Towards Gaze-Controlled Platform Games

Jorge Muñoz, Georgios N. Yannakakis, Fiona Mulvey, Dan Witzner Hansen, German Gutierrez, Araceli Sanchis

***Abstract*—This paper introduces the concept of using gaze as a sole modality for fully controlling player characters of fast-paced action computer games. A user experiment is devised to collect gaze and gameplay data from subjects playing a version of the popular Super Mario Bros platform game. The initial analysis shows that there is a rather limited grid around Mario where the efficient player focuses her attention the most while playing the game. The *useful grid* as we name it, projects the amount of meaningful visual information a designer should use towards creating successful player character controllers with the use of artificial intelligence for a platform game like Super Mario.**

Information about the eyes' position on the screen and the state of the game are utilized as inputs of an artificial neural network, which is trained to approximate which keyboard action is to be performed at each game step. Results yield a prediction accuracy of over 83% on unseen data samples and show promise towards the development of eye-controlled fast-paced platform games. Derived neural network players are intended to be used as assistive technology tools for the digital entertainment of people with motor disabilities.

I. Introduction

The use of alternate modalities (to standard keyboard and mouse) for interacting with computers and game consoles is growing and it becomes beneficial for both the user and the interaction design; typical examples of this trend include the *Wii*[1] console and the PrimeSense[2] sensor technology. On one side, gaze, speech, and video modalities (among others) may engage larger masses of players with dissimilar abilities and game needs. On the other side, games can benefit from alternate modalities of user input since such input can assist in building better gameplay mechanics and realize the development of more believable artificial intelligence (AI) in games.

Eye tracking technology offers information on where and what players are looking at while playing and opens up the aforementioned possibilities for both the player and the game. Such information may reveal hidden knowledge of the appropriate input to the AI that controls player characters and assist in building more accurate and believable game AI grounded on realistic human visual perception of the game. Thus, the process of designing intelligent controllers in games can only benefit from modern low-cost and efficient eye tracking technology. Furthermore, and more importantly, fast-paced action game characters can be fully controlled by people with severe motor disabilities by utilizing eye tracking as alternative means of player character control.

This paper introduces the concept of achieving full control of player characters grounded solely on gaze and in-game data which is built using an artificial neural network (ANN) trained via standard back-propagation. For that purpose, a user study of 12 participants is devised on the popular *Mario Bros* platform game and gaze data, gameplay data and player actions are collected. This initial study focuses on the data of the best-performing player out of the 12 available. The first analysis on the data shows that there exists a small grid area around the player character, Mario, we name *useful grid* within best players are mostly looking while playing the game. This information grid drives the design of ANNs that we train to control the player character, Mario, built on eye tracking input and the current state of the game. The ANNs are assessed via 10-fold cross validation. It is worth noting that we do not consider a direct mapping between the information obtained from the gaze tracking system and the controls of the game. Instead, the paper investigates the underlying function between gaze information, in-game information and human player controls that a machine learner would be able to approximate in order to perform human-like movements of the player character. In other words, we design AI that imitates human player keyboard actions by processing real-time gaze position information of the player.

The paper presents two sets of experiments where the impact of ANN input types and the information grid size on the performance of the ANN is examined. Results obtained show that it is possible to predict unseen keyboard strokes with an accuracy of around 87% based on real-time perception of eye tracked coordinates, level geometry, enemies and objects position, and player character state. Experiments across different information grid sizes show that the information grid incorporating the 90% of the gaze samples, defined by 35 cells around the character, is the most efficient and computationally balanced information with respect to network performance.

The paper structure is as follows: in Section II we provide background for the state of the art on the eye tracking and games. Then in Section III we describe the platform game used for our experimentation followed by a description of the eye tracking system utilized in this paper (see Section IV) and the data collection experiment followed (Section V). In Section VI we go through the data pre-

J. Muñoz, G. Gutierrez, A. Sanchis are with the Computer Science Department, Universidad Carlos III de Madrid, Avda. de la Universidad 30, 28911 Leganés, Spain (emails: { jmfuente, ggutierr, masm }@inf.uc3m.es). G. N. Yannakakis is with the Center for Computer Games Research, IT University of Copenhagen, Rued Langgaards Vej 7 2300 København S, Denmark (email: yannakakis@itu.dk). F. Mulvey and D. Witzner Hansen are with the Innovative Communication Group, IT University of Copenhagen (emails: { fbmu, witzner }@itu.dk)

[1]http://wii.com/

[2]http://www.primesense.com/

978-1-4577-0011-8/11/$26.00 ©2011 IEEE

processing procedure followed and the inputs and outputs of the controller. All experiments held and obtained results are presented in Section VII while conclusions about the experiments (Section IX) and discussion about future steps of this work (Section VIII) conclude this paper.

II. Using Gaze in Games

Eye trackers are devices capable of determining the real-time positioning of the user's eyes on a screen of a human-computer interactive system. Since gaze position is calculated on any 2D plane (usually a computer screen) gaze can also be used for computer interaction tasks and for games [1], [2]. The majority of studies relating gaze and video games investigate the use of gaze directly as an alternative (or supplementary) input modality to the games. The emphasis is on comparing the mouse, keyboard and gaze for game control [3], [4], [5]. For example Dorr et al. [5] show that gaze control has an advantage over mouse control when playing Pong.

In [6] gaze-based interaction is established for specific locomotion, fighting and equipment tasks in *World of Warcraft*; the resulting gaze-controller performs at a beginner player's skill level. On the same basis, Smith and Graham [7] investigate the use of gaze as a interaction modality in dissimilar games; in that study aiming is controlled via gaze in *Quake II* and a fast-paced arcade shooting game.

Alternatively, the impact of gaze interaction on player experience is examined in [8]. According to that study, gaze — when replacing the mouse in a FPS (First Person Shooter) game — increases player immersion but, nevertheless, lowers the efficiency and accuracy of the interaction.

In comparison to other studies, we embed full-control over the player's character by using gaze *indirectly* as a source of implicit and strategic information about the game play rather than as a direct control mechanism. As far as we are aware, this is the first study to take advantage of the implicit and strategic information available in eye movement data. It makes sense that eye movement data would provide useful information to the NN to predict the proper actions, since eye movements represent the strategic, moment to moment gathering of information for the actively processing brain, effected both by the bottom-up properties of the stimulus and the higher order goals of winning the game [9]. Therefore, the eye movement information stream represents a trace of which visual information the brain selected as most informative for the goals of game play, with temporal resolution of over a hundred samples per second. This rich information stream offers many opportunities to model play behaviour. However, gaming and interaction technologies which take advantage of this aspect of eye movements data are absent or only emerging. In this study, gaze pointing is utilized as an alternative input to a machine learner that would be able to accurately control the game character.

Furthermore, the tasks of aiming, locomotion in 3D environments and accessing the inventory reported in the literature are rather simple for a good eye-tracking device to control in real-time. Full gaze-control of a player character in a platform game like Super Mario Bros is a far more challenging task with respect to reaction times and fine-grained movements required of the player [10].

III. Test-bed game: Mario Bros

The game we have selected as a benchmark for our experiments is a version of the popular platform game *Mario Bros* (Fig. 1). In particular, we utilize the *Infinite Mario Bros*[3] version developed by Markus Persson. Infinite Mario Bros allows for procedural generation of non-deterministic game levels based on a random seed and a difficulty parameter corresponding to the number of gaps and enemies in the level. This version of Mario Bros has been already used in the Mario AI competition [4]. This competition provides the state of the art of AI-controlled Mario players; however, none of the Mario players generated for the competition attempts to imitate a human playing or uses alternate modalities of control, so results obtained in this study cannot not be comparable.

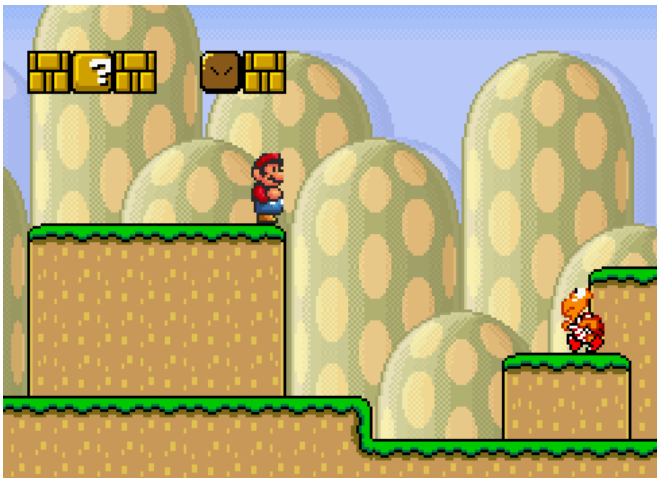

Fig. 1. Screenshot from the Infinite Mario Bros test-bed game.

The gameplay in Mario Bros consists of moving the player-controlled character, Mario, through two-dimensional levels, which are viewed from the side. Mario can walk and run to the right and left, duck, jump, and (depending on which state he is in) shoot fireballs. Gravity acts on Mario, making it necessary to jump over holes (or gaps) to get past them. Mario can be in one of three states: Small (at the beginning of a game), Big (can crush some objects by jumping into them from below), and Fire (can shoot fireballs).

The main goal of each level is to get to the end of the level, which means traversing it from left to right. Auxiliary goals include collecting as many coins as possible, which are scattered around the level, clearing the level as fast as possible, and obtaining the highest score, which in part depends on the number of collected coins and killed enemies.

[3] http://www.mojang.com/notch/mario/
[4] http://www.marioai.org/

The interested reader can find an extended description of the game in [11].

The game screen refresh frequency (game step) is set at 24 Hz corresponding to 24 actions per second (which in turn categorizes Mario Bros as a fast-paced game). Available Mario actions are: move left, move right, run, jump and duck (only when in Big or Fire mode). The player performs the actions by pressing or releasing the keys of the keyboard. Any machine learning mechanism applied should be able to perform the same actions efficiently in real-time.

Mario Bros is picked as the initial test-bed game for the study of indirect gaze control of action game characters due to its popularity as a game and its increasing popularity as an AI benchmark as well as its simplicity with regards to gameplay and mechanics. As already mentioned, the task of imitating human playing behavior in a fast-paced action game, which requires 24 actions per second, using gaze input, is rather challenging; any additional complication forced into the machine learner via a more complex 3D game would be inappropriate at this initial stage.

IV. Gaze Tracking System

The *EyeFollower*[5] gaze tracker is used for the experiments presented in this paper. The EyeFollower is a remote gaze tracker with two infrared cameras mounted on a pan-tilt head, mounted under the monitor, and a webcam on top of the monitor to track head position (see Fig. 2). The device is able to locate the PoR (point of regard) on the monitor, and allows for totally free head movement without the need to wear headgear. The eye tracker detects the pupil and the reflections of IR (infra red) light sources on the cornea to be able to determine the PoR. Gaze trackers need to be calibrated to each individual prior to use by having the user look at a set of on-screen targets. Calibration is needed to be able to infer person-specific parameters. Without calibration the PoR estimates could vary several degrees.

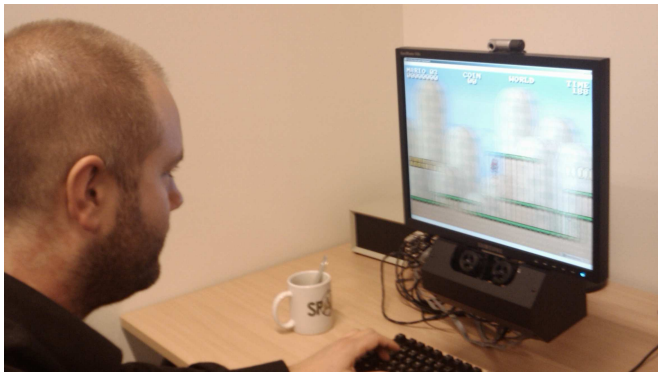

Fig. 2. A subject playing Super Mario Bros.

The EyeFollower is highly accurate, able to sample gaze position and the pupil size at a rate of 120 samples per second (60 Hz per eye). This is equivalent to approximately 5 samples per game tick/step. Note that the game sampling frequency (24 Hz) is five times lower than that of the eye-tracker system (120 Hz); this indicates the need for data pre-processing prior to training the controller on data from both modalities.

[5]developed by *LC Technologies, Inc.* — http://www.eyegaze.com/

V. Data Collection Experiment

This section presents the experimental protocol used for user data collection. Twelve subjects participated in the experiment reported in this study. Subjects start the experiment by filling in an on-line demographic questionnaire. Information is asked about age, gender, potential vision problems as well as game skill questions relating to computer games in general and specifically to Mario Bros. Then each subject plays a set of 6 games of different difficulty levels; the first three games are played in increasing difficulty order whereas the remaining three games are played in all possible difficulty order permutations. Given the proposed experimental protocol twelve subjects cover all possible permutations of three games (i.e. 6 permutations) twice.

The reason for adopting such a two-phase experiment scheme is both to train all players in the same (increasing difficulty) first phase gameplay scenario and, during the second phase, to minimize order effects with respect to learnability and gameplay difficulty. Gaze 2D coordinates and pupil size are stored at a frequency of 60 Hz per each eye reaching the maximum sampling rate (120 Hz) of the eye tracker used.

Each person has its own vision characteristics and properties which make the eyes behave dissimilarly under the same environment conditions [1]. These dissimilarities increase the level of noise perceived through the eye tracker hardware. Although the gaze system used is very robust with respect to hardware noise reduction, there is still data noise caused or data lost due to hardware setup, environmental causes and real-time vision particularities. Usual reasons for missing data include eye blinking and various forms of light reflection which prevent the eye tracker system to retrieve gaze data. In addition, data is lost due to the fact that the game and the eye gaze system run under the same CPU. Resulting gaps in the data streams obtained vary from 200 milliseconds to maximum 1 second and accumulatively amount to less than 1% of the total gaze position samples. We consider this data loss percentage rather insignificant. Since any form of data interpolation is not appropriate for gaze data, those small sections of corresponding gameplay data are not considered for further investigation.

In addition to missing data there are also valid data samples that do not provide any useful information. For instance, it is well known that there is no information processing by the human brain during a saccade (movement between two fixations) [1]. In a fast-paced game like Mario Bros players use their gaze mid-term or long-term planning of future actions to be performed. In such occasions the player is fixing her gaze at a particular point on the screen while performing actions to a far distant point; this gaze behavior happens often in Mario Bros players since real-time event prediction is crucial for playing the game successfully. Such gaze-data

particularities challenge any machine learning approach that attempts to imitate human playing behavior based on human visual perception.

Delays in gaze sampling may reach 25 milliseconds and occur due to 1) processing time of the eye image taken and 2) the use of sockets for enabling data communication between the game and the eye tracker. Although these delays are important in real time applications, we have to bear in mind that delays also happen in the brain, and also, that the game-step (≈ 40 milliseconds) is greater than the delay.

VI. Imitating Mario via Gaze

As previously mentioned, our desire is for users to be able to control the Mario character solely by using information retrieved from the eye tracking system and in-game data. On that basis, the behavior of Mario needs to be similar to a human playing behavior and the Mario controller needs to encapsulate the actions a human would perform in certain situations. Imitating Mario players on the grounds on eye position information is a rather complex task. We believe that the underlying function between eye positioning, in-game state information and keyboard actions could be approximated via the use of an ANN trained on data retrieved from human players. There are other supervised learning algorithms that one could attempt; however, ANN is a good initial match for this problem due to their good real time performance and their universal approximation capability [12], [13].

A. Data pre-processing

The first step we have followed towards achieving this goal is to retrieve a good set of human playing patterns to train the ANN which is obtained via data pre-processing. The first and last 4 recorded seconds of each level are removed to minimize noisy data retrieved while the screen fades in and out from a black screen to the level. In addition, levels with very few number of samples are not taken into account. The samples when the player looks out of the game are not considered either.

From the 12 subjects that played the game, we eliminate players with significant amounts of missing data due to malfunction of the eye tracker, and players showcasing poor playing performance. Thus, data retrieved from the best performing player that depicts no significant eye position data loss are used for the experiments presented in this initial study. Player performance is assessed via the total number of times the player died, hurts while on Fire or Large mode and enemies killed in all 6 levels played. The player picked as *best* died and was hurt 3 and 9 times, respectively (both numbers being the minimum among all players), and killed the maximum amount of enemies (45). In addition to space considerations the reason for selecting this particular player in this initial analysis is obvious: we feel it is very important to report on the type and amount of visual information retrieved when a *hardcore* gamer plays Mario Bros levels since both the game AI and the gaze tracking studies can benefit from such knowledge. Furthermore, we decided to use the first four levels of the best performing player since the last two levels played were way too complex for the player and eye position data are lost more often in those two levels due to the tiredness of the player.

Real-time eye positioning when playing games is of vital importance for a player controller since it may determine, up to a good degree, the amount and type of information necessary to control the player character efficiently. For this purpose we discretize the screen in cells of the same size of Mario and analyze the times a human would look at those cells. Fig. 3 presents the game cells, relative to Mario, the best player looks at when playing levels of varying difficulty. Results show that such a player focuses in the vast majority of gameplay time in a small grid around Mario which we name *useful grid*. A hill-climbing search algorithm — the cell around the current selected cells with more samples is selected next — yields 22, 27 and 36 cells around Mario that cover more than 80%, 85% and 90% of the samples respectively. Eye position relative to Mario suggests the existence of important information in those cells that an imitation algorithm can exploit in the process of learning to play Mario. Thus it is our intention to use information about enemies, the environment (gaps and blocks) and objects (mushrooms, coins, flowers) represented as information grids relative to Mario [14] within the *useful grid* as depicted in Fig. 3.

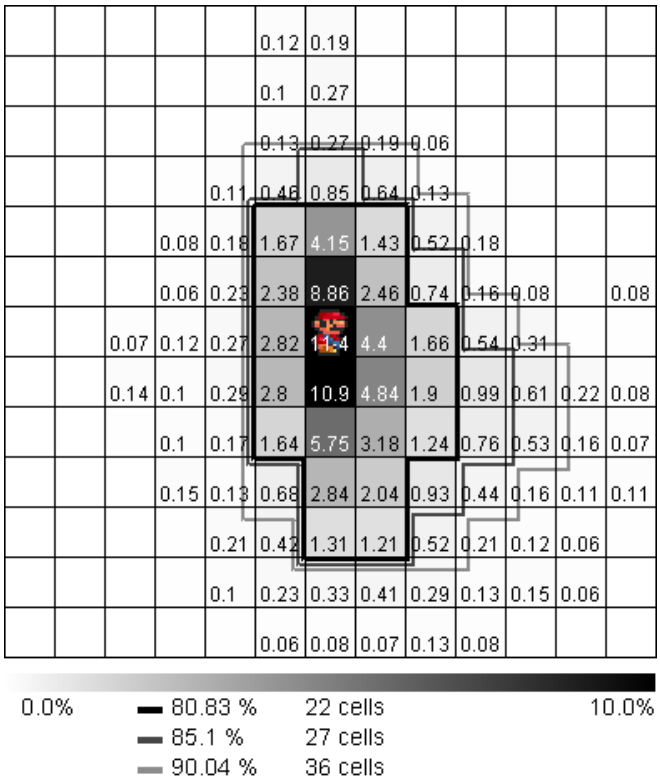

Fig. 3. Grid showing the percentage of gaze samples in each cell relative to the main character, Mario. The cell's size equals Mario's size (16×16 pixels). Regions containing the 80%, 85% and 90% of the gaze samples are also depicted.

B. Inputs and Outputs of the ANN

This section discusses the input and output representation of the ANN considered in this study. In particular, a multi-layer perceptron employing sigmoid (logistid) activation functions at each neuron is trained via back-propagation as discussed later in Section VII. The outputs of the ANN are defined as the actions that Mario can take and are as follows:

- Move to the left
- Move to the right
- Run (when Mario is in fire mode this action also fires a fireball)
- Jump

The duck action, that can be only executed when Mario is in big mode, has been removed from the outputs of the network since it is rarely used from the Super Mario players of our study. In particular, the best player we examine here has never used the duck action.

An action is executed when the output of the ANN for this action is above the threshold 0.5; note that the outputs fire a value between 0 and 1 (neurons employ the logistic activation function).

On the other hand, the inputs of the ANN should somehow relate to the gaze information and the current state of the game; the remaining of this section discusses possible inputs that an ANN controller could use in order to predict accurately human playing actions (keystrokes).

One class of possible inputs for the ANN is the cell where the player is looking at with respect to Mario consisting of the x and y coordinates of the point. Note that if the eye samples are not valid it is assumed that the player is looking at Mario. Other four possible ANN inputs include the type of information existent in the point that the player is looking at: there could be an enemy, the type of the enemy (whether it can be killed via stomping on it or not) or a useful object.

The current state of Mario could also be considered by the ANN. In addition to the 3 states Mario could be (small, big and fire), we also consider a state if Mario is up in the air (due to a previously executed jump), and an additional state if Mario is carrying a shell resulting to five possible Mario states (5 possible inputs for the ANN).

Landscape information appears to be a vital input for the ANN since it would allow the ANN to take an action decision based on the current game landscape (e.g. gaps, hills etc.). Landscape information is given to the ANN as a binary representation of grid cells relative to Mario's position; each cell represents a binary value which equals 1 if there is ground on the corresponding cell or 0 otherwise. The number of inputs is dependent on the size of the grid a designer considers. Enemies may form another information grid showing the existence of enemies with respect to the position of Mario. Each cell contains the value of 1 if there is one or more enemies, or 0 otherwise. Likewise, objects that appear on the screen can be represented in a similar fashion: the value of 1 in a cell represents the existence of an object on that particular cell. Objects can be one of the following: coins, mushrooms, flowers and blocks which contain objects that have to be hit by Mario to release the object.

VII. Experiments

This section summarizes the main results obtained from our experimentation attempts. We run two types of experiments. In the first set of experiments we use different types of ANN input derived from a constant grid around Mario whereas in the second set of experiments we explore the impact of the size and shape of the grid on prediction accuracy.

For this study we use fully-connected feed-forward ANN [12] trained through the standard backpropagation algorithm. There are a total of 6902 data patterns for training the ANN derived from the best-performing player. For all the experiments presented here we split the training data in 10 folds and use 10 fold cross-validation to assess the performance of the trained ANNs.

The folds were created dividing the data into groups of 24 consecutive samples (1 second of game play), shuffling the data and grouping it into 10 folds with same number of samples. The groups of 24 consecutive samples were made in order to not split the actions of the player. Since the game state and the action of the human player do not almost change between two consecutive game steps, we keep consecutive samples of the game in the same fold instead of using consecutive game samples for both train and test sets. We chose one second as a reasonable number of samples for the initial groups; experiments with smaller and larger time windows led to similar results.

The number of epochs each ANN is trained on equals 1000 with a constant learning rate of 0.001. The ANN have one hidden layer with 10 neurons. The selection of these parameters is based on experiments held for maximizing the performance of the ANNs.

A. Experiments across different ANN input vectors

For the first set of experiments we use a rectangular grid based on the *useful grid* information depicted in Fig. 3. and investigate the impact of dissimilar types of input to the performance of the ANNs. The grid chosen is of size 6×7 cells with Mario in the middle as illustrated in Fig. 4 covering the vast majority (86.1%) of gaze samples (see Fig. 3).

Table II shows the input sets used for each experiment held with this information grid. The first column, *Experiment*, provides an encoded name for the experiment and the last

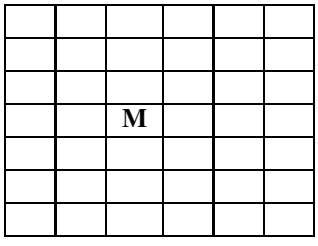

Fig. 4. The 6×7 grid used; **M** represents Mario.

column, *Inputs*, represents the number of ANN inputs. The remaining columns of the table represent the various classes of ANN inputs used: *Eyes* are the x and y coordinates of the cell the player is looking at relative to Mario; *Grid* represents the information grids for *Landscape*, *Enemies* and *Objects* containing 41 inputs each (the cell where Mario is, is not considered); *Object at view* is information relative to what the player is looking at (4 inputs – see Section VI-B); and, finally, *Mario State* represents the five possible state of Mario.

Table III shows the results of the experiments with the inputs described in Table II. The table presents 10-fold cross-validation accuracy for each output of the ANN and on average (*Average* column) for all outputs. The last column, *Std. Dev.*, depicts the standard deviation values of the average performances. Bottom rows of the Table III are a default behavior called *always right* where only the Right action is executed all the time and a *random behavior*, both included for comparison purposes.

By observing the performances of the different ANNs attempted the left and run action are predicted correctly around 90% and 93% of the time respectively also in the *always right* behavior. On the other hand, the right action is predicted correctly between 63% and 67% of the time and the jump action is predicted correctly between 75% and 80% of the time. It is also apparent that the two actions with the highest variation in the prediction accuracy are the right and jump actions.

A corrected resampled t-test [15] over each of the outputs in the experiments of the Table II shows that there is not a significant difference among the experiments for the left and run actions. However, there is statistical significance for the right action between the experiments that uses the eyes input and the complete grid of the landscape, objects and enemies (*e_leo*, *e_leo_l*, *e_leo_m*, *e_leo_l_m*) and the remaining experiments (*e*, *e_l*, *e_e*, *e_o*, *leo_m*). For the jump action there is also a statistical significance between the experiments whichs uses the mario state as input (*e_leo_m*, *e_leo_l_m*, *leo_m*) and the remaining experiments.

Table I presents the percentage of time each available Mario action is taken for the best-performing player. As it can be seen, left and run actions are fired less than 10% of the time.

The low frequency of the left and run actions make them very well predicted by ANNs, with accuracy of more than 90%. On the other hand, the jump action is predicted well (accuracy is significantly higher than 75%) when the ANN's input vector includes the state of Mario. For the right action, it is worth noticing that ANNs incorporating the eye coordinates and the three grids (landscape, enemies and objects) as inputs reach accuracy values higher than 66%. These results indicate both the difficulty in predicting the right and jump actions (which are the most frequent in Mario Bros) but also that the existence of particular types of ANN perception, such as eye tracked coordinates and level landscape which are crucial for the successful prediction of player actions. Overall, the best average ANN performance (82.63%) is obtained when all the inputs are included (experiment code: *e_leo_l_m_134*).

TABLE I
BEST-PERFORMING HUMAN: PERCENTAGE OF TIME EACH ACTION IS EXECUTED.

Left	Right	Jump	Run
9.14	49.45	24.73	6.71

B. Experiments across different information grid sizes

The second set of experiments presented in this paper [6] investigates the impact of the information grid size to the performance of the ANNs. Table IV shows the performance of the ANN containing all inputs, across different grid sizes. In particular, we utilize a 6×7, a 4×5 and a 8×9 grid with Mario placed in the middle of the grid as depicted in Fig. 4, as well as the 80%, the 85% and the 90% *useful grids* depicted in Fig. 3.

While large performance differences among the different grids are not observed, there appears to be a slight improvement in some ANNs for the right action. The only statistically significancet difference can be found in right action between the 4×5 grid and the other grids. A grid bigger than 4×5 seems to be neccesary to achieve a good perforance for the right output.

C. Contingency Table

Table V presents the contingency table of the best ANN (experiment codename *e_leo_l_m* with the 6×7 grid). The table shows the number of correctly and wrongly classified samples for each action. In particular there are four possibilities:

- *True Positive*: the action is executed and the ANN predicts it.
- *True Negative*: the action is not executed and the ANN predicts it.
- *False Positive*: the action is not executed but the ANN predicts it is executed.
- *False Negative*: the action is executed and the ANN predicts it is not executed.

Table V also shows the *precision* value (rate of correct positive predictions among all times the ANN predicts the action is fired) and the *recall* value (rate of correct postive predictions among all times the action is fired).

The accuracy of *Left* and *Run* actions is more than a 90% but their recall values are small (high number of false negative samples). There are so many samples where the action is executed but the ANN does not predict it. The ANN does not learn it and almost always predicts that the action is not executed, most likely because there are very few samples for training the ANN where the action is executed compared with the total number of samples (see Table I).

[6] A different fold set were used in these experiments, therefore the value of the 6×7 grid experiment is slightly different in Table III and IV

TABLE II

ANN INPUT SETS USED IN OUR EXPERIMENTS A 6×7 INFORMATION GRID IS USED FOR THE EXPERIMENTS PRESENTED HERE.

Experiment	Eyes	Grid			Player Looking	Mario State	Number of Inputs
		Landscape	Enemies	Objects			
e_2	x						2
e_l_43	x	x					43
e_e_43	x		x				43
e_o_43	x			x			43
e_leo_125	x	x	x	x			125
e_leo_l_129	x	x	x	x	x		129
e_leo_m_130	x	x	x	x		x	130
e_leo_l_m_134	x	x	x	x	x	x	134
leo_m_123		x	x	x			123

TABLE III

ANN ACCURACY (%) OVER EACH POSSIBLE MARIO ACTION AND ON AVERAGE. A 6×7 INFORMATION GRID IS USED FOR ALL EXPERIMENTS PRESENTED HERE. VALUES IN BOLD INDICATE THE HIGHEST ACCURACY OBTAINED FOR THE PARTICULAR ACTION OR ON AVERAGE. BOTTOM ROWS ARE THE DEFAULT BEHAVIOR OF ALWAYS MOVING RIGHT AND A RANDOM BEHAVIOR.

Experiment	Left	Right	Jump	Run	Average	Std. Dev.
e_2	90.41	60.40	75.54	93.13	79.87	1.33
e_l_43	90.49	63.48	75.38	93.13	80.66	1.97
e_e_43	90.41	62.52	75.55	93.13	80.40	1.35
e_o_43	90.41	62.65	75.55	93.13	80.43	1.46
e_leo_125	90.80	66.96	75.27	93.02	81.51	1.69
e_leo_l_129	90.36	66.28	75.67	93.13	81.36	1.81
e_leo_m_130	90.37	66.52	**79.92**	92.99	82.45	2.20
e_leo_l_m_134	90.66	**67.58**	79.27	93.00	**82.63**	2.15
leo_m_123	90.57	64.64	79.81	92.91	81.98	2.27
always right	90.86	49.45	75.27	93.29	77.22	-
random	49.23	48.28	50.12	49.91	49.39	-

TABLE IV

ANN ACCURACY (%) ACROSS DIFFERENT GRID SIZES; ANN INPUT CONSISTS OF ALL INPUTS CONSIDERED (EXPERIMENT CODENAME *e_leo_l_m*). VALUES IN BOLD INDICATE THE HIGHEST ACCURACY OBTAINED FOR THE PARTICULAR ACTION OR ON AVERAGE.

Grid	ANN Inputs	Left	Right	Jump	Run	Average	Std. Dev.
4×5	68	90.76	65.78	80.66	93.23	82.61	1.36
80%	74	**91.05**	66.26	80.98	93.21	82.87	1.27
85%	89	91.02	68.31	80.47	93.19	83.25	1.61
90%	116	90.80	**68.36**	80.63	93.02	83.20	1.58
6×7	134	91.02	68.14	80.80	93.15	**83.28**	1.28
8×9	241	90.81	66.26	**81.08**	93.19	82.83	1.49

TABLE V

CONTINGENCY TABLE FOR THE BEST ANN (EXPERIMENT CODENAME *e_leo_l_m* WITH THE 6 × 7 GRID, SAME AS TABLE IV). THE CELLS SHOW THE NUMBER OF TRUE POSITIVE (TP), TRUE NEGATIVE (TN), FALSE POSITIVE (FP) AND FALSE NEGATIVE (FN) CLASSIFICACIONS OF THE ANN; AND THE VALUES OF PRECISSION ($\frac{TP}{TP+FP}$), RECALL ($\frac{TP}{TP+FN}$) AND ACCURACY ($\frac{(TP+TN)\cdot 100}{TP+TN+FP+FN}$).

Action	Correct Predictions		Wrong Predictions		Precision	Recall	Accuracy (%)
	TP	TN	FP	FN			
Left	31	6251	10	610	0.7561	0.0484	91.02
Right	2335	2368	1141	1058	0.6717	0.6882	68.14
Jump	666	4911	322	1003	0.6741	0.3990	80.80
Run	6	6423	15	458	0.2857	0.0129	93.15

The accuracy of *Right* and *Jump* actions is lower than the accuracy for the other two actions but their recall and precision values are closer to the best value. Unlike the *Left* and the *Run* actions where the ANN learnt the actions that are not executed, for the *Right* and the *Jump* action the ANN extracted some information and is able to generalize and predict if the action is executed or not with an accuracy of 68.14% and 80.8%, respectively. The samples are more balanced for these actions (see Table I).

VIII. Discussion and future work

This study introduces the concept of fully gaze-controlled games realizing playability of fast-paced platform games for people with severe motor disabilities. Evidently the gameplay experience alters for the players that are able to use the keyboard and the mouse [16], [8]; nevertheless using eye tracking as an alternative control modality introduces new possibilities for game design.

The first obvious step for improving the generality of our findings is to consider more player data which will allow us to examine different gameplay styles and their affect to the training of the ANN; but also to investigate differences in eye movement patterns among the differen players. By observing the way different people play Mario Bros and look at different elements of the game we expect that a number of dissimilar Mario controllers will be required. Thus prior to allocating a controller for a specific player, a player modeler (e.g. via emergent self-organizing maps [17]) could be trained to accurately classify the player. Player modeling under this study constitutes an important research step per se: that is, to be able to recognize playing behavior and skill based solely on gaze data.

Furthermore, we plan to increase the number of training samples the ANN is trained on so that the samples are, if possible, evenly distributed among different actions. This will guide the ANN training to treat all possible Mario actions in a fair manner and learn to execute them correctly.

Another obvious step to take is the use of recurrent (instead of feed-forward) ANNs for imitating human play thereby exploiting the ability of recurrent ANN in learning spatio-temporal relationships which appear to be significant in our case study. Moreover, one may attempt to learn combinations of key presses rather than the key press per se. Doing so will decrease the sample size and might ease the learning task.

IX. Conclusions

This initial study towards building gaze-controlled platform game characters revealed a number of important conclusions for future research. First, it appears that there is an vital grid area around the player character that the player looks at most of the time which furthermore assists in imitating human playing style. Thus, there is a consistency between where players are looking at when playing a game and the information an AI controller would require to learn to play that game. Second, it appears that both real-time eye coordinates and gameplay data are beneficial as ANN inputs for the training of the player controllers. Third, ANNs are performing well on the task of imitating a well-playing human player grounded on both eye information an gameplay data. Results reveal a 83.3% accuracy on unseen data which provides evidence for the appropriateness of ANNs as predictors of human playing actions.

References

[1] A. Duchowski, *Eye tracking methodology: Theory and practice*. Springer-Verlag New York Inc, 2007.

[2] D. W. Hansen and Q. Ji, "In the eye of the beholder: A survey of models for eyes and gaze," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 99, no. 1, to appear.

[3] P. Isokoski and B. Martin, "Eye tracker input in first person shooter games," in *Proceedings of the 2nd Conference on Communication by Gaze Interaction; COGAIN Conference*, 2006, pp. 76–79.

[4] P. Isokoski, A. Hyrskykari, S. Kotkaluoto, and B. Martin, "Gamepad and eye tracker input in FPS games; data for the first 50 minutes," in *Proceedings of the 3rd Conference on Communication by Gaze Interaction; COGAIN Conference*, 2007, pp. 11–15.

[5] M. Dorr, M. Böhme, T. Martinetz, and E. Barth, "Gaze beats mouse: a case study," in *Proceedings of the 3rd Conference on Communication by Gaze Interaction; COGAIN 2007*, 2007, pp. 16–19.

[6] H. Istance, A. Hyrskykari, S. Vickers, and T. Chaves, "For Your Eyes Only: Controlling 3D Online Games by Eye-Gaze," in *Proceedings of the 12th IFIP TC 13 International Conference on Human-Computer Interaction: Part I*. Springer, 2009, p. 327.

[7] J. Smith and T. Graham, "Use of eye movements for video game control," in *Proceedings of the 2006 ACM SIGCHI international conference on Advances in computer entertainment technology*. ACM, 2006, p. 20.

[8] L. Nacke, S. Stellmach, D. Sasse, and C. Lindley, "Gameplay experience in a gaze interaction game," in *The 5th Conference on Communication by Gaze Interaction–COGAIN 2009: Gaze Interaction For Those Who Want It Most*. The COGAIN Association, 2009.

[9] J. Findlay and I. Gilchrist, *Active vision: The psychology of looking and seeing*. Oxford University Press New York, 2003.

[10] P. Isokoski, M. Joos, O. Spakov, and B. Martin, "Gaze controlled games," *Universal Access in the Information Society*, pp. 323–337, 2009.

[11] C. Pedersen, J. Togelius, and G. N. Yannakakis, "Modeling Player Experience in Super Mario Bros," in *Proceedings of the IEEE Symposium on Computational Intelligence and Games*. Milan, Italy: IEEE, September 2009, pp. 132–139.

[12] S. Haykin, *Neural networks: a comprehensive foundation*. Prentice Hall, 2008.

[13] J. Muñoz, G. Gutierrez, and A. Sanchis, "Controller for torcs created by imitation," in *IEEE Symposium on Computational Intelligence and Games.*, September 2009, pp. 271–278.

[14] J. Togelius, S. Karakovskiy, J. Koutnık, and J. Schmidhuber, "Super Mario Evolution," in *IEEE Symposium on Computational Intelligence and Games.*, September 2009.

[15] R. Bouckaert and E. Frank, "Evaluating the replicability of significance tests for comparing learning algorithms," *Advances in Knowledge Discovery and Data Mining*, pp. 3–12, 2004.

[16] T. Gowases, R. Bednarik, and M. Tukainen, "Gaze vs. Mouse in Games: The Effects on User Experience," in *Proceedings of the International Conference on Advanced Learning Technologies, Open Contents and Standards (ICCE)*, 2008, pp. 773–777.

[17] A. C. A. Drachen and G. N. Yannakakis, "Player Modeling using Self-Organization in Tomb Raider: Underworld," in *Proceedings of the IEEE Symposium on Computational Intelligence and Games*. Milan, Italy: IEEE, September 2009, pp. 1–8.