model founded on the EUDN decision process and on Reinforcement Learning. A limitation of this pipeline is its reliance on good performance by the CNN, which was not an issue in our case but may be degraded depending on the EOG dataset used[3] and the number of classes defined.

The use of RL has resulted in a small but important improvement in the number of collisions. Its influence in assisting a user in this type of interface will probably be more relevant with longer usage times, which would allow each user-specific action policy to be trained more thoroughly. The *shared-control* Reinforcement Learning mode that was validated can point both to the deployment of EOG-based interfaces onto real-world applications using a similar control paradigm, such as assisted wheelchair navigation [30], or to better EOG-based interaction with HMIs used, for example, for controlling smart homes or assisted living interfaces.

ACKNOWLEDGMENT

The authors would like to thank all participants in the experiments.

REFERENCES

[1] G. Pires, M. Torres, N. Casaleiro, U. Nunes, and M. Castelo-Branco, "Playing Tetris with non-invasive BCI," in *Proc. IEEE 1st Int. Conf. Serious Games Appl. Health (SeGAH)*, Nov. 2011, pp. 1–6.

[2] R. Parafita, G. Pires, U. Nunes, and M. Castelo-Branco, "A spacecraft game controlled with a brain-computer interface using SSVEP with phase tagging," in *Proc. IEEE 2nd Int. Conf. Serious Games Appl. Health (SeGAH)*, May 2013, pp. 1–6.

[3] K. D. O. Andrade, G. Fernandes, G. A. P. Caurin, A. A. G. Siqueira, R. A. F. Romero, and R. D. L. Pereira, "Dynamic player modelling in serious games applied to rehabilitation robotics," in *Proc. Joint Conf. Robot., SBR-LARS Robot. Symp. Robocontrol*, Oct. 2014, pp. 211–216.

[4] M. Vidal, A. Bulling, and H. Gellersen, "Analysing EOG signal features for the discrimination of eye movements with wearable devices," in *Proc. 1st Int. Workshop Pervasive Eye Tracking Mobile Eye-Based Interact.*, Sep. 2011, pp. 15–20.

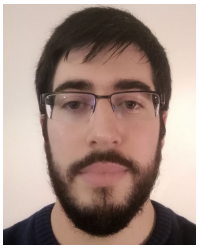
[5] M. R. Kim and G. Yoon, "Control signal from EOG analysis and its application," *World Acad. Sci., Eng. Technol., Int. J. Elect., Electron. Sci. Eng.*, vol. 7, no. 10, pp. 830–834, 2013.

[6] A. Bulling, D. Roggen, and G. Tröster, "EyeMote—Towards context-aware gaming using eye movements recorded from wearable electrooculography," in *Proc. Int. Conf. Fun Games*. Springer, 2008, pp. 33–45.

[7] C.-T. Lin, J.-T. King, P. Bharadwaj, C.-H. Chen, A. Gupta, W. Ding, and M. Prasad, "EOG-based eye movement classification and application on HCI baseball game," *IEEE Access*, vol. 7, pp. 96166–96176, 2019.

[3]The dataset of annotated RGB images of EOG data, and the Python code to train and validate a CNN based on those images, or to load a CNN model when evaluating transformed EOG input, can be found at https://github.com/luisgarrote/EOG_RL_KartGame

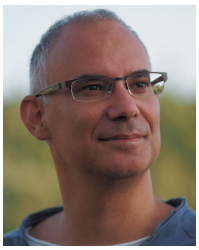
[8] M. Nakanishi and Y. Mitsukura, "Wheelchair control system by using electrooculogram signal processing," in *Proc. 19th Korea-Jpn. Joint Workshop Frontiers Comput. Vis.*, Jan. 2013, pp. 137–142.
[9] L. Y. Deng, C.-L. Hsu, T.-C. Lin, J.-S. Tuan, and S.-M. Chang, "EOG-based human–computer interface system development," *Expert Syst. Appl.*, vol. 37, no. 4, pp. 3337–3343, 2010.
[10] A. B. Usakli and S. Gurkan, "Design of a novel efficient human–computer interface: An electrooculagram based virtual keyboard," *IEEE Trans. Instrum. Meas.*, vol. 59, no. 8, pp. 2099–2108, Aug. 2010.
[11] A. Ubeda, E. Ianez, and J. M. Azorin, "An integrated electrooculography and desktop input bimodal interface to support robotic arm control," *IEEE Trans. Hum.-Mach. Syst.*, vol. 43, no. 3, pp. 338–342, May 2013.
[12] K.-R. Lee, W.-D. Chang, S. Kim, and C.-H. Im, "Real-time 'eye-writing' recognition using electrooculogram," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 25, no. 1, pp. 37–48, Jan. 2017.
[13] S. He and Y. Li, "A single-channel EOG-based speller," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 25, no. 11, pp. 1978–1987, Nov. 2017.
[14] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2011.
[15] K. Tsiakas, M. Dagioglou, V. Karkaletsis, and F. Makedon, "Adaptive robot assisted therapy using interactive reinforcement learning," in *Social Robotics*. Cham, Switzerland: Springer, 2016.
[16] A. Bulling, J. A. Ward, H. Gellersen, and G. Tröster, "Eye movement analysis for activity recognition using electrooculography," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 4, pp. 741–753, Apr. 2011.
[17] C. M. Bishop, *Neural Networks for Pattern Recognition*. London, U.K.: Oxford Univ. Press, 1995.
[18] R. Barea, L. Boquete, J. M. Rodriguez-Ascariz, S. Ortega, and E. López, "Sensory system for implementing a human–computer interface based on electrooculography," *Sensors*, vol. 11, no. 1, pp. 310–328, Dec. 2010.
[19] R. Barea, L. Boquete, M. Mazo, E. L. Guillén, and L. M. Bergasa, "EOG guidance of a weelchair using spiking neural networks," in *Proc. ESANN*, 2000, pp. 233–238.
[20] A. Güven and S. Kara, "Classification of electro-oculogram signals using artificial neural network," *Expert Syst. Appl.*, vol. 31, no. 1, pp. 199–205, Jul. 2006.
[21] Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time series," *Handbook Brain Theory Neural Netw.*, vol. 3361, no. 10, p. 1995, 1995.
[22] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.
[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
[24] X. Zhu, W.-L. Zheng, B.-L. Lu, X. Chen, S. Chen, and C. Wang, "EOG-based drowsiness detection using convolutional neural networks," in *Proc. IJCNN*, 2014, pp. 128–134.
[25] S. Hoppe and A. Bulling, "End-to-end eye movement detection using convolutional neural networks," 2016, *arXiv:1609.02452*. [Online]. Available: http://arxiv.org/abs/1609.02452
[26] J. Perdiz, L. Garrote, G. Pires, and U. J. Nunes, "Measuring the impact of reinforcement learning on an electrooculography-only computer game," in *Proc. IEEE 6th Int. Conf. Serious Games Appl. Health (SeGAH)*, May 2018, pp. 1–8.
[27] J. E. Bresenham, "Algorithm for computer control of a digital plotter," *IBM Syst. J.*, vol. 4, no. 1, pp. 25–30, 1965.
[28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: https://arxiv.org/abs/1412.6980
[29] J. Perdiz, A. Cruz, U. J. Nunes, and G. Pires, "A hybrid brain-computer interface fusing P300 ERP and electrooculography," in *Proc. Medit. Conf. Med. Biol. Eng. Comput.* Cham, Switzerland: Springer, 2019, pp. 1755–1766.
[30] A. Cruz, G. Pires, A. Lopes, C. Carona, and U. J. Nunes, "A self-paced BCI with a collaborative controller for highly reliable wheelchair driving: Experimental tests with physically disabled individuals," *IEEE Trans. Hum.-Mach. Syst.*, vol. 51, no. 2, pp. 109–119, Apr. 2021, doi: 10.1109/thms.2020.3047597.
[31] S. G. Hart and L. E. Stavenland, "Development of NASA-TLX (task load index): Results of empirical and theoretical research," in *Human Mental Workload*, P. A. Hancock and N. Meshkati, Eds. Amsterdam, The Netherlands: Elsevier, 1988, ch. 7, pp. 139–183.
[32] A. F. Siegel, "Robust regression using repeated medians," *Biometrika*, vol. 69, no. 1, pp. 242–244, 1982.



JOÃO PERDIZ (Graduate Student Member, IEEE) was born in Coimbra, Portugal, in 1990. He received the M.Sc. degree in physics engineering from the University of Coimbra, in 2013. He is currently pursuing the Ph.D. degree with the University of Coimbra, with a Project focused on the recognition of human activities using biosignals.

He has published works on recognition of EOG signals for the development of a game interface and the augmentation of EEG functionality of a speller through the use of EOG signals. He has also taken part in testing navigational interfaces that fuse EOG inputs with environmental awareness to train a Reinforcement Learning Model for indoors navigation of a robotic wheelchair.

LUÍS GARROTE (Student Member, IEEE) received the master's degree in electrical and computer engineering from the University of Coimbra, Portugal. He is currently pursuing the Ph.D. degree with the Institute of Systems and Robotics, University of Coimbra, Portugal. His research interests include deep learning, perception, and mobile robotics.



GABRIEL PIRES (Member, IEEE) received the Ph.D. degree in electrical and computer engineering from the University of Coimbra, Portugal, in 2012. He is currently an Adjunct Professor with the Engineering Department, Polytechnic Institute of Tomar. He is also an Integrated Researcher with the Institute of Systems and Robotics, University of Coimbra. He has been responsible and a member of several funded projects in the areas of human–computer interfaces and assistive mobile robotics. His main research interests include EEG-based brain–computer interfaces (BCIs) and biosignal processing, seeking to improve the reliability and usability of BCIs. He is also the Coordinator of the research Laboratory VITA.IPT–Life Assisted by Intelligent Environments.

URBANO J. NUNES (Senior Member, IEEE) received the Ph.D. degree in electrical engineering from the University of Coimbra, Portugal, in 1995. He is currently a Full Professor with the Department of Electrical and Computer Engineering, University of Coimbra (UC), and a Senior Researcher with the Institute for Systems and Robotics. He has been involved with/responsible for several funded projects at both national and international levels in the areas of human-centered mobile robotics and intelligent vehicles. He has been active in the organization of conferences, such as the General Chair of the 2010 IEEE Intelligent Transportation Systems Conference, the General Chair of the 2012 IEEE/RSJ Intelligent Robots and Systems, and the General Chair of the 2017 IEEE International Symposium on Robot and Human Interactive Communication. He is also an Associate Editor of the IEEE TRANSACTIONS ON INTELLIGENT VEHICLES.