Embedded Control System for Smart Walking Assistance Device

Matevž Bošnak and Igor Škrjanc, *Member, IEEE*

***Abstract*—This paper presents the design and implementation of a unique control system for a smart hoist, a therapeutic device that is used in rehabilitation of walking. The control system features a unique human–machine interface that allows the human to intuitively control the system just by moving or rotating its body. The paper contains an overview of the complete system, including the design and implementation of custom sensors, dc servo motor controllers, communication interfaces and embedded-system based central control system. The prototype of the complete system was tested by conducting a 6-runs experiment on 11 subjects and results are showing that the proposed control system interface is indeed intuitive and simple to adopt by the user.**

***Index Terms*—Control design, user-intention-based control, walk assistance device, walk rehabilitation.**

I. Introduction

HOIST is a therapeutic device that was developed for the use in rehabilitation of walking after injuries or neural impairments. Rehabilitation of walking is a multi-step process that is aimed to return the freedom of motion to the patient. Is is a complex undergoing, mostly based on repetitive tasks execution [1].

It starts with intense therapy of the muscular system and proceeds with the supervised static and dynamic balance training [2]. The dynamic balance training is typically performed in presence of at least two expert therapist who manually assist the subject to walk and maintain the balance at the same time. Several technical solutions have been proposed to relieve the therapists from this physically intensive engagement, such as walking canes, simple static hoists, treadmill-based devices [3], robotic limb manipulators [4], movable support platforms [2] etc.

The popularity of the assisted living research topics resulted in presentation of multiple similar devices that were designed for walking assistance to elderly people and those with motor disabilities. Such devices are usually based on a movable platform that is either actively steered [5] or fully motorized and may combine additional features, such as active assistance for standing up and sitting down [6], or even help with other everyday tasks, such as picking up items [7]. Most of these systems are controlled with the use of steerable handle bars or static handles, equipped with force sensors. Since gait and balance instability is one of the most common sources of fall-induced injuries [8], it is essential that the falls are prevented during the rehabilitation – systems that can not provide the support for patient's full body weight during a failure event (loss of balance, tripping, stumbling, etc.) are not preferred since constant supervision and presence of the physiotherapist is required. The *Hoist* device prototype presented in this article provides a fail-safe and patient-engaging approach to gait rehabilitation.

The paper is structured as follows. Section II introduces the designed walking assist control system, then a detailed description of the rehabilitation platform that is built-upon in this research is given in the Section III. The description of the base platform, hardware upgrades, embedded system and sensory system is given in separate sub-sections. The paper is concluded with the experimental results in Section IV and a short discussion of the results and the possible upgrades is included in the end.

II. Walking Assist Control

Our research is focused on functional extension of an existing commercial rehabilitation platform named *THERA-Trainer e-go* [9], which was designed to offer the support for a strapped-in patient (device user) during the clinical rehabilitation process. Previous work on this platform [2] exposed a potential for a new, intuitive user-machine interface, that will be presented in this paper.

The device itself is built as a stable chassis with four caster wheels, equipped with battery power supply, electronics and two additional electrically driven wheels that enable it to move as a two-wheel robot (two of the caster wheels are lifted once driving wheels are attached). The interface between the device chassis and the vertical support frame is made of a ball joint, equipped with adjustable coaxial springs that have limited range of motion in terms of off-vertical deflection angles. The interface enables the user a certain degree of freedom in motion, but it also limits the user's motion if needed in order to prevent injuries in cases of tripping, stumbling or falling.

The idea behind the *Hoist* project was to augment the manual control mode of the existing walking assist system [illustrated in Fig. 1(a)] by observing the patient and adapting the control strategy accordingly. For this purpose, user position determination and intention-based-control system was designed that allows the patient to control the motion of the device by perturbing its position in regards to the platform base. This approach equates to a very intuitive way of controlling the device,

Manuscript received July 16, 2015; revised January 18, 2016; accepted February 29, 2016. Funded by Slovenian Research Agency . Project number L2-5471, Intelligent robot for walking training.

The authors are with Faculty of Electrical Engineering, University of Ljubljana, 1000 Ljubljana, Slovenia (e-mail: matevz.bosnak@fe.uni-lj.si).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNSRE.2016.2553369

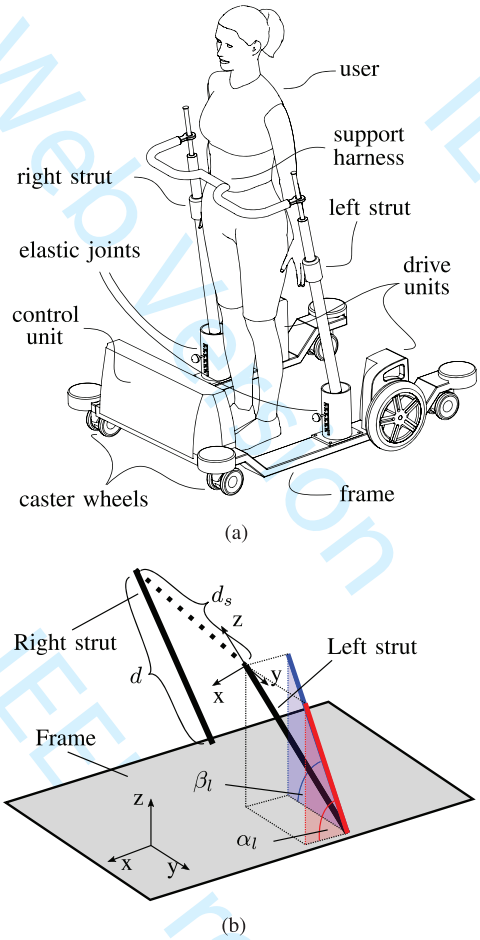


Fig. 1. Illustration of the *Hoist* platform. (a) Parts of the *Hoist* system. (b) Strut angles illustration.

since the control system works towards maintaining the neutral position of the user in regards to the platform. The final result is a system that follows the motion of the user, while providing a constant fail-safe support for the user.

A. User Position Determination

User's position in regards to the platform is determined by observing the angles between the left/right (vertical) struts and the base frame [illustrated in Fig. 1(b)]. For this purpose, struts and frame are equipped with bespoke tilt sensors (detailed in chapter Section III-C), allowing the relative angles between the frame and struts to be determined. The x-axis of the platform frame coordinate system is oriented towards the front of the device [towards the *control unit* in Fig. 1(a)] and the z-axis is oriented vertically in the up direction, while the y-axis is oriented towards the left side of the frame, creating a right-handed Cartesian coordinate system.

While at rest, the tilt sensor coordinate system axes are aligned with the platform frame coordinate system. Deflection angles of the vertical support struts are mechanically limited to approximately 15° from the vertical direction and the angular velocities of the struts are relatively low due to the long length of the struts. By using the small angle approximation of the trigonometric functions, rotation transformations between the platform frame coordinate system to tilt sensor coordinate system can be handled individually for the rotations around the x and y axes. Let α denote the angle of rotation around the sensor's y axis that rotates the platform frame coordinate system to a tilt sensor coordinate system and β denote the angle of rotation around the sensor's x axis that rotates the platform frame coordinate system to a tilt sensor coordinate system.

Perturbations of the user harness attachment point origin $(\Delta x, \Delta y)$ and orientation $\Delta\psi$ is therefore calculated using the following equations:

$$\Delta x = \frac{d(\tan\alpha_l + \tan\alpha_r)}{2} \tag{1}$$

$$\Delta y = -\frac{d(\tan\beta_l + \tan\beta_r)}{2} \tag{2}$$

$$\Delta\psi = \arctan\frac{d(\tan\alpha_r - \tan\alpha_l)}{2d_s} \tag{3}$$

where d is the vertical distance between the elastic joints and horizontal user support strut, d_s is the length of that strut (i.e., horizontal distance between the two vertical struts), while α_l, β_l, α_r and β_r are α and β angles for the left and right strut, respectively. Since the deflection angle of the struts is limited, the above equations can be simplified by replacing the tan functions with the linear small-angle approximation, which results in $\Delta x \approx d(\alpha_l + \alpha_r)/2$, $\Delta y \approx -d(\beta_l + \beta_r)/2$ and $\Delta\psi \approx d(\alpha_r - \alpha_l)/2d_s$.

These perturbations are used for estimating the position of the user in regards to the platform, which directly indicates the intention of the user for motion. This is justified by the following assumptions.

- When the user intends to move in the forward direction, the user will initiate a move in that direction, causing its relative position in regards to the platform to move forward.
- When the user intends to speed up or slow down, user's speed will increase or decrease, respectively. This directly results in relative position of the user in regards to the platform to change to more forward (user is accelerating) or more backwards (user is slowing down) position.
- When the user intends to change the direction of motion, the user will rotate his/her pelvis in that direction [10].

Based on these assumptions, the values of Δx and $\Delta\psi$ can be used directly as an estimation of user intent for acceleration and turning, while the side motion Δy can be used as a measurement of the user's walking (gait) quality [2], [11].

B. Control Algorithm

The control system must be robust enough to ignore normal oscillations in the signals that result from walking dynamics, however special care must be taken for allowing the user to execute controlled rapid stopping manoeuvres. Thus, the controller combines a regular P-controller and a filtered P-controller (PfP controller), a combination that allows the user to have the feeling of an immediate control (regular P part) and smooth changes in the average speed of the platform (filtered P part, that simulates the effect of an integrator. Such controller has a continuous transfer function of $H(s) = K_P + K_{P,f}(\tau s + 1)^{-1}$. Input dead-band with variable width was added to the controller to tackle the problem of measurement (input) noise in vicinity

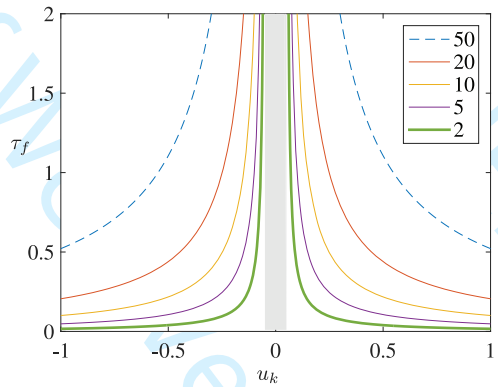


Fig. 2. Exponential braking ($|u_k| > D_x$ and $u_k \cdot v_{k-1} < 0$) – decay time constant at $D_x = 0{,}05$ (shaded area) and L_{dec} varying from 2,0 to 50.

of the user's neutral position and allow the user to keep the platform stationary when needed. Motion direction reversal situation is also handled separately, improving the deceleration during braking action.

The continuous transfer function of the controller has been discretized by zero-order hold method using the step time of $\Delta T = 10$ ms and the resulting difference equation can be written as

$$v_k = K_P \cdot u_k + v_{f,k} \tag{4}$$

$$v_{f,k} = K_{f,1} \cdot v_{f,k-1} + K_{f,2} \cdot (K_{P,f} \cdot u_k) \tag{5}$$

where $K_{f,1} = e^{-(\Delta T/\tau_f)}$. The time constant of the filter τ_f is defined as follows:

$$\tau_f = \begin{cases} \tau_{dec}, & |u_k| \le D_x \\ \Delta T \frac{L_{dec}}{|u_k| - D_x}, & |u_k| > D_x, u_k \cdot v_{k-1} < 0 \\ \tau_n, & |u_k| > D_x, u_k \cdot v_{k-1} \ge 0 \end{cases} \tag{6}$$

and $K_{f,2}$ is defined as

$$K_{f,2} = \begin{cases} 0, & |u_k| \le D_x \\ K_{2,r}, & |u_k| > D_x, u_k \cdot v_{k-1} < 0 \\ \left(1 - e^{-(\Delta T/\tau_n)}\right), & |u_k| > D_x, u_k \cdot v_{k-1} \ge 0 \end{cases}. \tag{7}$$

The three cases above define the parameters in three operating modes of the controller: dead-band operation ($u_k \le D_x$), motion reversal or braking operation ($u_k > D_x$ and $u_k \cdot v_{k-1} < 0$) and normal operation ($u_k > D_x$ and $u_k \cdot v_{k-1} \ge 0$).

In dead-band operation, input value is discarded (hence $K_{f,2}$ is set to 0), while the time constant of the filtered part τ_f is set to a fixed value of τ_{dec}. The motion direction reversal (i.e., braking) operation must allow the user to rapidly stop the motion of the platform, which is realized by cancelling the slow response rate of the implemented low-pass filter. Time constant of the filter τ_f in this operation mode is therefore dependent on the value of the input u_k and parameters D_x and L_{dec} (Fig. 2), that define the dead-band width and time constant function shape, respectively. Since the controller operates in this mode only outside the dead-band, a fixed value of τ_{dec} is used for the time constant inside it.

The parameter $K_{2,r}$ is selected to have a relatively low value ($0 < K_{2,r} \ll (1 - e^{-(\Delta T/\tau_n)})$) to obtain the deceleration towards the standstill position, while its value has to be greater than zero in order for the motion direction to truly reverse—without it, the condition for normal operation $u_k \cdot v_{k-1} \ge 0$ would never be met after initiating a direction reversal manoeuvre.

Although the choice of controller parameters greatly depends on the walking ability of the current user (mostly dead-band D_x, gains K_P, $K_{P,f}$, the shape of braking curve L_{dec} and maximum value of the filtered part $v_{f,\max}$), generic set of values for the parameters has been determined in experimental runs. By using these values, test subjects were able to control the speed of the platform in precise and smooth way, as demonstrated in the experimental study (Section IV).

Algorithm 1 Translational motion controller

procedure TRANSPI(u_k, v, v_f)
  if $|u_k| < D_x$ **then**
    $v_f \leftarrow e^{-(\Delta T/\tau_{dec})} \cdot v_f$
    $v \leftarrow v_f$
  else
    $u'_k \leftarrow u_k \cdot K_{P,f}$
    if $u_k \cdot v < 0$ **then**
      $v_f \leftarrow e^{-((|u_k| - D_x)/L_{dec})} \cdot v_f + K_{2,r} \cdot u'_k$
    else
      $v_f \leftarrow e^{-(\Delta T/\tau_n)} \cdot v_f + (1 - e^{-(\Delta T/\tau_n)}) \cdot u'_k$
    $v_f \leftarrow \text{limit}\{v_f, [-v_{f,\max}, v_{f,\max}]\}$
    $v \leftarrow K_P \cdot u_k + v_f$

The implemented algorithm is listed as Algorithm 1 (translational motion controller), where the following variables are used.

u_k	Input variable – normalized perturbation of the user's position in forwards–backwards direction (values in range $[-1{,}0; 1, 0]$).
v	Output variable that controls the reference speed for the servo motors, driving the wheels.
v_f	Filtered part of the output variable.
$v_{f,\max}$	Maximum value of the filtered part of the output variable that determines the maximum accumulated velocity.
D_x	Dead-band of the input variable (5% of the u_k range).
τ_n	Time constant of the filter in normal operation ($\tau_n = 10$ s).
τ_{dec}	Time constant of the filter in dead-band operation ($\tau_{dec} = 2$ s).
L_{dec}	Parameter that determines the gain of the exponential braking function ($L_{dec} = 2{,}0$).

$K_{2,r}$ See description in text ($K_{2,r} = 0{,}0001$).

K_P Proportional gain of the normal part of the controller ($K_P = 2{,}0$).

$K_{P,f}$ Proportional gain of the filtered part of the controller ($K_{P,f} = 50{,}0$).

Similar controller was designed for rotational motion. It is also based on the PfP concept without the direction reversal logic. The implementation of the algorithm is given below.

Algorithm 2 Rotational motion controller

1: **procedure** ROTPI($u_{\psi,k}, \omega, \omega_f$)
2: **if** $|u_{\psi,k}| < D_\psi$ **then**
3: $\omega_f \leftarrow e^{-(\Delta T/\tau_{\psi,dec})} \cdot \omega_f$
4: **else**
5: $u'_{\psi,k} \leftarrow K_{\psi,P,f} \cdot u_{\psi,k}$
6: $\omega_f \leftarrow e^{-(\Delta T/\tau_{\psi,n})} \cdot \omega_f + (1 - e^{-(\Delta T/\tau_{\psi,n})}) \cdot u'_{\psi,k}$
7: $\omega \leftarrow K_{\psi,P} \cdot u_{\psi,k} + \omega_f$

The following variables are used in algorithm 2 (rotational motion controller):

$u_{\psi,k}$ Input variable—normalized perturbation of the user's pelvis orientation (values in range $[-1{,}0; 1, 0]$).

ω Output variable that controls the reference rotational velocity for the servo motors, driving the wheels.

ω_f Value (state) of the filtered part of the controller.

D_ψ Dead-band of the input variable (5% of the $\Delta\psi$ range).

$\tau_{\psi,n}$ Time constant of the filter in normal operation ($\tau_{\psi,n} = 2$ s).

$\tau_{\psi,dec}$ Time constant of the filter in dead-band operation ($\tau_{\psi,dec} = 0{,}1$ s).

$K_{\psi,P}$ Proportional gain of the normal part of the controller ($K_{\psi,P} = 2{,}0$).

$K_{\psi,P,f}$ Proportional gain of the filtered part of the controller ($K_{\psi,P,f} = 50{,}0$).

III. Description of the Rehabilitation Platform *Hoist*

As noted in the previous section, modified commercial rehabilitation device *THERA-Trainer e-go* [9] is used as the platform in this research. The original device has been modified by replacing all electronics with bespoke components. The device has been equipped with multiple sensors that observe the device's and user's motion and position. For the purpose of odometry based position determination, each driven wheel has been equipped with a rotational encoder that measures speed and position of each wheel. In addition, three MEMS (*Micro-Electro-Mechanical*) angular rate and linear acceleration sensors are used to measure the chassis and user's support struts orientation (Fig. 3 shows the device during demonstration and testing). The data collection, processing and control is done on a Odroid XU3 embedded system, running Ubuntu linux distribution. The embedded system runs ROS (*Robot Operating System*) on top of Ubuntu operating system, with ROS taking care of inter-device communication tasks, data gathering, data processing synchronization, etc.

Fig. 3. *Hoist* system demonstration and testing.

A. Electrical and Drive System

Main power supply consists of two 12 V, 18 Ah lead-acid batteries that power the motor drivers via an emergency cut-off relay, the embedded computer via a 5 V switching power supply and CAN (*Controller Area Network*) devices via separate 12 V switching power supply.

The propulsion system basically mimics a two-wheeled robot with a differential drive. Each driven wheel has its own 100 W geared electric motor with a shaft encoder and a dedicated speed controller that is attached to the shared CAN bus on the device. Each motor driver executes a closed-loop speed control algorithm with adjustable current limits. Motor parameters (e.g., position, speed, motor current, etc.) are adjustable and observable using the CAN bus protocol.

B. CAN Communication Bus

CAN is a broadcasting digital bus designed to operate at preselected speeds from 20 kbit/s to 1 Mbit/s with the transmission rate selected depending on the bus length, topography and transceiver capabilities. CAN is an attractive solution for embedded control systems because of its low cost, light protocol management, the deterministic resolution of the contention, and the built-in features for error detection and retransmission [12].

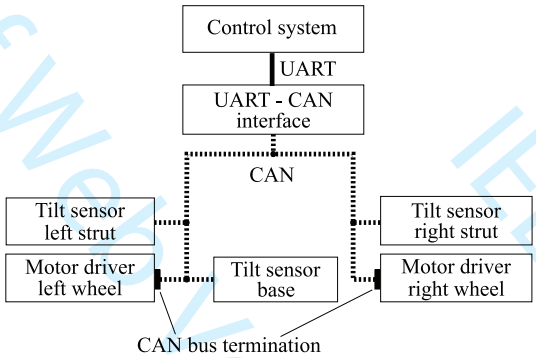


Fig. 4. CAN network on *Hoist* .

CAN has a proven reliability in automotive and process industry [13], [14] and was therefore a preferred option for a network between different low-level subsystems in the *Hoist* system. A relatively conservative bus speed of 250 kbit/s was chosen for the CAN bus in this project (illustrated in Fig. 4) that has a total length of 4 m with two 1-m-long stubs (unterminated bus ends). Split-type bus termination was used on left and right motor driver CAN nodes, as suggested in [15].

A simple bi-directional gateway between CAN bus and UART (Universal Asynchronous Receiver/Transmitter) was designed in order to simplify the access to the CAN bus from the high-level control system, which lacks support for CAN bus communication.

C. Tilt Sensors

The device is equipped with three tilt sensors, two of which are mounted on the vertical support struts and one mounted on the base (platform chassis). This sensor arrangement provides a simple method of measuring the relative orientation between the support frame struts and the base. Since the motion in each axis is restricted mechanically to approximately $\pm 15^\circ$ from vertical and corresponding angular velocities are low, tilt angles are calculated independently for each axis [producing angles α and β, as indicated in Fig. 1(b)]. Since the support struts are mechanically restricted from rotating around the third (vertical) axis, estimation of the third orientation angle is not required for this application.

The custom built tilt sensors host power supply system, microcontroller and MEMS-based gyroscope and accelerometer. As gyroscope output (angular rate) is biased and has to be integrated over time to obtain the orientation angle, dual simplified one-axis sensor fusion algorithms were implemented in the microcontroller of each tilt sensor, resulting in two independent orientation angle estimates per sensor (angles α and β).

The fusion algorithm for one axis is based on static Kalman filter. If the following states are selected for the state vector x_k

$$x_k = [\,\phi_k \quad \dot{\phi}_{b,k} \quad \theta_k \quad \dot{\theta}_{b,k}\,]^T \tag{8}$$

where ϕ_k, θ_k are the values of the angles ϕ, θ at the moment k, $\dot{\phi}_{b,k}$, $\dot{\theta}_{b,k}$ are gyroscope output biases at the moment k and input vector u_k contains current angular rates measurement

$$u_k = [\,\dot{\phi}_{m,k} \quad \dot{\theta}_{m,k}\,]^T \tag{9}$$

the system is presented in the following state space form:

$$x_k = \mathbf{A}x_{k-1} + \mathbf{B}u_k \tag{10}$$

$$y_k = \mathbf{C}x_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} x_k \tag{11}$$

$$\mathbf{A} = \begin{bmatrix} 1 & -\Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -\Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} \Delta t & 0 \\ 0 & 0 \\ 0 & \Delta t \\ 0 & 0 \end{bmatrix}. \tag{12}$$

Kalman filter can be evaluated offline because of the fixed state-transition, input and output matrices along with fixed state and measurement noise covariance matrices ($\mathbf{V}$ and $\mathbf{N}$, respectively). In order to obtain static Kalman filter gain matrix K_s, the following steps are evaluated iteratively until the K_k converges to K_s (*a posteriori* covariance matrix $\hat{\mathbf{P}}_k$ is initialized before starting the iteration)

1) $\mathbf{P}_k^* = \mathbf{A}\hat{\mathbf{P}}_{k-1}\mathbf{A}^T + \mathbf{V}$;
2) $\mathbf{K}_k = \mathbf{P}_k^*\mathbf{C}^T[\mathbf{C}\mathbf{P}_k^*\mathbf{C}^T + \mathbf{N}]^{-1}$;
3) $\hat{\mathbf{P}}_k = \mathbf{P}_k^* - \mathbf{K}_k\mathbf{C}\mathbf{P}_k^*$.

Based on the approximate values of the signal variances and some manual fine-tuning, the following values were used in our implementation:

$$\begin{aligned} &\Delta t = 0.01\ \text{s} \quad \hat{\mathbf{P}}_0 = \mathbf{I} \\ &\mathbf{V} = \text{diag}(0.1\ \text{rad}^2, 1\ \frac{\text{rad}^2}{\text{s}^2}, 0.1\ \text{rad}^2, 1\ \frac{\text{rad}^2}{\text{s}^2}) \\ &\mathbf{N} = \text{diag}(800\ \text{rad}^2, 800\ \text{rad}^2) \end{aligned} \tag{13}$$

which resulted in the following static Kalman gain K_s

$$K_s = \begin{bmatrix} 0.0102 & -0.0013 & 0 & 0 \\ 0 & 0 & 0.0102 & -0.0013 \end{bmatrix}^T. \tag{14}$$

The notation $\text{diag}(\lambda)$ was used to denote a diagonal matrix with the elements of vector λ on the diagonal. Static Kalman filter was then implemented in the microcontroller in the iterative form with time step of $\Delta t = 0.01$ s, consisting of the prediction step

$$\hat{x}_k = \mathbf{A}x_{k-1}^* + \mathbf{B}u_{k-1} \tag{15}$$

and the correction step

$$x_k^* = \hat{x}_k + K_s\left([\,\phi_{s,k} \quad \theta_{s,k}\,]^T - \mathbf{C}\hat{x}_k\right) \tag{16}$$

where the current sensor angles $\phi_{s,k}$ and $\theta_{s,k}$ were calculated from the acceleration measurements $a_{x,k}$, $a_{y,k}$ and $a_{z,k}$

$$\phi_{s,k} = \arctan\frac{-a_{z,k}}{a_{y,k}} \tag{17}$$

$$\theta_{s,k} = \arctan\frac{-a_{z,k}}{a_{x,k}}. \tag{18}$$

D. Servo Drives

Original motor drivers for the 40 V, 100 W geared electric motors with built-in shaft encoders were embedded into the control box hardware in front of the platform. Since there was no external interface available to control them, various commercial compact servo motor controllers were initially evaluated for

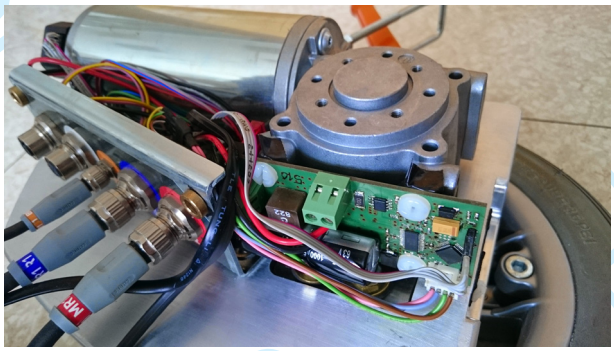


Fig. 5. Servo drive circuit fitted directly to the gearbox.

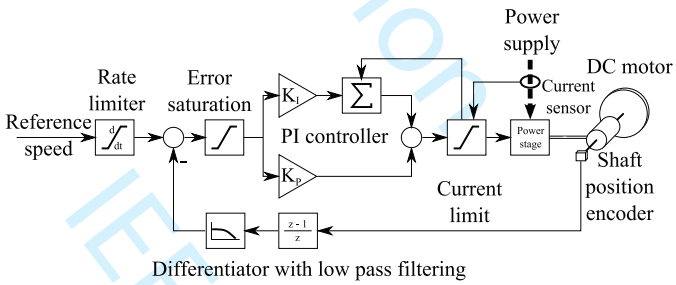


Fig. 6. Servo control loop.

this application, but most failed to work properly at low operating speeds. Therefore, a custom servo motor controller was designed, which operates appropriately at low rotational speeds and also fits into the tight space in the drivetrain housing itself (Fig. 5).

The application running on the servo controller allows the access to all controller parameters, current position, speed and motor current via CAN bus messages.

1) Electronics Design: The servo controller is built around Cortex-M0 microcontroller with an integrated CAN transceiver and smart H-bridge MOSFET (Metal–Oxide–Semiconductor Field-Effect Transistor) driver with external power MOSFET H-bridge. Since the microcontroller has the CAN bus functionality built-in, the driver is directly accessible from the control system for configuration and control. The physical dimensions of the designed servo controller allow the board to be mounted directly onto the gearbox, which also acts as a heatsink for the driver's PCB (Printed Circuit Board).

2) Servo Control Loop: The servo control loop is shown on Fig. 6. It is a semi-cascaded controller with current limit in internal loop and velocity-mode PI servo controller in external loop. Current shaft rotational velocity is obtained by differentiating the shaft position and filtering the result using a first order low-pass filter. The reference speed as the input variable is rate-limited to limit the maximum motor acceleration in order to reduce the mechanical stress of the components and limit the electric current spikes. There is an additional feedback loop from current limit controller back to integral of the PI controller to avoid saturation effects. The servo control loop is executed at a fixed frequency of 1 kHz.

E. Embedded System

Initially a compact embedded system BeagleBone Black was selected as the main computational platform for the project, but was later replaced by a more capable Odroid XU3 embedded system. Additionally, a small wireless router was used to provide a reliable WiFi connection between the *Hoist* system and the external monitoring and control computers. To interface the CAN bus devices, a USB-UART converter was used to talk to our custom UART-CAN gateway. Battery voltages, bumper switches and other configuration switches were connected to a PoKeys57U USB device, which is interfaced by the embedded computer with the help of a cross-platform open-source communication library PoKeysLib [16].

We used Ubuntu 14.04 LTS as the operating system of the embedded computer with meta-operating system ROS (Robot Operating System) version Indigo. ROS (Robot Operating System) is a meta-operating system, containing libraries and tools to help software developers create robot applications. It provides hardware abstraction, device drivers, libraries, visualizers, message-passing, package management and more (details are available in online ROS documentation [17]).

Our application was divided into a set of ROS nodes that can be started individually and used on different hosts as long as all the hosts are configured with the same ROS environment and network settings. Main ROS node was designed as an interface between ROS system and the physical hoist device. It uses CAN bus communication (using the CAN gateway) to communicate with servo motor drivers and tilt sensors, uses analog inputs of the PoKeys57U device to monitor the battery voltage and digital inputs to monitor the state of bumper switches. Motion commands are forwarded to wheel motor drivers, while wheel odometry data is generated from the reported encoder positions and exported along with sensor reading and switch statuses to ROS environment in the form of standard ROS message types.

F. Environment Sensing

Three parts of the primary hoist system are used to sense the physical environment—wheels odometry information, bumper switch sensors, and platform tilt sensor.

Each driven wheel of the hoist is equipped with an encoder, whose position is constantly tracked by the servo drive (for the purpose of closed-loop speed control and position tracking) and broadcasted over CAN bus. The main ROS node receives both left $\Delta E_{L,k}$ and right $\Delta E_{R,k}$ wheel encoder information from servo drives and combines this information into odometry data. The following equations are evaluated periodically for every step k:

$$\Delta d_k = \frac{\Delta E_{L,k} + \Delta E_{R,k}}{2} K_{p\to m} \tag{19}$$

$$\Delta \phi_k = \frac{\Delta E_{R,k} - \Delta E_{L,k}}{d_{ww}} K_{p\to m} \tag{20}$$

$$p_{x,k+1} = p_{x,k} + \Delta d_k \cos p_{\phi,k} \tag{21}$$

$$p_{y,k+1} = p_{y,k} + \Delta d_k \sin p_{\phi,k} \tag{22}$$

$$p_{\phi,k+1} = p_{\phi,k} + \Delta \phi_k \tag{23}$$

where $K_{p\to m}$ is a factor that translates the encoder change to driven wheel distance, d_{ww} is the distance between the driven

TABLE I
EXPERIMENTAL RUNS
(* PLATFORM WAS CONTROLLED BY THE SUBJECT)

	Run 1 □	2	3*	4 □	5	6*
Trajectory shape	□	☍	☍	□	☍	☍
Control by subject			✓			✓
Injury simulation				✓	✓	✓

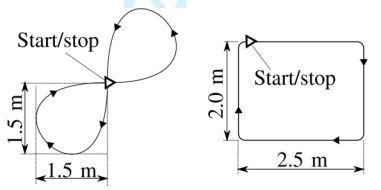


Figure-8 pattern ☍ Square pattern □

Fig. 7. Trajectory shapes used in the experiments.

wheels, while $p_{x,k}$, $p_{y,k}$ and $p_{\phi,k}$ are components of the position state vector, describing positions in axes x, y and rotation around vertical axis ϕ, respectively.

IV. RESULTS

The platform and proposed control system approach were tested both in simulated environment (using ROS and Gazebo with a mathematical and visual model of the platform) and in experimental study in laboratory environment, mainly testing the turning control of the platform. The experimental results will be presented next.

A. Experimental Results

Because the presented device is a prototype, the experiment was limited to 11 healthy subjects in six different runs (see Table I) in the confined space of about 5×5 m. The subjects were 10 male and 1 female volunteers (aged between 25 and 65), who had none or very limited experience with the presented device.

The experiment was executed in two stages, as it is imagined the device to be used in true rehabilitation sessions. First, the remote control functionality of the presented system was employed, where operator of the device forced the subject to follow the hoist in the prescribed pattern, drawn on the floor. During this first step, the subjects were only told to follow the motion of the platform. In the second stage, the subjects were instructed to control the system by following the same pattern. Each run consisted of at least two executions of the prescribed pattern.

Two patterns were used for trajectory shape during experiments (Fig. 7), while only the *figure of eight* pattern (denoted as ☍) was used for autonomous operation by the subject. The experimental runs 1 (square pattern □) and 2 (*figure of eight* pattern ☍) were first conducted (system controlled by the operator using remote control), then experimental run 3 (pattern ☍) followed, where the subject had full control of the system.

TABLE II
VALUES OF THE CROSS-CORRELATION FACTOR FOR ALL EXPERIMENTAL RUNS AT ZERO LAG WITH THE AVERAGE VALUE $\bar{C}$ FOR ALL SUBJECTS PER EACH RUN IN THE LAST ROW – HIGHER VALUE IS BETTER
(* PLATFORM WAS CONTROLLED BY THE SUBJECT)

	Run 1 □	2	3*	4 □	5	6*
1	0,09	0,60	0,95	-	-	-
2	-0,05	0,23	0,91	-	-	-
3	0,05	0,42	0,86	0,11	0,47	0,92
4	0,15	0,76	0,91	0,10	0,88	0,91
5	0,06	0,49	0,77	0,10	0,65	0,81
6	0,01	0,18	0,70	0,14	0,69	0,92
7	0,09	0,39	0,89	0,11	0,45	0,87
8	0,05	0,24	0,91	-	-	-
9	0,10	0,70	0,86	-	-	-
10	0,12	0,35	0,86	0,12	0,51	0,93
11	0,12	0,56	0,95	0,11	0,77	0,91
$\bar{C}$	0,07	0,45	0,87	0,11	0,63	0,90

TABLE III
VALUES OF NORMALIZED ISE COST FUNCTION J_N FOR ALL EXPERIMENTAL RUNS WITH THE AVERAGE VALUE $\bar{J}$ FOR ALL SUBJECTS PER EACH RUN IN THE LAST ROW – LOWER VALUE IS BETTER
(* PLATFORM WAS CONTROLLED BY THE SUBJECT)

	Run 1 □	2	3*	4 □	5	6*
1	30,35	9,42	3,78	-	-	-
2	32,38	8,41	5,88	-	-	-
3	30,57	29,73	3,56	30,92	5,58	2,12
4	36,93	4,06	3,41	110,63	11,12	3,61
5	50,06	6,80	17,80	31,17	13,30	9,87
6	29,83	3,58	7,05	72,55	3,63	7,79
7	30,34	5,01	7,29	32,15	12,39	4,43
8	10,64	6,84	5,05	-	-	-
9	31,79	24,76	4,78	-	-	-
10	34,96	14,36	6,62	40,52	31,22	7,40
11	74,01	12,20	8,68	78,78	8,58	7,72
$\bar{J}$	35,62	11,38	6,72	56,67	12,26	6,13

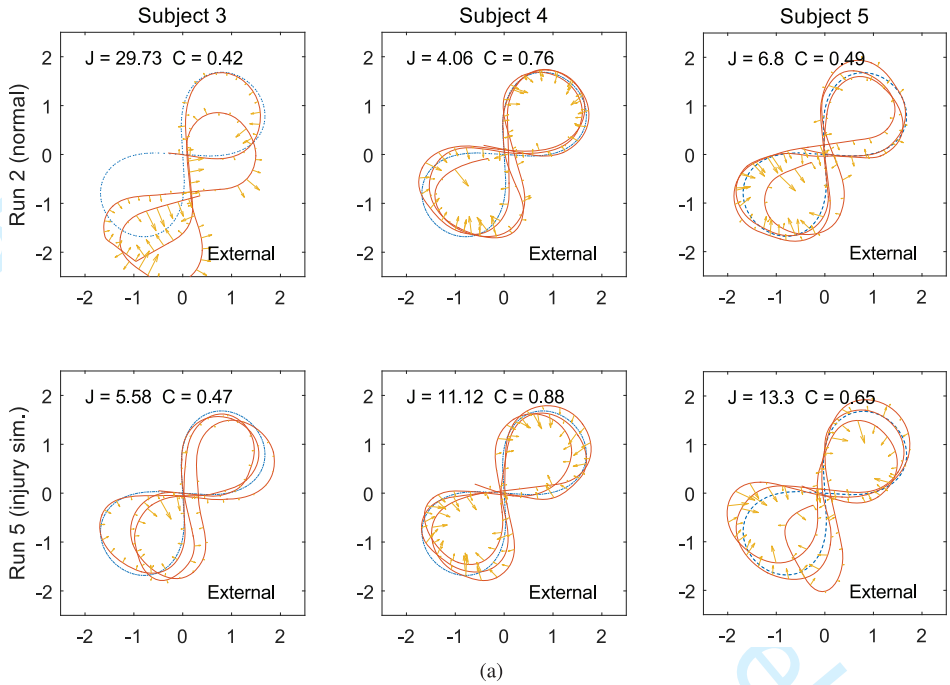
In the runs 4, 5, and 6, the subjects were equipped with leg-mounted contraption that simulated an injury to the leg's muscles or nerve system [18], which allowed us to evaluate the suitability of the proposed control system for the real patients in the clinical rehabilitation process. Since some subjects rejected installing the orthosis, there is some missing data (denoted by −).

For the purpose of results evaluation, all sensor measurements were recorded on-board the presented experimental system. This includes calibrated odometry data for evaluation of the tracking quality, remote control commands and sensory data, that was used for the user position assessment and user-input based control of the system.

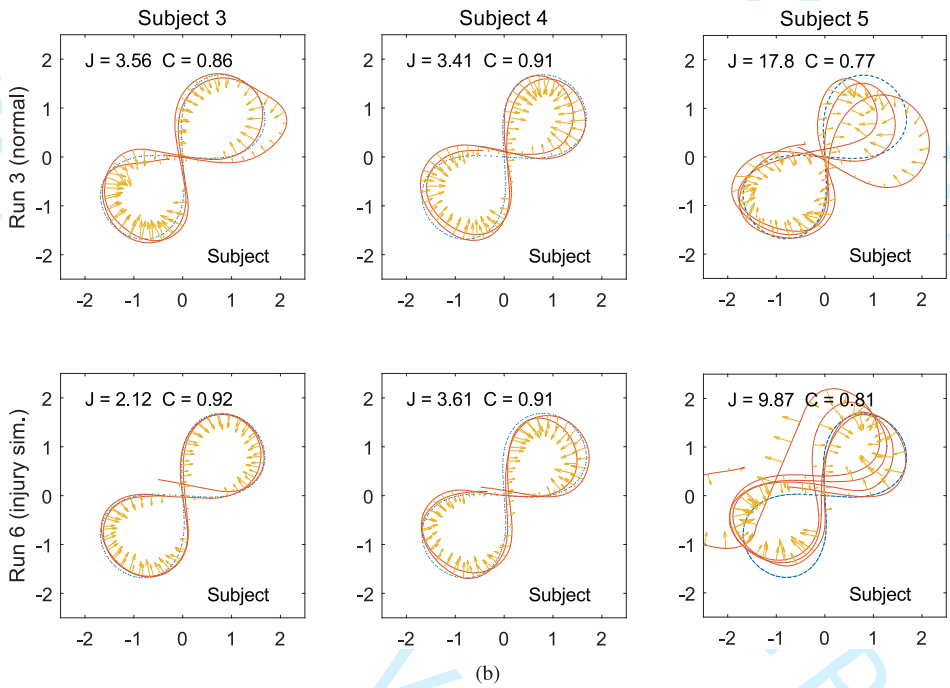
The results were evaluated both in terms of execution precision of the reference trajectory shape and in terms of cross-correlation between the $\Delta\psi$ (obtained from struts tilt angles) and rotational velocity around the vertical axis Ω_r of the reference trajectory.

Before executing the cross-correlation, the data sequences were normalized so that the autocorrelations at zero lag equal 1 (MATLAB's function *xcorr*). The results are gathered in Table II.

The execution precision of the reference trajectory shape was evaluated using the normalized ISE (integral square error) cost function J_N of the pose error, defined as a difference between the actual (x_i, y_i, ψ_i) and reference pose samples



(a)



(b)

Fig. 8. Experimental results of runs. (a) External (therapist) control without (first row) and with injury simulation (second row). (b) Subject (user) control without (first row) and with injury simulation (second row).

$(x_{r,i}, y_{r,i}, \psi_{r,i})$, consisting of both the position and the orientation parts

$$J_N = \frac{1}{L}\sum_{i=1}^{N}\left((x_i - x_{r,i})^2 + (y_i - y_{r,i})^2\right) + \frac{1}{L}\sum_{i=1}^{N}(\psi_i - \psi_{r,i})^2. \quad (24)$$

The normalization value L is the total length of the reference path, calculated from the odometry data. Values of cost functions for all runs are gathered in Table III.

Except for runs 1 and 4 (with a square shaped reference trajectory), in average results show a good correlation between the user's pelvis rotation angle $\Delta\psi$ and reference rotational velocity Ω_r. The most surprising are the correlation results of run 2, where subjects were being externally commanded (using a remote control feature) and were following the *figure of eight* pattern for the first time. Although they were told only to follow the motion induced by the machine, the results show that they were indicating the change of the direction by the additional rotation of the pelvis into the direction of the turn. This fact strengthens the idea of the proposed system being indeed very intuitive to

use. By comparing the results between the runs 2 and 5, latter being executed after runs 1 to 4, it is generally true that the indicative pelvis motion has been amplified after just a few experimental runs (taking 1–2 min each), additionally confirming the intuitive learning nature of the presented control system.

An interesting view on this matter can be given by observing the values of the cost function that was used to estimate the execution precision of the reference trajectory shape. The results [both numerically in Table III and graphically in Fig. 8(a) and (b)] show that subjects in control of the system were superior to external remote control in following the prescribed curve with and without injury simulation, resulting in lower value of the cost function in runs 3 and 6. Fig. 8(a) and (b) illustrates the results for subjects 3, 4, and 5 in runs 2, 3, 5, and 6 by overlaying odometry-based trajectories of the platform (solid line) over the reference trajectory (dotted line) with arrows illustrating the intended turning direction and amplitude, as estimated from the user intent of turning (chapter Section II-A).

The results for subject 5 show greater cost values and hence worse execution precision of the reference trajectory shape, which is the result of subject overpowering the machine, causing the systems drive wheels to slip and loose the correct odometry data.

No subject had problems with commanding the translational velocity of the platform, easily adapting it to the user's needs or abilities.

V. Conclusion

The prototype of walk rehabilitation system with the implementation of the presented control system has been proven in experimental study for being very intuitive and easy to adopt by the users. Most indicative are the experimental runs executed in *External control* mode (remote control by the operator), which produced data, that clearly show a positive correlation between the user's intention (pelvic rotation) and the rotational velocity along the prescribed reference trajectory. The subjects were only instructed to follow the motion of the platform along the trajectory and since the pelvic rotation clearly exceeded the platform's orientation changes along the path, we assume that the presented approach to the prediction of the user's intention is valid and allows natural (intuitive) motion control of the platform. Precision of the reference trajectory shape execution improved by a large margin when the control was handed over to the subjects themselves, indicating their ability of precise control over the motion of the system. Interestingly, although leg injury simulation was used in run 6, results generally improved over those from run 3. This can be explained by additional experience gained during runs 4 and 5 and system's ability to cope with stray input signals, originating from the injury-induced changes in their walking patterns.

Based on the encouraging experimental test results on healthy subjects, the proposed control system will be evaluated on subjects with reduced motor and/or cognitive abilities, where similar results are expected. This will enable us to further check the validity of the above claims and adequacy of the default control system parameters (Section II-B) for a larger set of users.

References

[1] M. G. Bowden, A. E. Embry, L. A. Perry, and P. W. Duncan, "Rehabilitation of walking after stroke," *Curr. Treat. Opt. Neurol.*, vol. 14, no. 6, pp. 521–530, 2012.

[2] A. Olenšek *et al.*, "Adaptive dynamic balance training during overground walking with assistive device," in *Proc. IEEE RAS EMBS Int. Conf. Biomed. Robot. Biomechatron.*, 2012, pp. 1066–1070.

[3] X. Zhang, X. Kong, G. Liu, and Y. Wang, "Research on the walking gait coordinations of the lower limb rehabilitation robot," in *2010 IEEE Int. Conf. Robot. Biomimetr.*, 2010, pp. 1233–1237.

[4] J. F. Veneman *et al.*, "Design and evaluation of the LOPES exoskeleton robot for interactive gait rehabilitation," *IEEE Trans. Neural Syst. Rehabil. Eng*, vol. 15, no. 3, pp. 379–386, Sep. 2007.

[5] A. J. Rentschler, R. Simpson, R. A. Cooper, and M. L. Boninger, "Clinical evaluation of Guido robotic walker," *J. Rehabil. Res. Develop.*, vol. 45, no. 9, pp. 1281–1293, 2008.

[6] D. Chugo, T. Asawa, T. Kitamura, J. Songmin, and K. Takase, "A motion control of a robotic walker for continuous assistance during standing, walking and seating operation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2009, pp. 4487–4492.

[7] B. Graf, M. Hans, and R. D. Schraft, "Care-O-bot II—Development of a next generation robotic home assistant," *Auton. Robots*, vol. 16, no. 2, pp. 193–205, 2004.

[8] B. T. Tyson, M. T. Pham, N. T. Brown, and T. R. Mayer, "Patient safety considerations in the rehabilitation of the individual with cognitive impairment," *Phys. Med. Rehabil. Clin. N. Am.*, vol. 23, no. 2, pp. 315–334, May 2012.

[9] THERA-Trainer e-go 2015 [Online]. Available: http://www.thera-trainer.de/english/professional/gait/thera-trainer-e-go/index.html

[10] Z. Matjačić and A. Olenšek, Apparatus for training dynamic balance and turning manoeuvres during walking WO2014081400, 2014, Pat.Nr. WO2014081400.

[11] I. Cikajlo, J. Oblak, and Z. Matjačić, "Haptic floor for virtual balance training," in *Proc. IEEE World Haptics Conf.*, 2011, pp. 179–184.

[12] M. Di Natale, H. Zeng, P. Giusto, and A. Ghosal, *Understanding and Using the Controller Area Network Communication Protocol*. New York: Springer, 2012.

[13] G. I. Mary, Z. C. Alex, and L. Jenkins, "Reliability analysis of controller area network based systems — A review," *Int. J. Commun., Netw. Syst. Sci.*, vol. 6, pp. 155–166, 2013.

[14] K. H. Johansson, M. Törngren, and L. Nielsen, "Vehicle applications of controller area network," *Handbook Netw. Embed. Control Syst.*, vol. VI, pp. 741–765, 2005.

[15] S. Corrigan, Controller area network physical layer requirements Texas Instrum., Tech. Rep. SLLA270, 2008.

[16] PoKeys Library PoKeysLib PoLabs, 2015 [Online]. Available: http://blog.poscope.com/pokeyslib-pokeys-library/

[17] Documentation–ROS Wiki ROS [Online]. Available: http://wiki.ros.org/

[18] A. Olenšek, Z. Matjačić, and T. Bajd, "Muscle contracture emulating system for studying artificially induced pathological gait in intact individuals," *J. Appl. Biomechan.*, vol. 21, no. 4, pp. 348–358, 2005.

Matevž Bošnak received the B.S. and Ph.D. degrees in electrical engineering from the Faculty of Electrical Engineering, University of Ljubljana, Ljubljana, Slovenia, in 2009 and 2013, respectively,

.

He is currently a researcher and an assistant at the Faculty of Electrical Engineering, University of Ljubljana, Ljubljana, Slovenia. His main research area is autonomous mobile systems localization and control.

Igor Škrjanc is Professor at Faculty of Electrical Engineering, University of Ljubljana, Ljubljana, Slovenia. He is lecturing the basic control theory at graduate and advanced intelligent control at post-graduate study. His main research areas are adaptive, predictive, neuro-fuzzy and fuzzy adaptive control systems. His current research interests include also the field of autonomous mobile systems in sense of localization, direct visual control and trajectory tracking control. He has published 79 papers with SCI factor and 25 other journal papers. He is co-author and author of eight chapters in international books and co-author of scientific monograph with the title "Predictive Approaches to Control of Complex Systems" (Springer). He also serves as an Associated Editor the *Evolving Systems* journal.

Prof. Škrjanc he received the award for the first place, in 2007, at the competition organized by IEEE Computational Society, Learning from the data in the frame of IEEE World Congress on Computational Intelligence 2012, Brisbane, Avstralija: Solving the sales prediction problem with fuzzy evolving methods, and in 2013 the best paper award at IEEE International Conference on Cybernetics in Lausanne, Switzerland. In 2008 he received the most important Slovenian research award for the work in the area of computational intelligence in control—Zois award. In year 2009, he received a Humboldt research award for long term stay and research at University of Siegen. He is also a member of IEEE CIS Standards Committee and Slovenian Modelling and Simulation society and Automation Society of Slovenia. He also serves as an Associated Editor for IEEE TRANSACTION ON NEURAL NETWORKS AND LEARNING SYSTEM and IEEE TRANSACTION ON FUZZY SYSTEMS.

Embedded Control System for Smart Walking Assistance Device

Matevž Bošnak and Igor Škrjanc, *Member, IEEE*

***Abstract*—This paper presents the design and implementation of a unique control system for a smart hoist, a therapeutic device that is used in rehabilitation of walking. The control system features a unique human–machine interface that allows the human to intuitively control the system just by moving or rotating its body. The paper contains an overview of the complete system, including the design and implementation of custom sensors, dc servo motor controllers, communication interfaces and embedded-system based central control system. The prototype of the complete system was tested by conducting a 6-runs experiment on 11 subjects and results are showing that the proposed control system interface is indeed intuitive and simple to adopt by the user.**

***Index Terms*—Control design, user-intention-based control, walk assistance device, walk rehabilitation.**

I. Introduction

HOIST is a therapeutic device that was developed for the use in rehabilitation of walking after injuries or neural impairments. Rehabilitation of walking is a multi-step process that is aimed to return the freedom of motion to the patient. Is is a complex undergoing, mostly based on repetitive tasks execution [1].

It starts with intense therapy of the muscular system and proceeds with the supervised static and dynamic balance training [2]. The dynamic balance training is typically performed in presence of at least two expert therapist who manually assist the subject to walk and maintain the balance at the same time. Several technical solutions have been proposed to relieve the therapists from this physically intensive engagement, such as walking canes, simple static hoists, treadmill-based devices [3], robotic limb manipulators [4], movable support platforms [2] etc.

The popularity of the assisted living research topics resulted in presentation of multiple similar devices that were designed for walking assistance to elderly people and those with motor disabilities. Such devices are usually based on a movable platform that is either actively steered [5] or fully motorized and may combine additional features, such as active assistance for standing up and sitting down [6], or even help with other everyday tasks, such as picking up items [7]. Most of these systems are controlled with the use of steerable handle bars or static handles, equipped with force sensors. Since gait and balance instability is one of the most common sources of fall-induced injuries [8], it is essential that the falls are prevented during the rehabilitation – systems that can not provide the support for patient's full body weight during a failure event (loss of balance, tripping, stumbling, etc.) are not preferred since constant supervision and presence of the physiotherapist is required. The *Hoist* device prototype presented in this article provides a fail-safe and patient-engaging approach to gait rehabilitation.

The paper is structured as follows. Section II introduces the designed walking assist control system, then a detailed description of the rehabilitation platform that is built-upon in this research is given in the Section III. The description of the base platform, hardware upgrades, embedded system and sensory system is given in separate sub-sections. The paper is concluded with the experimental results in Section IV and a short discussion of the results and the possible upgrades is included in the end.

II. Walking Assist Control

Our research is focused on functional extension of an existing commercial rehabilitation platform named *THERA-Trainer e-go* [9], which was designed to offer the support for a strapped-in patient (device user) during the clinical rehabilitation process. Previous work on this platform [2] exposed a potential for a new, intuitive user-machine interface, that will be presented in this paper.

The device itself is built as a stable chassis with four caster wheels, equipped with battery power supply, electronics and two additional electrically driven wheels that enable it to move as a two-wheel robot (two of the caster wheels are lifted once driving wheels are attached). The interface between the device chassis and the vertical support frame is made of a ball joint, equipped with adjustable coaxial springs that have limited range of motion in terms of off-vertical deflection angles. The interface enables the user a certain degree of freedom in motion, but it also limits the user's motion if needed in order to prevent injuries in cases of tripping, stumbling or falling.

The idea behind the *Hoist* project was to augment the manual control mode of the existing walking assist system [illustrated in Fig. 1(a)] by observing the patient and adapting the control strategy accordingly. For this purpose, user position determination and intention-based-control system was designed that allows the patient to control the motion of the device by perturbing its position in regards to the platform base. This approach equates to a very intuitive way of controlling the device,

Manuscript received July 16, 2015; revised January 18, 2016; accepted February 29, 2016. Funded by Slovenian Research Agency . Project number L2-5471, Intelligent robot for walking training.

The authors are with Faculty of Electrical Engineering, University of Ljubljana, 1000 Ljubljana, Slovenia (e-mail: matevz.bosnak@fe.uni-lj.si).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNSRE.2016.2553369

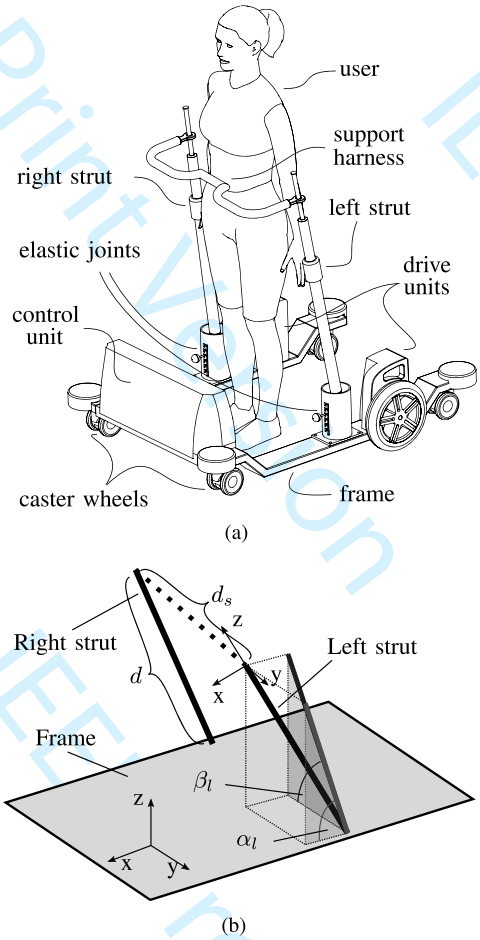


Fig. 1. Illustration of the *Hoist* platform. (a) Parts of the *Hoist* system. (b) Strut angles illustration.

since the control system works towards maintaining the neutral position of the user in regards to the platform. The final result is a system that follows the motion of the user, while providing a constant fail-safe support for the user.

A. User Position Determination

User's position in regards to the platform is determined by observing the angles between the left/right (vertical) struts and the base frame [illustrated in Fig. 1(b)]. For this purpose, struts and frame are equipped with bespoke tilt sensors (detailed in chapter Section III-C), allowing the relative angles between the frame and struts to be determined. The x-axis of the platform frame coordinate system is oriented towards the front of the device [towards the *control unit* in Fig. 1(a)] and the z-axis is oriented vertically in the up direction, while the y-axis is oriented towards the left side of the frame, creating a right-handed Cartesian coordinate system.

While at rest, the tilt sensor coordinate system axes are aligned with the platform frame coordinate system. Deflection angles of the vertical support struts are mechanically limited to approximately 15° from the vertical direction and the angular velocities of the struts are relatively low due to the long length of the struts. By using the small angle approximation of the trigonometric functions, rotation transformations between the platform frame coordinate system to tilt sensor coordinate system can be handled individually for the rotations around the x and y axes. Let α denote the angle of rotation around the sensor's y axis that rotates the platform frame coordinate system to a tilt sensor coordinate system and β denote the angle of rotation around the sensor's x axis that rotates the platform frame coordinate system to a tilt sensor coordinate system.

Perturbations of the user harness attachment point origin $(\Delta x, \Delta y)$ and orientation $\Delta\psi$ is therefore calculated using the following equations:

$$\Delta x = \frac{d(\tan\alpha_l + \tan\alpha_r)}{2} \tag{1}$$

$$\Delta y = -\frac{d(\tan\beta_l + \tan\beta_r)}{2} \tag{2}$$

$$\Delta\psi = \arctan\frac{d(\tan\alpha_r - \tan\alpha_l)}{2d_s} \tag{3}$$

where d is the vertical distance between the elastic joints and horizontal user support strut, d_s is the length of that strut (i.e., horizontal distance between the two vertical struts), while α_l, β_l, α_r and β_r are α and β angles for the left and right strut, respectively. Since the deflection angle of the struts is limited, the above equations can be simplified by replacing the tan functions with the linear small-angle approximation, which results in $\Delta x \approx d(\alpha_l + \alpha_r)/2$, $\Delta y \approx -d(\beta_l + \beta_r)/2$ and $\Delta\psi \approx d(\alpha_r - \alpha_l)/2d_s$.

These perturbations are used for estimating the position of the user in regards to the platform, which directly indicates the intention of the user for motion. This is justified by the following assumptions.

- When the user intends to move in the forward direction, the user will initiate a move in that direction, causing its relative position in regards to the platform to move forward.
- When the user intends to speed up or slow down, user's speed will increase or decrease, respectively. This directly results in relative position of the user in regards to the platform to change to more forward (user is accelerating) or more backwards (user is slowing down) position.
- When the user intends to change the direction of motion, the user will rotate his/her pelvis in that direction [10].

Based on these assumptions, the values of Δx and $\Delta\psi$ can be used directly as an estimation of user intent for acceleration and turning, while the side motion Δy can be used as a measurement of the user's walking (gait) quality [2], [11].

B. Control Algorithm

The control system must be robust enough to ignore normal oscillations in the signals that result from walking dynamics, however special care must be taken for allowing the user to execute controlled rapid stopping manoeuvres. Thus, the controller combines a regular P-controller and a filtered P-controller (PfP controller), a combination that allows the user to have the feeling of an immediate control (regular P part) and smooth changes in the average speed of the platform (filtered P part, that simulates the effect of an integrator. Such controller has a continuous transfer function of $H(s) = K_P + K_{P,f}(\tau s+1)^{-1}$. Input dead-band with variable width was added to the controller to tackle the problem of measurement (input) noise in vicinity

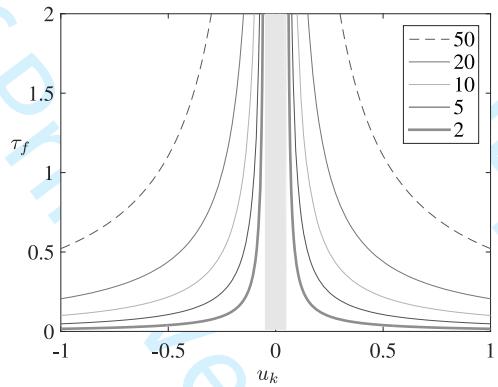


Fig. 2. Exponential braking ($|u_k| > D_x$ and $u_k \cdot v_{k-1} < 0$) – decay time constant at $D_x = 0{,}05$ (shaded area) and L_{dec} varying from 2,0 to 50.

of the user's neutral position and allow the user to keep the platform stationary when needed. Motion direction reversal situation is also handled separately, improving the deceleration during braking action.

The continuous transfer function of the controller has been discretized by zero-order hold method using the step time of $\Delta T = 10$ ms and the resulting difference equation can be written as

$$v_k = K_P \cdot u_k + v_{f,k} \tag{4}$$

$$v_{f,k} = K_{f,1} \cdot v_{f,k-1} + K_{f,2} \cdot (K_{P,f} \cdot u_k) \tag{5}$$

where $K_{f,1} = e^{-(\Delta T/\tau_f)}$. The time constant of the filter τ_f is defined as follows:

$$\tau_f = \begin{cases} \tau_{dec}, & |u_k| \le D_x \\ \Delta T \frac{L_{dec}}{|u_k| - D_x}, & |u_k| > D_x, u_k \cdot v_{k-1} < 0 \\ \tau_n, & |u_k| > D_x, u_k \cdot v_{k-1} \ge 0 \end{cases} \tag{6}$$

and $K_{f,2}$ is defined as

$$K_{f,2} = \begin{cases} 0, & |u_k| \le D_x \\ K_{2,r}, & |u_k| > D_x, u_k \cdot v_{k-1} < 0 \\ \left(1 - e^{-(\Delta T/\tau_n)}\right), & |u_k| > D_x, u_k \cdot v_{k-1} \ge 0 \end{cases}. \tag{7}$$

The three cases above define the parameters in three operating modes of the controller: dead-band operation ($u_k \le D_x$), motion reversal or braking operation ($u_k > D_x$ and $u_k \cdot v_{k-1} < 0$) and normal operation ($u_k > D_x$ and $u_k \cdot v_{k-1} \ge 0$).

In dead-band operation, input value is discarded (hence $K_{f,2}$ is set to 0), while the time constant of the filtered part τ_f is set to a fixed value of τ_{dec}. The motion direction reversal (i.e., braking) operation must allow the user to rapidly stop the motion of the platform, which is realized by cancelling the slow response rate of the implemented low-pass filter. Time constant of the filter τ_f in this operation mode is therefore dependent on the value of the input u_k and parameters D_x and L_{dec} (Fig. 2), that define the dead-band width and time constant function shape, respectively. Since the controller operates in this mode only outside the dead-band, a fixed value of τ_{dec} is used for the time constant inside it.

The parameter $K_{2,r}$ is selected to have a relatively low value ($0 < K_{2,r} \ll (1 - e^{-(\Delta T/\tau_n)})$) to obtain the deceleration towards the standstill position, while its value has to be greater than zero in order for the motion direction to truly reverse—without it, the condition for normal operation $u_k \cdot v_{k-1} \ge 0$ would never be met after initiating a direction reversal manoeuvre.

Although the choice of controller parameters greatly depends on the walking ability of the current user (mostly dead-band D_x, gains K_P, $K_{P,f}$, the shape of braking curve L_{dec} and maximum value of the filtered part $v_{f,\max}$), generic set of values for the parameters has been determined in experimental runs. By using these values, test subjects were able to control the speed of the platform in precise and smooth way, as demonstrated in the experimental study (Section IV).

Algorithm 1 Translational motion controller

procedure TRANSPI(u_k, v, v_f)
  if $|u_k| < D_x$ **then**
    $v_f \leftarrow e^{-(\Delta T/\tau_{dec})} \cdot v_f$
    $v \leftarrow v_f$
  else
    $u'_k \leftarrow u_k \cdot K_{P,f}$
    if $u_k \cdot v < 0$ **then**
      $v_f \leftarrow e^{-((|u_k| - D_x)/L_{dec})} \cdot v_f + K_{2,r} \cdot u'_k$
    else
      $v_f \leftarrow e^{-(\Delta T/\tau_n)} \cdot v_f + (1 - e^{-(\Delta T/\tau_n)}) \cdot u'_k$
    $v_f \leftarrow \text{limit}\{v_f, [-v_{f,\max}, v_{f,\max}]\}$
    $v \leftarrow K_P \cdot u_k + v_f$

The implemented algorithm is listed as Algorithm 1 (translational motion controller), where the following variables are used.

u_k	Input variable – normalized perturbation of the user's position in forwards–backwards direction (values in range $[-1{,}0; 1, 0]$).
v	Output variable that controls the reference speed for the servo motors, driving the wheels.
v_f	Filtered part of the output variable.
$v_{f,\max}$	Maximum value of the filtered part of the output variable that determines the maximum accumulated velocity.
D_x	Dead-band of the input variable (5% of the u_k range).
τ_n	Time constant of the filter in normal operation ($\tau_n = 10$ s).
τ_{dec}	Time constant of the filter in dead-band operation ($\tau_{dec} = 2$ s).
L_{dec}	Parameter that determines the gain of the exponential braking function ($L_{dec} = 2{,}0$).

$K_{2,r}$ See description in text ($K_{2,r} = 0{,}0001$).

K_P Proportional gain of the normal part of the controller ($K_P = 2{,}0$).

$K_{P,f}$ Proportional gain of the filtered part of the controller ($K_{P,f} = 50{,}0$).

Similar controller was designed for rotational motion. It is also based on the PfP concept without the direction reversal logic. The implementation of the algorithm is given below.

Algorithm 2 Rotational motion controller

1: **procedure** ROTPI($u_{\psi,k}$, ω, ω_f)
2: **if** $|u_{\psi,k}| < D_\psi$ **then**
3: $\omega_f \leftarrow e^{-(\Delta T/\tau_{\psi,dec})} \cdot \omega_f$
4: **else**
5: $u'_{\psi,k} \leftarrow K_{\psi,P,f} \cdot u_{\psi,k}$
6: $\omega_f \leftarrow e^{-(\Delta T/\tau_{\psi,n})} \cdot \omega_f + (1 - e^{-(\Delta T/\tau_{\psi,n})}) \cdot u'_{\psi,k}$
7: $\omega \leftarrow K_{\psi,P} \cdot u_{\psi,k} + \omega_f$

The following variables are used in algorithm 2 (rotational motion controller):

$u_{\psi,k}$ Input variable—normalized perturbation of the user's pelvis orientation (values in range $[-1{,}0; 1, 0]$).

ω Output variable that controls the reference rotational velocity for the servo motors, driving the wheels.

ω_f Value (state) of the filtered part of the controller.

D_ψ Dead-band of the input variable (5% of the $\Delta\psi$ range).

$\tau_{\psi,n}$ Time constant of the filter in normal operation ($\tau_{\psi,n} = 2$ s).

$\tau_{\psi,dec}$ Time constant of the filter in dead-band operation ($\tau_{\psi,dec} = 0{,}1$ s).

$K_{\psi,P}$ Proportional gain of the normal part of the controller ($K_{\psi,P} = 2{,}0$).

$K_{\psi,P,f}$ Proportional gain of the filtered part of the controller ($K_{\psi,P,f} = 50{,}0$).

III. Description of the Rehabilitation Platform *Hoist*

As noted in the previous section, modified commercial rehabilitation device *THERA-Trainer e-go* [9] is used as the platform in this research. The original device has been modified by replacing all electronics with bespoke components. The device has been equipped with multiple sensors that observe the device's and user's motion and position. For the purpose of odometry based position determination, each driven wheel has been equipped with a rotational encoder that measures speed and position of each wheel. In addition, three MEMS (*Micro-Electro-Mechanical*) angular rate and linear acceleration sensors are used to measure the chassis and user's support struts orientation (Fig. 3 shows the device during demonstration and testing). The data collection, processing and control is done on a Odroid XU3 embedded system, running Ubuntu linux distribution. The embedded system runs ROS (*Robot Operating System*) on top of Ubuntu operating system, with ROS taking care of inter-device communication tasks, data gathering, data processing synchronization, etc.

Fig. 3. *Hoist* system demonstration and testing.

A. Electrical and Drive System

Main power supply consists of two 12 V, 18 Ah lead-acid batteries that power the motor drivers via an emergency cut-off relay, the embedded computer via a 5 V switching power supply and CAN (*Controller Area Network*) devices via separate 12 V switching power supply.

The propulsion system basically mimics a two-wheeled robot with a differential drive. Each driven wheel has its own 100 W geared electric motor with a shaft encoder and a dedicated speed controller that is attached to the shared CAN bus on the device. Each motor driver executes a closed-loop speed control algorithm with adjustable current limits. Motor parameters (e.g., position, speed, motor current, etc.) are adjustable and observable using the CAN bus protocol.

B. CAN Communication Bus

CAN is a broadcasting digital bus designed to operate at preselected speeds from 20 kbit/s to 1 Mbit/s with the transmission rate selected depending on the bus length, topography and transceiver capabilities. CAN is an attractive solution for embedded control systems because of its low cost, light protocol management, the deterministic resolution of the contention, and the built-in features for error detection and retransmission [12].

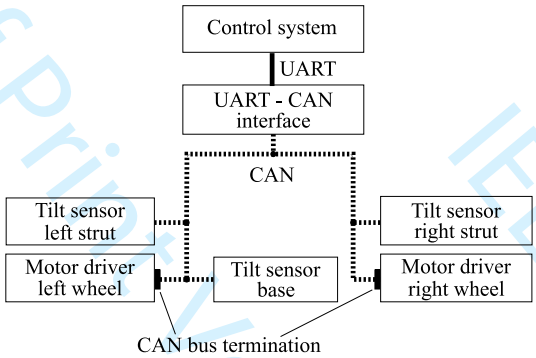


Fig. 4. CAN network on *Hoist* .

CAN has a proven reliability in automotive and process industry [13], [14] and was therefore a preferred option for a network between different low-level subsystems in the *Hoist* system. A relatively conservative bus speed of 250 kbit/s was chosen for the CAN bus in this project (illustrated in Fig. 4) that has a total length of 4 m with two 1-m-long stubs (unterminated bus ends). Split-type bus termination was used on left and right motor driver CAN nodes, as suggested in [15].

A simple bi-directional gateway between CAN bus and UART (Universal Asynchronous Receiver/Transmitter) was designed in order to simplify the access to the CAN bus from the high-level control system, which lacks support for CAN bus communication.

C. Tilt Sensors

The device is equipped with three tilt sensors, two of which are mounted on the vertical support struts and one mounted on the base (platform chassis). This sensor arrangement provides a simple method of measuring the relative orientation between the support frame struts and the base. Since the motion in each axis is restricted mechanically to approximately $\pm 15^\circ$ from vertical and corresponding angular velocities are low, tilt angles are calculated independently for each axis [producing angles α and β, as indicated in Fig. 1(b)]. Since the support struts are mechanically restricted from rotating around the third (vertical) axis, estimation of the third orientation angle is not required for this application.

The custom built tilt sensors host power supply system, microcontroller and MEMS-based gyroscope and accelerometer. As gyroscope output (angular rate) is biased and has to be integrated over time to obtain the orientation angle, dual simplified one-axis sensor fusion algorithms were implemented in the microcontroller of each tilt sensor, resulting in two independent orientation angle estimates per sensor (angles α and β).

The fusion algorithm for one axis is based on static Kalman filter. If the following states are selected for the state vector x_k

$$x_k = [\,\phi_k \quad \dot{\phi}_{b,k} \quad \theta_k \quad \dot{\theta}_{b,k}\,]^T \tag{8}$$

where ϕ_k, θ_k are the values of the angles ϕ, θ at the moment k, $\dot{\phi}_{b,k}$, $\dot{\theta}_{b,k}$ are gyroscope output biases at the moment k and input vector u_k contains current angular rates measurement

$$u_k = [\,\dot{\phi}_{m,k} \quad \dot{\theta}_{m,k}\,]^T \tag{9}$$

the system is presented in the following state space form:

$$x_k = \mathbf{A}x_{k-1} + \mathbf{B}u_k \tag{10}$$

$$y_k = \mathbf{C}x_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} x_k \tag{11}$$

$$\mathbf{A} = \begin{bmatrix} 1 & -\Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -\Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} \Delta t & 0 \\ 0 & 0 \\ 0 & \Delta t \\ 0 & 0 \end{bmatrix}. \tag{12}$$

Kalman filter can be evaluated offline because of the fixed state-transition, input and output matrices along with fixed state and measurement noise covariance matrices ($\mathbf{V}$ and $\mathbf{N}$, respectively). In order to obtain static Kalman filter gain matrix K_s, the following steps are evaluated iteratively until the K_k converges to K_s (*a posteriori* covariance matrix $\hat{\mathbf{P}}_k$ is initialized before starting the iteration)

1) $\mathbf{P}_k^* = \mathbf{A}\hat{\mathbf{P}}_{k-1}\mathbf{A}^T + \mathbf{V}$;
2) $\mathbf{K}_k = \mathbf{P}_k^*\mathbf{C}^T[\mathbf{C}\mathbf{P}_k^*\mathbf{C}^T + \mathbf{N}]^{-1}$;
3) $\hat{\mathbf{P}}_k = \mathbf{P}_k^* - \mathbf{K}_k\mathbf{C}\mathbf{P}_k^*$.

Based on the approximate values of the signal variances and some manual fine-tuning, the following values were used in our implementation:

$$\begin{aligned} &\Delta t = 0.01\ \text{s} \quad \hat{\mathbf{P}}_0 = \mathbf{I} \\ &\mathbf{V} = \text{diag}(0.1\ \text{rad}^2, 1\ \frac{\text{rad}^2}{\text{s}^2}, 0.1\ \text{rad}^2, 1\ \frac{\text{rad}^2}{\text{s}^2}) \\ &\mathbf{N} = \text{diag}(800\ \text{rad}^2, 800\ \text{rad}^2) \end{aligned} \tag{13}$$

which resulted in the following static Kalman gain K_s

$$K_s = \begin{bmatrix} 0.0102 & -0.0013 & 0 & 0 \\ 0 & 0 & 0.0102 & -0.0013 \end{bmatrix}^T. \tag{14}$$

The notation $\text{diag}(\lambda)$ was used to denote a diagonal matrix with the elements of vector λ on the diagonal. Static Kalman filter was then implemented in the microcontroller in the iterative form with time step of $\Delta t = 0.01$ s, consisting of the prediction step

$$\hat{x}_k = \mathbf{A}x_{k-1}^* + \mathbf{B}u_{k-1} \tag{15}$$

and the correction step

$$x_k^* = \hat{x}_k + K_s\left([\,\phi_{s,k} \quad \theta_{s,k}\,]^T - \mathbf{C}\hat{x}_k\right) \tag{16}$$

where the current sensor angles $\phi_{s,k}$ and $\theta_{s,k}$ were calculated from the acceleration measurements $a_{x,k}$, $a_{y,k}$ and $a_{z,k}$

$$\phi_{s,k} = \arctan\frac{-a_{z,k}}{a_{y,k}} \tag{17}$$

$$\theta_{s,k} = \arctan\frac{-a_{z,k}}{a_{x,k}}. \tag{18}$$

D. Servo Drives

Original motor drivers for the 40 V, 100 W geared electric motors with built-in shaft encoders were embedded into the control box hardware in front of the platform. Since there was no external interface available to control them, various commercial compact servo motor controllers were initially evaluated for

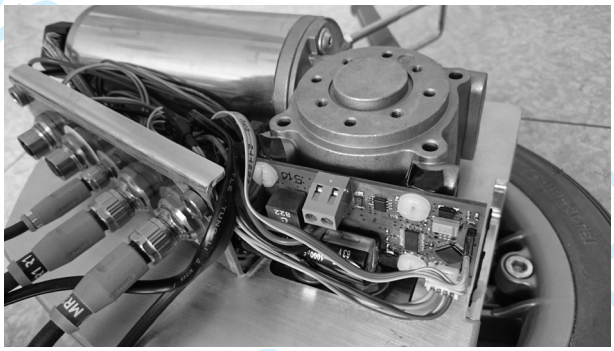


Fig. 5. Servo drive circuit fitted directly to the gearbox.

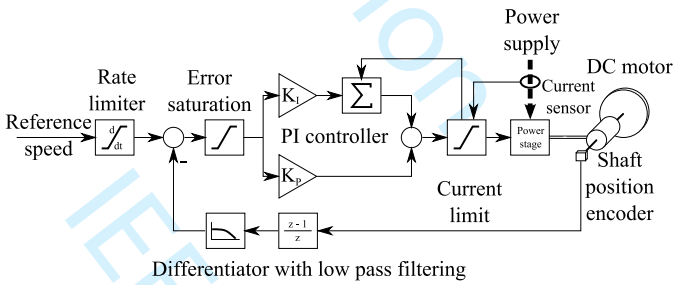


Fig. 6. Servo control loop.

this application, but most failed to work properly at low operating speeds. Therefore, a custom servo motor controller was designed, which operates appropriately at low rotational speeds and also fits into the tight space in the drivetrain housing itself (Fig. 5).

The application running on the servo controller allows the access to all controller parameters, current position, speed and motor current via CAN bus messages.

1) Electronics Design: The servo controller is built around Cortex-M0 microcontroller with an integrated CAN transceiver and smart H-bridge MOSFET (Metal–Oxide–Semiconductor Field-Effect Transistor) driver with external power MOSFET H-bridge. Since the microcontroller has the CAN bus functionality built-in, the driver is directly accessible from the control system for configuration and control. The physical dimensions of the designed servo controller allow the board to be mounted directly onto the gearbox, which also acts as a heatsink for the driver's PCB (Printed Circuit Board).

2) Servo Control Loop: The servo control loop is shown on Fig. 6. It is a semi-cascaded controller with current limit in internal loop and velocity-mode PI servo controller in external loop. Current shaft rotational velocity is obtained by differentiating the shaft position and filtering the result using a first order low-pass filter. The reference speed as the input variable is rate-limited to limit the maximum motor acceleration in order to reduce the mechanical stress of the components and limit the electric current spikes. There is an additional feedback loop from current limit controller back to integral of the PI controller to avoid saturation effects. The servo control loop is executed at a fixed frequency of 1 kHz.

E. Embedded System

Initially a compact embedded system BeagleBone Black was selected as the main computational platform for the project, but was later replaced by a more capable Odroid XU3 embedded system. Additionally, a small wireless router was used to provide a reliable WiFi connection between the *Hoist* system and the external monitoring and control computers. To interface the CAN bus devices, a USB-UART converter was used to talk to our custom UART-CAN gateway. Battery voltages, bumper switches and other configuration switches were connected to a PoKeys57U USB device, which is interfaced by the embedded computer with the help of a cross-platform open-source communication library PoKeysLib [16].

We used Ubuntu 14.04 LTS as the operating system of the embedded computer with meta-operating system ROS (Robot Operating System) version Indigo. ROS (Robot Operating System) is a meta-operating system, containing libraries and tools to help software developers create robot applications. It provides hardware abstraction, device drivers, libraries, visualizers, message-passing, package management and more (details are available in online ROS documentation [17]).

Our application was divided into a set of ROS nodes that can be started individually and used on different hosts as long as all the hosts are configured with the same ROS environment and network settings. Main ROS node was designed as an interface between ROS system and the physical hoist device. It uses CAN bus communication (using the CAN gateway) to communicate with servo motor drivers and tilt sensors, uses analog inputs of the PoKeys57U device to monitor the battery voltage and digital inputs to monitor the state of bumper switches. Motion commands are forwarded to wheel motor drivers, while wheel odometry data is generated from the reported encoder positions and exported along with sensor reading and switch statuses to ROS environment in the form of standard ROS message types.

F. Environment Sensing

Three parts of the primary hoist system are used to sense the physical environment—wheels odometry information, bumper switch sensors, and platform tilt sensor.

Each driven wheel of the hoist is equipped with an encoder, whose position is constantly tracked by the servo drive (for the purpose of closed-loop speed control and position tracking) and broadcasted over CAN bus. The main ROS node receives both left $\Delta E_{L,k}$ and right $\Delta E_{R,k}$ wheel encoder information from servo drives and combines this information into odometry data. The following equations are evaluated periodically for every step k:

$$\Delta d_k = \frac{\Delta E_{L,k} + \Delta E_{R,k}}{2} K_{p\to m} \tag{19}$$

$$\Delta \phi_k = \frac{\Delta E_{R,k} - \Delta E_{L,k}}{d_{ww}} K_{p\to m} \tag{20}$$

$$p_{x,k+1} = p_{x,k} + \Delta d_k \cos p_{\phi,k} \tag{21}$$

$$p_{y,k+1} = p_{y,k} + \Delta d_k \sin p_{\phi,k} \tag{22}$$

$$p_{\phi,k+1} = p_{\phi,k} + \Delta \phi_k \tag{23}$$

where $K_{p\to m}$ is a factor that translates the encoder change to driven wheel distance, d_{ww} is the distance between the driven

TABLE I
EXPERIMENTAL RUNS
(* PLATFORM WAS CONTROLLED BY THE SUBJECT)

	Run 1 □	2	3*	4 □	5	6*
Trajectory shape	□	☍	☍	□	☍	☍
Control by subject			✓			✓
Injury simulation				✓	✓	✓

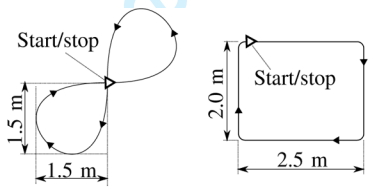


Figure-8 pattern ☍ Square pattern □

Fig. 7. Trajectory shapes used in the experiments.

wheels, while $p_{x,k}$, $p_{y,k}$ and $p_{\phi,k}$ are components of the position state vector, describing positions in axes x, y and rotation around vertical axis ϕ, respectively.

IV. RESULTS

The platform and proposed control system approach were tested both in simulated environment (using ROS and Gazebo with a mathematical and visual model of the platform) and in experimental study in laboratory environment, mainly testing the turning control of the platform. The experimental results will be presented next.

A. Experimental Results

Because the presented device is a prototype, the experiment was limited to 11 healthy subjects in six different runs (see Table I) in the confined space of about 5×5 m. The subjects were 10 male and 1 female volunteers (aged between 25 and 65), who had none or very limited experience with the presented device.

The experiment was executed in two stages, as it is imagined the device to be used in true rehabilitation sessions. First, the remote control functionality of the presented system was employed, where operator of the device forced the subject to follow the hoist in the prescribed pattern, drawn on the floor. During this first step, the subjects were only told to follow the motion of the platform. In the second stage, the subjects were instructed to control the system by following the same pattern. Each run consisted of at least two executions of the prescribed pattern.

Two patterns were used for trajectory shape during experiments (Fig. 7), while only the *figure of eight* pattern (denoted as ☍) was used for autonomous operation by the subject. The experimental runs 1 (square pattern □) and 2 (*figure of eight* pattern ☍) were first conducted (system controlled by the operator using remote control), then experimental run 3 (pattern ☍) followed, where the subject had full control of the system.

TABLE II
VALUES OF THE CROSS-CORRELATION FACTOR FOR ALL EXPERIMENTAL RUNS AT ZERO LAG WITH THE AVERAGE VALUE $\bar{C}$ FOR ALL SUBJECTS PER EACH RUN IN THE LAST ROW – HIGHER VALUE IS BETTER
(* PLATFORM WAS CONTROLLED BY THE SUBJECT)

	Run 1 □	2	3*	4 □	5	6*
1	0,09	0,60	0,95	-	-	-
2	-0,05	0,23	0,91	-	-	-
3	0,05	0,42	0,86	0,11	0,47	0,92
4	0,15	0,76	0,91	0,10	0,88	0,91
5	0,06	0,49	0,77	0,10	0,65	0,81
6	0,01	0,18	0,70	0,14	0,69	0,92
7	0,09	0,39	0,89	0,11	0,45	0,87
8	0,05	0,24	0,91	-	-	-
9	0,10	0,70	0,86	-	-	-
10	0,12	0,35	0,86	0,12	0,51	0,93
11	0,12	0,56	0,95	0,11	0,77	0,91
$\bar{C}$	0,07	0,45	0,87	0,11	0,63	0,90

TABLE III
VALUES OF NORMALIZED ISE COST FUNCTION J_N FOR ALL EXPERIMENTAL RUNS WITH THE AVERAGE VALUE $\bar{J}$ FOR ALL SUBJECTS PER EACH RUN IN THE LAST ROW – LOWER VALUE IS BETTER
(* PLATFORM WAS CONTROLLED BY THE SUBJECT)

	Run 1 □	2	3*	4 □	5	6*
1	30,35	9,42	3,78	-	-	-
2	32,38	8,41	5,88	-	-	-
3	30,57	29,73	3,56	30,92	5,58	2,12
4	36,93	4,06	3,41	110,63	11,12	3,61
5	50,06	6,80	17,80	31,17	13,30	9,87
6	29,83	3,58	7,05	72,55	3,63	7,79
7	30,34	5,01	7,29	32,15	12,39	4,43
8	10,64	6,84	5,05	-	-	-
9	31,79	24,76	4,78	-	-	-
10	34,96	14,36	6,62	40,52	31,22	7,40
11	74,01	12,20	8,68	78,78	8,58	7,72
$\bar{J}$	35,62	11,38	6,72	56,67	12,26	6,13

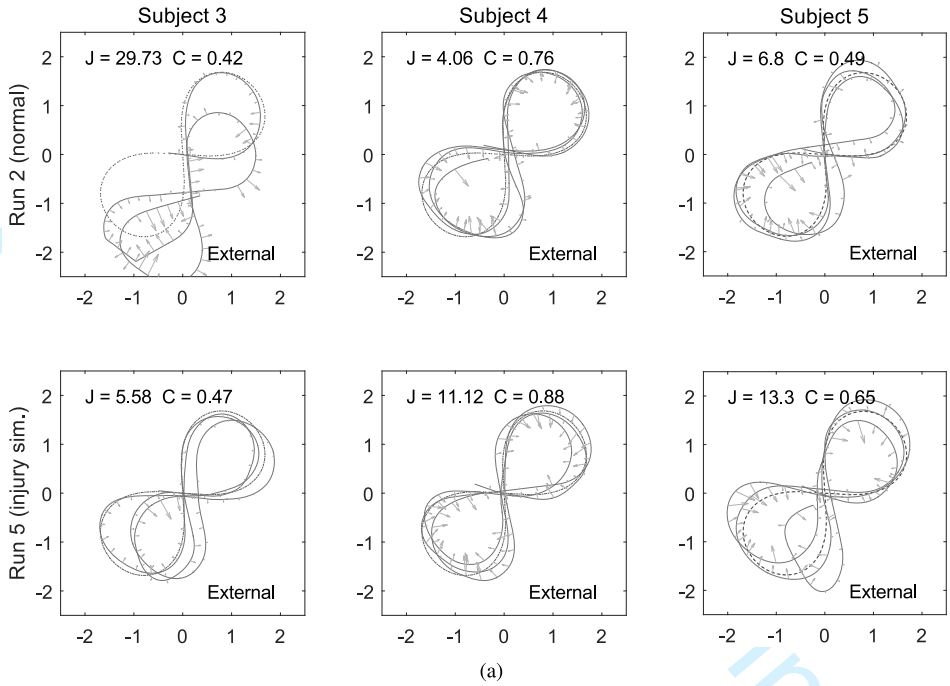
In the runs 4, 5, and 6, the subjects were equipped with leg-mounted contraption that simulated an injury to the leg's muscles or nerve system [18], which allowed us to evaluate the suitability of the proposed control system for the real patients in the clinical rehabilitation process. Since some subjects rejected installing the orthosis, there is some missing data (denoted by −).

For the purpose of results evaluation, all sensor measurements were recorded on-board the presented experimental system. This includes calibrated odometry data for evaluation of the tracking quality, remote control commands and sensory data, that was used for the user position assessment and user-input based control of the system.

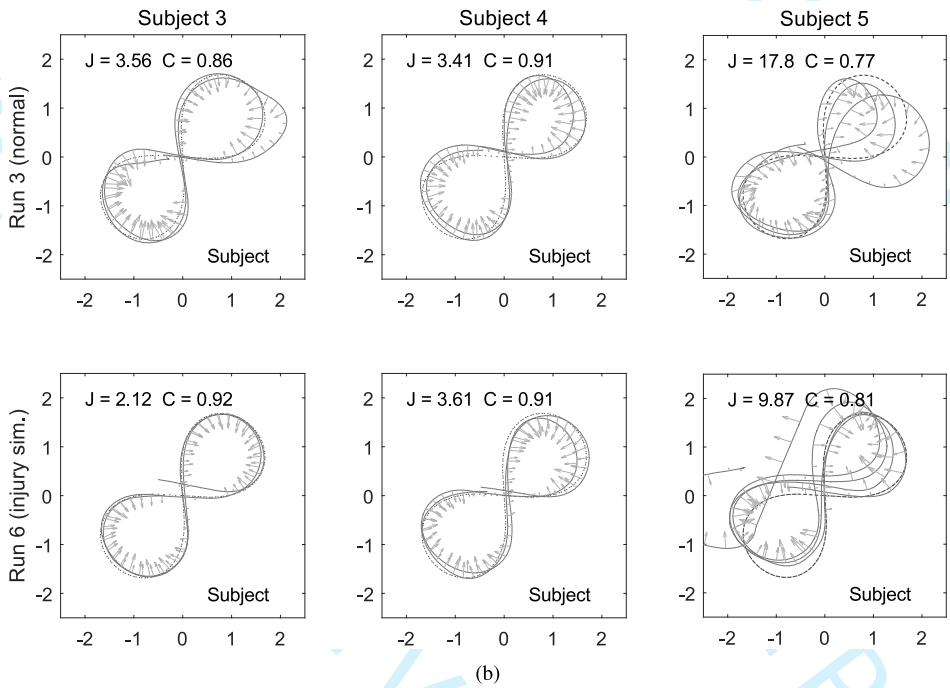
The results were evaluated both in terms of execution precision of the reference trajectory shape and in terms of cross-correlation between the $\Delta\psi$ (obtained from struts tilt angles) and rotational velocity around the vertical axis Ω_r of the reference trajectory.

Before executing the cross-correlation, the data sequences were normalized so that the autocorrelations at zero lag equal 1 (MATLAB's function *xcorr*). The results are gathered in Table II.

The execution precision of the reference trajectory shape was evaluated using the normalized ISE (integral square error) cost function J_N of the pose error, defined as a difference between the actual (x_i, y_i, ψ_i) and reference pose samples



(a)



(b)

Fig. 8. Experimental results of runs. (a) External (therapist) control without (first row) and with injury simulation (second row). (b) Subject (user) control without (first row) and with injury simulation (second row).

$(x_{r,i}, y_{r,i}, \psi_{r,i})$, consisting of both the position and the orientation parts

$$J_N = \frac{1}{L}\sum_{i=1}^{N}\left((x_i - x_{r,i})^2 + (y_i - y_{r,i})^2\right) + \frac{1}{L}\sum_{i=1}^{N}(\psi_i - \psi_{r,i})^2. \quad (24)$$

The normalization value L is the total length of the reference path, calculated from the odometry data. Values of cost functions for all runs are gathered in Table III.

Except for runs 1 and 4 (with a square shaped reference trajectory), in average results show a good correlation between the user's pelvis rotation angle $\Delta\psi$ and reference rotational velocity Ω_r. The most surprising are the correlation results of run 2, where subjects were being externally commanded (using a remote control feature) and were following the *figure of eight* pattern for the first time. Although they were told only to follow the motion induced by the machine, the results show that they were indicating the change of the direction by the additional rotation of the pelvis into the direction of the turn. This fact strengthens the idea of the proposed system being indeed very intuitive to

use. By comparing the results between the runs 2 and 5, latter being executed after runs 1 to 4, it is generally true that the indicative pelvis motion has been amplified after just a few experimental runs (taking 1–2 min each), additionally confirming the intuitive learning nature of the presented control system.

An interesting view on this matter can be given by observing the values of the cost function that was used to estimate the execution precision of the reference trajectory shape. The results [both numerically in Table III and graphically in Fig. 8(a) and (b)] show that subjects in control of the system were superior to external remote control in following the prescribed curve with and without injury simulation, resulting in lower value of the cost function in runs 3 and 6. Fig. 8(a) and (b) illustrates the results for subjects 3, 4, and 5 in runs 2, 3, 5, and 6 by overlaying odometry-based trajectories of the platform (solid line) over the reference trajectory (dotted line) with arrows illustrating the intended turning direction and amplitude, as estimated from the user intent of turning (chapter Section II-A).

The results for subject 5 show greater cost values and hence worse execution precision of the reference trajectory shape, which is the result of subject overpowering the machine, causing the systems drive wheels to slip and loose the correct odometry data.

No subject had problems with commanding the translational velocity of the platform, easily adapting it to the user's needs or abilities.

V. Conclusion

The prototype of walk rehabilitation system with the implementation of the presented control system has been proven in experimental study for being very intuitive and easy to adopt by the users. Most indicative are the experimental runs executed in *External control* mode (remote control by the operator), which produced data, that clearly show a positive correlation between the user's intention (pelvic rotation) and the rotational velocity along the prescribed reference trajectory. The subjects were only instructed to follow the motion of the platform along the trajectory and since the pelvic rotation clearly exceeded the platform's orientation changes along the path, we assume that the presented approach to the prediction of the user's intention is valid and allows natural (intuitive) motion control of the platform. Precision of the reference trajectory shape execution improved by a large margin when the control was handed over to the subjects themselves, indicating their ability of precise control over the motion of the system. Interestingly, although leg injury simulation was used in run 6, results generally improved over those from run 3. This can be explained by additional experience gained during runs 4 and 5 and system's ability to cope with stray input signals, originating from the injury-induced changes in their walking patterns.

Based on the encouraging experimental test results on healthy subjects, the proposed control system will be evaluated on subjects with reduced motor and/or cognitive abilities, where similar results are expected. This will enable us to further check the validity of the above claims and adequacy of the default control system parameters (Section II-B) for a larger set of users.

References

[1] M. G. Bowden, A. E. Embry, L. A. Perry, and P. W. Duncan, "Rehabilitation of walking after stroke," *Curr. Treat. Opt. Neurol.*, vol. 14, no. 6, pp. 521–530, 2012.

[2] A. Olenšek *et al.*, "Adaptive dynamic balance training during overground walking with assistive device," in *Proc. IEEE RAS EMBS Int. Conf. Biomed. Robot. Biomechatron.*, 2012, pp. 1066–1070.

[3] X. Zhang, X. Kong, G. Liu, and Y. Wang, "Research on the walking gait coordinations of the lower limb rehabilitation robot," in *2010 IEEE Int. Conf. Robot. Biomimetr.*, 2010, pp. 1233–1237.

[4] J. F. Veneman *et al.*, "Design and evaluation of the LOPES exoskeleton robot for interactive gait rehabilitation," *IEEE Trans. Neural Syst. Rehabil. Eng*, vol. 15, no. 3, pp. 379–386, Sep. 2007.

[5] A. J. Rentschler, R. Simpson, R. A. Cooper, and M. L. Boninger, "Clinical evaluation of Guido robotic walker," *J. Rehabil. Res. Develop.*, vol. 45, no. 9, pp. 1281–1293, 2008.

[6] D. Chugo, T. Asawa, T. Kitamura, J. Songmin, and K. Takase, "A motion control of a robotic walker for continuous assistance during standing, walking and seating operation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2009, pp. 4487–4492.

[7] B. Graf, M. Hans, and R. D. Schraft, "Care-O-bot II—Development of a next generation robotic home assistant," *Auton. Robots*, vol. 16, no. 2, pp. 193–205, 2004.

[8] B. T. Tyson, M. T. Pham, N. T. Brown, and T. R. Mayer, "Patient safety considerations in the rehabilitation of the individual with cognitive impairment," *Phys. Med. Rehabil. Clin. N. Am.*, vol. 23, no. 2, pp. 315–334, May 2012.

[9] THERA-Trainer e-go 2015 [Online]. Available: http://www.thera-trainer.de/english/professional/gait/thera-trainer-e-go/index.html

[10] Z. Matjačić and A. Olenšek, Apparatus for training dynamic balance and turning manoeuvres during walking WO2014081400, 2014, Pat.Nr. WO2014081400.

[11] I. Cikajlo, J. Oblak, and Z. Matjačić, "Haptic floor for virtual balance training," in *Proc. IEEE World Haptics Conf.*, 2011, pp. 179–184.

[12] M. Di Natale, H. Zeng, P. Giusto, and A. Ghosal, *Understanding and Using the Controller Area Network Communication Protocol*. New York: Springer, 2012.

[13] G. I. Mary, Z. C. Alex, and L. Jenkins, "Reliability analysis of controller area network based systems — A review," *Int. J. Commun., Netw. Syst. Sci.*, vol. 6, pp. 155–166, 2013.

[14] K. H. Johansson, M. Törngren, and L. Nielsen, "Vehicle applications of controller area network," *Handbook Netw. Embed. Control Syst.*, vol. VI, pp. 741–765, 2005.

[15] S. Corrigan, Controller area network physical layer requirements Texas Instrum., Tech. Rep. SLLA270, 2008.

[16] PoKeys Library PoKeysLib PoLabs, 2015 [Online]. Available: http://blog.poscope.com/pokeyslib-pokeys-library/

[17] Documentation–ROS Wiki ROS [Online]. Available: http://wiki.ros.org/

[18] A. Olenšek, Z. Matjačić, and T. Bajd, "Muscle contracture emulating system for studying artificially induced pathological gait in intact individuals," *J. Appl. Biomechan.*, vol. 21, no. 4, pp. 348–358, 2005.

Matevž Bošnak received the B.S. and Ph.D. degrees in electrical engineering from the Faculty of Electrical Engineering, University of Ljubljana, Ljubljana, Slovenia, in 2009 and 2013, respectively,

.

He is currently a researcher and an assistant at the Faculty of Electrical Engineering, University of Ljubljana, Ljubljana, Slovenia. His main research area is autonomous mobile systems localization and control.

Igor Škrjanc is Professor at Faculty of Electrical Engineering, University of Ljubljana, Ljubljana, Slovenia. He is lecturing the basic control theory at graduate and advanced intelligent control at post-graduate study. His main research areas are adaptive, predictive, neuro-fuzzy and fuzzy adaptive control systems. His current research interests include also the field of autonomous mobile systems in sense of localization, direct visual control and trajectory tracking control. He has published 79 papers with SCI factor and 25 other journal papers. He is co-author and author of eight chapters in international books and co-author of scientific monograph with the title "Predictive Approaches to Control of Complex Systems" (Springer). He also serves as an Associated Editor the *Evolving Systems* journal.

Prof. Škrjanc he received the award for the first place, in 2007, at the competition organized by IEEE Computational Society, Learning from the data in the frame of IEEE World Congress on Computational Intelligence 2012, Brisbane, Avstralija: Solving the sales prediction problem with fuzzy evolving methods, and in 2013 the best paper award at IEEE International Conference on Cybernetics in Lausanne, Switzerland. In 2008 he received the most important Slovenian research award for the work in the area of computational intelligence in control—Zois award. In year 2009, he received a Humboldt research award for long term stay and research at University of Siegen. He is also a member of IEEE CIS Standards Committee and Slovenian Modelling and Simulation society and Automation Society of Slovenia. He also serves as an Associated Editor for IEEE TRANSACTION ON NEURAL NETWORKS AND LEARNING SYSTEM and IEEE TRANSACTION ON FUZZY SYSTEMS.