View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Biblioteca Digital da Produção Intelectual da Universidade de São Paulo (BDPI/USP)

Universidade de São Paulo

Biblioteca Digital da Produção Intelectual - BDPI

Departamento de Engenharia Mecânica - EESC/SEM

Comunicações em Eventos - ICMC/SCC

2014-10

Dynamic player modelling in serious games applied to rehabilitation robotics

Joint Conference on Robotics and Intelligent Systems; Brazilian Robotics Symposium, 2th; Latin American Robotics Symposium, 11th; Workshop on Applied Robotics and Automation, 6th, 2014, São Carlos.
http://www.producao.usp.br/handle/BDPI/48643

Downloaded from: Biblioteca Digital da Produção Intelectual - BDPI, Universidade de São Paulo

Dynamic Player Modelling in Serious Games applied to Rehabilitation Robotics

Kleber de O. Andrade[1], Guilherme Fernandes[2], Glauco A. P. Caurin[1], Adriano A. G. Siqueira[1], Roseli A. F. Romero[3] and Rogerio de L. Pereira[4]

***Abstract*— This article proposes a reinforcement learning approach to dynamically model the player skills in applications that integrate games and rehabilitation robotic. The approach aims to match the game difficulty to the player skills, keeping proper motivation (flow) during a rehabilitation process. The traditional rehabilitation process involves repetitive exercises. Robots and serious games provide new means to improve user motivation and commitment during treatment. Each person shows different skills when facing the challenges posed by computer games. Thus, the game difficulty level should be adjusted to each player skill level. The Q-Learning algorithm was adapted in this context to modify game parameters and to assess user skills based on a performance function. This function provides a path to an individual difficulty adjustment and consequently a tool to keep the user exercising. Experiments with thirty minutes duration are presented, involving four players, and the results obtained indicate the proposed approach is feasible for modeling the user behaviour getting to capture the adaptations and trends for each player according to the game difficulties.**

I. INTRODUCTION

A worldwide concern and reality, the population aging is directly correlated with the increase in the number of post-stroke patients that will need some form of motor rehabilitation. However, conventional rehabilitation constitutes a labor intensive, tedious and boring process, from the patient's perspective [1].

The integration of rehabilitation robots with serious games [2] brings state-of-the-art instrumentation technology to measure real-time values relative to the patient performance (range of motion, speed, strength), and even actively interacts in the process. The obtained information is more accurate and deterministic, allowing a better comparison to the established parameters that are used to evaluate the patient progress, replacing reasoned but subjective opinion of the professional. Combined, robots and games are efficient in delivering routine therapy activities and storing patient records that improve and simplify analysis and diagnosis. Nevertheless this integration is not a trivial task.

According to the flow theory [3], a game difficulty level should balance the proposed challenges with the users skills in order to keep the motivation level high. This definition assumes that the system is able to measure the player's abilities, ie, it makes explicit the importance of modeling the player. In this study, we used a strategy of adjustment difficulty of a game to model the behavior of the player. On the other hand, if an extremely hard game impose demands beyond the users skills, it will be a frustrating experience for the user [4]. Usually, in commercial computer games, the players are able to adjust the difficulty level, statically.

In section V a proposal for a flexible difficulty adjustment is presented in together with a proposal for expanding the concept of adaptive robot therapy. The approach is based on reinforcement learning concepts implemented with a Q-Learning algorithm. This paper presents the implementation of adaptive games as key feature towards the creation of a motivating therapy environment from the user perspective. Experiments with 4 volunteers were carried out using a robotic device described in the next section.

II. THE ROBOTIC REHABILITATION DEVICE

The robot was designed allowing the implementation of active and passive therapy. A therapy is classified active when only the patient is responsible for the efforts that results movements, i.e. the work performed by the robot is null ($WR = 0$). Passive therapy occurs when both robot and patient produce efforts that results in movements; in this case, the robot produces work ($WR \neq 0$). A more detailed description of the wrist rehabilitation robot prototype (left top corner of Figure 1) can be found in found in [5] and [6].

The device is designed for therapy exercises with a single degree of freedom at a time: flexion/extension; adduction/abduction; pronation/supination.

This mechanical characteristic simplifies the robot structure, weight and increases its reliability. Mechanical setups are responsible for the choice of each wrist motion. This simplification is supported by the evidence that simultaneous therapy exercises with more than one degree of freedom do not provide significant improvements when compared to single movement exercises [7].

The mechanical structure of the device is composed by two aluminum links connected to a rotatory joint. One link is connected to a bracket[1]. During the therapy sessions, the

*This work was supported in part by CAPES process DS 00011/07-0.

[1]Kleber de O. Andrade, Glauco A. P. Caurin and Adriano A. G. Siqueira are with the Department of Mechanical Engineering, EESC, University of São Paulo at São Carlos, São Carlos, SP, Brazil `pdjkleber@sc.usp.br, gcaurin@sc.usp.br, siqueira@sc.usp.br`

[2]Guilherme Fernandes. Toradex - AG, Switzerland `guilherme.fernandes@toradex.com`

[3]Roseli A. F. Romero is with Computer Science Department, ICMC, University of Sao Paulo, São Carlos, SP, Brazil `rafrance@icmc.usp.br`

[4]Rogerio de L. Pereira is with the Department of Computer Science, State University of Maringá, Maringá, PR, Brazil `rlpereira@uem.br`

[1]Commercial Bracing Axilo Palmar Salvape® Lite Supp.

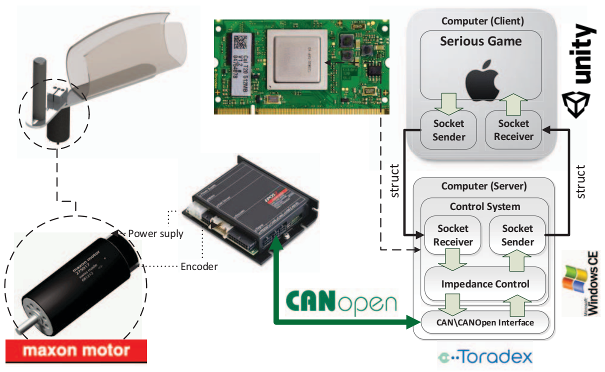

Fig. 1. Overview of the system.

bracket is attached to the user forearm. The second link carries the input handle. The rotatory axis is directly coupled to a DC servomotor without gearing providing the system with backdriveability. The servomotor has an integrated 2000 ppt encoder. Low-level control is implemented using a maxon EPOS 24/5 control driver. A CANOpen[2] network links the information exchange between servomotor control driver and the communication layer.

The communication layer, between the game and the robot, is implemented using TCP/IP sockets (Transmission Control Protocol). The server runs on a Toradex Colibri T20 computer with an ARM Cortex A9 processor and Windows Embedded Compact 7 RTOS. It acts as a middleware [3] monitoring the changes at the robot joints and reporting them to the client (game) at an update rate of $200Hz$. The client (MAC mini 2.3GHz-quad core with 4GB, 1.6MHz SDRAM DDR3 and an Intel HD Graphics 4000 graphics card) renders model images at $60Hz$. Data transmission is implemented serializing a structure that encapsulates the robot state variables. The information is shared by the robot and the game.

Conceptually, the device is simple and portable. It is a prototype aiming the development of future home therapy devices.

III. THE REHABILITATION GAME

We extend the definition of serious games and have applied it to healthcare. It this context serious game is defined as: “a mental play performed in a computer, according to specific rules that uses entertainment as a form of achieving rehabilitation goals”. According to our definition, serious games also incorporate entertainment aspects and combine them with the clinical objectives.

Figure 2 displays a screenshot of the game developed here. The game is called “Nuts Catcher” and was developed and implemented in 3D virtual environment[4]. It is presented to the user in a side view perspective. The player controls the movements of a squirrel that walks on the horizontal axis (right and left). The squirrel collects nuts falling down from the trees. The game was developed considering a robot attached to the user wrist as input device. The system was conceived to accept additional inputs from commercial video game input devices.

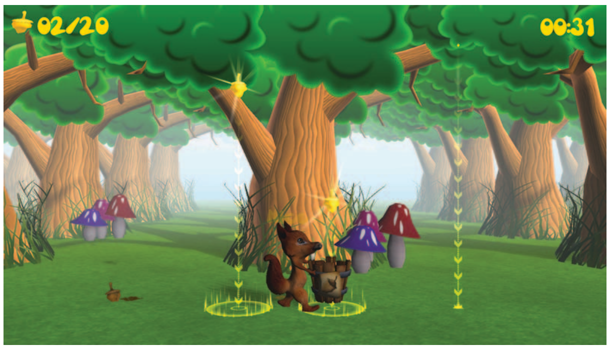

Fig. 2. Nuts Catcher game screenshot developed.

Rehabilitation games focus on biomechanical aspects, i.e. it should help to improve the user’s ability to perform movements or the user visual-motor coordination [8]. Several requirements apply to the games processes. The game characteristics should be tailored to fit the different behavior and preferences of the users. The game difficulty level should be adjusted avoiding frustrating conditions when the patient is requested to perform exercises beyond its capacity. The difficulty level should also be enhanced if the patient is insufficiently challenged. Patients skills vary greatly according to the tasks proposed by different games, therefore game difficulty should be adapted according to the user limitations and also to their accomplishments.

Some of the game parameters that are suitable interfere in the difficulty level perceived by the player are: the fall down velocity (v) of the nuts; the appearing frequency (f) of the nuts on the screen two or more nuts may appear simultaneously; the initial distance (d) between an appearing nut and the squirrel; the size (b) of the basket carried by the squirrel.

IV. REINFORCEMENT LEARNING

Reinforcement Learning (RL) is a learning technique that allows an agent to learn from its interaction with the environment through reinforcement and punishment mechanisms. The learning process takes place based on the knowledge of the current agent state (s) in the environment, the action performed by this agent (a) and the observation of state change arising from the action (s’). These are the basic elements of a reinforcement learning process known as Q-Learning [9]. Q-Learning may also be interpreted as a Markov decision process (MDP) with unknown probabilities and rewards.

[2]http://www.can-cia.org/

[3]Middleware describes usually a software binder, or mediator, between two existing and independent program codes. Its function is provide the applications with the independence of transmission systems.

[4]The game was developed using Unity 3D. More information at `http://unity3d.com`

A. Q-Learning

The Q-learning algorithm [10] consists in updating discounted values of expected rewards, $Q(s,a)$. At each iteration with the environment, the Q-values are updated according to Eq. 1.

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right] \quad (1)$$

where γ is the discount factor used to ensure that the values of Q are finite and α is the learning constant, and $0 < \alpha \leq 1$ and $0 < \gamma \leq 1$.

Performing an action a, the agent changes from state s to the state s', and receives an immediate reward r. In state s' a search is made within the available actions to find the action a' that leads the agent to a state with the highest reward value, represented as $max_{a'} Q(s',a')$ in equation 1.

One advantage of the Q-Learning approach is it enables the implementation of an agent based online learning process. However, the optimal convergence of the actions can be slow, depending on the adopted model [11].

B. SARSA

A powerful variation for the Q-Learning is implemented holding the action update value at each step [12] as described below:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma Q(s',a') - Q(s,a) \right] \quad (2)$$

If the chosen action a' is $max_{a'} Q(s',a')$, the algorithm becomes equivalent to the standard Q-Learning algorithm. Additionally, the SARSA algorithm allows a' to be randomly chosen using a predefined probability. Eliminating the *max* operator from the actions makes SARSA faster than the standard Q-learning, especially for applications with high cardinality actions sets. A common choice consists in adopting the maximum reward value for 70% of the evaluations and a random reward value for the remaining 30% cases. The procedure is useful to avoid local maximum.

V. REINFORCEMENT LEARNING APPROACH FOR "NUTS CATCHER"

The problem here may be summarized in inducing the agent to learn, at runtime, how the player responds to changes in the game difficulty level. For this modeling process, it is necessary to define: the set of possible game states s, representing part of the environment, the set of actions $A(s)$, the form of the reward, r; and the function that evaluates the player performance at each state, $P(s)$.

As explained in the next section, the performance is a serious game measure composed not only by the game scores but also by the amount of exercise (movement) the user executes. The goal is to maximize the performance function keeping the game a challenging and entertaining task while the user executes repetitive movements.

Variables that act directly on the game difficulty were chosen to serve as environment states. In the performed experiments only two parameters were adopted for difficulty adjustment purposes, namely the "nut drop rate" (v) discretized in m values and the "distance to nut" (d), discretized in n states. The combination of these two variables define a matrix of possible states and the number of states ($s_{i,j}$) and $s_{total} = m \times n$. For experimental reasons, in this work we set $m = 5$ and $n = 5$, so that during the therapy sessions, most of the states can be visited by the software agent.

In this specific case, the agent navigates through the states towards the direction of the most difficult game level ($s_{m,n}$). However, as difficulty increases, it becomes harder to the player to collect the same number of nuts. The actions that change the game difficulty are defined as:

$$A = \{left\ (\leftarrow), up\ (\uparrow), right\ (\rightarrow), down\ (\downarrow)\} \quad (3)$$

where, the $\leftarrow$ action represents a reduction of the distance ($s_{i,j-1}$), $\uparrow$ reduces the velocity ($s_{i-1,j}$), $\rightarrow$ increases distance ($s_{i,j+1}$) and $\downarrow$ increases the velocity ($s_{i+1,j}$). Actions leading to undefined states, outside the predefined ranges are not allowed.

Each agent action changes the game difficulty. The parameters are updated and kept for a period that corresponds to the release of (η) of nuts. Currently this set $\eta = 2$ nuts. After this period, a performance value (P) is taken as an estimate of the player behavior/adaptation to the new therapy exercise condition.

$$P(s) = \alpha_\theta \cdot \tau_\theta + \alpha_v \cdot \tau_v + \alpha_e \cdot \tau_e \quad (4)$$

where, α_θ , α_v and α_e are weighting variables for the importance of the three performance components.

τ_θ measures if the player is using the maximum range of motion. $0 \leq \tau_\theta \leq 1$ and it is given by:

$$\tau_\theta = \frac{\theta_{extesion} + \theta_{flexion}}{\theta_{max}} \quad (5)$$

where $\theta_{extension}$ is the maximum wrist extension value (radians), $\theta_{flexion}$ is the maximum wrist flexion value and θ_{max} is the some of the flexion and extension values found in the literature [13].

τ_v gives a time measure of the synchronization between the nut movement and the users movement and it is given by Eq. 6. This is measured indirectly if the player has to wait until the nuts fall on the squirrel basket. The idea behind this performance component is to induce the player to execute more movements.

$$\tau_v = \eta \cdot \frac{nuts_y^{start} - ground_y}{v_i} - \sum_{j=1}^{\eta} \Delta t_j^{nuts} \quad (6)$$

where $nuts_y^{start}$ and $ground_y$ are constants representing the y coordinate (height) on the screen where the nut appears and the y coordinate of the ground, respectively, v_i is the nut falling velocity and $\sum_{j=1}^{\eta} \Delta t_i^{nuts}$ is the accumulated sum of the time periods when the player waited for the nut and did not move the handle.

τ_e represents the relative error in x screen coordinates between a nut fall position d_i and the current character position (imposed by the user) (p_x), $0 \leq \tau_e \leq \eta \frac{d_n}{2}$ being τ_e given by Eq. 7 and p_x calculated by using Eq. 8. This performance component privileges the player motion. The farther the nuts fall more the player will need to move the robot handle.

$$\tau_e = max\left(\eta \cdot \frac{d_n}{2} - \sum_{j=1}^{\eta} \left|\frac{d_i}{2} - p_x\right|, 0\right) \tag{7}$$

$$p_x = \begin{cases} \left(\frac{\theta_{motor}}{\theta_{extesion}}\right) \cdot d, & if \theta_{motor} \geq 0 \\ \left(\frac{\theta_{motor}}{\theta_{flexion}}\right) \cdot d, & otherwise \end{cases} \tag{8}$$

$$\theta_{motor} = \frac{pulse * 360}{2000} \tag{9}$$

where θ_{motor} is the angular value provided by the encoder (Eq. 9).

The immediate reward given at each step interaction is calculated using the inverse of the Euclidean distance from the current state to the final state (Eq. 10). The reward represents how far is the player from the hardest game level. Although the player can performance poorly in the higher difficulty levels, it is necessary to test how much challenge that player may handle. The reward is chosen in the attempt to maximize challenge trying the user for larger wrist displacements while avoiding to reduce the player scores.

$$r = \frac{1}{\sqrt{(v_m - v_i)^2 + (d_n - d_j)^2}} \tag{10}$$

The Q-Learning approach algorithm has been adapted and implemented to work with the "Nuts Catcher" game and it is presented in Algorithm 1.

Algorithm 1 Q-Learning for "Nuts Catcher"

```
Require: Load or Initialize $Q(s,a)$ with arbitrary values
  Calibrate $\theta_{extension}$ and $\theta_{flexion}$ of the player
  for all episodes do
    Initialize $s$ randomly
    for $step \leftarrow 1$ to 5 do
      for all A(s') do
        Execute action $a$ with $\eta$ nuts
        Observe the player for each $s'$
        $P(s) = \alpha_\theta \cdot \tau_\theta + \alpha_v \cdot \tau_v + \alpha_e \cdot \tau_e$
      end for
      Choose action $a$ in state $s$, using $\varepsilon$-greedy policy
      $Q(s,a) \leftarrow Q(s,a) + \alpha\,[r + \gamma Q(s',a') - Q(s,a)]$
      $s \leftarrow s'$
    end for
  end for
```

During the game play, the user interacts with the virtual environment through the robotic device. The game keeps record of the current state, the rules set, provides feedback about the next difficulty state, the validity of a particular action and the reward for a particular action. Using the adopted policy, the Q values are calculated, consequently updating the game state and then selecting a new action (new speed and distance values for the game). Figure 3 shows this process that happens throughout the game.

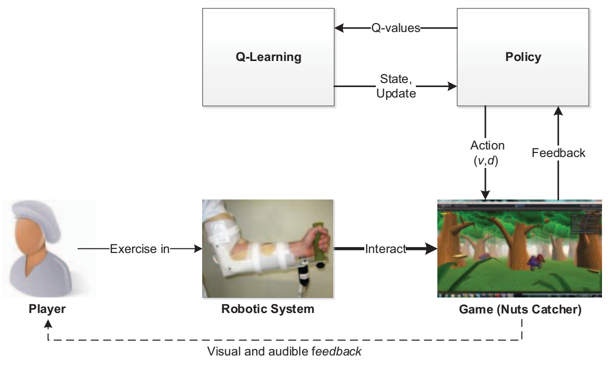

Fig. 3. Block diagram of the user interaction with the game.

The algorithm updates the value Q after every episode, and after each update new speeds and distance conditions are tested, observing the player performance and thus improving the information about user skills. It is important to notice that, the algorithm randomly selects the start state, thus the game may start at either an easy level or a very difficult level. The players did not receive training before the conduction of the experiments.

VI. EXPERIMENTS

For the tests, an experimental setup was built. It is composed by a robotic device with one degree of freedom and the development of single player adaptive game. Experiments with thirty minutes duration were conducted with 4 healthy volunteers. The robot attached to the user wrist captured extension and flexion movements and forwarded them as inputs to the rehabilitation game. The robot system also collected the motion data during the experiment. In Fig. 4 is shown one of the volunteers playing.

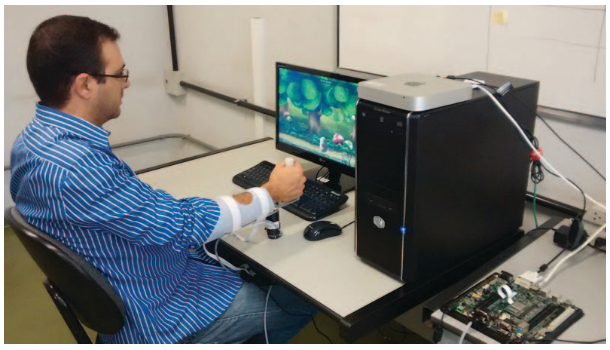

Fig. 4. Healthy volunteer playing with the robotic system handle the game "Nuts Catcher".

A. Game settings

The first time the player uses the game, he is puts your personal data, generating a log file. This file encompasses:

user name, gender, age and handedness (left or right handed). After that, an evaluation is performed providing individual movement limits. The maximal flexion ($\theta_{flexion}$) and extension ($\theta_{extension}$) wrist amplitudes are introduced and stored at this movement. This procedure allows amplitude calibration every time the game initiates, whereas personal data are input only once, being queried to additional sessions.

The amplitude of player's movement is used to calculate the positions where nuts may appear and fall. Fig. 5 shows how these coordinates are configured.

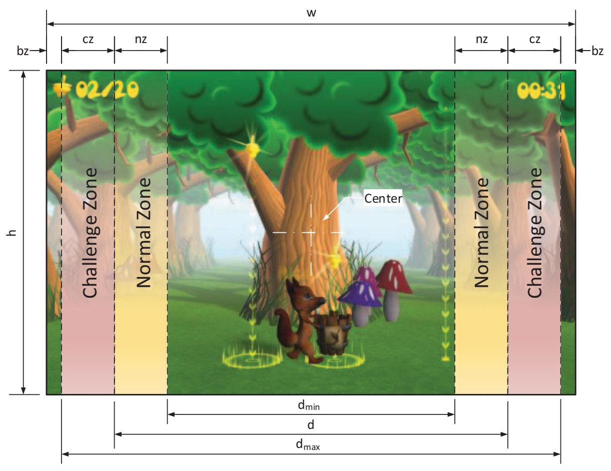

Fig. 5. Configuration of distances on screen, in relation to player's amplitude.

$$\begin{cases} bz = \sigma_{bz}\dfrac{w}{2} & \text{(11a)} \\ cz = \sigma_{cz}\dfrac{w}{2} & \text{(11b)} \\ nz = \sigma_{nz}\dfrac{w}{2} & \text{(11c)} \\ d = w - 2(bz+cz) & \text{(11d)} \\ d_{min} = d - 2nz & \text{(11e)} \\ d_{max} = d + 2cz & \text{(11f)} \end{cases}$$

where, bz is a distance constraint that presents the creation of nuts outside the screen and w is the screen size in pixels. Two zones are created: a challenge zone (cz) that exceeds the player's movement reach and a normal zone (nz), that is compatible with the player's maximum range of motion. Player's movement distance d defines, in pixels, how the maximum range of motion is represented on screen.

The σ variables are constant defined either by the programmer or by a therapist. In this case, $\sigma_{bz} = 0.04$, $\sigma_{nz} = 0.3$ and $\sigma_{bz} = 0.3$ if $\theta_{flexion} + \theta_{extension} < \theta_{max}$. Otherwise, $\sigma_{bz} = 0$ because the additional amplitude increase would not be feasible.

The nut fall velocity range is defined empirically using visual inspection. Minimum (v_{min}) and maximum (v_{max}) velocities were assigned to $3m/s$ and $20m/s$.

B. Results and discussion

During the game, the parameters of the game were adapted according Algorithm 1. In Figure 6 shows the performances of each individual player and the corresponding tendency curve. The number of episodes ranged from 54 to 83. This number is a result of the nuts dropping down during the 30 minutes experiment. One may observe that players show distinct responses to different difficulty levels. For example players 2 and 3 shows a more stable behaviour while players 1 and 4 performances oscillate.

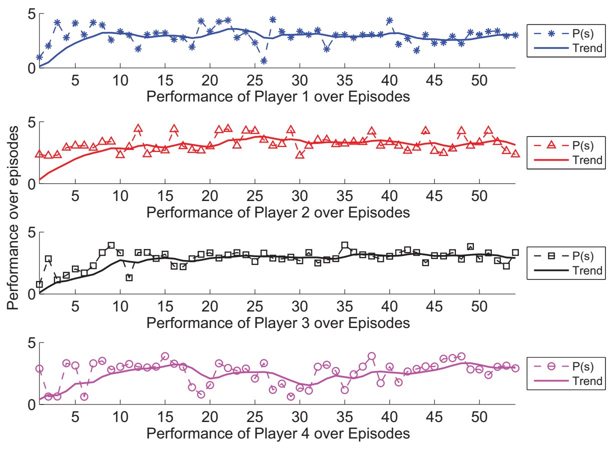

Fig. 6. Performance of players throughout the episodes in a single session.

Figure 7 displays the percentage of nuts captured by each player. The percentage of captured nuts stabilize for all players after approximately the fifteenth episode. The percentage of captured nuts is connected to the response each player gives to each game difficulty level, the more difficult the game, the less nuts are captured, conversely the easier game , higher is number of captured nuts. In this case the players captured between 65% and 85% of the available (released) nuts.

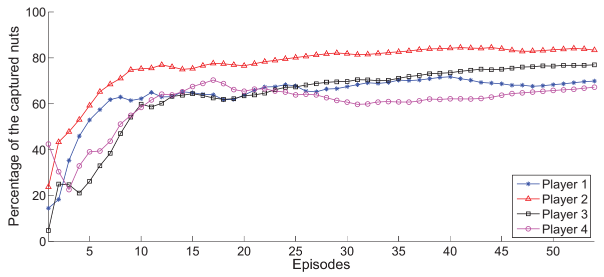

Fig. 7. Percent taken along nuts of the episodes.

Figure 8 shows the cumulative rewards each player had over the episodes. The greater the received reward, the higher the selected game difficulty. Player 4 has accumulated the highest amount of rewards. He remained at more difficult game levels and captured a smaller number of nuts. Probably he is the more skilled player.

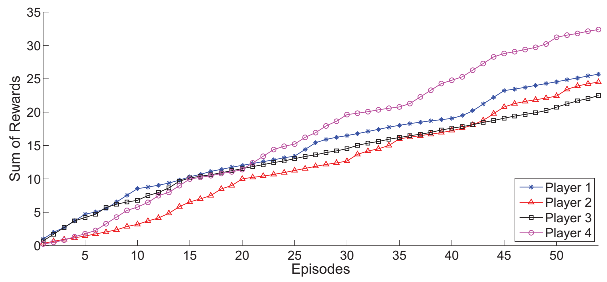

Fig. 8. Sum of rewards obtained during episodes.

Figure 9 shows an state array ($velocity \times distance$) for player 4, with the corresponding Q values. The state with the speed range of 2 and 4 present the highest Q values. The array demonstrates how the approach maps difficulty states with respect to the player skills

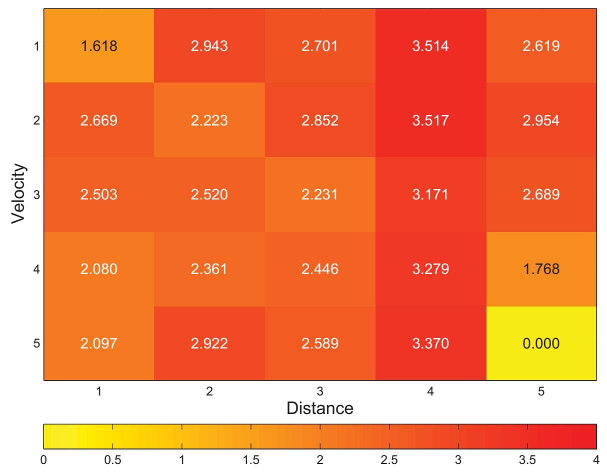

Fig. 9. Matrix of Q-values for each state.

VII. CONCLUSIONS

In this work, the Q-learning algorithm was implemented to map how a player responds to variations in the game difficulty parameters. Experiments were carried out covering the state space formed by two game parameters: one related to motion amplitude and another parameter related to velocity. The results indicated the capacity of the approach to dynamically adapt the game difficulty according to each player, indirectly modeling the player skills. The approach stimulates the player disturbing the game conditions, measures the responses using performance functions and tries to find trends for each player.

Although the experiments are still preliminary and a larger number of samples and tests are necessary, the results obtained so far indicates that the approach is feasible for modeling the user behavior and encourage us to extend the studies to experiments with clinical subjects. The obtained map may be used as a guideline on how to make the game easier or harder for each individual player.

It also important to notice, that Q-learning is one of the simplest reinforcement learning mechanisms. Therefore we intent to compare it with other RL approaches, such as R-Learning, H-Learning and Z-Learning. As future work, we are also planning to include robot impedance as an additional parameter for game difficulty adaption.

ACKNOWLEDGMENT

We thanks Ricardo Ranzani (`https://soundcloud.com/ricardo-ranzani`) for creating the music and sound effects and Rogério Alan do Prado (`http://alanart.net/`) from the virtual environment and character modeling of the game "Nuts Catcher".

REFERENCES

[1] G. C. Burdea, "Virtual rehabilitation–benefits and challenges." *Methods of information in medicine*, vol. 42, no. 5, pp. 519–523, 2003. [Online]. Available: http://dx.doi.org/10.1267/meth03050519

[2] J. W. Burke, M. D. J. McNeill, D. K. Charles, P. J. Morrow, J. H. Crosbie, and S. M. McDonough, "Optimising engagement for stroke rehabilitation using serious games," *Vis. Comput.*, vol. 25, no. 12, pp. 1085–1099, Oct. 2009. [Online]. Available: http://dx.doi.org/10.1007/s00371-009-0387-4

[3] M. Csikszentmihalyi, *Flow: The Psychology of Optimal Experience*, first edition ed. Harper Perennial, Mar. 1991. [Online]. Available: http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20\&path=ASIN/0060920432

[4] R. Hunicke and V. Chapman, "AI for Dynamic Difficulty Adjustment in Games," 2004. [Online]. Available: http://cs.northwestern.edu/~hunicke/pubs/Hamlet.pdf

[5] K. Andrade, G. Fernandes, J. Martins, J., V. Roma, R. Joaquim, and G. Caurin, "Rehabilitation robotics and serious games: An initial architecture for simultaneous players," in *Biosignals and Biorobotics Conference (BRC), 2013 ISSNIP*, 2013, pp. 1–6.

[6] K. O. Andrade, G. G. Ito, R. C. Joaquim, B. Jardim, A. A. G. Siqueira, G. A. P. Caurin, and M. Becker, "A robotic system for rehabilitation of distal radius fracture using games," in *Proceedings of the 2010 Brazilian Symposium on Games and Digital Entertainment*, ser. SBGAMES '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 25–32. [Online]. Available: http://dx.doi.org/10.1109/SBGAMES.2010.26

[7] H. Krebs, B. Volpe, D. Williams, J. Celestino, S. Charles, D. Lynch, and N. Hogan, "Robot-aided neurorehabilitation: A robot for wrist rehabilitation," *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 15, no. 3, pp. 327 –335, sept. 2007.

[8] N. Hogan, H. I. Krebs, B. Rohrer, J. J. Palazzolo, L. Dipietro, S. E. Fasoli, J. Stein, R. Hughes, W. R. Frontera, D. Lynch, and B. T. Volpe, "Motions or muscles? Some behavioral factors underlying robotic assistance of motor recovery." *Journal of rehabilitation research and development*, vol. 43, no. 5, pp. 605–618, 2006. [Online]. Available: http://view.ncbi.nlm.nih.gov/pubmed/17123202

[9] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998. [Online]. Available: http://www.cs.ualberta.ca/\%7Esutton/book/ebook/the-book.html

[10] C. J. C. H. Watkins, "Learning from delayed rewards," Ph.D. dissertation, King's College, Cambridge, UK, May 1989. [Online]. Available: http://www.cs.rhul.ac.uk/~chrisw/new_thesis.pdf

[11] S. Singh, P. Norvig, D. Cohn, and H. Inc, "How to make software agents do the right thing: An introduction to reinforcement learning," Adaptive Systems Group, Harlequin Inc, Tech. Rep., 1996.

[12] R. S. Sutton, "Generalization in reinforcement learning: Successful examples using sparse coarse coding," in *Advances in Neural Information Processing Systems 8*. MIT Press, 1996, pp. 1038–1044.

[13] W. Karwowski and K. Karwowski, *International Encyclopedia of Ergonomics and Human Factors, Second Edition - 3 Volume Set*. CRC, Mar. 2006.