Real-time Control of Whole-body Robot Motion and Trajectory Generation for Psychotherapeutic Juggling in VR

Pouya Mohammadi[1*], Milad Malekzadeh[1], Jindrich Kodl[2], Albert Mukovskiy [2], Dennis L. Wigand[3], Martin Giese[2] and Jochen J. Steil[1]

***Abstract*— Motor rehabilitation is in increasingly high demand to deal with minor functional motor impairments resulting from stroke, cerebellar ataxia, or Parkinson's disease. Juggling physiotherapy has shown to induce brain plasticity and to improve coordination and balance in this context. The physiotherapy, however, relies on large number of repetitions to be effective which prompts to deploy robots to release the burden on therapists both in terms of time as well as physical strain. This paper provides a framework to enable juggling games for patients in interacting with robots through Virtual Reality (VR). A set of throwing motions is recorded from the therapist and is retargeted to the humanoid robot COMAN's wrist. The respective whole-body motion is then solved in a stack of Quadratic Programs (QP) in a real-time architecture that integrates OROCOS and Gazebo. The resulting motion is finally streamed to VR for animation of the robot and the thrown ball, which the user can catch in VR using a controller device. We regard the VR setting as an essential step towards physiotherapeutic robotic juggling, because it ensures safety of the patients and effective testing of the methods and already has potential for actual therapeutic intervention. The control framework, however, is already validated in this paper for switching to full real-time operation on the physical robot.**

I. Introduction

The demand for specific physiotherapy procedures which deal with functional motor impairments continuously grows due to the aging world population and the increasing number of patients who survive medical emergencies like stroke or suffer from cerebellar ataxia and Parkinson's disease. Such procedures aim to enhance the functional ability and to restore involvement in daily social life, but they need to employ an intensive intervention to be effective [1]. To realize the latter is challenging because it requires task-specificity and high-level of motivation, and is primarily based on rote exercise. A rehabilitation therapy thus often involves daily one-on-one interactions with the therapist and can last for extended periods of time. Overall, the physiotherapy process places a significant burden on the therapists and on the healthcare systems, prompting researchers and clinicians to increasingly consider robotic devices in motor rehabilitation.

This work is funded by the European Community's Horizon 2020 robotics program ICT-23-2014 under grant agreement 644727 - CogIMon.

*Corresponding author Email: p.mohammadi@tu-bs.de

[1]Institute for Robotics and Process Control (IRP), TU Braunschweig Muehlenpfordtstr. 23, 38106 Braunschweig - Germany

[2]Section for Computational Sensomotorics, Department of Cognitive Neurology, HIH, CIN, University Clinic Tbingen, Tbingen, Germany

[3]Technical Faculty, Bielefeld University, Germany

Fig. 1. User equipped with Head Mounted Display and controllers performing the virtual ball catching task.

The advantages of using robotic systems are their ability to provide continuous repetitive high-intensity therapy, detailed and accurate measurement of patients performance, precise reaction to patients motion and continual assessment of changes in motor function through performance measures. The proper understanding of the patient's responses can then be used to encourage correct movements and improve motor function and discourage incorrect movements, because the latter can lead to a deterioration of the motor behavior. Robotic assistance in the rehabilitation task enables the therapist to fully focus on assessing the patients movement or simply to treat more patients effectively. Current robotic systems for motor impairment rehabilitation are custom-built devices focusing mostly on the upper extremity. Many of these robotic rehabilitation devices utilize gamification to engage and motivate the patient throughout the intensive procedure. A wide range of systems with various DOF (1 to 18) have been introduced for hand rehabilitation (wrist and fingers). Burdet et al. [2] conducted a detailed review of these using 8 out of 30 reported robotic devices. They classified the devices into different categories according to their design, usage, DOF, therapy schedule, results and other features. The clinical studies indicate that robot-assisted hand rehabilitation can provide a reduction of the motor impairments in arm and hand and improves the functional use of the affected hand.

They emphasize that the outcomes are preliminary but very promising. The robotic devices, however, are rather simple and do not require a complex controller.

Several papers have reviewed the performance and efficiency of robot-assisted therapy [3], [4]. They indicate that successful motor rehabilitation requires early training of upper body which is an intensive and task-specific activity. Early robotic training can be a suitable alternative. The 3-DOF NeReBot, for example, is programmed to perform repetitive assistive movements of the proximal upper limb (shoulder and elbow) in [4].

Introducing a humanoid robot to the robotic rehabilitation process could help to further increase the patients motivation by introducing the social aspect to the process. In [5], [6], the humanoid robot Nao is utilized for children with cerebral palsy and arm impairment to engage them in a movement imitation game. Nao enhanced motivation and thus supported rehabilitation.

A well established therapeutic training process that could benefit from robotic assistance is one that utilizes coordinative games through throwing and catching balls, a goal-directed functional task that is familiar and motivating, yet challenging and intensive. It has been shown that this task is not only motivating, demanding and induces brain plasticity [7], [8], but since it requires the coordination of arm movement and postural control, it improves coordination and balance [1]. Furthermore, studies exploiting observation and response to natural ball movement, enhance the arm motion and trunk-arm coordination of Parkinson's patients [9], [10]. Therefore, ball throwing and catching tasks appear to be an appropriate training procedure to improve patients' motor functions.

An important feature of this therapy is that the complexity can be scaled by using up to three balls, varying the catching technique (under- and over-hand), employing deceptive throwing, or by altering the order of throwing and catching when using multiple balls. The therapist has too assess the patient's performance and adjust the throws to vary the difficulty of the task throughout the rehabilitation procedure accordingly. Assessment and creation of variation in throwing difficulty and accuracy could be effectively reproduced and possibly improved by the robotic system.

This paper introduces the technological means for our longer-term goal to realize human-humanoid therapeutic juggling, a term we use to summarize the described coordinative games through throwing and catching balls. While in the first stage we achieve real-time synthesis of robot throwing in Virtual Reality, we meet from the outset the requirements that arise from scaling up the system to the real patient-robot interaction (Sec. II). Thus the VR setup allows to test feasibility and physical constraints safely and effectively before integrating the real robot, but with the actual real-time controller already in-the-loop. Furthermore, this intermediate VR-system has already significant value in itself for experimentation, verification and possibly actual therapy, as we will discuss in the text and the conclusion section. Note that this goes significantly beyond plain VR systems, where for instance automatic coaching of physical training [11] was proposed, but respective avatars do not need to implement physically realistic and feasible robot motion.

Our system retargets demonstrations of a therapist's throwing trajectories to be kinematically feasible for COMAN. It uses the retargeted motions as input to a generative learning system that enables real-time synthesis of throwing motions for particular targets (Sec. III). The synthesized motions serve as reference input in a stack of quadratic programs (Sec. IV) that provide the COMAN whole-body motion in real-time and in a control framework that allows for transparent switching between dynamic simulation and the real robot. The resulting physically feasible and realistic motion is finally animated in VR (Sec. V). Finally, we provide performance validations and illustration of the complete system in Sec. VI).

II. Requirements and system architecture

Given the ambitious goal of realizing therapeutic intervention with actual patients, a significant number of requirements arise on functional and technological levels.

In short these requirements can be listed as

R0 Safety: The system must be safe for the user at all times and under all conditions.

R1 Flexibility: The robot throwing must be versatile and generated on-the-fly in real-time, in order to provide the capability for reactive and variable behavior.

R2 Realtime: The motion generation and robot control must run at 1 kHz for the real robot. Furthermore, Humans' perceive visual cues as fast as 13 milliseconds [12]. While this varies on individual basis, the VR backend must run at a minimal frequency of 60 Hz to evoke the impression of natural movement.

R3 Platform independency: Switching between robot simulation and real robot as well as easy adaption to new robots is desired without imposing structural changes in the system architecture.

R4 Reusability: Given the complexity of the overall system, the architecture must support modular implementation and easy deployment and integration of components.

Figure 2 depicts a sketch of the main system components and their connections. We describe their setup and interplay with respect to the requirements.

Patient safety **(R0)** is naturally of utmost importance. In the VR scenario, there is not direct interaction with the robot and the patients can stand or sit dependent on their impairments. Additional guarding mechanism for boundary limitations are possible and should be implemented according to patients's mobility. For the real robot, a sufficient safety margin will be enforced by mechanical restraints, such that it can not touch a patient even in the case of a technical failure.

The motion generation part (Figure 2, left) provides the necessary flexibility **(R1)** to vary the throwing as required in the therapeutic scenario. It implements a real-time capable motion synthesis that can produce input wrist trajectories for the robot controller within the control cycle as was demonstrated in [13]. It is, however, non-trivial to provide kinematically feasible and on-target throws, as will be described further in Sec. III. The system can use either offline

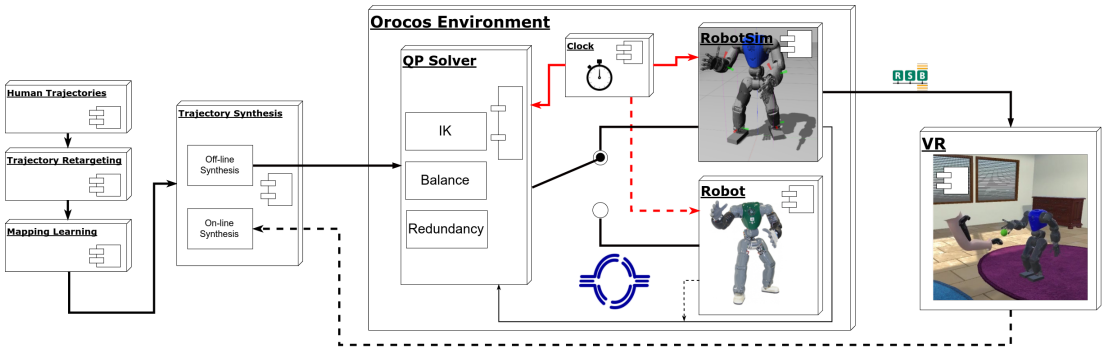

Fig. 2. Proposed architecture: All communications are managed using real-time framework "Orocos-RTT". This implementation allows transparent switching between simulated and real robot.

determined targets or, in later stages, evaluate feedback online (dashed line in Figure 2). While the latter is not integrated in the system yet, the feedback pipeline has already been tested independently [13] and will be added in the future development. For the reported performance tests, we simply generate a set of trajectories offline and replay them. It adds further flexibility that in principle any motion generator could be used to provide input to the robot controller, given it matches the requirements on real-time capability. Ideally, such motion generation will then be included and deployed as a component in the central, real-time component based controller.

Regarding real-time **(R2)**, the motion generation system, the central real-time controller and the VR communicate via the robotics service bus RSB [14], where the VR is soft-real-time and currently running at 90 Hz. The actual QP-based controller is realized in CoSiMA[1], an Orocos [15] based execution framework developed in the H2020 project CogIMon[2]. It features a component-based architecture, where all inter-component communications are handled by RTT.

CoSiMA facilitates transparent switching between the real and simulated robot (**R3**) by providing identical interfaces for both by means of Orocos components that wrap the respective low level drivers. In principle, it is also possible to simply load a different robot model into CoSiMA and the VR environment without any structural changes to the architecture **(R3)**.

Finally, reusability **(R4)** is supported through using the Orocos component framework, which allows to load entire controller components, robot models, and simulation tools from a Orocos Component Library (ocl). This greatly facilitates reuse and sharing of components, and the actual deployment. Note that also domain-specific language (DSL) based programming tools for this part of the system are available to further simplify the system design using CoSiMA [16], but their discussion is beyond the scope of this paper.

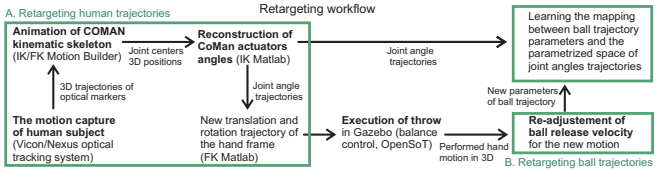

Fig. 3. Retargeting workflow: Human captured kinematic trajectories for the robot (A). Offline re-evaluation of motion using robot constraints and the retargeting of the ball trajectory (B). Machine learning of the mapping from ball trajectory parameters to parametric space of kinematic motion primitives for generative motion synthesis (C).

III. Trajectory Generation

Therapeutic juggling requires a very flexible trajectory generation that serves as input for the whole-body real-time motion control. This is highly non-trivial: if the throw is on target, but kinematically infeasible, then the robot cannot track it precisely and would miss the target. If however the motion is retargeted to be feasible for the robot, it will not be on the target as in the demonstration. Furthermore, we assume that human-like throws are beneficial for the patient and therefore make the trajectories as human-like as possible. That is actually a useful hypothesis, which we can test only if such learning is realized. Therefore, trajectory generation is performed in several steps shown in Figure 3. These together guarantee that a motion generator can be learned so that it produces feasible, on-target human-like trajectories in real-time for some given throwing target as input to the whole-body motion control (cmp. Figure 2).

Motion capture from human throws: Subject's underhand ball throwing movements were captured using a Vicon (Vicon Motion Systems, Oxford, UK) optical tracking system, exploiting the software NEXUS. The full body trajectories were cropped to the interval between the start of the arm motion from the relaxed downwards position until the moment of ball release.

Retargeting of human trajectories: Arm trajectories were retargeted to the robot using a two-stage process. As first step, we retargeted the whole body trajectory to the COMAN

[1]http://cogimon.github.io/
[2]http://cogimon.eu

skeleton using Motion Builder (MB, Autodesk) exploiting the built-in IK and geometry rescaling tools of the Motion Builder. The results are joint-center trajectories of the COMAN skeleton that reproduce the timing of the human subjects, however ignoring joint limits. As second step, given the joint centers positions for each time point, we re-estimated trajectories compatible with the COMAN kinematic chain using MATLAB with a customized IK tool that takes into account constraints defined by the joint angle limits. The results are trunk and right arm angle values for every time step. Next, this right arm and shoulder trajectory was played back using the same kinematic model in MATLAB, keeping all other body joints constant. In addition, the starts of the trajectories were matched to the angle values of the parking position of the robot. These hand frame trajectories then form the reference input for the whole body controller in the robot simulator. From the resulting new arm trajectories the resulting hand position, orientation and corresponding velocities at the moment of ball release were determined as critical parameters that control the ball trajectory.

Retargeting of ball trajectories: Since the robot's geometric parameters differ from the human ones, but the timing of the movements is preserved, the ball release velocity must be rescaled accordingly. Additionally, the robot's hand orientation at the moment of release differs from the human one after retargeting of the arm trajectory (though the rotational velocities often are very similar). We simplified the ball retargeting by approximating the release velocity through a) rescaling of absolute ball velocity and b) reproduction of the initial flying direction at the release moment. We used as scaling factor of the absolute ball velocity the ratio of the absolute velocities of the hand center of the robot (retargeted motion) and of the human (original data). To re-compute the ball direction vector we estimated the rotational coordinate transform between the hand frames of original and retargeted motion at the moment of ball release. We then applied this transformation to the direction vector. As next step, we re-simulated a simple parabolic flight trajectory of the ball that is compatible with the computed initial velocity. This simulation neglects air-drag forces. The workflow block scheme on the Figure 3 depicts the relations between different stages of the retargeting procedures.

Learning of the generative model: For learning, the arm trajectories are represented as time series of joint angle values, starting from the parking position until the moment of ball release. The arm trajectories were represented by an anechoic mixture model, a linear weighted mixture of time-delayed basis functions ('source functions' [17]). We learn the shapes of source functions and the parameters of anechoic mixture model by our customized algorithm [18] and save all these parameters together with the real-time value from the moment of motion start until ball release. For the generation of throwing trajectories we learn statistically the map from the target parameters onto the mixing parameter space for the motion primitives, i.e. the space of anechoic mixture weights and mean values. The resulting mapping is constructed in two steps. First, we learn the map from the space of target parameters onto the release timing and corresponding ball velocity. Second, we learn the map from the space of target parameters extended by these two parameters (timing and ball velocity) onto the space of anechoic mixture weights and mean values. The underlying function is learned by Locally Weighted Linear Regression (LWLR) [19]. Once this offline learning is performed, the evaluation of a new motion for desired target parameters can easily be executed in real-time.

IV. QP-BASED WHOLE-BODY MOTION GENERATION

This section describes the actual real-time control of the (simulated) robot (cmp. Fig.2, central block), whereas it receives a reference input trajectory for the wrist point as provided in the previous section. The (simulated) robot executes the joint motion and releases the ball at the desired time, which then follows a parabolic trajectory. The robot's base link position and orientation, the joint angles, and the ball coordinates are provided as output in each control cycle and further processed by the VR module as described in the next Section.

The controller solves a cascade of QP problems which take into account a number of tasks or constraints. We first introduce the notation for defining constraints for the whole-body control of the COMAN and elaborate on the concrete selection of priorities subsequently.

Definitions and notation: Classical approaches to inverse kinematics problem such as [20] cannot deal with inequality constraints and they are susceptible to ill-conditioned Jacobians. In recent years, it has therefore become popular to solve constraint quadratic programs instead [21], [22], which belong to the general class of convex optimization problems [23]. We formulate QP at velocity level[3]: for an n-DOF robot and given desired end-effector velocity $\dot{\boldsymbol{x}}$, calculate the required joint velocities $\dot{\boldsymbol{q}}$ to minize the kinetic energy

$$\underset{\dot{\boldsymbol{q}}}{\operatorname{argmin}} \frac{\dot{\boldsymbol{q}}^{\mathrm{T}} \boldsymbol{H} \dot{\boldsymbol{q}}}{2} \tag{1}$$

$$\text{s. t.} \quad \boldsymbol{J}\dot{\boldsymbol{q}} = \dot{\boldsymbol{x}} \tag{2}$$

$$\dot{\boldsymbol{\xi}}^{-} \leq \dot{\boldsymbol{q}} \leq \dot{\boldsymbol{\xi}}^{+}, \tag{3}$$

where $\boldsymbol{H} \in \mathbb{R}^{n \times n}$ is the inertia matrix. It yields the minimum norm solution for $\boldsymbol{H} = \boldsymbol{I}$. (1) is the cost function and (2) and (3) are the equality and inequality constraints.

The bilateral constraint represented in (3) by $\dot{\boldsymbol{\xi}}^{-}$ and $\dot{\boldsymbol{\xi}}^{+}$ is the secondary lower and upper limits of joint velocity considering the real limits of joint position ($\boldsymbol{q}^{-}$, $\boldsymbol{q}^{+}$) and joint velocities ($\dot{\boldsymbol{q}}^{-}$, $\dot{\boldsymbol{q}}^{+}$) as

$$\begin{aligned} \dot{\boldsymbol{\xi}}^{-} &= \max(\dot{\boldsymbol{q}}^{-}, \mu(\boldsymbol{q}^{-} - \boldsymbol{q})) \\ \dot{\boldsymbol{\xi}}^{+} &= \min(\dot{\boldsymbol{q}}^{+}, \mu(\boldsymbol{q}^{+} - \boldsymbol{q})). \end{aligned} \tag{4}$$

The parameter μ tunes how the joint limits will be avoided.

An equivalent formulation has been proposed in [24] and [22] by eliminating the equality constraint (2) because the

[3]QP also exists in the acceleration level [21] to minimize the torque or acceleration norm which is out of focus of this paper.

Jacobian in equation (2) is sometimes rank-deficient:

$$\dot{\boldsymbol{q}} = \underset{\dot{\boldsymbol{q}} \in \mathcal{S}}{\operatorname{argmin}} \frac{1}{2} \|\boldsymbol{J}\dot{\boldsymbol{q}} - \dot{\boldsymbol{x}}\|. \tag{5}$$

The set $\mathcal{S}$ is expressed in closed-form as

$$\mathcal{S} = \{\boldsymbol{J}^{\dagger}\dot{\boldsymbol{x}} + (\boldsymbol{I} - \boldsymbol{J}^{\dagger}\boldsymbol{J})\boldsymbol{\nu}, \boldsymbol{\nu} \in \mathbb{R}^n\}, \tag{6}$$

where $\boldsymbol{J}^{\dagger}$ is the pseudo-inverse of the Jacobian. $\boldsymbol{\nu}$ is an arbitrary vector projected on the null-space of $\boldsymbol{J}$.

Hierarchy of multiple constraints: When more than one cost function needs to be minimized, the solution to one constraint is either as important as some other solutions (soft constraint) or must lie in the null-space of previous solutions (hard constraint).

A set of generic tasks including equality constraints $T_i = \langle \boldsymbol{J}_i, \dot{\boldsymbol{x}}_i \rangle$ along with inequality constraints $C_i = \langle \boldsymbol{A}_i, \boldsymbol{b}_i \rangle$ can be put together to form a 'cascade of QP problems'

$$\underset{\dot{\boldsymbol{q}}}{\operatorname{argmin}} \frac{1}{2} \|\dot{\boldsymbol{q}}\| \quad \text{s.t.} \tag{7}$$

$$T_i \coloneqq \quad \dot{\boldsymbol{q}}_i = \underset{\dot{\boldsymbol{q}} \in \mathcal{S}_i}{\operatorname{argmin}} \frac{1}{2} \|\boldsymbol{J}_i\dot{\boldsymbol{q}} - \dot{\boldsymbol{x}}_i\| \tag{8}$$

$$C_i \coloneqq \quad \boldsymbol{A}_i\dot{\boldsymbol{q}} \leq \boldsymbol{b}_i \tag{9}$$

where $\boldsymbol{J}_i$, $\dot{\boldsymbol{x}}_i$ and $\dot{\boldsymbol{q}}$ represent the corresponding Jacobian and desired Cartesian and joint velocity vectors of the i^{th} equality constraint (T_i). $\boldsymbol{A}_i$ and $\boldsymbol{b}_i$ determine the i^{th} inequality constraint (C_i, if there is any)[4]. In practice, the bilateral joint-velocity limits (4) are the outcome of the unification process of equation (9) where, $\boldsymbol{A}_i = \boldsymbol{I}$ and $\boldsymbol{b}_i = \{\boldsymbol{b}_i^l,\ \boldsymbol{b}_i^u\}$.

Hard constraint: A fundamental observation from the closed-loop expression of (6) is that the vector $\boldsymbol{\nu}$ gives some freedom in the control of the system. A potential secondary objective can be satisfied in the null-space of the primary task without affecting it [20]. This leads to a hierarchy of kinematic tasks of decreasing priority.

Soft constraint: If p tasks have the same priority level, all respective equality constraints $\langle \boldsymbol{J}_1, \dot{\boldsymbol{x}}_1 \rangle \dots \langle \boldsymbol{J}_p, \dot{\boldsymbol{x}}_p \rangle$ have to be realized simultaneously. The solution is then obtained by stacking them into a single task $\langle \underline{\boldsymbol{J}}, \underline{\dot{\boldsymbol{x}}} \rangle$ where

$$\underline{\boldsymbol{J}} = \begin{bmatrix} \boldsymbol{J}_1 \\ \vdots \\ \boldsymbol{J}_p \end{bmatrix}, \quad \underline{\dot{\boldsymbol{x}}} = \begin{bmatrix} \dot{\boldsymbol{x}}_1 \\ \vdots \\ \dot{\boldsymbol{x}}_p \end{bmatrix}. \tag{10}$$

The same rule applies on inequality constraints $\langle \boldsymbol{A}_1, \boldsymbol{b}_1 \rangle \dots \langle \boldsymbol{A}_p, \boldsymbol{b}_p \rangle$ resulting in $\langle \underline{\boldsymbol{A}}, \underline{\boldsymbol{b}} \rangle$ where

$$\underline{\boldsymbol{A}} = \begin{bmatrix} \boldsymbol{A}_1 \\ \vdots \\ \boldsymbol{A}_p \end{bmatrix}, \quad \underline{\boldsymbol{b}} = \begin{bmatrix} \boldsymbol{b}_1 \\ \vdots \\ \boldsymbol{b}_p \end{bmatrix}. \tag{11}$$

The resulting QP is solved while satisfying all of the tasks simultaneously. A weighting matrix $\boldsymbol{\psi}$ is often used to give more importance to some constraints.

A. Stack of tasks for throwing

Figure 4 shows the structure of hard and soft constraints and their level of priority.

[4]Calling these functions as *task* or *constraint* is only a matter of context.

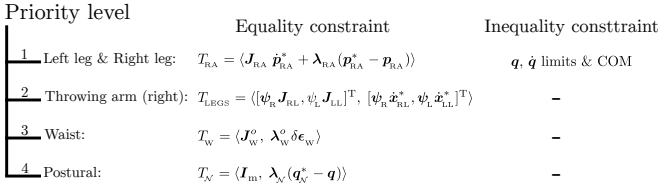

Fig. 4. A graphical scheme of different levels of task priorities and their corresponding equality and inequality constraints.

Legs task: In our setting, it is crucial that the tasks for the robot legs have the highest priority in the cascade of QP problems since slipping compromises the stability. The tasks for left and right legs share the same hard priority level while all the remaining tasks are solved in the solution space of this level. That is, the legs have a hard-constraint relationship with other tasks, however, between each other they form a soft-constraint relationship

$$T_{\text{LEGS}} = \langle [\boldsymbol{\psi}_{\text{R}}\boldsymbol{J}_{\text{RL}}, \psi_{\text{L}}\boldsymbol{J}_{\text{LL}}]^{\text{T}},\ [\boldsymbol{\psi}_{\text{R}}\dot{\boldsymbol{x}}^*_{\text{RL}}, \boldsymbol{\psi}_{\text{L}}\dot{\boldsymbol{x}}^*_{\text{LL}}]^{\text{T}} \rangle. \tag{12}$$

$\boldsymbol{\psi}_{\text{R}}$ and $\boldsymbol{\psi}_{\text{L}}$ in (12) are two diagonal matrices that determine a relative, soft priority between left and right legs. The desired velocity vectors $\dot{\boldsymbol{x}}^*_{\text{LL}}$ and $\dot{\boldsymbol{x}}^*_{\text{RL}}$ are treated similar to Cartesian component of (13).

At this level, all inequality constraints are also applied, i.e. they are always satisfied even during the subsequent priority levels. For the COMAN whole-body motion joint limits, velocity limits and constraints on the Center of Mass (CoM) have to be fulfilled at the top level (or equivalently every level). The constraints on CoM limit its height above ground and used bounds to ensure that the projection of CoM on the ground is within the convex hull of the support polygon.

Right arm task: The right arm task, responsible for throw, has the next priority level and is defined as

$$T_{\text{RA}} = \langle \boldsymbol{J}_{\text{RA}},\ \dot{\boldsymbol{p}}^*_{\text{RA}} + \boldsymbol{\lambda}_{\text{RA}}(\boldsymbol{p}^*_{\text{RA}} - \boldsymbol{p}_{\text{RA}}) \rangle, \tag{13}$$

where $\boldsymbol{J}_{\text{RA}}$ is the task Jacobian at the end-effector, $\dot{\boldsymbol{p}}^*_{\text{RA}}$ is the desired linear and angular velocity of the task, and $(\boldsymbol{p}^*_{\text{RA}} - \boldsymbol{p}_{\text{RA}})$ forms an error vector between the desired and the real pose of the end-effector and prevents error accumulation. $\boldsymbol{\lambda}_{\text{RA}}$ is a diagonal matrix determining positive gain parameters.

Waist task: To maintain upright orientation of the torso (waist), rotation along x, y axis is prevented while rotation along z is remains unconstrained. This is based on the assumption that keeping torso orientation fixed in roll and pitch results in a more human-like motion. It is realized through the following orientation-only constraint

$$T_{\text{W}} = \langle \boldsymbol{J}^o_{\text{W}},\ \boldsymbol{\lambda}^o_{\text{W}}\delta\boldsymbol{\epsilon}_{\text{W}} \rangle, \tag{14}$$

where $\boldsymbol{J}^o_{\text{W}}$ represents the rotational part of waist Jacobian and $\delta\boldsymbol{\epsilon}_{\text{W}}$ is the orientation error of waist expressed in unit quaternions. $\boldsymbol{\lambda}^o_{\text{W}}$ represents a diagonal matrix of the gains. Assume quaternion representation of the waist orientation $\boldsymbol{o}_{\text{W}} = [\eta_{\text{W}}, \boldsymbol{\epsilon}_{\text{W}}]$, where η_{W} and $\boldsymbol{\epsilon}_{\text{W}}$ are respectively the scalar and the vector part. According to [25], the orientation error in (14) w.r.t the desired orientation (represented by superscript

'*') can be obtained form

$$\delta\boldsymbol{\epsilon}_{\mathrm{w}} = \eta^*_{\mathrm{w}}\boldsymbol{\epsilon}_{\mathrm{w}} - \eta_{\mathrm{w}}\boldsymbol{\epsilon}^*_{\mathrm{w}} + skew(\boldsymbol{\epsilon}^*_{\mathrm{w}})\boldsymbol{\epsilon}_{\mathrm{w}}, \tag{15}$$

where '$skew$' represents a skew symmetric matrix.

Postural task: To ensure a natural throw, it is crucial to consider a postural task in the null-space of Jacobian of the throwing arm (e.g., shoulder-down). This is done by considering a hard-constraint relationship between the throwing arm task and the postural task. This task is defined in the joint-space as

$$T_{\mathcal{N}} = \langle \boldsymbol{I}_{\mathrm{m}},\ \boldsymbol{\lambda}_{\mathcal{N}}(\boldsymbol{q}^*_{\mathcal{N}} - \boldsymbol{q})\rangle, \tag{16}$$

where $\boldsymbol{I}_m \in \mathbb{R}^{m\times m}$ is the identity matrix, m is the number of DOF of the throwing arm and $\boldsymbol{q}^*_{\mathcal{N}}$ is the desired joint behavior.

Other tasks such as capture point [26] and regularization of centroidal linear/angular momentum [27] were tested but did not improve robot movements. This is in part due to the fact that the target of propose architecture are patients with limited motor functions and throws are generated by rather small Cartesian velocities. However, if faster throws are needed, then capture point for recovery through stepping and centroidal momentum for improved balance proves to be useful. In the attached video submission some of these simulations are presented.

V. Virtual Reality System

As long as throwing is only simulated, the respective robot and ball movements need to be presented to the patient (or more general: the user) in a meaningful manner for the physiotherapy to be effective. Our solution is to utilize a virtual environment (VE), where the patient can both observe the robot (Fig.7) and interact with the ball. The patient experiences the virtual environment via the HTC Vive system, which consists of head mounted display (HMD) and two controllers for motion tracking and ball catching action. One controller is securely attached to the patients right arm and second is held in right hand, where pressing a trigger executes the catching action in the VE.

The VR environment was realized using Unity 2017.3 game engine and consists of the following main building blocks:

- *Virtual room* A virtual environment that is presented to the patient. Great effort was made to design the room in order to enhance the effect of immersion and to be stimulating but not distracting to the user.
- *Robot and ball animator* A direct output of the current scene state as produced by the simulator and described in the previous Section. The virtual robot is an exact match of the URDF and meshes that are also used in the simulator. The robot and ball are animated using the data received via the RSB communicator (see also Fig. 2). The data transfer latency was measured to be approximately 1 ms, which is way below the critical level of about 10ms to running the VR system. The RSB link for the ball is broken on the successful catch of the ball by the user, when the control of the ball is taken over by the user or Unity physics engine until the start of the next throw. Also note that the robot and the ball motion are proportionally scalable such that the user can experience larger and smaller robots accordingly.
- *User controller* An interpreter of patient's actions. The movement and actions of the patient are interpreted using SteamVR plugin, running with approximately 5 ms latency.

Altogether, the VR environment runs sufficiently fast to create a realistic, real-time animation. It is also sufficiently fast to provide feedback to the motion generation for creation of new targets in the loop [13].

VI. Performance validation

The proposed architecture and controller are put to the test. Two different aspects are validated: the functionality of the building blocks in the CoSiMA/Orocos environment and of the fully integrated system and the timing.

We used the following libraries and tools during in this validation run (note that the modularity of the system would allow for exchange of these tools):

- quadratic programming Online Active SEt Strategy (qpOASES) as the solver [28]
- Rigid Body Dynamics Library (RBDL) as a real-time-safe tool for kinematic and dynamic computations [29]. It is suitable for computations regarding the humanoid robot since it supports kinematic trees and floating bases.
- OpenSoT as an open-source real-time-safe interface library to facilitate the definition of equality and inequality constraints [30]. It interfaces with RBDL and qpOASES and provides an easy-to-use chain for creating constraints for robot description (e.g., URDF and SRDF).
- Gazebo as a dynamic simulation environment [31] using ODE as physics engine [32]

The variation of cycle times for two important and computationally demanding components of the system, *the robot simulator* and *qp IK*, are depicted in Figure 5. The cycle time of each component was tested over 2000 data samples. The repeatability of the generated joint motion in 10 consecutive throws was examined. Figure 6 shows the Cartesian trajectories of the thrown balls from the release point until the first ground contact. The grasp and release of the ball is simulated using a two-state (on/off) gripper realized by grasp plugin[5]. Detailed simulation of ball release is beyond the scope of this work.

The overall system integration was verified in a comprehensive setup shown in Figure 1. We captured 23 underhand ball-throwing movements performed by a single right-handed subject and generated the corresponding retargeted hand trajectories. To test the full pipeline and the VR setting with the user in the loop, one motion was repeated successfully more than 50 times. Then, a number of different throwing trajectories were generated offline and stored to create a repeatable and demonstrable system, It can load the trajectories, run them in the simulator and present them in VR to a subject wearing HMD. Preliminary tests show that users

[5]https://goo.gl/ETJ44s

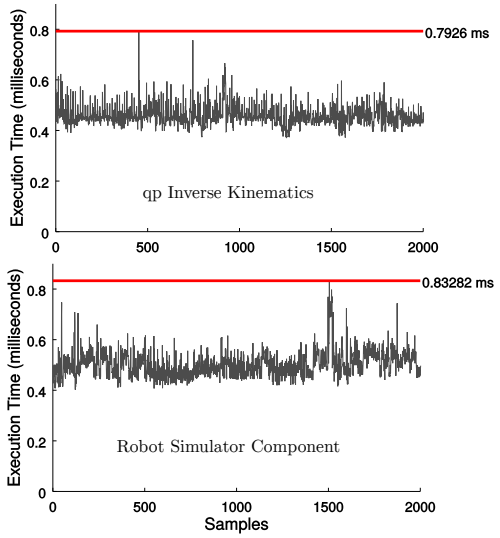

Fig. 5. Timing specification for qp-IK solver (top) and robot component (bottom). The red line shows the worst cycle time.

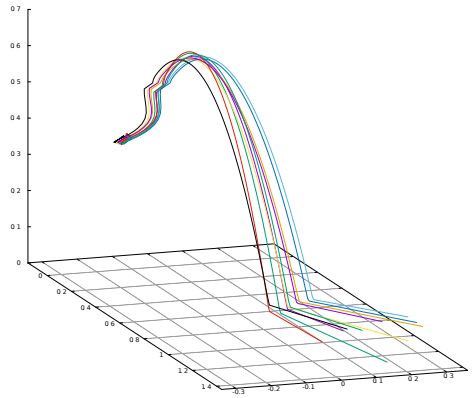

Fig. 6. 10 throws of the ball: Given a suitable release, the main sources of error are small variations in the end-effector pose and velocity of the throw arm at the release moment.

are able to interact with the virtual representation of the robot in simulation which is manifested through successful catches of the ball. A representative example of one interaction trial is depicted in Figure 7. The possibility robot scaling in VR is demonstrated in Figure 8, which we assume useful to accommodate variance for the user's body height.

VII. Conclusion

The paper presented an unprecedented and novel combination of real-time motion generation based on retargeting human motion, real-time robot control and VR for therapeutic juggling. The current system provides a rather intermediate, but already fully functional stage towards the long-term goal to deploy it with the real robot in actual therapeutic intervention. But it already meets the technological requirements for such advanced usage and the presented runtime experiments show that the system reaches the required real-time performance. The control already is suitable for switching to the real robot. However, there are still crucial questions to solve. The likely most difficult part will be to realize a reproducible release of the ball with a real robot hand, although in a real therapeutic game a certain variance will be tolerable. But what is acceptable and useful for different kinds of patients and impairments will certainly need careful investigation.

The presented system nevertheless provides the technological basis to conduct a number of interesting experiments. These can advance our knowledge about human-robot interaction in VR in general and provide information how to set up possible therapeutic intervention in particular. For example, possible studies are: (i) Do humans actually perceive the naturalness of the generated robot motion in comparison to purely kinematic animation that does not respect the feedback from the robot dynamics, or animation of an avatar? Our initial subjective experience with the system suggests so. (ii) Do humans perceive the difference between retargeted human throwing motion, as it is proposed in this paper, and analytically generated throwing motion? (iii) How do patients perceive the robot and will they engage in the throwing game? (iv) Is it important to scale the robot and the motion with respect to the body height of the user? (v) How to organize and exploit feedback from the VR for actual gamification of the interaction and how will it influence the user's engagement? These are exiting perspectives already, but ultimately the system is meant to be used in therapeutic intervention and shall be evaluated according to the usual clinical standards. We are convinced that we have laid the technological foundation for this feat.

References

[1] S. T. Rodrigues, P. F. Polastri, G. C. Gotardi, S. A. Aguiar, M. R. Mesaros, M. B. Pestana, and F. A. Barbieri, "Postural control during cascade ball juggling," *Perceptual and Motor Skills*, vol. 123, no. 1, pp. 279–294, aug 2016.

[2] S. Balasubramanian, J. Klein, and E. Burdet, "Robot-assisted rehabilitation of hand function," *Current opinion in neurology*, vol. 23, no. 6, pp. 661–670, 2010.

[3] N. Norouzi-Gheidari, P. S. Archambault, and J. Fung, "Effects of robot-assisted therapy on stroke rehabilitation in upper limbs: systematic review and meta-analysis of the literature," *Journal of rehabilitation research and development*, vol. 49, no. 4, p. 479, 2012.

[4] S. Masiero and M. Armani, "Upper-limb robot-assisted therapy in rehabilitation of acute stroke patients: focused review and results of new randomized controlled trial," *Journal of rehabilitation research and development*, vol. 48, no. 4, p. 355, 2011.

[5] N. A. Malik, H. Yussof, F. A. Hanapiah, R. A. A. Rahman, and H. H. Basri, "Human-robot interaction for children with cerebral palsy: Reflection and suggestion for interactive scenario design," *Procedia Computer Science*, vol. 76, pp. 388–393, 2015.

[6] A. Guneysu, B. Arnrich, and C. Ersoy, "Children's rehabilitation with humanoid robots and wearable inertial measurement units," in *Proc. of the 9th Int. Conf. on Pervasive Computing Technologies for Healthcare*, 2015, pp. 249–252.

[7] B. Draganski, C. Gaser, V. Busch, G. Schuierer, U. Bogdahn, and A. May, "Changes in grey matter induced by training," *Nature*, vol. 427, no. 6972, pp. 311–312, 2004.

[8] C. Sampaio-Baptista, N. Filippini, C. J. Stagg, J. Near, J. Scholz, and H. Johansen-Berg, "Changes in functional connectivity and GABA levels with long-term motor learning," *NeuroImage*, vol. 106, pp. 15–20, feb 2015.

[9] M. J. Majsak, T. Kaminski, A. M. Gentile, and A. M. Gordon, "Effects of a moving target versus a temporal constraint on reach and grasp in patients with parkinson's disease," *Experimental Neurology*, vol. 210, no. 2, pp. 479–488, apr 2008.

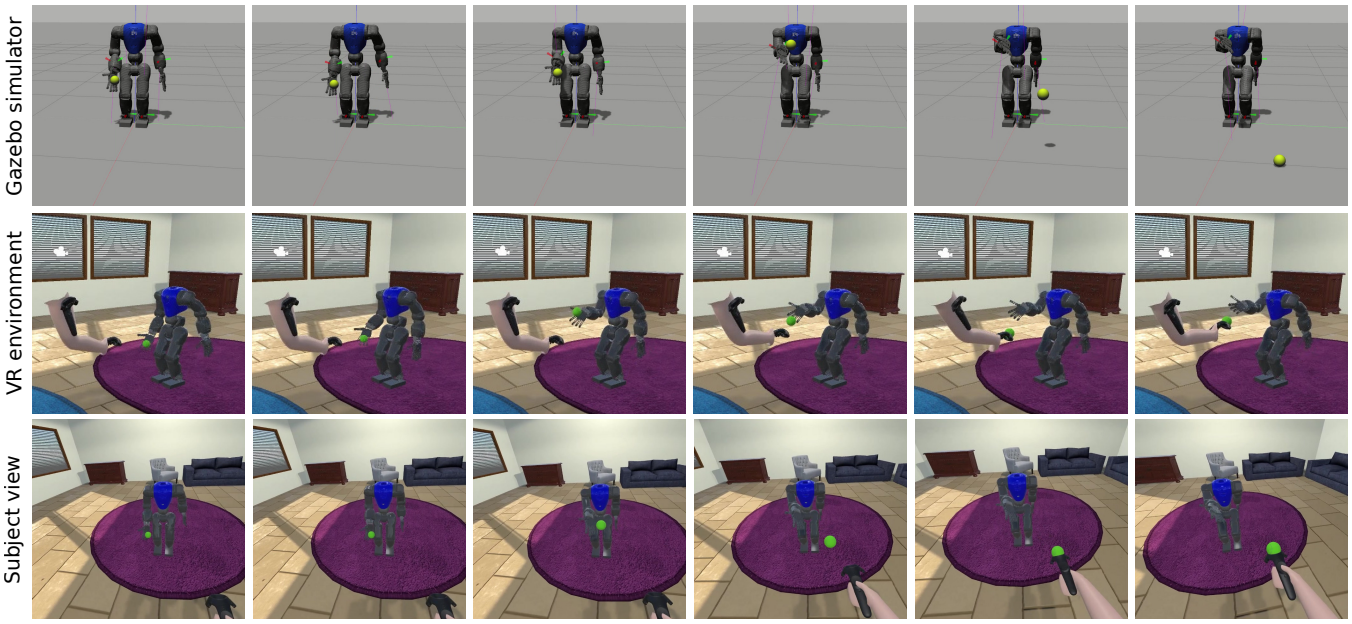

Fig. 7. Example of a throwing trial: selected frames from Gazebo simulator (top row), virtual environment (middle row) and subject view (bottom row). The leftmost column shows the start of the throwing. The mismatch of the ball positions for simulator and VR in the two rightmost columns are due to subject's interaction with the ball that is realized only in the VR.

Fig. 8. The COMAN avatar in VR can be scaled by different factors.

[10] H.-I. Ma, W.-J. Hwang, C.-Y. Wang, J.-J. Fang, I.-F. Leong, and T.-Y. Wang, "Trunk–arm coordination in reaching for moving targets in people with parkinson's disease: Comparison between virtual and physical reality," *Human Movement Science*, vol. 31, no. 5, pp. 1340–1352, 2012.

[11] F. Hülsmann, S. Kopp, and M. Botsch, "Automatic error analysis of human motor performance for interactive coaching in virtual reality," *CoRR*, vol. abs/1709.09131, 2017.

[12] C. E. Hagmann and M. C. Potter, "Ultrafast scene detection and recognition with limited visual information," *Visual Cognition*, vol. 24, no. 1, pp. 2–14, jan 2016.

[13] J. Kodl, A. Mukovskiy, T. Dijkstra, D. Brötz, N. Ludolph, N. Taubert, and M. Giese, "Ball throwing games in virtual reality for motor rehabilitation," in *Assistive Technology, IX Iberoamerican Congress in*. Iberdiscap2017, 2017, pp. 489–496.

[14] J. Wienke and S. Wrede, "A middleware for collaborative research in experimental robotics," in *Proc. IEEE/SICE Int. Symp. System Integration (SII)*, Dec. 2011, pp. 1183–1190.

[15] H. Bruyninckx, "Open robot control software: the orocos project," vol. 3, pp. 2523 – 2528 vol.3, 02 2001.

[16] D. L. Wigand, A. Nordmann, N. Dehio, M. Mistry, and S. Wrede, "Domain-specific language modularization scheme applied to a multi-arm robotics use-case," *Journal of Software Engineering for Robotics*, vol. 8, no. 1, pp. 45–64, 2017.

[17] A. Mukovskiy, W. M. Land, T. Schack, and M. A. Giese, "Modeling of predictive human movement coordination patterns for applications in computer graphics," 2015.

[18] E. Chiovetto, A. d'Avella, and M. Giese, "A unifying framework for the identification of motor primitives," *arXiv preprint arXiv:1603.06879*, 2016.

[19] C. G. Atkeson, A. W. Moore, and S. Schaal, "Locally weighted learning for control," in *Lazy learning*. Springer, 1997, pp. 75–113.

[20] B. Siciliano and J.-J. Slotine, "A general framework for managing multiple tasks in highly redundant robotic systems," in *Fifth International Conference on Advanced Robotics*. IEEE, 1991, pp. 1211–1216.

[21] Y. Zhang, S. S. Ge, and T. H. Lee, "A unified quadratic-programming-based dynamical system approach to joint torque optimization of physically constrained redundant manipulators," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 34, no. 5, pp. 2126–2132, 2004.

[22] A. Escande, N. Mansard, and P.-B. Wieber, "Hierarchical quadratic programming: Fast online humanoid-robot motion generation," *The International Journal of Robotics Research*, vol. 33, no. 7, pp. 1006–1028, 2014.

[23] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[24] O. Kanoun, F. Lamiraux, and P.-B. Wieber, "Kinematic control of redundant manipulators: Generalizing the task-priority framework to inequality task," *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 785–792, 2011.

[25] J. S. Yuan, "Closed-loop manipulator control using quaternion feedback," *IEEE Journal on Robotics and Automation*, vol. 4, no. 4, pp. 434–440, Aug 1988.

[26] J. Pratt, J. Carff, S. Drakunov, and A. Goswami, "Capture point: A step toward humanoid push recovery," in *2006 6th IEEE-RAS International Conference on Humanoid Robots*, Dec 2006, pp. 200–207.

[27] D. E. Orin and A. Goswami, "Centroidal momentum matrix of a humanoid robot: Structure and properties," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept 2008, pp. 653–659.

[28] H. Ferreau, C. Kirches, A. Potschka, H. Bock, and M. Diehl, "qpOASES: A parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, vol. 6, no. 4, pp. 327–363, 2014.

[29] M. Felis, "Rbdl: an efficient rigid-body dynamics library using recursive algorithms," vol. 41, 06 2016.

[30] A. Rocchi, E. Mingo Hoffman, D. G. Caldwell, and N. Tsagarakis, "Opensot: A whole-body control library for the compliant humanoid robot coman," vol. 2015, pp. 6248–6253, 06 2015.

[31] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, vol. 3, 2004, pp. 2149–2154 vol.3.

[32] R. Smith *et al.*, "Open dynamics engine," 2005.