Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

Control of a Smart Walker for Training Using Interaction-Energy and Personalized Parameters ⋆

Denis Stogl * **Daniel Zumkeller** * **Manuel Muth** * **Björn Hein** *,**

* *Institute for Anthropomatics and Robotics, Intelligent Process Automation and Robotics Lab (IAR-IPR), Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany (e-mail: denis.stogl, bjoern.hein@kit.edu, daniel.zumkeller, manuel.muth@student.kit.edu).*
** *Karlsruhe University of Applied Science, Karlsruhe, Germany, (e-mail: bjoern.hein@hs-karlsruhe.de)*

Abstract: Smart walkers with admittance controller usually have limited dynamics and low maximal velocity to provide stable and safe behavior. To enable physically challenging training with smart walker control strategies enabling faster dynamics is needed. However, in certain situations, this can lead to instabilities, which further complicates the finding of suitable parameters for the envisioned training functionalities. To overcome these issues, we have introduced an interaction-energy limiter and developed a strategy to automatically determine individual user parameters for an adaptive admittance controller. The energy limiter bounds the controller and training elements to avoid uncomfortable and dangerous situations. These training elements are placed on a 2D map of a training environment. If the user passes these training elements with our smart walker – RoboTrainer – they are triggered. Therefore we call them spatial control actions. We can show that interaction-energy limiter successfully avoids instabilities and dangerous situations when spatial control actions are triggered. The evaluation with 22 users, which used RoboTrainer with and without individualized parameters, successfully demonstrate the benefits of the parameterization method. The presented method could be generally valuable for the implementation of smart walkers in everyday life since it provides a solution for dealing with users with different skills and a solution for safe interaction with smart walkers using high-dynamic control.

Keywords: Human-machine interface, Human-machine interaction, Admittance control, Model-based control, Human-centered design, Nonlinear gain, Mobile robots

1. INTRODUCTION

Technological developments in recent decades enabled robotic systems to emerge in our everyday life. Smart walkers (SWs) (i.e., motorized and sensor-equipped walkers) are one example of those systems. Recent research and development on smart walkers focus on their design and application with elderly, disabled, or injured persons. However, control of smart walkers in these scenarios is done by sharply limiting the walker's dynamics (e.g., PAMM in Yu et al. (2003) interacts with forces of approx. 20 N). Such a solution is not suitable for an active force-based and physically challenging interaction with a user, which is needed to realize training with a smart walker.

Research on dynamic control of smart walkers provides a variety of methods, which can be mainly classified by the type of input device used and the walker's low-level control. Studies that use a force sensor as an input device and control the velocity of a walker use an admittance control approach (Dubowsky et al. (2000); Yu et al. (2003); Chuy et al. (2007)). Another challenge is the robust control of a smart walker covering a variety of persons with different physical strength and interaction preferences. There are few proposals in the literature to adapt walkers' parameters to a user (e.g., Chuy et al. (2005); Martins et al. (2014)).

In our previous works Stogl et al. (2019a,b), we proposed a smart walker – RoboTrainer – for strength and motor training which uses an admittance controller with fixed parameters. Here we describe an extended control approach considering interaction-energy between a user and RoboTrainer and a method to determine a personalized control behavior for each user.

The main contributions are (I) an extension of the energy limit approach providing better performance compared to state of the art; (II) an interaction-energy approach for avoiding unwanted behaviors when using active training elements – spatial control actions (SCAs); (III) a method for allowing personalized interaction forces and (IV) a non-linear adaption function for the user's forces considering the actual speed of RoboTrainer. We extend the interaction-energy limiter approach (i.e., passivity) introduced by Chuy et al. (2007) for use with a walker with faster dynamics and, at the same time recognizing the user's intention when suddenly changing driving direction. The interaction-energy approach for SCAs implements two concepts: passivity and safety. The passivity concept reduces the influence of SCAs at lower interaction forces, and the safety protects the user by limiting RoboTrainer from moving unintentionally in his direction. The method for individualized interaction forces considers input forces separately in all three

⋆ The research presented in this paper is supported by Federal Ministry of Education and Research of Germany within the project *Learning Robot-Based Systems for the Neuro-Muscular Training - RoSylerNT* (Nr. 16SV7855).

degrees of freedom and adjusts the speed of the RoboTrainer to the actual walking speed of a user. The method does not need any external sensor nor markings in the training environment. This method, as well as the non-linear adaption function, are evaluated in an experiment with 22 persons between 20 to 40 years.

Section 2 gives an overview of relevant related work on admittance control approach and its parameterization for smart walkers. Section 3 explains the concepts in detail and section 4 presents relevant software components of our SW – RoboTrainer. Section 5 presents experimental results for the interaction-energy limiting approach and outcomes from the evaluation of the parameterization approach with the users. Section 6 finalizes the paper with a conclusion and an outlook.

2. RELATED WORK

An active research community on smart walkers (SW) exists already for over two decades, starting with the PAM-AID by Lacey and M. Dawson-Howe (1998), which uses a joystick as an input device. Later, researchers began to use force sensors as a human-machine interface in similar walking aid systems (e.g., Dubowsky et al. (2000)), and admittance control became a sort of a standard (e.g., Yu et al. (2003); Chuy et al. (2005)) for control of SWs. Force-torque sensors are a widespread and intuitive human-machine interface for a SW and therefore used in many devices Martins et al. (2012); Solenne et al. (2016).

Yu et al. (2003) proposes a velocity-dependent adaption of the damping factor of the admittance rule:

$$\boldsymbol{M} \cdot \ddot{\boldsymbol{x}}(t) + \boldsymbol{D} \cdot \dot{\boldsymbol{x}}(t) = \boldsymbol{F_h}(t) \tag{1}$$

where the $\boldsymbol{M}$ is the desired mass and $\boldsymbol{D}$ the desired damping of the smart device while $\boldsymbol{F_h}$ being the input force from a user. The proposed damping adaption is shown in (2) where damping is reduced with increasing speed V between maximal d_{max} and minimal d_0 value. V_{max} represents the maximal speed of the walking aid (i.e., PAMM).

$$D = d_{max} - \frac{d_{max} - d_0}{V_{max}}|V| \tag{2}$$

We used this approach as a basis for the individual adaption of the input forces to RoboTrainer's speed. We propose an extension from linear to non-linear adaption.

During interaction with RoboTrainer, we expect that users use stronger forces, especially when spatial control actions (SCAs) are active. In that case, a user needs to react actively to changes in RoboTrainer's behavior. If the user then stiffens his arms, this can lead to an oscillatory behavior, as described in Chuy et al. (2007). The authors also propose a solution for this by modifying the admittance rule (1) to:

$$\boldsymbol{M} \cdot \ddot{\boldsymbol{x}}(t) + \boldsymbol{D} \cdot \dot{\boldsymbol{x}}(t) = k \cdot \boldsymbol{F_h}(t) \tag{3}$$

where factor $k \in [0\ 1]$ adjusts the dynamics of a SW to achieve stable behavior of the human-walker system. The authors also propose an approach for the detection of oscillatory situations by monitoring the energy of the system:

$$E_{system} = \int_0^{t_{curr}} \boldsymbol{F_h} \cdot \dot{\boldsymbol{x}} dt \tag{4}$$

where E represents the energy, $\boldsymbol{F_h}$ the input force from a user and $\dot{\boldsymbol{x}}$ the velocity of the SW. The energy is calculated at each control step from the system start $t = 0$ and until current time $t = t_{curr}$. Exact detail for adjusting factor k from (3) are not provided, other than $k < 1$.

In this paper, we implement, extend, and evaluate the approach from Chuy et al. (2007) to detect the oscillatory behavior of RoboTrainer. This extension (c.f. Stogl et al. (2019b)) solves a drawback of the original method, which accumulates too much energy over time and therefore delays the detection of oscillations. Furthermore, we add an intention recognition mechanism, which avoids the reduction of scaling factor k when sudden, aperiodic changes in the input force happen.

For the second part of this paper about user-specific control parameters for SW, there is not much relevant literature. There is a calibration strategy for novel handle interface, presented in Martins et al. (2014), and an approach to adapt SW's behavior by shifting its center of rotation (CoR) Chuy et al. (2005). Other publications use fuzzy and state-space control, but these are not relevant to our work. The calibration strategy from Martins et al. (2014) is similar to our approach since we also introduced a parameterization sequence to measure the user's input and derive from this the controller parameters. Martins et al. (2014) considers only forward movements of a smart walker and does not enable automatic calculation of individualized controller parameters. The approach from Chuy et al. (2005) needs specific reference paths in the environment of the SW to calculate the adaption of CoR. Our approach does not have such requirements and is used in an open space.

3. CONCEPT

This section describes the interaction-energy limiting approach, and per-user parameterization implemented and evaluated with RoboTrainer. The first part considers a situation in which comes to unintended oscillations between user and SW caused by user suddenly changing input force vector or stiffness of its arms. The second part details the approach for limiting active forces caused by spatial control actions (SCAs) and the reasoning behind it. The last two subsections explain a process for the automatic determination of per-user maximal forces and force scaling factors according to RoboTrainer's velocity.

3.1 Intention-Energy Limiting

During the previous evaluations of RoboTrainer's functionalities by inexperienced users, we have observed that they are often using powerful forces and stiff their hands to try to hold RoboTrainer in place. This behavior decreases the stability of the user-SW system and may cause oscillations. These oscillations endanger a user and may break some parts of RoboTrainer's hardware. A very similar case is already observed by Chuy et al. (2007). Therefore, at first, we implemented their approach for the calculation of interaction-energy. This approach did not provide satisfactory results, probably because of the different dynamics of Chuy et al. (2007)'s system and RoboTrainer. An example result of this algorithm is shown in Fig. 2, where it can be seen that the approach from Chuy et al. (2007) accumulates the energy of the system very fast, which is not able to fall under zero in the case of oscillations. A phase shift of approximately $\frac{1}{4}$ of the period between the input force and RoboTrainer's velocity in the case of oscillations causes the short negative power of the system (c.f. 4) at each period. In our experiment, this leads to a slow increase Chuy et al. (2007)'s energy instead of a decrease.

Based on these observations, we have adapted the energy calculation from (4) to sliding-discrete form (5), where only local energy of the last N measurements with period Δt between them,

is calculated. The sliding form provides a faster decrease of the calculated system energy in case of oscillations (cf. Fig. 2). Based on this observation and proposal from Chuy et al. (2007), we implemented a linear decrease of factor k from (3) when the negative energy is detected, and linear increase in case of the positive energy. We did this for each degree of freedom, for which also a desired adaption factor k_d in (6) is defined. In this context, h is the proportional gain which models disturbance force feedback caused by user's stiff arms, M_a is the actual mass of the RoboTrainer and M_d desired mass. The desired adaption factor defines minimal factor to damp input forces in case of the oscillatory behavior.

$$\boldsymbol{E}_{system,n} = \sum_{i=n-N}^{n} \boldsymbol{F}_{h,i} \dot{\boldsymbol{X}}_i \Delta t \tag{5}$$

$$k_d = 1/\left(\frac{h * M_a}{M_d} - h\right) \tag{6}$$

Differing from many SWs, RoboTrainer can move freely in all directions. Therefore it is possible to suddenly change its movement direction for $180\,\mathrm{deg}$ (i.e., from forward to backward, or from left to right). Preparing for the evaluation of this work, we realized that the dynamic of the system slows down when suddenly switching directions. This velocity decline is caused by the sliding integral approach for energy calculation. The sliding integral approach from (5) is very sensitive to a large negative power, which leads to the reduction of the scaling factor (6). This effect is shown in Fig. 3 and explained in detail in section 5.

To achieve the targeted scenario with RoboTrainer, we had to further extend the method for the adaptation to recognize the user's intention. The recognition system detects one or multiple direction changes in a specified time interval. During the first direction change, even if the energy of the system is negative, the scaling factor is not reduced, but the controller waits on the further reaction of the user. If another direction change is detected or the energy is negative also after the time interval, the scaling factor is reduced. If there is no further direction change, as soon as the energy is positive again, the scaling factor is reset to its maximal value (i.e., 1). In this way, RoboTrainer does not limit its dynamic on sudden direction change, but it is still able to detect oscillatory behavior based on negative system energy.

3.2 Energy Limiting for Spatial Control Actions

The goal of energy limiting for SCAs is to achieve passive behavior and increase safety when behavior modifiers are influencing RoboTrainer. SCAs are introduced in Stogl et al. (2019a) (called *control modalities*) and explained in detail in Stogl et al. (2019). In short, the SCAs are virtual elements which are defined along a training path to modify RoboTrainer's behavior to make training more versatile and challenging. They are activated if the user is in their influence radius. In general, SCAs can be defined and configured arbitrarily, unrelated from user's characteristics (e.g., interaction forces and walking speed) and controller parameters. As described in Stogl et al. (2019), SCAs have there own dynamics which enables better control over the influence on the main controller, but also makes the control over all parameters more complex. For these reasons, it is possible that the virtual force, and consequently, virtual velocity, is higher than the user's input, which could lead to dangerous situations for a user. This effect is due to the influence of SCAs on the output velocity of RoboTrianer (7).

$$\dot{\boldsymbol{X}}_{n,out} = \dot{\boldsymbol{X}}_{n,main} + \sum_{a}^{A} \dot{\boldsymbol{X}}_{n,a} \tag{7}$$

n is the discrete controller step, $\dot{\boldsymbol{X}}_{n,main}$ is the velocity from the main controller using (8) and $\dot{\boldsymbol{X}}_{n,a}$ is the output velocity of the SCA a from the set of active SCAs A. RoboTrainer main controller is time-discrete admittance rule (1):

$$\dot{\boldsymbol{X}}_n = \boldsymbol{F}_{n-1} \cdot \boldsymbol{K} \cdot (1 - e^{-\frac{1}{r\boldsymbol{T}}}) + \dot{\boldsymbol{X}}_{n-1} \cdot e^{-\frac{1}{r\boldsymbol{T}}} \tag{8}$$

where $r = 1/\Delta t$ is the sampling rate, $\boldsymbol{K} = 1/\boldsymbol{D}$ and $\boldsymbol{T} = \boldsymbol{M}/\boldsymbol{D}$. More details about it we gave in Stogl et al. (2019b).

For better understanding, dangerous situations are explained exemplary on the *Virtual Forces* SCA. This is a type of SCAs where on a predefined path, a force vector with influence radius, direction, and strength is defined (c.f. Stogl et al. (2019a,b)). The virtual force disturbs a user during the training by pulling him away from the path. The user needs to "feel" this force and act to it to keep RoboTrainer on the predefined path.

In a case that *Virtual Forces* SCA is not configured properly, the virtual force may exceed the user's input force, which can lead to situations where (i) user releases the handles or (ii) tries to overcome the virtual force. At the current state of the implementation, both cases lead to dangerous situations for a user. In the first case, RoboTrainer will start to move into the direction of the virtual force on its own, and in the second case, RoboTrainer potentially pushes the user backward from the virtual force field.

We introduce two concepts for protecting a user in those situations. The first one is inspired by the energy limiting concept, which defines that no movement of the RoboTrainer is allowed without the user's intention. For this specific case, we adapted the SCAs where we want a user to "feel" virtual forces. We implemented a rule where the absolute value of virtual velocity caused by SCAs does $\dot{x}_{SCA}$ not exceeds the absolute value of velocity caused by the user $\dot{x}_h$, i.e., $\|\dot{x}_h\|_2 \geq \|\dot{x}_{SCA}\|_2$. In practice, this functionality is active only if user's input force is lower than some predefined minimal force $\|F_h\|_2 < F_{min}$, then the equation for calculating RoboTrainer's velocity $\dot{x}_R$ is:

$$\dot{\boldsymbol{X}}_{n,out} = \dot{\boldsymbol{X}}_{n,main} + \sum_{a}^{A} \dot{\boldsymbol{X}}_{n,a} \cdot f_{n,a} \; ; \; f_{n,a} = \frac{\|\dot{x}_{n,h}\|_2}{\|\dot{x}_{n,a}\|_2} \tag{9}$$

$f_{n,a}$ is the scaling factor, $\dot{x}_{n,h}$ is velocity resulting from the user's force input and $\dot{x}_{n,a}$ is virtual velocity at at controller step n and for an SCA a.

The second concept, safety, is designed to protect a user from non-intentional RoboTrainer movements in the user's direction. For this, a safety angle of 30° from RoboTrainer toward the user, i.e., in a backward direction, is defined. This concept protects a user when using stronger forces than the predefined minimal force F_{min}, where the SCA can generally produce stronger virtual force in one dimension, but should not push a user backward. Concrete, this concept checks if the virtual force is stronger than the user's input force and if it is inside the safety angle, and if so, (9) is applied.

3.3 Adaption of Personalized Force-Limits

Different users have a potentially vast difference in interaction forces with RoboTrainer and natural walking speed. An admittance controller with fixed parameters provides a fixed ratio

between a user's maximal force and maximal velocity of RoboTrainer, which is generally not the case. To adapt RoboTrainer towards an individual user, we developed a method for personalizing maximal interaction forces, which adapts indirectly mass and damping of the admittance rule from (1). This method determines the maximal user's interaction forces and adjusts the admittance parameters to correspond to the maximal feasible velocity of RoboTrainer. This maximal interaction force is in the following called the base force.

In the procedure to determine personalized force-limits, a user pushes RoboTrainer against virtual spring for each movement direction separately. Parameters for backward movement are calculated indirectly by scaling-down parameters for the forward direction to avoid the risky situation where users would pull the RoboTrainer towards itself. At the beginning of each step, the movement of RoboTrainer is limited to the dimension of the to-be-determined force, e.g., only forward movement is allowed. The interaction with a user is done by flashing the LEDs on RoboTrainer. So the green light visualizes the direction in which a user should give the force. When a user pushes against the virtual spring with factor k for a distance x from the starting point, it generates the opposing force is $F_{opp} = k{*}x$. This force is subtracted from the user's input force and sent to the admittance controller with fixed parameters to generate RoboTariner's movement. A user is then required to push the RoboTrainer against the spring as much as he feels safe to handle and to keep the force and position stable for at least one second in a predefined range of a few centimeters. After reaching this state, the average input force during the last second is calculated and stored as the maximum force value for the to-be-determined movement direction. A red flashing light informs a user about the completion of this step and the RoboTrainer's intention to move autonomously. As soon as a user releases the handles, RoboTrainer starts slowly to move back to the starting position. After the maximal forces are determined for forward, left and right translation, and clockwise and counterclockwise rotation, the next personalization step is to be done.

3.4 Non-linear Adjustment of Admittance Parameters According to Velocity

Steady-state velocity output of an admittance controller (8) is only defined by the damping constant: $\mathbf{V}_{ss} = \mathbf{F}_h/\mathbf{D}$ (cf. Yu et al. (2003)). To move RoboTrainer with a constant velocity, the user needs to input a constant force. To reduce user's effort when walking at a constant speed, damping constant needs to be reduced. Nevertheless, if damping is too small, the SW gets too reactive, and the user-SW system tends to oscillate. This is observed by Yu et al. (2003), which proposed the linear adaption of damping constant shown in (2).

After the initial implementation of this concept, we also investigated a linear adaption of the virtual mass of the RoboTrainer. Changing only damping or mass constant, we ended up with either oscillatory or overall too heavy SW. Because of the high-dynamics of our system, none of the solutions provided satisfactory results. Therefore, we decided to change both constants at the same ratio to keep the dynamic profile of the system the same but change the response time. For the reason of simplicity, we decided to implement this by manipulating the user's input force. Another positive effect of this approach is that the influence of the spatial control actions changes accordingly,

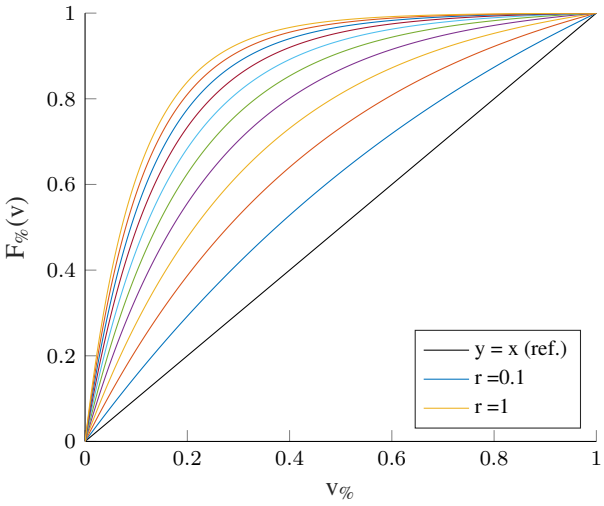

Fig. 1. Non-linear scaling of user's input force in respect to velocity (11), with $r \in [0.05, 1.0]$, $s = 10.0$.

without the need for adapting their internal dynamic model. We are adapting the maximal force limit, which scales the user's input force in the interval $[0, 1]$ at the start of the control loop. Depending of RoboTrainer's current velocity $\mathbf{v}_{\%}(t)$, we adapt the user's maximum force limit $\mathbf{F}_{bu}$ between two extremes $\mathbf{f}_0 \in [0.35, 1.0]$ and $\mathbf{f}_{v_{max}} \in [1.0, 2.0]$ for standstill and maximum velocity:

$$\mathbf{F}^*_{max}(t) = (\mathbf{f}_{v_{max}} + (\mathbf{f}_{v_{max}} - \mathbf{f}_0)\mathbf{F}_{\%}(\mathbf{v}(t))) \cdot \mathbf{F}_{bu} \qquad (10)$$

where scaling factor $\mathbf{F}_{\%}(\mathbf{v}(t))$ is an arbitrary function. In this work, we define it as a non-linear function with parameters curvature r and steepness s:

$$\mathbf{F}_{\%}(\mathbf{v}(t)) = \frac{1.0 - (\mathbf{v}_{\%}(t) \cdot r + 1.0)^{-s}}{1.0 - (r + 1.0)^{-s}} \qquad (11)$$

The non-linear scaling function (11) is shown in Fig. 1 for parameters $r \in [0.05, 1.0]$ and $s = 10.0$. This adaption model provides higher granularity at the lower velocities to get better maneuverability in curves and around obstacles, and results in lower effort when the user is moving faster. This parameters are chosen experimentally, $s = 10$ based on curve plots as in Fig. 1 and $r = 0.55$ based on analysis of its influence on SW's behavior.

In general, the scaling parameter for zero velocity f_0 should be kept at values ≤ 1.0 to give the system more robustness at lower velocities. The parameter for maximum velocity $f_{v_{max}}$ should be held at values > 1.0 to lower the user's force input for keeping the pace at higher velocities. The parameters are chosen to decrease the SW's sensitivity for factor three and to decrease the needed force for the maximal velocity for a half..

For calculation of scale parameters f_0 and $f_{v_{max}}$ we developed a second parameterization method. This method consists of two steps, in which RoboTrainer can move only forward and backward. The user is asked to walk two times for approximately $8\,\mathrm{m}$ straight forward and then backward to the start position. During this movement, the average distance between the RoboTrainer and the user's lower-legs is determined. In the first run, the RoboTrainer is using the admittance controller with fixed parameters and maximal force value determined during the previous (i.e., maximal interaction force) parameterization process. From this run, an average leg distance $dist_{base}$ for

the specific user is determined and used as a baseline for the second step. In the second run RoboTrainer uses the velocity based force adaption (10) with default parameters and the quadratic distance difference in distance towards the first run $\delta_d(t) = \|dist(t) - dist_{base}\|_2$ to tune the parameters. If the measured user-SW distance is larger than the distance from the previous run, the parameters are tuned to provide more restrictive behavior (12)(13). If the distance is smaller, the parameters are tuned to provide more agile behavior. This tuning is done online and stops when the same distance as the baseline is recognized for five consecutive measurements.

$$f_m = \begin{cases} f_m + \delta_d(t) \cdot (1 - f_m) & \text{if } \mathbf{F}_\% < 1 \text{ and } \delta_d(t) < 0 \\ f_m - \delta_d(t) \cdot (1 - f_m) & \text{if } \mathbf{F}_\% < 1 \text{ and } \delta_d(t) > 0 \end{cases} \quad (12)$$

$$f_0 = \begin{cases} f_0 + \delta_d(t) \cdot (1 - f_0) & \text{if } \mathbf{F}_\% > 1 \text{ and } \delta_d(t) > 0 \\ f_0 - \delta_d(t) \cdot (1 - f_0) & \text{if } \mathbf{F}_\% > 1 \text{ and } \delta_d(t) < 0 \end{cases} \quad (13)$$

Upon completion of this parameterization step, RoboTrainer is free to move in all directions. The obtained maximum force limits and velocities are used as controller parameters, and the velocity-adaptive control is activated using the calculated adaption factors.

4. ROBOTRAINER'S CONTROL ARCHITECTURE

The general concept and technical details of the RoboTrainer device are already described in our previous works, Stogl et al. (2014); Stogl et al. (2019a). Here, we give more concrete information on components relevant to the control concepts. The software of RoboTrainer is realized using the *Robot Operating System (ROS)* Quigley et al. (2009) as a middleware for communication between different components. For control *ros_control*-concept is used, where controllers are programmed in C++ and abstracted from used robot hardware. The controllers are compiled in dynamic libraries that can be loaded and unloaded during run-time without a need for a new start of a RoboTrainers's hardware.

Controllers are implemented in a hierarchy so that the base controller presented in Stogl et al. (2019a) is extended by controller implementing the interaction-energy limiter, which is further extended by the adaptive admittance controller. To demonstrate different functionalities, we use *dynamic_reconfigure* ROS-package to change parameters during run-time and enable functionalities. All functionalities in this paper are implemented directly in the controllers, except the shin tracking algorithm used for calculating the distance between a user and RoboTrainer. The leg-tracking algorithm implements the shin tracking approach from Lee et al. (2011) in a separate ROS-Node.

5. EVALUATION

For the evaluation of our extension of the energy limiting approach from Chuy et al. (2007), a user was intentionally using very the results are only shown for one dimension, i.e., the longitudinal axis of the system. The quality of individualized parameters and non-linear interaction adaption was evaluated in a study with 22 participants on test parkour shown in Fig. 9. The complete details about evaluation are given in the following subsections.

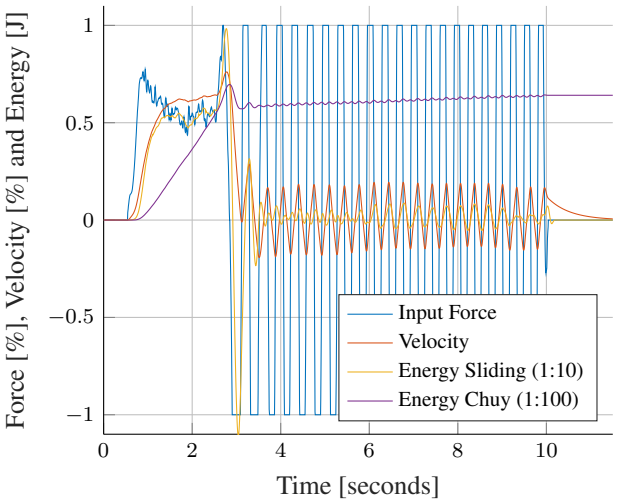

Fig. 2. Comparison of energy calculation with approach from Chuy et al. (2007) ("Energy Chuy") and our extension ("Energy Sliding").

5.1 Energy Limiting

Fig. 2 shows an experimental comparison of the approach from Chuy et al. (2007) and our extended approach. The user was advised to move for a few seconds forward and then to provoke an oscillatory behavior by stiffing his arms. The Chuy et al. (2007) approach does not function properly with our system. The main reason for this is that one could have long periods when energy is positive, during which Chuy's energy achieves high values. These values are, in the case of oscillatory behavior (at 3 s), not able to fall under zero. The second issue with Chuy's energy is that during oscillations for a short period, the power of the system is positive, i.e., input force and velocity show in the same direction, which leads to the slow increase of it. Our approach to calculating energy using a sliding integral shows a faster reaction to oscillatory behavior and keeps low values during oscillations.

To show and to evaluate the issue of reducing RoboTrainer's dynamics when the user suddenly changes directions, we defined the following scenario: (I) the user moves with RoboTrainer approximately two meters forwards; (II) suddenly switches direction and moves back to the start; (III) the user moves forwards again and stiffs his arms; (IV) the user waits until the oscillatory behavior is in given limits and then removes his hands from RoboTrainer's handles. The first two steps evaluate the influence of the energy limiter on sudden direction changes, and the second two steps examine RoboTrainer's behavior when an oscillation happens.

Fig. 3 shows the influence of the sliding-integral-energy-calculation on the reduction of the scaling factor. When oscillatory behavior happens (at 10 s), the scaling factor of the input force is damped as intended, and the output velocity of RoboTrainer is stabilized. At 4 s and 6 s, this approach impairs the dynamics of RoboTrainer when the user suddenly inverts the direction. Fig. 4 shows the results regarding our adapted method for changing the scaling factor by detection of the user's intention. The scaling factor is reduced only for a short time when the user changes the direction (approx. 3 s and 6 s), while the reliable detection of oscillations is still present. At 10 s, the

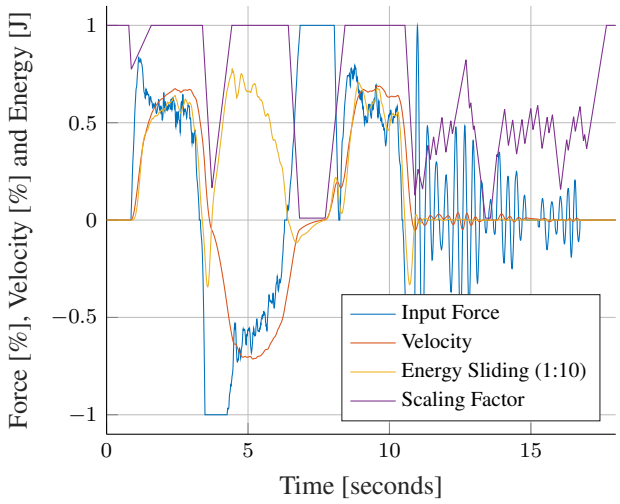

Fig. 3. Influence of the sliding integral on a sudden change of RoboTrainer's direction.

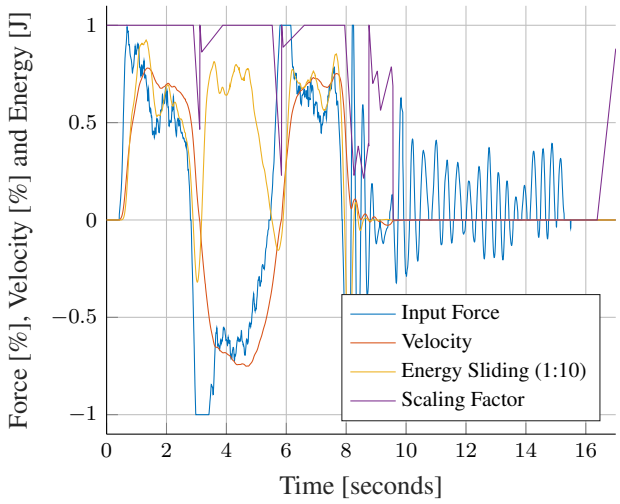

Fig. 4. Adapted method for scaling factor with the detection of the user's intention to keep dynamics of the system when the user suddenly changes direction.

oscillatory behavior takes for a second, and the scaling factor is set to zero to protect RoboTrainer's hardware.

5.2 Passivity and Safety of Spatial Control Actions

Potentially dangerous situations without the energy limiting concepts for SCAs are shown in Fig. 5 and 7. The effect of these concepts is the same for different types of SCAs. Therefore, we present here only the evaluation with *Virtual Forces* SCA. For the clear representation, figures show only one degree of freedom, i.e., the longitudinal axis of RoboTrainer. In the evaluation scenario, we configured the virtual force of *Virtual Forces* SCA to be oriented towards the user and stronger than his maximal input force.

Fig. 5 depicts a situation where the user is overstrained with the virtual force and decides to remove his hands from RoboTrainer's handlebars (at 5 s). At that moment, RoboTrainer starts to move towards the user, i.e., continues its movements according to the velocity generated by the virtual force field (at 6 s) until the influence of the field disappears (at 8 s).

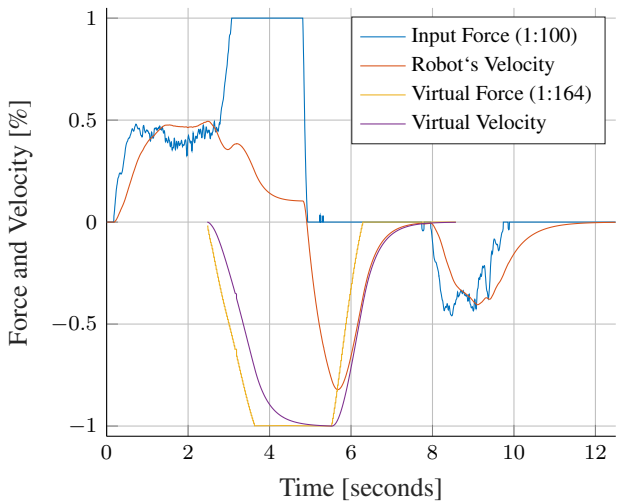

Fig. 5. SCAs without the passivity concept. The user is overstrained with the virtual force and lets RoboTrainer go (at 5 s). RoboTrainer starts to move towards the user (6 s) until the influence of the virtual force disappears (at 8 s). After that, the user moves RoboTrainer again.

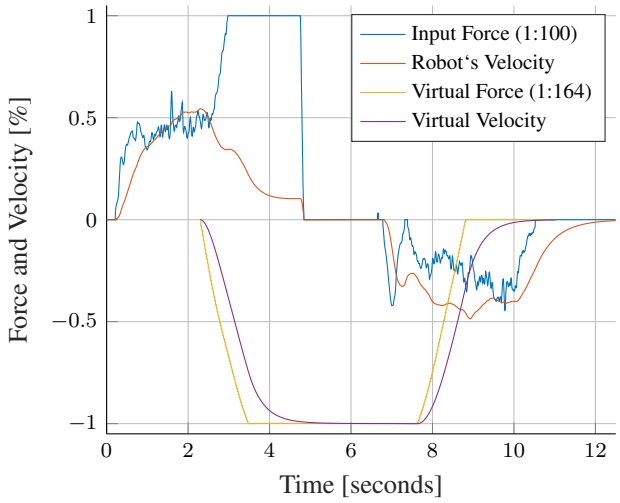

Fig. 6. SCAs with the passivity concept. When the user removes his hands from the handles (at 5 s), RoboTrainer stops. Even in a virtual force field, RoboTrainer does not move (5 s to 7 s) and starts its movement in the user's direction only when the user intends this.

When the passivity concept for SCAs is integrated and used, RoboTrainer stops immediately when the user releases its handles (Fig. 6 at 5 s to 7 s). It starts to move when the user's input force present again.

The scenario when the user tries to overcome a too strong virtual force, is depicted in Fig. 7 for the deactivated safety concept and in Fig. 8 for the activated safety concept. In the first case, the user is pushed our from the virtual force field (i.e., negative RoboTrainer's velocity at 5 s to 7 s) against his will (i.e., the user's input force is pushing the robot away from him). In the second case, the resulting velocity of the robot stays around 0 and, therefore, does not endanger the user.

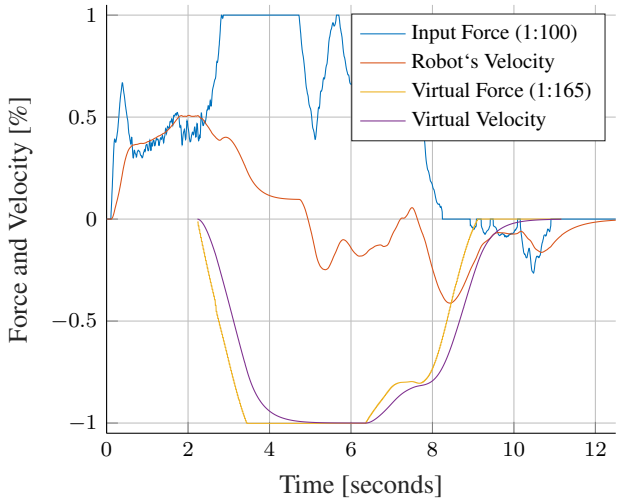

Fig. 7. The user tries to overcome a too strong virtual force. Without any limitations, the user is being pushed back from the virtual force field (negative robot velocity at 5 s to 7 s) against its intention (input force is positive).

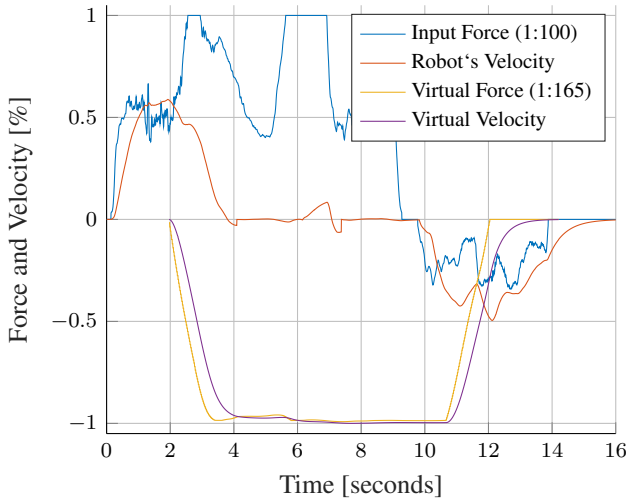

Fig. 8. The safety concept for SCAs. Even if the virtual force is stronger than the user's input force, RoboTrainer does not move towards the user against its intention. The robot's velocity is stopped at zero and becomes negative only when the user intends to move RoboTrainer towards himself.

5.3 Individualized Controller Parameters

The approach for the parameterization of personalized control parameters is evaluated in a user study with 22 participants (3 females) between 20 to 40 years of age. The participants were university students and researchers, which responded to our call-for-participation on the institute's mailing list. 13 of them had very little or no previous experience with RoboTrainer, 4 average experience, and 5 were experienced and very experienced with it. In the latter group are authors of this paper and former students, who worked with RoboTrainer in the past.

We evaluated the following variables: (1) subjective complexity, intuitiveness, and practical use of parameterization processes; (2) subjective and objective influence on the control performance with and without personalized forces; (3) subjective and objective influence of non-linear adaptive control versus controller with fixed parameters; (4) subjective influence of the position of the center of rotation (CoR) on control of RoboTrainer. Subjective evaluations are done in the form of questionaries and objective ones by measuring average velocity, forces, and deviation from the given paths. The change of RoboTrainer's CoR is done by transforming the user's input force in the coordinate system in the middle of the RoboTrainer's front edge, i.e., further away from the user. Therefore, there is no detailed explanation about it in the concept section. In the evaluation, we used the RoboTrainer v1 device and parkour from Fig. 9.

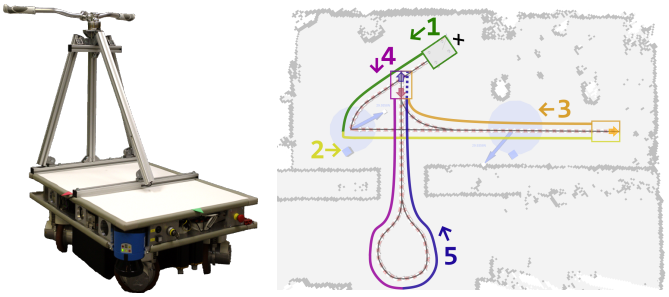

Fig. 9. Left: RoboTrainer v1; Right: The training paths for the user-study marked with numbered sections. The users pulled RoboTrainer backward in section 3. In other sections, the user is moving forwards.

Upfront the study, we explained to the participants the study goals, gathered demographic data, and got consent to record and process their data. As a warm-up task, the participants moved along the straight line ("1" in Fig. 9) forward and backward. First, the participants did user-force parameterization process, after which we ask the following questions: (1) "How complex was the parameterization process for You?"; (2) "How intuitive was the parameterization process?". The participants answered on a scale with five-level Likert items from "1 - very complex / not intuitive at all" to "5 - very simple / very intuitive". The answers are summarized in the second and third columns of Table 1.

The usefulness of this process is evaluated by two consecutive repetitions of the parkour "1-2-3" (section "3" backward movement). The pre-defined controller with maximal linear force of 100 N in both dimensions and maximal rotational moment of 30 N m is compared to the user-specific maximum force/torque values. Those two settings were chosen in random order between the repetitions to avoid the influence of growing experience with RoboTrainer. Afterward, we asked uses which repetition was easier for them to accomplish.

72.7 % of the users rated the individual-force parameterization as easy and very easy, while only one user rated it as difficult. 68.2 % graded the process as intuitive and very intuitive, while 18.2 % were undecided or found the process unintuitive (cf. Table 1). The cause for this is probably a too high minimal-reaction-force limit, which was an issue for some very careful first-time-users. 19 of the 22 users preferred their maximal force, while 3 of them found the standard values to be more comfortable. For both of these runs, we calculated the average RoboTrainer's velocity, user's input force, and path deviation. All three measured parameters tend to be higher using standard values, with minimal, i.e., <10 %, and statistically insignificant differences.

Table 1. The number of users' responses per grades on the Likert scale regarding their experience with the personalization strategies and shift of the CoR.

	Individual User Force		**Individual Velocity Adaption**		**CoR Front**
Grade	Complexity	Intuitiveness	Complexity	Intuitiveness	Complexity
1	0	0	0	0	2
2	1	4	0	1	1
3	5	3	2	3	3
4	9	6	7	3	7
5	7	9	13	15	9
Scale: 5 - very simple, very intuitive; 5 - very complex, very unintuitive					

The process for characterization of non-linear velocity adaption parameters was evaluated by participants using the same two questions as for the previous parameterization. The results are presented in the fourth and fifth columns of Table 1. Over $90.9\,\%$ of the users found the parameterization for the velocity adaption factors easy and very easy, while two users rated it as average and none as difficult. The method was rated as intuitive and very intuitive by $81.8\,\%$ of the users, while only one participant rated it as unintuitive.

For comparison of the non-linear adaptive controller to the admittance-controller with fixed parameters, the users repeated the "1-2-3" parkour with activated virtual forces twice. The controllers are chosen again in random order. Three participants found the admittance-controller with fixed parameters to be more comfortable, while 19 of them preferred the velocity-adaptive control. Users were faster using the adaptive controller and much faster compared to the repetitions from individual-force parameterization. Probably a growing experience with RoboTrianer caused this. The average force and its standard deviation increased, which means that the users' input forces were widespread.

The shift of the center of rotation (CoR) was evaluated using a "4-5" parkour, where the participants had to do $180°$ turn on a predefined radius. Most of the users preferred shifting the center of rotation further away from themselves. $72\,\%$ of the users rated it as simple and very simple, three users were undecided, and the remaining three prefer the center of rotation to be closer to them (see Table 1).

6. CONCLUSIONS AND OUTLOOK

Admittance control is one of the most commonly used control strategies in smart walkers (SWs) when a force sensor is the chosen input device. We present an interaction-energy observer to achieve better performance in detecting oscillatory behavior during the interplay of the user and the SW, especially taking the user's intention into account. This concept is also applied to a training scenario where spatial control actions, i.e., active modifiers of RoboTrainer's behavior, are applied to prevent potentially dangerous situations for a user. Furthermore, we propose a method for individual per-user parameterization of an adaptive admittance controller. The evaluation shows that users prefer our control strategy, being able to incorporate individual user parameters while using a suitable adaptation strategy.

The goal of our upcoming research work with RoboTrainer is to provide better intention recognition, especially in a case where scaling factors should not be changed at all when a user abruptly changes movement direction. Furthermore, we want to enable updates of adaption parameters also during the interaction with RoboTrainer to react to changes in the user's behavior.

ACKNOWLEDGEMENTS

Sincere thanks to all researchers and students from our institute who took part in the evaluation for this paper.

REFERENCES

Chuy, O., Hirata, Y., and Kosuge, K. (2005). Online approach in adapting user characteristic for robotic walker control. In *9th International Conference on Rehabilitation Robotics, 2005. ICORR 2005.*, 139–142.

Chuy, O., Hirata, Y., and Kosuge, K. (2007). Active type robotic mobility aid control based on passive behavior. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 165–170.

Dubowsky, S., Genot, F., Godding, S., Kozono, H., Skwersky, A., Yu, H., and Yu, L.S. (2000). Pamm - a robotic aid to the elderly for mobility assistance and monitoring: a "helping-hand" for the elderly. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, volume 1, 570–576 vol.1.

Lacey, G. and M. Dawson-Howe, K. (1998). The application of robotics to a mobility aid for the elderly blind. *Robotics and Autonomous Systems*, 23, 245–252.

Lee, G., Jung, E., Ohnuma, T., Chong, N.Y., and Yi, B. (2011). Jaist robotic walker control based on a two-layered kalman filter. In *2011 IEEE International Conference on Robotics and Automation*, 3682–3687.

Martins, M., Santos, C., and Frizera, A. (2012). Online control of a mobility assistance smart walker. In *2012 IEEE 2nd Portuguese Meeting in Bioengineering (ENBENG)*, 1–6.

Martins, M., Santos, C., Seabra, E., Frizera, A., and Ceres, R. (2014). Design, implementation and testing of a new user interface for a smart walker. In *2014 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, 217–222.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A.Y. (2009). ROS: an open-source Robot Operating System. In *ICRA workshop on open source software*, 3.2, 1–6.

Solenne, P., Saint-Bauzel, L., Rumeau, P., and Pasqui, V. (2016). Smart walkers: an application-oriented review. *Robotica*, -1, 1–20.

Stogl, D., Armbruster, O., Mende, M., Hein, B., Wang, X., and Meyer, P. (2019a). Robot-Based Training for People With Mild Cognitive Impairment. *IEEE Robotics and Automation Letters*, 4(2), 1916–1923.

Stogl, D., Hein, B., and Mende, M. (2019b). Overview of a Robot for a Neuromuscular Training – RoboTrainer. In *2019 European Conference on Mobile Robots (ECMR)*, 1–6.

Stogl, D., Hein, B., Meyer, P., Armbruster, O., Irgenfried, S., and Wörn, H. (2014). A Technical system for physical activation of persons with mild cognitive impairment. In *Technische Unterstützung für Menschen mit Demenz : Symposium 30.09. - 01.10.2013. Hrsg.: T. Schultz.*

Stogl, D., Wern, P., Hein, B., and Hartmann, D. (2019). Spatial Control Actions for a Physical Training with a Smart Walker. In *2019 61st International Symposium ELMAR (ELMAR)*.

Yu, H., Spenko, M., and Dubowsky, S. (2003). An adaptive shared control system for an intelligent mobility aid for the elderly. *Autonomous Robots*, 15(1), 53–66.