Development of Rehabilitation System (ReHabgame) through Monte-Carlo Tree Search Algorithm

Shabnam Sadeghi Esfahlani, *Fellow, IET,* George Wilson, *Member, IEEE*,

Abstract—Computational Intelligence (CI) in computer games plays an important role that could simulate various aspects of real life problems. CI in real-time decision-making games can provide a platform for the examination of tree search algorithms. In this paper, we present a rehabilitation serious game (ReHabgame) in which the Monte-Carlo Tree Search (MCTS) algorithm is utilized. The game is designed to combat the physical impairment of post stroke/brain injury casualties in order to improve upper limb movement. Through the process of ReHabgame the player chooses paths via upper limb according to his/her movement ability to reach virtual goal objects. The system adjusts the difficulty level of the game based on the player's quality of activity through MCTS. It learns from the movements made by a player and generates further subsequent objects for collection. The system collects orientation, muscle and joint activity data and utilizes them to make decisions. Players data are collected through Kinect Xbox One and Myo Armband. The results show the effectiveness of the MCTS in the ReHabgame that progresses from highly achievable paths to the less achievable ones, thus configuring and personalizing the rehabilitation process.

Index Terms—Monte Carlo Tree Search, Gesture Recognition, Myo Armband, Kinect XBox One, Motion Capture.

1 Introduction

Recent advances in low cost depth sensing technology have led to the creation of consumer electronics devices that can sense the user′s motion [12]. These devices have rapidly become a major source of motivation for researchers who immediately recognized the potential use of the new technology in experimental rehabilitation [11], [35]. It transforms the player into the controller, making games more intuitive to play, thus more accessible to a broader audience. Human-computer interaction systems such as Microsoft Kinect in 2010[1] and Thalmic Labs Myo in 2016[2] have been utilised in scientific research projects and commercial applications and are shown in Fig. 1. These sensors capture player movements in the real world/time and convey them inside the game as well as capturing the position, orientation, and muscle activities. Kinect depth and image sensors facilitate a robust interactive human body tracking facility which reads the environment through a randomized decision forest algorithm. Its single input depth image is segmented into a dense probabilistic body part labeling, with the parts defined being spatially localized near skeletal joints of interest [27]. A wireless Myo armband sensor provides data through eight electromyography (EMG) pods and a nine-axis inertial measurement unit (IMU) capturing real world movements through the use of a gyroscope and accelerometer.

A clinical application of a body based control game via sensor input is not a one-size-fits-all solution and needs to be customized according to the individual′s condition, needs, and range of ability [29]. The rehabilitation therapy requires real-time decision making and must be able to adapt to the difficulty level according to the rehabilitation goals. Artificial and computational intelligence have mostly focused on games where no information is hidden and all actions are deterministic [15]. However, rehabilitation games feature elements of hidden information, real-time decision making, and uncertainty which are non-deterministic. Hidden information occurs when a game is in one of the many possible states with uncertainties, and real-time decision making is required to balance the performance and gain score.

In this research, we propose the use of Monte-Carlo Tree Search (MCTS) in designing a serious game for rehabilitation purposes called RehabGame. This virtual reality therapy system exposes stroke or traumatic brain injury survivors to a 3D environment to monitor, measure and improve the kinematic movement with the focus on the upper limbs. It is done through motion capture sensors (Kinect and Myo) that are interfaced with a game engine (Unity3D) through plugins Microsoft SDK 2015[3] and Kinect tools and resources 2016[4]. It measures the kinetic and kinematic movements through reach and grab which is a real time rehabilitation therapy.

1.1 Background & Related Work

In the last few years, several Monte-Carlo based techniques have emerged in the field of computer games that have had significant success in various AI game playing algorithms [15]. They can be applied to a single-player games and planning problems, multi-player games, real-time games, and games with uncertainty or simultaneous moves. The

- *S. S. Esfahlani and G. Wilson are with the Department of Computing and Technology, Anglia Ruskin University, Cambridge, UK shabnam.sadeghi-esfahlani@anglia.ac.uk, george.wilson@anglia.ac.uk*

[1] http://www.xbox.com/en-US/xbox-one/accessories/kinect
[2] https://www.myo.com/
[3] http://wiki.etc.cmu.edu/unity3d/index.php/Microsoft
[4] https://developer.microsoft.com/en-us/windows/kinect/tools

MCTS method was developed by combining the random sampling of traditional Monte-Carlo methods with random simulations to search for the best outcome. Possible results are organized in a search tree to estimate the long-term potential, and with each iteration, the game tree is expanded and converged over time [9]. The technique specifies which actions are possible from each state and which states are terminal [39]. The approach is based on real-time planning that finds the best branch and plays the best arm within that branch. MCTS has been applied successfully to many board games including; the Asian board game GO [2], [26], General Game Playing where the rules of games used to evaluate techniques are not known in advance [7], [9], [10], imperfect information games where each player independently chooses an action and these actions are applied at the same time, such as Scrabble and Bridge [3], the arcade Ms Pac-Man game with repeated random sampling to obtain results [13], [17], [20], [23], [32]. mds

August 26, 2015

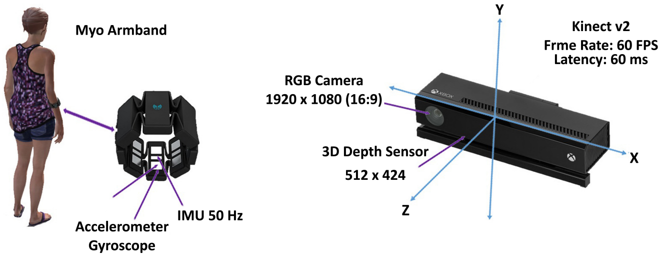


Fig. 1. Main components of the Kinect and Myo device.

In the last few years, several integrated solutions have been proposed for computer-based rehabilitation techniques [4], [8], [12], [14], [18], [19], [21], [22], [25], [28], [33], [34], [36], [37], [38]. Although there has been a lot of research in the area of virtual reality there are still outstanding challenges to apply MCTS technique to rehabilitation games in order to adapt it to patient′s status. [29] developed a middleware software framework called the Articulated Skeleton Toolkit to integrate full-body interaction with VR and video games. They presented a system architecture and described two case studies to evaluate the toolkit for controlling the first person video games. A study by [24] suggests that rehabilitation games should be integrated into general-purpose rehabilitation stations, adhere to the constraints posed by the clinical protocols, and adapt to patients status and progress. They used a Fuzzy system to monitor the execution of the exercises and provide direct feedback to the patients as well as a Quest Bayesian adaptive approach for real-time adaptation. [10] studied MCTS methods to be implemented in more complex video games. They investigated the use of a modified MCTS algorithm in video game playing using the general video game AI framework. In particular a modification to the core of MCTS known as UCT was performed and the results of their research showed the effectiveness and efficiency of the modified MCTS highly depends on the type of the game designed. [20], [23] have introduced MCTS for controlling the characters in the real time game Ms Pac-Man to find an optimal path for an agent at each turn. It determines the best move to make, based on the results of numerous randomized simulations.

2 Kinect and Myo Devices

The Kinect One $V2$ device consists of an infrared laser based IR emitter and a colored (RGB: red-green-blue) camera. It can detect the position and orientation of 25 individual joints (including thumbs), the weight put on each limb, a speed of joint movements, and track gestures performed with a standard controller. Body position is determoned in a two stage process; firstly computing a depth map and secondly infering participant body position. The depth map is constructed by analyzing a speckle pattern of infrared laser light [6]. Whilst body parts are inferred using a randomized decision forest learned from over one million training examples [16]. Kinect provides approximately 60 skeleton frames per second.

The Myo armband streams the accelerometer, gyroscope, orientation, and the electromyograph (EMG) at 200 Hz frequency and the inertial measurement unit (IMU) data at 50 Hz. It is made of three medical grade stainless steel EMG sensors that detect the electric impulses in the muscles. The armband is connected via a Bluetooth USB adapter and controls muscle activity related to movements of fingers, thumb, hand, wrist, and forearm. Myo detects the electrical signal produced by muscle exertion through electromyography. It is used to discover underlying electrical activity displayed as a waveform (an electromyogram). A three-axis accelerometer, magnetometer, and gyroscope senses motion and records and collects real-time data with high accuracy and precision.

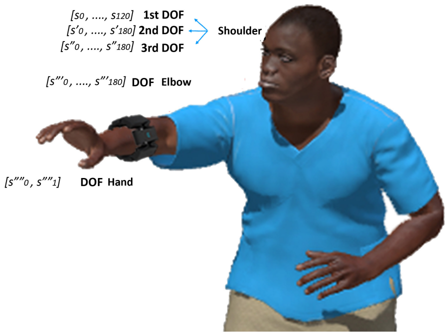

[s0, ..., s120] 1st DOF
[s′0, ..., s′180] 2nd DOF
[s″0, ..., s″180] 3rd DOF
Shoulder
[s‴0, ..., s‴180] DOF Elbow
[s⁗0, s⁗1] DOF Hand

Fig. 2. Character representation of the rehabilitation game with shoulder, elbow, and hand degree of freedom (DOF).

3 Development of ReHabgame system

A 3D character (avatar) is first constructed with 65 skeleton joints consists of one and three degree of freedom/s (DOF) developed in Mixamo[5] , [6]. One degree of freedom joints include the elbow and knee, whilst spherical three DOF joints include the shoulder, wrist, ankle, and pelvis. Fig. 2 shows the character with three DOF of shoulder, one DOF of elbow and three DOF of wrist.

[5] https://www.mixamo.com/fuse
[6] https://www.assetstore.unity3d.com

In order to employ the MCTS framework for rehabilitation therapy four enhancements are defined [13], [23] which are;

1) a constant tree depth with variable angles for the shoulder, elbow, and hand child elements in order to develop the parent upper limb tree search as shown in Fig. 2 and Fig. 4.
   Fig. 2 for example illustrates shoulder rotation of $120\,\mathrm{deg}$ in the x-direction, $180\,\mathrm{deg}$ in the y-direction, $180\,\mathrm{deg}$ in the z-direction, followed by rotation of the elbow by $90\,\mathrm{deg}$, and finally hand movement with thumb and four fingers.
2) search tree navigation and goal (virtual fruit) generation can be simulated according to the avatar's performance.
3) long term participant goals include achievement of high scores, improved performance, and motivation.
4) reuse of the search tree over several moves in a systematic way including backpropagation.

The environment of the game is made of a garden, an avatar, various fruits that are rotating around their axis in the scene and awaiting the avatar to reach and collect them in the basket. Fruit spawning, participant grab and release time and position are registered. A timer is defined for each fruit with the clock starting as soon as it is generated. The tree is constructed from the path movements of the hand in each session.

Fruits need to be released at the top of the basket where the inner surface of it changes to green when it is released in a right location as shown in Fig. 5. A warning is issued if the path is taken by a body and upper limb exceed constraints or the movement is outside the tree search. Position, angle, velocity and body constrains are specified as numerical values to minimize any risk of harm for users and at the same time encourage motor movement.

Although achieving high scores is the objective of the RehabGame the agent is designed based on player's performance and medical needs in order to take the player through a process that avoids frustration or harm. A randomness to fruit generation is introduced into the game to avoid pre-deterministic fixed paths during the game play.

The agent measures the distance between avatar to object utilizing a half arm span. Arm span or reach is the physical measurement of the length from one end of an individual's arms to the other when raised parallel to the ground at shoulder height at a $90\,\mathrm{deg}$ angle [30]. Distance vectors are accurately calculated including the shoulder to elbow (P_1P_2) vector, elbow to wrist (P_2P_3) vector, wrist to finger tips (P_3P_4) vector as well as shoulder to objects (P_1P_4) vector as illustrated in Fig. 3.

4 Monte-Carlo Tree Search

4.1 General MCTS

Monte-Carlo Tree Search is a probabilistic algorithm that uses lightweight random simulations to selectively grow a game tree. The method combines the precision of a tree search with the generality of random sampling [3]. The algorithm plans actions with a best first search technique that is based on stochastic models by Monte-Carlo simulations. In game play, this means that decisions are made based on the results of randomly simulated playouts [23]. Fig. 4 shows the mechanism of the MCTS algorithm applied to a round of the RehabGame.

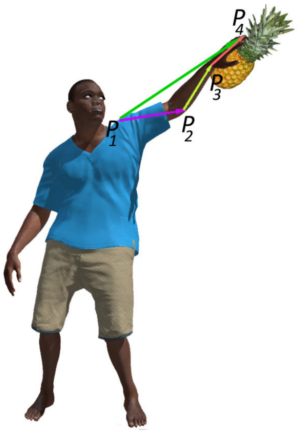


Fig. 3. Distances between body joints and fruits is calculated by the agent.

Decisions in the tree are the movements of upper limb followed by traversing the tree during the selection step. It means that the upper limb rotates in accordance to the selected nodes until playout starts based on the random path through the tree. The tree represents the sum of all possible arm movements during the game state. The root node starts with the rotation of the shoulder inside the XZ plane, or yaw, descending afterward to the second level where the decision resides in the degrees of rotation in the XY plane, or pitch. The third level is then reached, where the selection will determine the rotation of the arm around itself or roll. The final decision will determine the rotation of the elbow, which has only one degree of freedom. Having a complete path, from root to level four, which is a leaf, will determine the virtual position of the hand, which coincides with the desired position for the fruit placement. A path from the root node to a terminal node is then executed to reach that position. When descending the tree, the movement path of the avatar's upper limb is determined by the distances between the position of a wrist, elbow and shoulder components. Traversing the tree means that avatar's hand moves along the paths described by the current set of rotations until the final position is reached. The general MCTS algorithm is depicted in Table 1, with a combination of paths selected during the selection and playout steps form a single simulation. Here v_0 is the root node of the initial state s_0, and v_4 is the last node reached during the descending stage through the tree and corresponds to the final position reached with the pseudo-random choices. The choices are not completely random as $BestChild$ will stimulate a certain degree of exploration, to avoid duplicate random unfolding of the game. Δ is the reward for the terminal state, s_1, reached by running the default algorithm from state s_0. This reward reflects the accuracy of the position reached, as the General MCTS is only capable of finding a "best solution" in a limited pool of samples.

TABLE 1
Algorithm 1, General MCTS.

```
Function MonteCarloTreeSearch ($s_0$)
Create root node ($v_0$) within state ($s_0$)
$level \leftarrow 0$

While ($level$ < 4)
    $v_{level+1} \leftarrow$ BestChild($v_{level}$)
    $level \leftarrow level$ + 1
End While

$\Delta \leftarrow$ Accuracy ($v_4$)
BackPropagate ($\Delta$)
return $v_4$
```

As explained before, MCTS function has the capacity of looking ahead to obtain the accurate success probability of taking different paths (decision making). As for all real time games, the time constraint is very tight [31] expecting a player to make a decision towards an action within about 10 ms (milliseconds).
The next move is simulated based on the information taken from the current situation. Suppose that a fruit is located at (x, y) we need to go along a path d in order to collect it. The next object is located at position $(x\prime, y\prime)$ generated with an improved algorithm, related to the Upper Confidence bound for Tree, Eqn .2. This encourages movements just outside of the player′s comfort zone. Each node v in MCTS holds four pieces of data:

- Associated state $s(v)$
- Incoming action $a(v)$
- Total simulation reward $Q(v)$
- Visit count $N(v)$ (a non negative integer) [31]

The ratio of number of times the node has been visited Q(v), to the total reward of playouts N(v) is defined in Eqn .1, [3], [23], [31], [32].

$$\frac{Q(v)}{N(v)} \tag{1}$$

The return or score of the overall search in the tree is $a(BestChild(v_0, 0))$ with the action a that leads to the child with successful results with the exploration parameter c. The score and time are backpropagated to $(x\prime, y\prime)$ and another simulation from (x, y) is started again. The mechanism of the MCTS algorithm is made of; selection, expansion, simulation and backpropagation [5] and [3].

1) Selection: The algorithm starts at the root node, builds a tree of possible future game states (children). Each node in the tree represents a state and each link represents an action taken in that state that leads to a new state [40]. While the state is built in the tree, the next action is chosen according to the statistics stored, in a way that balances between exploitation and exploration. The task is either selecting an action that leads to the best results so far called exploitation or a less promising actions still needs to be explored known as; exploration, due to the uncertainty of the evaluation.
2) Expansion: More state (child nodes) needs to be added as new nodes to expand the tree. The tree is expanded by one node for each simulated game according to the available actions. The first action is chosen by an expansion strategy and subsequently simulated. This results in a new game state, for which a new node is created.
3) Simulation (Playout): After expansion, a rollout is done from the new node, which means that a simulated playout is run from the new node applying random actions until a predefined stop criterion is met or the game ends. The satisfactory weighting of action selection probabilities has a significant effect on the level of play. If all legal actions are selected with equal probability, then the strategy played is often weak, and the level of the Monte-Carlo program is suboptimal [5].
4) Backpropagation (back up): The result of the simulated playout is propagated immediately from the rollout (selected node) back up to the root node. This means that the reward is saved to the visited nodes then a new iteration starts.

Backpropagation updates node statistics that are utilized for future decision makings using Default Policy, [3]. It playouts the domain from a given non-terminal state to produce a value estimation (simulation).

4.2 Upper Confidence Bound for Tree

The modified version of MCTS is an effective variant and selection strategy (UCB applied to trees algorithm-UCT) known as the upper confidence bound for trees. The UCT is derived from the UCB function for maximizing the rewards of a multi-armed bandit [3], [7], [10], [13]. UCT balances the choice between poorly explored actions with a high uncertainty and actions that have been explored extensively with a higher value with a child node j. If child nodes for all actions of the current node are added into the tree, actions are selected in order to maximize a UCT on the action value. UCT works by doing many multi-phase playouts:

$$UCT = \bar{X}_j + C_p\sqrt{\frac{\ln n}{n_j}} \tag{2}$$

where C_p, is a bias parameter which defines the proportion of exploitation and exploration. If $C_p = 0$, the UCB policy becomes a greedy policy [35]. $\bar{X}_j$ is the average reward from child j and it is the exploitation part of the algorithm and the second part is exploration part. n is the number of times the current node (parent) has been played, n_j is the number of times child j has been visited and $C_p > 0$ is a constant (often set to $\sqrt{2}$) that shifts priority from exploration to exploitation. By increasing C_p, the priority is shifted to exploration which means; states that have been visited less will be visited with a higher priority than states that have been visited more. A decrease shifts the priority to exploitation and means that; states which have a high value are visited more in order to maximize reward. If more than one child node has the same maximal value, the tie

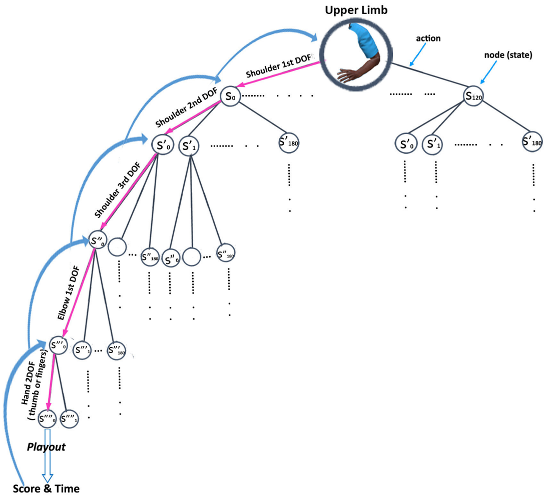


Fig. 4. Mechanism of Monte Carlo Tree Search algorithm of RehabGame.

is usually broken randomly. The values of $X_{i,t}$ and $\bar{X}_j$ are within $[0,1]$. UCT value yields ∞, as per $n_j = 0$. The traditional expansion strategy is to explore each action at least once in each node. After all, actions have been expanded, the node applies the selection strategy for further exploration. After a rollout, the reward is backed up, which means that the estimated value for every node that has been visited in this iteration is updated with the reward of this simulation. Usually, the estimated value of a node is the average of all rewards backed up to that node. The UCT selection policy is used when the visit count of all children is above a threshold. When one or more of the children's visit counts are below this threshold, a random uniform selection is made.

4.3 Markov Decision Making process

Markov Decision Processes (MDPs) are a fundamental modeling approach in decision theory and planning. They are used extensively within the artificial intelligence and operations research communities to model problems that require sequential decision making in an uncertain environment [41]. An MDP is generally defined by the subsequent elements: S is a finite set of fully-observable possible states, A is a finite set of possible actions depending on the states, $S \times A \times [0,1]$. It is the transition function where $P(s,a,s\prime)$ is the probability of moving to state $s\prime$ when action a is applied in state s. R is a real-valued reward function where $R(s,a)$ represents the expected reward for taking action a in state s. A policy π:$S \rightarrow A$ specifies an action $a = \pi(s)$ to be taken when in state s. The value function is defined as in Eqn.3:

$$V_\pi(s) = R(s,a) + \gamma . \sum_{s\prime \in S} P(s,a,s\prime).V_\pi(s\prime) \tag{3}$$

This represents the long term expected reward of executing a policy where $0 \leq \gamma \leq 1$ is a discount rate on future rewards. An optimal policy that maximizes the long term expected rewards is defined via the optimal value function shown in Eqn.4:

$$V^\star(s) = max_{a \in A} R(s,a) + \gamma . \sum_{s\prime \in S} P(s,a,s\prime).V^\star(s\prime) \tag{4}$$

5 Monte Carlo Tree Search for RehabGame with decision-making process

The main mechanism of the RehabGame resides in the fruit generator algorithm. This algorithm is an alternative variation of the UCT. The difference is that the statistical improvements that are calculated during the back-propagation phase, are also being propagated horizontally through the tree. This is achieved through the presence of virtual paths, or prospects. These prospects represent the nodes that have not been explored yet throughout the tree, which are grouped in layers. The statistical probabilities are added to these inexistent nodes using Eqn. 5:

$$P_x = (x^2 + 1)^{-k} \tag{5}$$

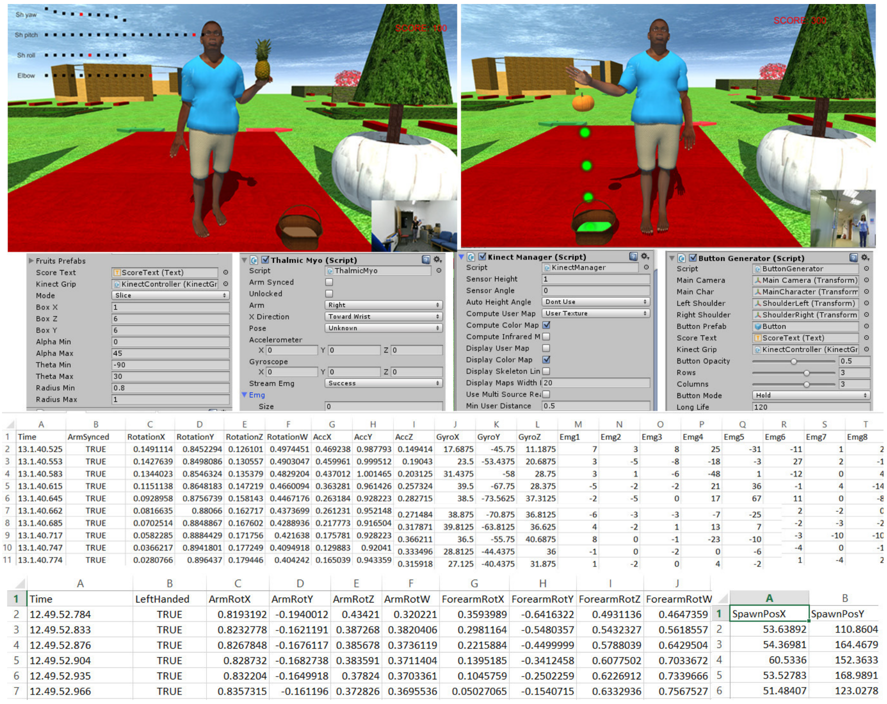
P_x represents the prospect for the x^{th} neighbor of the visited node, and k represents the degree of propagation. The larger the value, the more localized the back-propagation is.

The Score is achieved depending on the avatar's activity. If the fruit is collected and placed in a right place: $+1$. If the fruit is not reached or collected: -1. And if the fruit is collected but not placed in the basket or in the right location: 0.

$$M^i_{Score} = \begin{cases} M^i_{success} & if\ Score\ is\ +1 \\ M^i_{unsuccess} & if\ Score\ is\ 0 \\ M^i_{fail} & if\ Score\ is\ -1 \end{cases} \tag{6}$$

M^i_{Score} is the score of the i^{th} play during the selection, backpropagation, and decision-making process as defined in Eqn. 6. This Score is multiplied by the time efficiency, which is determined by confronting the player's time with a liner interpolation between a best possible time (max score) and a maximum time allowed (0 scores). Once the Score is determined, it is propagated upwards through the tree and expanded into the prospects of the neighbors, based on Eqn. 5.

When a new fruit needs to be spawned the agent analyses and extracts information from the avatar's past movements then selects the move with the most ambiguous success rate and based on that, it decides on the next step. By selecting the routes with the lowest absolute value, we are targeting the positions that have not been explored yet, or



▶ Fruits Prefabs
Score Text: ScoreText (Text)
Kinect Grip: KinectController (KinectGr
Mode: Slice
Box X: 1
Box Z: 6
Box Y: 6
Alpha Min: 0
Alpha Max: 45
Theta Min: -90
Theta Max: 30
Radius Min: 0.8
Radius Max: 1

☑ Thalmic Myo (Script)
Script: ThalmicMyo
Arm Synced: □
Unlocked: □
Arm: Right
X Direction: Toward Wrist
Pose: Unknown
Accelerometer: X 0 Y 0 Z 0
Gyroscope: X 0 Y 0 Z 0
Stream Emg: Success
Emg
Size: 0

☑ Kinect Manager (Script)
Script: KinectManager
Sensor Height: 1
Sensor Angle: 0
Auto Height Angle: Dont Use
Compute User Map: User Texture
Compute Color Map: ☑
Compute Infrared M: □
Display User Map: □
Display Color Map: ☑
Display Skeleton Lin: □
Display Maps Width I: 20
Use Multi Source Rea: □
Min User Distance: 0.5

☑ Button Generator (Script)
Script: ButtonGenerator
Main Camera: Main Camera (Transform)
Main Char: MainCharacter (Transform
Left Shoulder: ShoulderLeft (Transform)
Right Shoulder: ShoulderRight (Transform
Button Prefab: Button
Score Text: ScoreText (Text)
Kinect Grip: KinectController (KinectGr
Button Opacity: 0.5
Rows: 3
Columns: 3
Button Mode: Hold
Long Life: 120

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	Time	ArmSynced	RotationX	RotationY	RotationZ	RotationW	AccX	AccY	AccZ	GyroX	GyroY	GyroZ	Emg1	Emg2	Emg3	Emg4	Emg5	Emg6	Emg7	Emg8
2	13.1.40.525	TRUE	0.1491114	0.8452294	0.126101	0.4974451	0.469238	0.987793	0.149414	17.6875	-45.75	11.1875	7	3	8	25	-31	-11	1	2
3	13.1.40.553	TRUE	0.1427639	0.8498086	0.130557	0.4903047	0.459961	0.999512	0.19043	23.5	-53.4375	20.6875	3	-5	-8	-18	-3	27	2	-1
4	13.1.40.583	TRUE	0.1344023	0.8546324	0.135379	0.4829204	0.437012	1.001465	0.203125	31.4375	-58	28.75	3	1	-6	-48	1	-12	0	4
5	13.1.40.615	TRUE	0.1151138	0.8648183	0.147219	0.4660094	0.363281	0.961426	0.257324	39.5	-67.75	28.375	-5	-2	-2	21	36	-1	4	-14
6	13.1.40.645	TRUE	0.0928958	0.8756739	0.158143	0.4467176	0.263184	0.928223	0.282715	38.5	-73.5625	37.3125	-2	-5	0	17	67	11	0	-8
7	13.1.40.662	TRUE	0.0816635	0.88066	0.162717	0.4373699	0.261231	0.952148	0.271484	38.875	-70.875	36.8125	-6	-3	-3	-7	-25	2	-2	0
8	13.1.40.685	TRUE	0.0702514	0.8848867	0.167602	0.4288936	0.217773	0.916504	0.317871	39.8125	-63.8125	36.625	4	-2	1	13	7	-2	-3	-2
9	13.1.40.717	TRUE	0.0582285	0.8884429	0.171756	0.421638	0.175781	0.928223	0.366211	36.5	-55.75	40.6875	8	0	-1	-23	-10	-3	-10	-10
10	13.1.40.747	TRUE	0.0366217	0.8941801	0.177249	0.4094918	0.129883	0.92041	0.333496	28.8125	-44.4375	36	-1	0	-2	0	-6	-4	0	-1
11	13.1.40.774	TRUE	0.0280766	0.896437	0.179446	0.404242	0.165039	0.943359	0.315918	27.125	-40.4375	31.875	1	-2	0	4	-2	1	-4	2

	A	B	C	D	E	F	G	H	I	J
1	Time	LeftHanded	ArmRotX	ArmRotY	ArmRotZ	ArmRotW	ForearmRotX	ForearmRotY	ForearmRotZ	ForearmRotW
2	12.49.52.784	TRUE	0.8193192	-0.1940012	0.43421	0.320221	0.3593989	-0.6416322	0.4931136	0.4647359
3	12.49.52.833	TRUE	0.8232778	-0.1621191	0.387268	0.3820406	0.2981164	-0.5480357	0.5432327	0.5618557
4	12.49.52.876	TRUE	0.8267848	-0.1676117	0.385678	0.3736119	0.2215884	-0.4499999	0.5788039	0.6429504
5	12.49.52.904	TRUE	0.828732	-0.1682738	0.383591	0.3711404	0.1395185	-0.3412458	0.6077502	0.7033672
6	12.49.52.935	TRUE	0.832204	-0.1649918	0.37824	0.3703361	0.1045759	-0.2502259	0.6226912	0.7339666
7	12.49.52.966	TRUE	0.8357315	-0.161196	0.372826	0.3695536	0.05027065	-0.1540715	0.6332936	0.7567527

	A	B
1	SpawnPosX	SpawnPosY
2	53.63892	110.8604
3	54.36981	164.4679
4	60.5336	152.3633
5	53.52783	168.9891
6	51.48407	123.0278

Fig. 5. The configuration interface, parameters, results, constraints definition and Screen shots of RehabGame in which the avatar collects and releases the fruits above the basket.

positions that a participant is training on, by migrating from achievable areas to unachievable areas.
In this process, the agent calculates the distances between avatar's joints and the fruits in the 3D scene using data collected from Kinect (position) and Myo (gyroscope). It generates the forthcoming fruits inside the player's comfort zone and gradually directs it out of this zone while avoiding frustration or causing any harm. The distances are calculated as a vector from one joint to the other as well as from the joints to the fruit's location. The data generated will be saved after every session and can be retrieved by the agent and clinician. Fig. 5 shows the interfaces and data generated in the background. The game can be set up for right or left upper limb, the orientation and muscle activity data in 3D space are accumulated in a file to be accessed by the agent and also for further investigation. The data produced are orientation and angle of rotations, muscles activity through 8 EMGs, triaxial accelerometer and gyroscope that can detect inertial forces on all three axes every frame.

Fig .6 shows the data taken from the sample while playing the game that illustrates EMG, accelerometer, gyroscope and EMGs data. This represents the muscle activity during the fruit pick up practice. The search path is variably determined by a distance limit T_{path}: $max_{P_1 P_4}$. According to [1] the length of the upper limb is in average in the range of 665 to 850 mm with an average hand length of 184 mm, forearm length of 260 mm and upper arm length of 285 mm. The threshold defined in RehabGame is the max length of the arm. It ensures safe movements and paths are taken with high confidence in the safety and less damage to the muscle. Fruits are generated based on the paths created by the upper limb and the agent's UCT simulation strategy based on various multi-phase playouts. A rewarding node in the tree might result in a loss at an attempt due to stochastic nature of the game. For that, a progressive expansion is performed for further exploration and exploitation of the tree.

6 Conclusion

RehabGame has the ability to modify and adapt its challenge level to match the skills of the player, avoiding frustration and encouraging engagement. It uses a variant of the Monte-Carlo Tree Search algorithm that involves the iterative building of search tree until some predefined computational constrains are achieved. At that point, the search is stopped and the best performing root action is returned. The current results may provide a starting point for other virtual reality studies and rehabilitation programs compatible with cognitive impairment. Future studies can use the current findings regarding differences in correlations as a way to generate future hypotheses and guide research. The study was conducted with two sample sizes. Future research should consider using a larger sample size, stroke, BI (brain injury) survivors and various ages.

References

[1] S. Adewusi, S. Rakheja, and P. Marcotte, "Biomechanical models of the human hand-arm to simulate distributed biodynamic responses for different postures," *International Journal of Industrial Ergonomics*, vol. 42, no. 2, pp. 249–260, 2012.

[2] D. Billings, A. Davidson, J. Schaeffer, and D. Szafron, "The challenge of poker," *Artificial Intelligence*, vol. 134, no. 1, pp. 201–240, 2002.

[3] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A survey of monte carlo tree search methods," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4, no. 1, pp. 1–43, 2012.

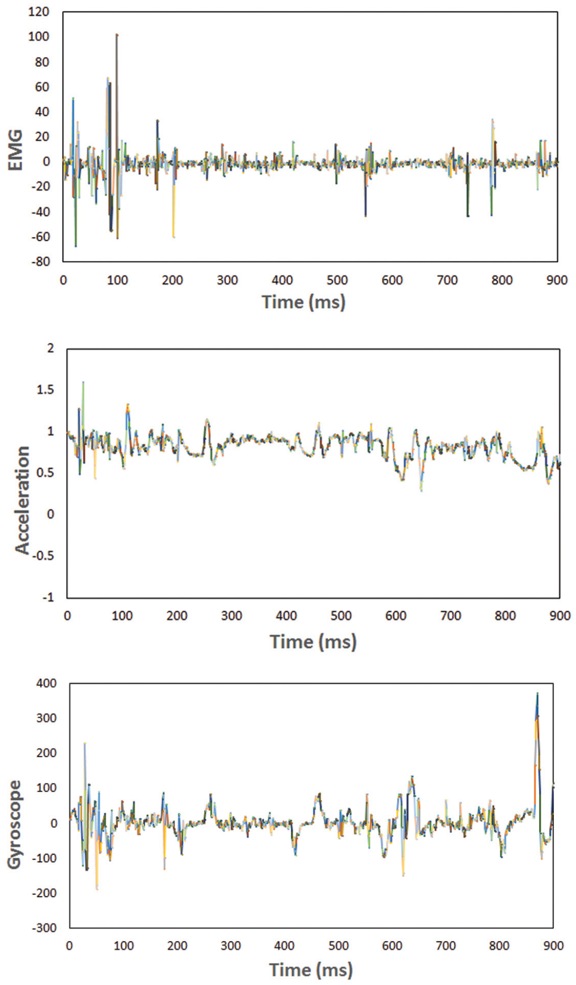


Fig. 6. Results taken from Myo (Gyroscope, Accelerometer and EMG).

[4] N. E. Carlozzi, V. Gade, A. S. Rizzo, and D. S. Tulsky, "Using virtual reality driving simulators in persons with spinal cord injury: three screen display versus head mounted display," *Disability and Rehabilitation: Assistive Technology*, vol. 8, no. 2, pp. 176–180, 2013.

[5] G. Chaslot, S. Bakkes, I. Szita, and P. Spronck, "Monte-carlo tree search: A new framework for game ai." in *AIIDE*, 2008.

[6] A. Criminisi, J. Shotton, and E. Konukoglu, "Decision forests for classification, regression, density estimation, manifold learning and semi-supervised learning," *Microsoft Research Cambridge, Tech. Rep. MSRTR-2011-114*, vol. 5, no. 6, p. 12, 2011.

[7] M. de Waard, "Monte carlo tree search with options for general video game playing," Ph.D. dissertation, Universiteit van Amsterdam, 2016.

[8] B. H. Dobkin, "Strategies for stroke rehabilitation," *The Lancet Neurology*, vol. 3, no. 9, pp. 528–536, 2004.

[9] H. Finnsson, "Generalized monte-carlo tree search extensions for general game playing." in *AAAI*, 2012.

[10] F. Frydenberg, K. R. Andersen, S. Risi, and J. Togelius, "Investigating mcts modifications in general video game playing," in *2015 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 2015, pp. 107–113.

[11] M. K. Holden, "Virtual environments for motor rehabilitation: review," *Cyberpsychology & behavior*, vol. 8, no. 3, pp. 187–211, 2005.

[12] M. K. Holden and T. Dyar, "Virtual environment training: a new tool for neurorehabilitation." *Journal of Neurologic Physical Therapy*, vol. 26, no. 2, pp. 62–71, 2002.

[13] N. Ikehata and T. Ito, "Monte-carlo tree search in ms. pac-man," in *2011 IEEE Conference on Computational Intelligence and Games (CIG'11)*. IEEE, 2011, pp. 39–46.

[14] M. F. Levin, P. L. Weiss, and E. A. Keshner, "Emergence of virtual reality as a tool for upper limb rehabilitation: incorporation of motor control and motor learning principles," *Physical therapy*, vol. 95, no. 3, pp. 415–425, 2015.

[15] S. M. Lucas, M. Mateas, M. Preuss, P. Spronck, and J. Togelius, "Artificial and computational intelligence in games: Integration (dagstuhl seminar 15051)," *Dagstuhl Reports*, vol. 5, no. 1, 2015.

[16] J. MacCormick, "How does the kinect work?" *Presentert ved Dickinson College*, vol. 6, 2011.

[17] S. Maycock and T. Thompson, "Enhancing touch-driven navigation using informed search in ms. pac-man," in *Computational Intelligence in Games (CIG), 2013 IEEE Conference on*. IEEE, 2013, pp. 1–2.

[18] A. Mirelman, P. Bonato, and J. E. Deutsch, "Effects of training with a robot-virtual reality system compared with a robot alone on the gait of individuals after stroke," *Stroke*, vol. 40, no. 1, pp. 169–174, 2009.

[19] S. D. Mohanty and S. Cantu, "Teaching introductory undergraduate physics using commercial video games," *Physics Education*, vol. 46, no. 5, p. 570, 2011.

[20] K. Q. Nguyen and R. Thawonmas, "Monte carlo tree search for collaboration control of ghosts in ms. pac-man," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 5, no. 1, pp. 57–68, 2013.

[21] R. L. O'Neil, R. L. Skeel, and K. I. Ustinova, "Cognitive ability predicts motor learning on a virtual reality game in patients with tbi," *NeuroRehabilitation*, vol. 33, no. 4, pp. 667–680, 2013.

[22] S. J. Page, S. Sisto, P. Levine, and R. E. McGrath, "Efficacy of modified constraint-induced movement therapy in chronic stroke: a single-blinded randomized controlled trial," *Archives of physical medicine and rehabilitation*, vol. 85, no. 1, pp. 14–18, 2004.

[23] T. Pepels, M. H. Winands, and M. Lanctot, "Real-time monte carlo tree search in ms pac-man," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 6, no. 3, pp. 245–257, 2014.

[24] M. Pirovano, R. Mainetti, G. Baud-Bovy, P. L. Lanzi, and N. A. Borghese, "Self-adaptive games for rehabilitation at home," in *2012 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 2012, pp. 179–186.

[25] G. Saposnik, R. Teasell, M. Mamdani, J. Hall, W. McIlroy, D. Cheung, K. E. Thorpe, L. G. Cohen, M. Bayley, S. O. R. C. S. W. Group *et al.*, "Effectiveness of virtual reality using wii gaming technology in stroke rehabilitation a pilot randomized clinical trial and proof of principle," *Stroke*, vol. 41, no. 7, pp. 1477–1484, 2010.

[26] B. Sheppard, "World-championship-caliber scrabble," *Artificial Intelligence*, vol. 134, no. 1, pp. 241–275, 2002.

[27] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore, "Real-time human pose recognition in parts from single depth images," *Communications of the ACM*, vol. 56, no. 1, pp. 116–124, 2013.

[28] G. V. Smith, K. H. Silver, A. P. Goldberg, and R. F. Macko, "task-oriented exercise improves hamstring strength and spastic reflexes in chronic stroke patients," *Stroke*, vol. 30, no. 10, pp. 2112–2118, 1999.

[29] E. A. Suma, D. M. Krum, B. Lange, S. Koenig, A. Rizzo, and M. Bolas, "Adapting user interfaces for gestural interaction with the flexible action and articulated skeleton toolkit," *Computers & Graphics*, vol. 37, no. 3, pp. 193–201, 2013.

[30] M. P. Tan, N. N. Wynn, M. Umerov, A. Henderson, A. Gillham, S. Junejo, and S. K. Bansal, "Arm span to height ratio is related to severity of dyspnea, reduced spirometry volumes, and right heart strain," *CHEST Journal*, vol. 135, no. 2, pp. 448–454, 2009.

[31] B. K. Tong and C. W. Sung, "A monte-carlo approach for ghost avoidance in the ms. pac-man game," in *2010 2nd International IEEE Consumer Electronics Society's Games Innovations Conference*. IEEE, 2010, pp. 1–8.

[32] B. K.-B. Tong, C. M. Ma, and C. W. Sung, "A monte-carlo approach for the endgame of ms. pac-man," in *2011 IEEE Conference on Computational Intelligence and Games (CIG'11)*. IEEE, 2011, pp. 9–15.

[33] E. Tsekleves, I. T. Paraskevopoulos, A. Warland, and C. Kilbride, "Development and preliminary evaluation of a novel low cost vr-based upper limb stroke rehabilitation platform using wii technology," *Disability and Rehabilitation: Assistive Technology*, vol. 11, no. 5, pp. 413–422, 2016.

[34] K. Ustinova, J. Perkins, W. Leonard, and C. Hausbeck, "Virtual reality game-based therapy for treatment of postural and co-ordination abnormalities secondary to tbi: A pilot study," *Brain injury*, vol. 28, no. 4, pp. 486–495, 2014.

[35] N. A. Vien, W. Ertel, V.-H. Dang, and T. Chung, “Monte-carlo tree search for bayesian reinforcement learning,” *Applied intelligence*, vol. 39, no. 2, pp. 345–353, 2013.

[36] M. Villiger, “Virtual reality rehabilitation in spinal cord injury patients,” Ph.D. dissertation, Diss., Eidgenössische Technische Hochschule ETH Zürich, Nr. 20482, 2012, 2012.

[37] M. Villiger, M.-C. Hepp-Reymond, P. Pyk, D. Kiper, K. Eng, J. Spillman, B. Meilick, N. Estévez, S. S. Kollias, A. Curt *et al.*, “Virtual reality rehabilitation system for neuropathic pain and motor dysfunction in spinal cord injury patients,” in *2011 International Conference on Virtual Rehabilitation*. IEEE, 2011, pp. 1–4.

[38] P. L. Weiss, H. Sveistrup, D. Rand, and R. Kizony, “Video capture virtual reality: a decade of rehabilitation assessment and intervention,” *Physical Therapy Reviews*, 2013.

[39] D. Whitehouse, “Monte carlo tree search for games with hidden information and uncertainty,” Ph.D. dissertation, University of York, 2014.

[40] M. H. Winands, Y. Björnsson, and J.-T. Saito, “Monte-carlo tree search solver,” in *International Conference on Computers and Games*. Springer, 2008, pp. 25–36.

[41] G. Zhu, “Real-time elective admissions planning for health care providers,” 2013.

Shabnam Sadeghi Esfahlani S. S. E. is a senior lecturer in Computer Gaming Technology, Anglia Ruskin University, Cambridge, UK. She received her Ph.D. in 2012 in Mechanical Engineering. She has research experience in virtual reality based rehabilitation game, model reliability, robustness and design optimization using applied Finite Element Analysis and statistical models. Her research interests include development of edutainment games for adults, and high-performance energy absorbing materials and meta-models.