33rd Annual International Conference of the IEEE EMBS
Boston, Massachusetts USA, August 30 - September 3, 2011

Smart Calibration for Video Game Play by People with a Movement Impairment

Sergi Perez, Raul Benitez, *Member, IEEE,* David J. Reinkensmeyer, *Member, IEEE*

Abstract— **People with movement impairment often cannot move with the range, speed, or acceleration required to play an off-the-shelf video game. This paper describes a smart calibration algorithm designed to facilitate video game play by people with movement impairment. The algorithm continuously adapts the calibration of the gaming input device by comparing the maximum range of motion measured in previous time periods, then adjusting the current required range of motion based on their difference. In several experiments with simple acceleration-based video games using a Nintendo Wiimote, we show that the algorithm adapts the calibration to allow healthy users to play the game with their full available range of acceleration without need for a special calibration protocol. Importantly, the algorithm described here can be used without altering the game software by inserting a hardware or software module between the gaming input device and the game console. Thus, the algorithm can be used with off-the-shelf video games without altering their source code.**

I. Introduction

Movement impairment arises due to neural injuries such as stroke, spinal cord injury, traumatic brain injury, and cerebral palsy, or due to progressive diseases such as multiple sclerosis, ALS, and Parkinson's disease [1]. Aging also reduces muscle strength [2,3], coordination [4], and range of motion (ROM) [5]. As a result, people with movement impairment often have difficulty playing video games, particularly if they cannot create the range, speed, or acceleration of movement necessary to reach movement targets defined by the game. This difficulty results in decreased access to video games for people with a movement impairment, which limits entertainment options and reduces social participation. Further, many rehabilitation clinics are interested in using video game consoles such as the Nintendo Wii [6,7] to motivate movement exercise. If a patient's movement impairment is too severe, then he or she cannot exercise by playing standard video games. In addition, as a patient's level of movement impairment changes during movement recovery, it would be desirable to have a game that automatically adapts to new movement ability, challenging the patient to improve further.

Previous work on developing games for movement training for people with a motor impairment have typically taken the approach of developing custom games that require a periodic calibration; the calibration procedure typically requires the participation of a supervising rehabilitation therapist [8,9]. The objective of the project was to develop an algorithm that continuously estimates the range of motion of a player given four design constraints: the algorithm receives no feedback from the game, the distribution of targets is unknown, the initial range of motion is unknown, and the range of motion can change during game play. Such an algorithm could be used without need for special calibration procedures or supervision from a rehabilitation therapist. It could also be used with existing video games without modifying their source code.

To satisfy these design constraints, we developed an algorithm that continuously adapts the calibration based on the difference in range of motion measured in previous time periods. We first describe the algorithm, then present experimental results that demonstrate that the algorithm allows users to play a simple acceleration-driven game with their full available range of acceleration.

II. Methods

A. Calibration Algorithm

We consider first a simple one-dimensional game that uses the Wiimote, an acceleration based sensor. The user must shoot an arrow to a target. The initial velocity of the arrow depends on the maximum acceleration of the z axis of the Wiimote, recorded during a fixed window of time of duration T, which begins after the vertical acceleration of the Wiimote passes a small threshold. The arrow is assumed to move with the dynamics of a point mass in earth's gravity field. The game presents multiple targets to the user at different positions of the screen.

Let $y \in D_y$ describe the input acceleration to the game and let $x \in D_x$ describe the sensor value the user generates (e.g. in our game, the maximum acceleration measured in a window of T seconds after the sensor reading exceeds the threshold x_{thres}), with:

$$D_y = \{y \in \mathbb{R}: 0 < y < G\}$$
$$D_x = \{x \in \mathbb{R}: 0 < x < R\} \quad (1)$$

G is the maximum input acceleration that the game can demand and R is the maximum input value that the user can create (e.g. maximum acceleration the user can create with a

Manuscript received April 15, 2011. This work was supported in part by the Balsells Fellowship program, the NIDRR Rehabilitation Engineering Research Center on Rehabilitation Robotics and Telemanipulation, H133E070013, NIH-R01HD062744-01 from NCMRR and Grant Number UL1 RR031985 from the National Center for Research Resources (NCRR), a component of the National Institutes of Health (NIH) and the NIH Roadmap for Medical Research.

Sergi Perez is with the Mechanical and Aerospace Department, University of California, Irvine CA 92697 USA (e-mail: sergi.perez@uci.edu).

Raul Benitez is with the Automatic Control Department, Universitat Politecnica de Catalunya, Comte Urgell 187 08036 Barcelona (e-mail: raul.benitez@upc.edu). R. Benitez acknowledges support from the Spanish Ministry of Education (MICINN) through project DPI 2009-06999.

David J. Reinkensmeyer is with the Departments of Mechanical and Aerospace, and Anatomy and Neurobiology University of California, Irvine CA 92697 USA (phone: (949) 824 5218, e-mail: dreinken@uci.edu).

Wiimote). In many games, G will be equal to R (e.g. our game requires the user to saturate the Wiimote accelerometer to hit a very high target by creating the maximum acceleration the accelerometer can read) but in other cases it may not. For example, if we design our game to require fine, slow control of the input device, then G may be less than the maximum value the sensor can read. The salient point is that every video game is designed to expect a range of sensor values. The algorithm presented here requires knowledge of this range. Note that this range can be measured by recording the sensor values (i.e. output of the gaming device) during normal game play for any game.

Now consider a user who cannot generate sensor values that are large enough to play the game, such that all the x the user generates satisfy $x \leq R < G$. This user will have limited success in playing the game. To assist the user we intercept the output of the sensor, scale it by a calibration parameter $a[i]$ and then send it to the game so that the user can play the game successfully, as seen in Fig. 1. The parameter i refers to the i^{th} iteration of the algorithm (i.e. the i^{th} update of a) where we define an iteration to occur every N throws of the arrow to the target. The relation between x and y is given by the linear transformation $f(x)\colon D_x \rightarrow D_y$, where:

$$y = f(x) = a[i]\,x \qquad (2)$$

If we know the maximum acceleration that the user can create then the calibration coefficient should be set to $a = G/R$. We assume that we don't know R; thus, the goal of the algorithm is to find the estimate $\hat{R}$, and find $\hat{a}$ such that the maximum acceleration that the user can generate is scaled to the maximum acceleration that the game requests for successful play:

$$G = \hat{a}\,\hat{R} \qquad (3)$$

We put the caret over a variable to distinguish the estimate from the real value.

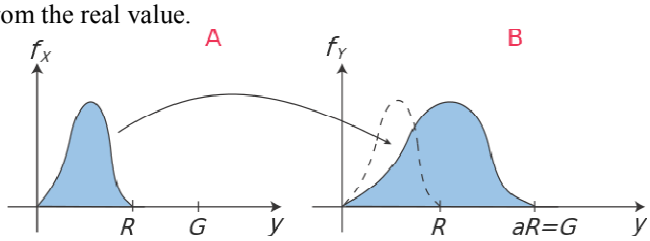

Fig. 1. Distribution of inputs to the game for an impaired subject. A. With no scaling, the subject can only reach those targets that are in the range $[0\ R]$. B. Using a linear transformation we can modify the distribution of inputs to the game and allow the injured subject to use the full ROM that the game requires, $y \in [0\ G]$.

One strategy for finding $\hat{a}$ is to ask the user to move as quickly as possible to measure $\hat{R}$. Then we choose $\hat{a} = G/\hat{R}$. However, this strategy requires a special calibration procedure that is independent of the game. Moreover, if the user's movement ability changes, due to motor learning, fatigue or disease progression or recovery, the calibration procedure must be performed again. Instead, we developed a strategy that continually estimates $\hat{R}$ during game play without requiring a special calibration. This estimate is used to calculate the current value of the calibration coefficient using $\hat{a} = G/\hat{R}$ and therefore is the maximum sensor value demanded by the game at the i[th] iteration of the algorithm. We expect the maximum value of x used in the game to increase as $\hat{R}$ increases until $\hat{R} = R$. When this happens the user will not be able to reach those targets that require $x > R$ as it is shown in the Fig. 1.

The algorithm for finding this estimate is as follows:

1) The initial estimation of $\hat{R}$ is G.

2) For each movement the user makes, the sensor sends x to the algorithm (i.e. in our game the maximum acceleration measured in a window of T seconds after the sensor reading exceeds the threshold x_{thres}).

3) If the number of x received (i.e. number of game movements made) in the current iteration i is smaller than N the algorithm uses the previous estimation $\hat{R}[i]$ and thus $\hat{a}$ to scale and sends $y = \hat{a}[i]x$ to the game.

4) If the number of x received in the current iteration i equals N the algorithm calculates $x_{(N)}$ which is the maximum value of x during the observation period of N throws of the arrow to the target, in our game.

$$x_{(N)} = \max_{iN+1<n\leq(i+1)N} x_n \qquad (4)$$

Let J be a cost function defined by:

$$J\big(\hat{R}[i+1]\big) = (K_1/2)\big(\,x_{(N)}[i] - \hat{R}[i+1]\,\big)^2 - K_2\,\hat{R}[i+1]\,|\,x_{(N)}[i] - x_{(N)}[i-1]| \qquad (5)$$

Now, the algorithm estimates $\hat{R}[i+1]$ by searching for the $\hat{R}[i+1]$ that minimizes (5). Then the calibration algorithm is set using $\hat{a} = (1+c)G/\hat{R}$, where c is defined as the "challenge factor". The purpose of the challenge factor is to provide the software designer with a parameter that allows precise control over how much the user is challenged, because c > 0 requires the user to use greater range of motion than the ongoing estimate of the actual range of motion.

To see the rationale for the cost function, consider the condition $\hat{a} < a$. In this case, the range of motion demanded by the game from the user will be greater than the range of motion of the user ($\hat{R} > R$). To see this, we note that in this case $\hat{R} = G/\hat{a} > G/a = R$. So we expect $x_{(N)}[i] = x_{(N)}[i-1] = R$. In this case, (5) reduces to:

$$J\big(\hat{R}[i+1]\big) = (K_1/2)\big(\,x_{(N)}[i] - \hat{R}[i+1]\,\big)^2 \qquad (6)$$

If we minimize (6) then our estimate $\hat{R}$ will be accurate ($\hat{R} = x_{(N)}[i] = R$). Now consider when $\hat{a} > a$. In this case, we expect that the user achieves the range of motion demanded by the game, so $x_{(N)} = \hat{R}$. Then the first term of (5) tries to make $\hat{R} = R$; in other words, it estimates the user's range of motion to be the maximum range achieved by the user in the previous iteration. However, we know that the user can generate a greater range of motion than we are currently measuring. Thus, the second term of the cost function is introduced. This term decreases the total cost as $\hat{R}[i+1]$ increases, and therefore causes the search for $\hat{R}[i+1]$ to move toward increasing values of $\hat{R}[i+1]$. The influence of this second term decreases when $x_{(N)}$ changes less between iterations, which we expect to happen when

$\hat{a} < a$. We expect this to happen when $\hat{R} > R$ since the overestimation of R implies that the user has to create higher values of x with more frequency and thus we can expect $x_{(N)}[i] \cong x_{(N)}[i-1]$.

Finding the $\hat{R}[i+1]$ that minimizes the cost function is a convex optimization problem. The minimum condition for $\hat{R}[i+1]$ can be ensured by accomplishing the Second Order Sufficient Conditions, given by (7).

$$\nabla J\big(\hat{R}[i+1]\big) = 0 \qquad \nabla^2 J\big(\hat{R}[i+1]\big) > 0 \qquad (7)$$

We solved (7) to find the $\hat{R}[i+1]$ (8) that will be a minimum if $K_1 > 0$:

$$\hat{R}[i+1] = x_{(N)}[i] + K_1/K_2 \left| x_{(N)}[i] - x_{(N)}[i-1] \right| \qquad (8)$$

Equation 8 thus serves as the update law for the estimate of the range of motion of the user. As can be seen, the estimated range of motion is the maximum acceleration achieved in the previous observation period (i.e. $x_{(N)}[i]$), incremented by the difference in measurements in the past two observation periods ($x_{(N)}[i]$ *and* $x_{(N)}[i-1]$).

B. Experiment Design

In order to evaluate the algorithm we developed the arrow shooting game described previously, using the Wiimote as the input device to play the game. Since this accelerometer (ADXL330) can be easily saturated by a non-impaired subject, we limited the accelerations applied to the Wiimote by attaching it to a lever. The users played the game by pushing on one end of the lever at a distance of 63 cm from the pivot. The Wiimote was attached 8 cm from the pivot on the same side as the lever. Thus, the acceleration measured by the Wiimote was approximately 50% of the maximum acceleration the Wiimote can sense.

We performed two experiments. In a first experiment we measured the evolution of the estimation of the range of motion during game play given a fixed distribution of targets. The data for this experiment was collected from 7 non-impaired subjects playing the Wiimote game throwing 20 arrows. We used the calibration algorithm with $K_1 = 3, K_2 = 1, c = 0.2$ and $N = 4$. The game shows the targets in 6 different positions: $Y = [15\ 30\ 46\ 61\ 76\ 92]$, with a frequency for each position given by the probability mass function $f_t = [1\ 2\ 3\ 3\ 4\ 4\ ]/17$. Before starting the games we asked every user to create the maximum acceleration possible to determine the real R. This value was not used in the algorithm, but we used it to analyze how well the algorithm converged on R.

In a second experiment, we analyzed how the convergence properties of the algorithm depended on the distribution of game targets. Let y_t be the required input to the game to precisely reach a target and let $x_t = y_t/\hat{a}$. If most of the targets are distributed at $y_t \ll G$, the x received during game play will be smaller than the real $R$of the user, and $\hat{R}$ will be underestimated. By modifying N we increase the number of x used for every iteration of the algorithm, and thus we increase the chances of receiving a $x_{(n)} \cong \hat{R}$.

4 subjects played the game by throwing 40 arrows to 40 targets. The position of the targets was calculated using 4 different distributions of y_t that are shown in the Fig. 2. For every distribution and subject we used 6 different values of $N = [1\ 2\ 3\ 4\ 5\ 6]$. We used the calibration algorithm with $K_1 = 3,\ K_2 = 1,\ c = 0.2$. The first 3 distributions were artificially generated to test the algorithm while the last distribution was calculated by playing successfully the game Wiisport golf, to add a representative distribution of a game.

We calculated the means of the error on the estimation (9) for the 4 subjects:

$$e = \sum_x \left| R - \hat{R} \right| \qquad (9)$$

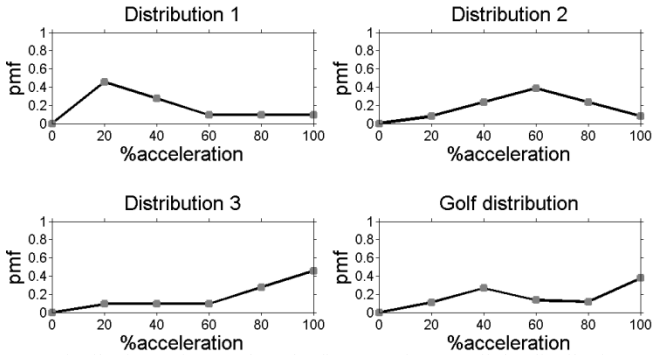

Fig. 2. Distributions of y_t used on the first experiment. All the distributions are discrete. In all games the targets are placed in 5 possible positions that correspond to 20-40-60-80-100% of G.

Note that for the experiments we considered the worst scenario and the game allows $y > G$. With this inequality, when $\hat{R} < R$ and the user creates an acceleration $x > \hat{R}$ he will see the arrow surpassing the position of the higher target and will reduce x on future throws. Since x is reduced we expect $x_{(n)}$ to be smaller than R and the error of the estimator will increase. By limiting the maximum value y that we send to the game to G we can avoid this problem.

III. Results

A. Experiment 1: Testing Convergence of Algorithm

Fig. 3 shows accelerations measured during a game of twenty throws for one subject. The algorithm started with the initial estimate of $\hat{R} = G$, and thus during the first 4 throws it was difficult for the subject to hit the targets as he needs to create $x_t > R$ to reach some targets. Since he cannot create those accelerations he creates $x \cong R$. After an observation period of N = 4 throws, the algorithm calculated a new value for $\hat{R}$ using the first 4 values of x it received. Now $\hat{R}$ is reduced and closer to R and thus the x_t required are smaller, allowing the user to be able to reach more targets.

Fig. 4 shows the mean of the errors on the estimation for the 7 subjects from this experiment. The mean of the means of the errors is -0.0137, indicating a global behavior towards a 0 error estimation given a distribution of targets representative of G.

B. Experiment 2: Effect of Target Distribution

In the second experiment, we examined how the target distribution affected the estimated error for 4 subjects as shown in Fig. 5. The distribution 1 is more restrictive for the algorithm than the rest of the distributions, given that most

$y_t \ll G$. As N increases the error on the estimation decreases since the probability of receiving a x representative of R increases. The distribution 3 is the most favorable as most $y_t = G$. For this distribution the total error remains constant as N increases.

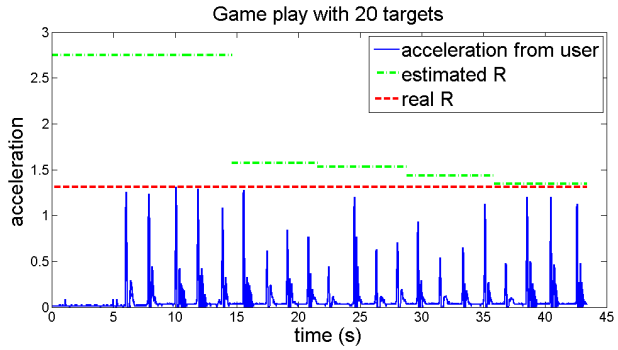

Fig. 3. Complete game play of one of the subjects. Every peak on the acceleration corresponds to an arrow thrown in the game. The unit of the acceleration is $g = 9.8\ m/s^2$.

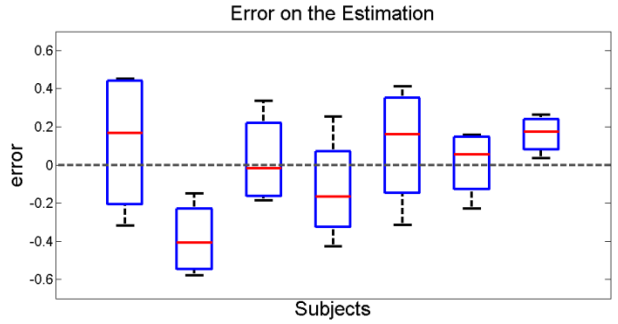

Fig. 4. Statistics of the residuals $\hat{R} - R$. A positive error indicates that the estimation of the range of motion was, on average, larger than the real range of motion, whereas a negative error implies underestimation. Every box represents a subject where the central mark represents the median, the edges of the box are the 25th and 75th percentiles and the whiskers extend to the extreme residuals.

IV. Discussion

This paper described an algorithm that estimates the capability of movement of a player in real time using sensor values created by the player. For therapeutic applications, this algorithm permits the use of already developed games (those which require sensor input to the game) as rehabilitation exercises, where the player can be challenged while playing without the aid of a therapist.

One limitation of the algorithm is related to the lack of feedback from the game. If the game requires high sensor values and the player can only create small values, the algorithm will receive these small values as inputs. If the game requires small sensor values and the player acts accordingly, the algorithm will also receive small values. In both situations the algorithm is receiving the same data but the cases are opposite. One possible way to address this issue would be to increase N for those games on which we do not expect the distribution to be representative of G. But increasing N also implies that the algorithm will adapt slowly for changes on the movement capabilities of the player. This tradeoff is significant for higher dimensions where we might be interested in following the position of the player and continuously adapting the estimated range of motion. Another possible limitation of the algorithm is that that frequently changing $\hat{R}$ may make it more difficult for the player to create an internal model of the scaling factor a. In other words, if the calibration is changed too rapidly, then the user may experience increased movement errors. Finding the right balance between adaptation and error is an important direction for future research.

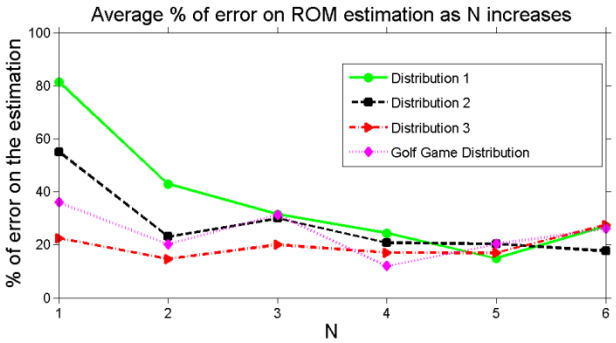

Fig. 5. Percentage of the mean error on ROM estimation for the Experiment 1. The error converges around 20% for $N \geq 4$. The origin of this error are the first N samples, where we assume $\hat{R} = G$. Since $G > R$ there is an absolute error of $|G - R|$ during the first N samples that increases the total error.

Future work will also estimate the distribution of targets $\hat{y_t}$ during game play to simulate feedback from the game, use an adaptive N, updating $\hat{R}$ every sample received, and extending the algorithm for multi-dimensional game play.

References

[1] B.H. Dobkin, *The clinical science of neurologic rehabilitation*, Oxford University Press, 2003.

[2] W.R. Frontera, V.A. Hughes, R.A. Fielding, M.A. Fiatarone, W.J. Evans, and R. Roubenoff, "Aging of skeletal muscle: a 12-yr longitudinal study," *J Appl Physiol*, vol. 88, 2000, pp. 1321-1326.

[3] L. Larsson, G. Grimby, and J. Karlsson, "Muscle strength and speed of movement in relation to age and muscle morphology," *J Appl Physiol*, vol. 46, 1979, pp. 451-456.

[4] M. H Woollacott, A. Shumway-Cook, and L. M Nashner, "Aging and posture control: changes in sensory organization and muscular coordination.," *International journal of aging human development*, vol. 23, 1986, pp. 97-114.

[5] S.L. Wolf, H.X. Barnhart, N.G. Kutner, E. McNeely, C. Coogler, and T. Xu, "Selected as the best paper in the 1990s: Reducing frailty and falls in older persons: an investigation of tai chi and computerized balance training.," *Journal of the American Geriatrics Society*, vol. 51, 2003, pp. 1794-803.

[6] J.E. Deutsch, M. Borbely, J. Filler, K. Huhn, and P. Guarrera-Bowlby, "Use of a low-cost, commercially available gaming console (Wii) for rehabilitation of an adolescent with cerebral palsy.," *Physical therapy*, vol. 88, Oct. 2008, pp. 1196-207.

[7] L. Yong Joo, T. Soon Yin, D. Xu, E. Thia, C. Pei Fen, C.W.K. Kuah, and K.-H. Kong, "A feasibility study using interactive commercial off-the-shelf computer gaming in upper limb rehabilitation in patients after stroke.," *Journal of rehabilitation medicine : official journal of the UEMS European Board of Physical and Rehabilitation Medicine*, vol. 42, May. 2010, pp. 437-41.

[8] L. Geurts, V. Vanden Abeele, J. Husson, F. Windey, M. Van Overveldt, J.-H. Annema, and S. Desmet, *Digital games for physical therapy*, New York, New York, USA: ACM Press, 2011.

[9] E.T. Wolbrecht, D.J. Reinkensmeyer, and J.E. Bobrow, "Pneumatic Control of Robots for Rehabilitation," *The International Journal of Robotics Research*, vol. 29, May. 2009, pp. 23-38.