HAL
open science

Compliant Robot Motion Regulated via Proprioceptive Sensor Based Contact Observer

Anastasia Bolotnikova, Sébastien Courtois, Abderrahmane Kheddar

To cite this version:

Anastasia Bolotnikova, Sébastien Courtois, Abderrahmane Kheddar. Compliant Robot Motion Regulated via Proprioceptive Sensor Based Contact Observer. Humanoids, Nov 2018, Beijing, China. pp.1-9, 10.1109/HUMANOIDS.2018.8624946 . lirmm-01895114v1

HAL Id: lirmm-01895114

https://hal-lirmm.ccsd.cnrs.fr/lirmm-01895114v1

Submitted on 13 Oct 2018 (v1), last revised 9 Dec 2019 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Compliant Robot Motion Regulated via Proprioceptive Sensor Based Contact Observer

Anastasia Bolotnikova[1,2], Sébastien Courtois[1], Abderrahmane Kheddar[2]

***Abstract*— In this paper, we present developments of a real-time compliant motion control for a personal humanoid robot. Our approach allows to interpret and react to human guidance through touch using only joint encoders measurements to monitor contact direction and intensity on both static and moving links. This novel method is developed with consideration of minimal sensor requirement of the hardware platform to meet high affordability criteria. We demonstrate performances in experiments with a humanoid robot Pepper.**

I. Introduction

In various fields, such as manufacturing or health care, an efficient automation may require work tasks to be performed by a robot in close-contact with a human. To achieve safe and efficient collaboration, it is important that agents understand each other's intentions and react to them appropriately. In this work, we focus on physical human robot interaction (pHRI) and, foremost, on the interpretation and correct online reaction of a humanoid robot to human touch.

The human-robot collaboration must ideally be safe, efficient and optimal in terms of cost. Thus, a minimal sensor set-up is preferable in order to reduce robot manufacturing and maintenance costs. As a result, we focus on more cost-efficient, but in practice more challenging way of contact sensing by using minimum of proprioceptive sensors. That is to say, we use only robot's joint encoders.

Recent developments in the area of proprioceptive sensor based contact monitoring have shown promising results for serial manipulators [1], humanoids [2] and also a special case of affordable personal robots, such as Pepper, used in this work, with high motor-joint backlash and friction [3]. In our previous work [4], we proposed a contact observer method that uses data-driven machine learning techniques to detect a colliding link and monitor contact direction and intensity. This method was developed to fit well into minimal sensor set-up, as only joint encoder measurements are exploited for contact detection for both moving and static links.

In this paper, we describe the modifications made to our initial method [4] in order to increase its accuracy and generalization properties in the case of multi-joint motions; we demonstrate its experimental performances in real-time feedback compliant motions regulation on the arm of a humanoid robot Pepper.

II. Background

In order to achieve active compliant motion of a robot, it is necessary to sense or estimate the external forces applied (anywhere) on the robot's links. A sophisticated yet expensive solution would require to cover the entire robot structure with a *haptic skin*. Such an approach, however, is not appropriate due to the cost of adaptation of additional sensors (even with cheap components) and their maintenance; the introduction of additional weight to the robot structure; additional power consumption... see eg. [5]. Thus, researches have directed their efforts towards achieving contact sensing by using must-be proprioceptive sensors only (i.e. those usually available on a robot platform), such as joint encoders, motors' electric current or torque sensors, etc.

A thorough overview of such contact sensing methods is presented in [1]. Methods, which were initially developed for serial manipulators, have been further developed to be applied on humanoid robots, e.g. [2], [6], [7]. In [6] a proprioceptive sensor based external force reconstruction was used as a feedback in the control loop to regulate robot interaction forces with the environment –under assumption of static conditions for external force reconstruction, whereas we aim at developing a method that works equally well for static and dynamic motions. Compliant reaction to human touch using proprioceptive sensors has been demonstrated in [3] in the experiment with a Romeo humanoid robot arm. However, this method could not be adapted for the current version of the Pepper platform. This is due to the inability to measure motor-link backlash and to measure or accurately estimate motor torque, which in turn prohibits to estimate external torque by means of the momentum observer method [8]. The latter method was used in robot control to achieve safe reaction to external collisions in pHRI context, see examples in [8], [9], [10], [11].

In order to address a more challenging task of contact sensing without using cover haptics, joint torque or even motors' electric current sensors; we developed a position tracking error based contact observer in [4]. This method is based on monitoring the discrepancy between the measured position tracking error and the *expected* one for a given desired free motion trajectory. First, a machine learning model is trained on a sample dataset with appropriate set of desired trajectory related features to predict the *expected* position tracking error value for collision-free joint motion. Then, for every iteration (of the control loop) the trained model is used to predict the expected position tracking error, assuming no contact has occurred. The contact is detected whenever there is significant discrepancies between measured and predicted (and therefore *expected*) error values. In our previous work, this method has shown promising results in contact detection experiments, and in monitoring both the contact intensity

[1]SoftBank Robotics Europe, Paris, France
[2]University of Montpellier–CNRS LIRMM, Interactive Digital Humans, Montpellier, France

and direction. Now, we exploit our contact observer in real-time control for regulating compliant motion of Pepper. The motivation behind this goal is to enable efficient physical interaction between human and a robot. In this work, we demonstrate that enhanced version of our contact observer can be exploited for the online interpretation of the human touch and regulation of compliant motion.

The use of disturbance observers for the regulation of robot compliant motion has been explored is several works. In [12] whole-body compliant motion in human-humanoid interaction settings was proposed for maintaining balance of the torque-controlled bipedal humanoid platform in the presence of unknown external forces. A momentum based disturbance observer was applied to the floating-base model of a humanoid robot in [13] to detect external forces and integrate them into a whole-body control for kinesthetic teaching and simultaneous compliant balancing. These methods, however, require a measurement of the joint torques, which may not be available and may also be challenging to estimate in case of highly geared and flexible joints encountered on many humanoid platforms.

In [14] disturbance observer based compliant humanoid motion control scheme, that can compensate for high joint elasticity, was proposed including the modeling of flexible joints that was performed using the measurement of motor-joint backlash angle. On the current position-controlled Pepper platform, access to the motor side encoder measurements is not available through robot's centralized memory, prohibiting to handle motor-joint backlash and joint flexibility. There are also no motor torque sensors on the platform. Furthermore, estimation of the motor torque from electric current measurements is not feasible, due to the measurements being passed to the centralized memory in absolute and down-sampled form. Nevertheless, in our work we try to overcome those constraints by exploiting the machine learning data-driven approach for disturbance observer.

We describe modifications of the contact observer w.r.t our previous work, with details about machine learning prediction model training process (Sec. III), and present the compliance control scheme and controller implementation details (Sec. IV). We demonstrate compliant motion results achieved with Pepper in pHRI experiments (Sec. V).

III. Contact Observer for Multi-Joint Motions

First we review the idea behind our methodology in [4]; then we present the details of the updated framework we propose in order to enhance both robustness and performance of our initial method for the case of multi-joint motions.

A. Review of the Contact Observer

In our settings, a robot is controlled in a task-space closed-loop by an acceleration resolved quadratic programming controller (QP) [15]. As for now, state feedback is not used to compute desired trajectories –assuming that no contact has occurred, because when contact occurs, resultant force estimation is not yet integrated in the QP model. Moreover, at the test time, when external collisions are introduced, the QP control without state feedback computes the desired trajectories that in its nature match those trajectories that were present in the collision-free training set. At each control loop iteration desired joint accelerations $\ddot{q}_d$ are computed as an output of QP. We obtain values of desired velocity $\dot{q}_d$ and position q_d through numerical integration of $\ddot{q}_d$, and compute desired torques τ_d from inverse dynamics. Note, that τ_d can also be obtained as an output of QP controller, by keeping torques as decision variables (not necessary in this context).

We define an *expected* position tracking error ϵ_{exp} to be the kind of tracking error that occurs when joint moves freely and no external forces are applied to any of the robot links (i.e. external joint torques are zero, $\tau_{\text{exp}} = 0$). Our goal is to predict the value of ϵ_{exp} given the information about the desired trajectory of the robot joints.

We have demonstrated in [4] that –under some conditions, ϵ_{exp} can be computed for a joint, actuated by a DC motor with PD control, as a linear combination of $\ddot{q}_d, \dot{q}_d, \tau_d$ and $\dot{\epsilon}_{\text{exp}}$ (assuming no motor-link backlash). However, due to modeling errors and the lack of possibility to model some critical non-linear effects, such as a large motor-link backlash –which in practice cannot be ignored, we use a non-linear model, such as binary tree, to learn the relationship between the appropriate set of available features ($\ddot{q}_d, \dot{q}_d, \tau_d$) and the value of position tracking error, ϵ, from a sample free-motion data. As a result, we obtain: $\epsilon^i_{\text{exp}} = \texttt{binary_tree}(\ddot{q}^i_d, \dot{q}^i_d, \tau^i_d)$ for every joint i[1]. The contact observer signal is defined as $r = \epsilon - \epsilon_{\text{exp}}$, where $\epsilon = q_d - q$, with q being a vector of measured joint positions (joint encoder readings).

When a contact occurs at a link in the chain after joint i, the value of r^i exceeds a predefined threshold $\pm\delta$. The magnitude and the sign of r^i signal encode the intensity and the direction of the external force respectively.

Interested reader may refer to original work in [4] for a more detailed explanation of the contact observer method.

B. Robustification of the Contact Observer

The use of r signal for contact monitoring has been demonstrated in several experiments with Pepper left and right arm joints, with one joint moving at a time, i.e. 1 degree of freedom (DoF) case [4]. Occasionally false positive (FP) and false negative (FN) contact detections occurred during the experiments. Furthermore, when the system was tested on more complex motions, e.g. with all arm joints moving simultaneously with arbitrary speed to random set-points, it turned out to be not general enough to adapt to such motions. The reason for that is mainly the fact that sudden motions of one joint can have significant influence on the tracking error value of some other joints, especially in the presence of a significant motor-link backlash.

Such false detection cases have to be eliminated or minimized in order to use r as a feedback signal in the active compliance control. We identified the main reasons for undesired false detection cases. First, due to the insufficient prediction accuracy of the binary tree prediction model for

[1] x^i being the ith element of vector x.

some new data (not used for the model training), r value can exceed threshold $\pm\delta$ even though external collision is not present, meaning that prediction model fails to generalize to "unseen" data and predict value of ϵ_{exp} correctly in those cases. Secondly, in some circumstances (e.g. singularity or near joint limits configurations), external contact force does not cause r to exceed $\pm\delta$, thus the contact remains unnoticed. Similarly, in case of light contacts, r does not exceed $\pm\delta$, which can be addressed by lowering the value of δ (which was set to $2.5°$ in previous work). However, to do that the overall prediction accuracy has to be improved, for both seen and unseen data, which may not be achievable due to the accuracy-generalization trade-off. And, finally, as mentioned previously, the prediction of expected tracking error of a joint depends on the motion of other joints. Therefore, *relevant* variables related to other joints' desired trajectories must be identified and included in the prediction feature vector.

The aim is to ensure that the trained models generalize well to a wide range of complex motions, while encountering fewest possible false positive contact detections. On the other hand, it is equally important that prediction is computed within the control time constraint for all joints. Therefore, prohibiting the use of very complex models and potentially sacrificing the maximum achievable prediction accuracy. The trade-off between model complexity and feature vector size and model accuracy must be properly handled.

In several application areas of machine learning, so called, ensemble based techniques have often shown to perform better in practice compared to single classification or regression models [16]. In particular, *boosting* is a machine learning technique for ensemble model training that is suitable for our goal to battle false positive contact detection cases [17].

According to the boosting methodology, a cascade of several models is trained successively; every new model $k = 2, \ldots, N$ (N being the total number of models in the ensemble) is trained taking into consideration the performance of the previously trained models $1, \ldots, k-1$. At every new iteration k, previously known inaccurate predictions of ϵ_{exp} are given more weight before training of kth model takes place. As a result, every new model in the ensemble is trained to make a better prediction, especially in those cases where previous models performed poorly. This results in minimization of the maximum prediction error, and thus less false positives in the r signal. We chose a *cascade of boosted decision trees* as our prediction model; we use `XGBoost` library implementation of this technique [18].

To achieve the best possible prediction model accuracy and generalization, while minimizing the required computation time, we make several important choices about model hyperparameters. The following developments are presented for *LShoulderRoll* joint, also referred to as joint nr. 6 or *LSRoll* for short. Analysis for other joints is performed similarly.

1) Feature Vector Components: First of all, on a sample recording of 45 minutes of collision-free data (with time step of 12 ms, i.e 225000 samples in total) we analyze the importance of different variables for ϵ^6_{exp} prediction to select the most relevant variables for the feature vector. We train a boosted ensemble of decision trees with a feature vector that includes the desired motion related variables for all the main left arm joints. The results of relative variable importance computed from the trained model are summarized in Fig. 1.

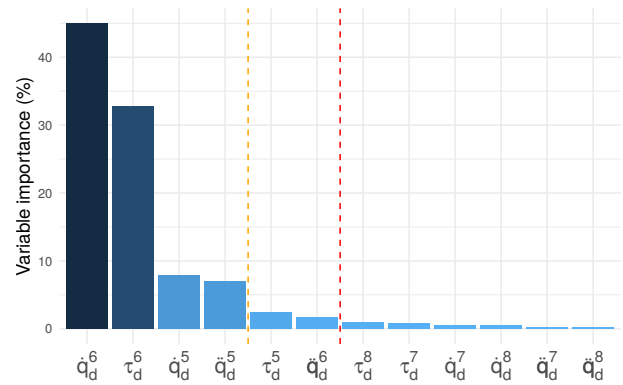

Fig. 1: Variable importance for ϵ^6_{exp} prediction.

As can be seen from the plot, the most important variables for ϵ^6_{exp} prediction are desired speed and torque of the target joint, which confirms our previous findings [4]. However, desired speed, acceleration and torque of the nearby joint, *LShoulderPitch* (joint nr. 5), also have significant weight for prediction of ϵ^6_{exp}, which even overweight importance of $\ddot{q}^6_d$. The variables after the orange dashed line on the plot are considered of medium-importance and variables after red dashed line are considered as non-important.

To make a more informed decision on the final size of the feature vector, we perform k-fold cross-validation with $k = 4$ (75% samples for training and 25% for validation) and analyze how the root-mean-square error (RMSE) and maximum absolute prediction error (MAX) on validation sets are affected as we incrementally include more of most important variables into prediction feature vector. The mean RMSE evaluated over 4 cross-validation folds only improves significantly if 3 to 4 most important variables are used (see Fig. 2). The mean MAX improves significantly until 6 most important variables are used and stops improving and even deteriorates after this point. Thus, we chose to include 6 most important variables in the final feature vector.

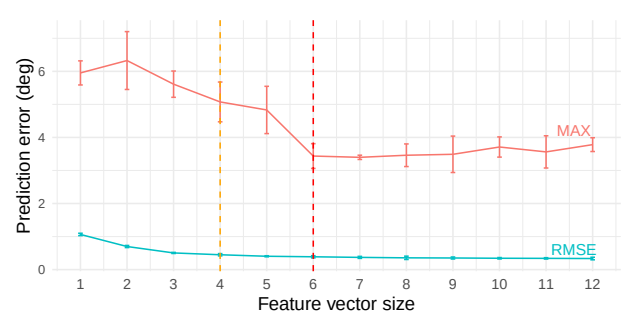

Fig. 2: Prediction error with various feature vector sizes.

2) Ensemble Size and Learning rate: Now that the best components of the feature vector have been set, we identify an optimal value for the ensemble size N, and a learning rate η of the model. Since we intend to use the trained model in the real-time application, we must select the model with the smallest possible ensemble size. Selecting a higher learning rate for the training allows to achieve good performance with fewer number of sub-models in the ensemble. However, higher learning rate may result in lower model accuracy, as in this case the training process is only allowed to make "big steps" towards an optimal solution. We analyze the mean cross-validation RMSE for various sizes of ensemble and for different values of the learning rate. The Fig. 3 illustrates the performance of models of various sizes trained with three different learning rates.

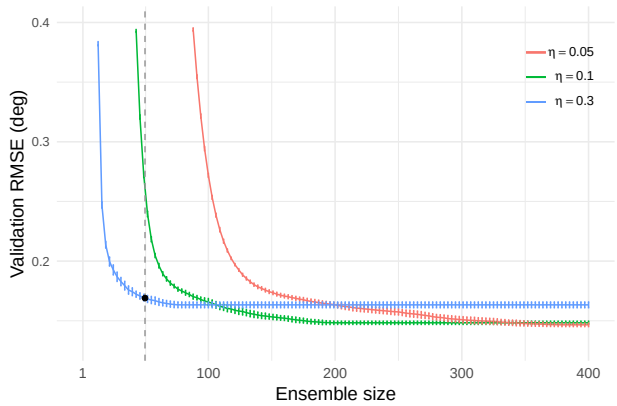

Fig. 3: Mean validation RMSE for various ensemble sizes and learning rates.

We see that lower η values do not result in significant improvement of performance in terms of the mean RMSE over k cross-validation folds as the size of the ensemble increases. The difference in performance between the best and worst performing models is $< 0.02°$. Therefore, we choose to use $\eta = 0.3$ with model size $N = 50$ as our optimal choice, since it provides a good performance while keeping ensemble size N relatively low.

3) Individual Decision Tree Maximum Depth: Now that other model hyperparameters are selected, we identify an optimal size of the individual decision trees. We train several models with $N = 50$ and $\eta = 0.3$ and analyze mean RMSE and MAX computed for training and validation folds. We also compute the rate of false positive detections as a total amount of absolute validation prediction errors exceeding the threshold $\pm\delta$ divided by the total number of the validation samples. The results are shown in Fig. 4.

The result is such that with maximum tree depth of 7 the false positive rate decreases to $< 0.1\%$. The values of RMSE for both the validation and the training data are acceptably low, always below soft and hard thresholds indicated as orange and red dashed lines on the plot respectively. The maximum absolute prediction error decreases for both seen and unseen data until maximum tree depth reaches 8, after

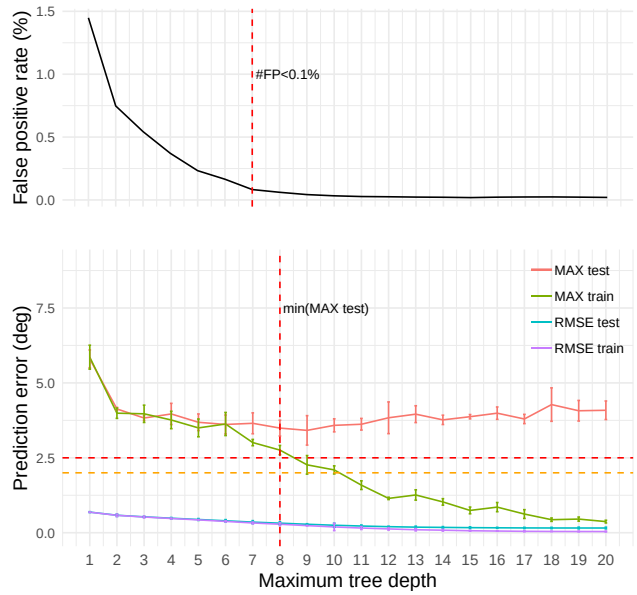

Fig. 4: Optimal max individual tree depth selection from false positive rate (top) analysis and training/test error (bottom).

this point, trained model becomes too complex and overfits the training data (training error decreases, while validation error remains the same or increases). Adapting regularization strategies did not help to improve the performance of the model on validation data. The final value for the maximum tree depth is, therefore, set to 8.

With this final decision on hyperparameters, we are able to achieve the performance of the contact observer that is acceptable for the use of r as a feedback in a real-time control for active compliance. The next section presents the compliant motion control implementation details.

IV. Compliant Motion Regulated via Monitoring of Joint Position Discrepancies

Here we describe the QP controller design that enables compliant motion using the contact observer signal as a feedback on collisions. The overview of the entire process is presented by pseudocode in Algorithm 1. For simplicity, consider a QP controller with one posture task in the objective function, and typical kinematics constraints, such as self-collision avoidance and joint limits. The posture task specifies a target posture, q_{target}[2]. The output, $\ddot{q}_d$, of the QP controller generates a whole-body motion that brings the robot joints closer to q_{target} while satisfying the kinematics constraints. The rate of convergence of the joints to q_{target} is regulated via the QP posture task stiffness gain.

Before the QP controller starts, we read the initial robot sensor values, q_{init}. For the first iteration of the controller execution, timestamp $t = 0$, we assume that no collision with the environment occurred, i.e. $r(0) = \mathbf{0}$. At this point, the posture task target position is equal to the values of the initial

[2]Note that q_{target} is the *final* target robot posture, whereas q_d is intermediate desired joints position command which iteratively and incrementally brings a robot from its initial state to q_{target}.

sensor reading ($q_{\text{target}} = q_{\text{init}}$). The QP controller computes $\ddot{q}_d$, that is integrated twice to have q_d that is sent to the low-level PD servo and brings progressively the robot joints closer to q_{target}. In the first iteration, since $q_{\text{target}} = q_{\text{init}}$ the robot does not move, i.e. $\ddot{q}_d \approx \mathbf{0}$.

Using known robot model and QP output, the desired torque τ_d is computed. Now the ϵ_{exp} can be computed using the trained prediction models described in the previous section. At the same time, the measured position tracking error is computed as $\epsilon = q_d - q_{\text{init}}$. Finally, the contact observer signal, $r = \epsilon - \epsilon_{\text{exp}}$, is computed and passed to the next QP iteration along with the new sensor readings.

Starting from the second iteration, if $|r| < \delta$ the posture task remains unchanged. Otherwise, if a discrepancy between ϵ and ϵ_{exp} has occurred (i.e. $|r| \geq \delta$), before the QP controller is inferred, the posture task has to be readjusted to comply in the direction of estimated disturbance. The compliant motion of every joint is expressed as follows:

$$q_{\text{target}}(t) = q_d(t-1) - K_c r + K_v \dot{q}_d(t-1) \qquad (1)$$

where K_c and K_v are compliance and velocity gains respectively, which are tuned for every joint. The posture task is updated in the QP objective function with new q_{target}, and the entire process repeats until the controller is stopped.

The compliance gain, K_c, regulates how much influence contact observer signal r has on updated target posture. Lower values of K_c allow the target posture to be updated only slightly, which means that QP will not compute high $\ddot{q}_d$ and robot will comply stiffly (little amount with lower speed). On the contrary, a high K_c causes the q_{target} of the posture task to change significantly resulting in high compliance and faster motion, as the $\ddot{q}_d$ is proportional to the QP task error. The velocity gain, K_v, regulates the proportional contribution of joint desired speed to facilitate compliance of moving joints (those that started to comply in the previous iterations).

Note that this basic strategy can be augmented with any other QP tasks (e.g. set point, visual servoing, force tasks, etc.), by adding them to the QP objective function via higher level planning algorithm either along with compliant posture task or within periods when $|r| < \delta$. In such case, a proper tuning of QP tasks' weight and stiffness values can influence greatly the overall performance of the complex QP controller.

Algorithm 1: The pseudocode of algorithm for the QP controller regulating joint compliance via r signal

```
q ← ROBOT.GET SENSORS()
r ← 0
CNTRL ← INITIALIZE CONTROLLER()
POSTURE TASK ← INITIALIZE POSTURE(q)
CNTRL ← INSERT TASK(POSTURE TASK)
CNTRL.RUN()
while CNTRL.RUNNING() do
    if |r| ≥ δ then
        q_target(t) ← COMPLY JOINTS(r, q_d(t−1), q̇_d(t−1))
        CNTRL.POSTURE TASK ← UPDATE TASK(q_target(t))
    end if
    q̈_d ← SOLVE QP(POSTURE TASK)
    q̇_d ← INTEGRATE(q̈_d)      q_d ← INTEGRATE(q̇_d)
    ROBOT.SETJOINTANGLES(q_d)
    τ_d ← INV. DYNAMICS(q̈_d, q̇_d, q_d, ROBOT MODEL)
    ε_exp ← PREDICT(q̈_d, q̇_d, q_d, τ_d)
    ε = q_d − q
    r = ε − ε_exp
    q ← ROBOT.GET SENSORS()
end while
```

V. Experimental Results

We use our contact observer methodology in two pHRI experimental setups. The threshold for contact detection δ is set to $2.5°$ for all joints in all experiments.

In the first scenario, a robot is making a motion that initiates the process of assisting a human to stand up from a sitting position. The robot moves its arm towards the human back and stops moving further if it detects a contact (i.e. K_c and K_v gains of Eq. 1 are set to zero). If no contact is detected the robot continues to move its arm until joint limits. The motion is repeated several times with varying values of the target *LShoulderPitch* (*LSPitch* for short) joint position, allowing to take contact lower or higher on the back of a human. The values of the desired joint positions and the contact observer of the *LSRoll* are shown in Fig. 5. When the contact observer signal exceeds the threshold, the commanded position for the *LSRoll* joint is altered such that robot stops instead of continuing to move forward.

In the second experimental scenario, the robot moves 4 joints of the left arm randomly. When the contact is detected, Eq. 1 with positive K_c and K_v gains, is used to guide the robot motions through human touch and comply the joints of the arm. The values for K_c and K_v gains are set manually for every joint; exact values used in the experiments are presented in Tab. I.

	LShoulderPitch	*LShoulderRoll*	*LElbowYaw*	*LElbowRoll*
K_c	17	5	5	17
K_v	0.08	0.02	0.02	0.02

TABLE I: Gain values for main left arm joints.

The plot segment of desired position of the shoulder roll joint and corresponding contact observer signals are shown in Fig. 6. Due to the random set-points being commanded to the robot during the free motion –to show that the method can be applied for movement of arbitrary position and speed, the compliance of the joints may not be obvious from the plot. However, looking closely one sees that at the moments of contacts (indicated with dashed blue lines) the desired joint position is being altered taking into account the direction in which the contact is applied.

No special handling of the contact/no-contact transition was done in aforementioned experiments, although the control method would benefit from meticulous handling of such transition, which is left for the future work. Extended presentation of the results of both experiments is included in the video accompanying this paper.

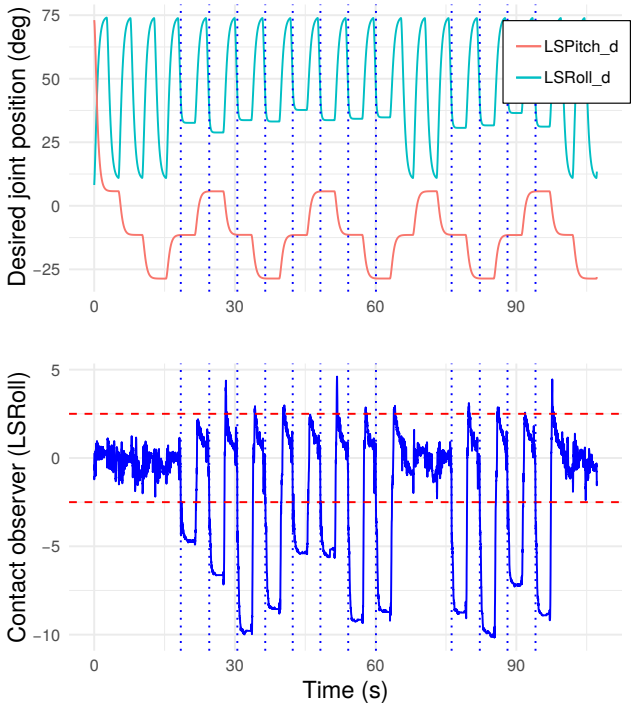

Fig. 5: Robot stops moving when in contact. The dotted blue lines show start of the contacts, red dashed lines show $\pm\delta$.

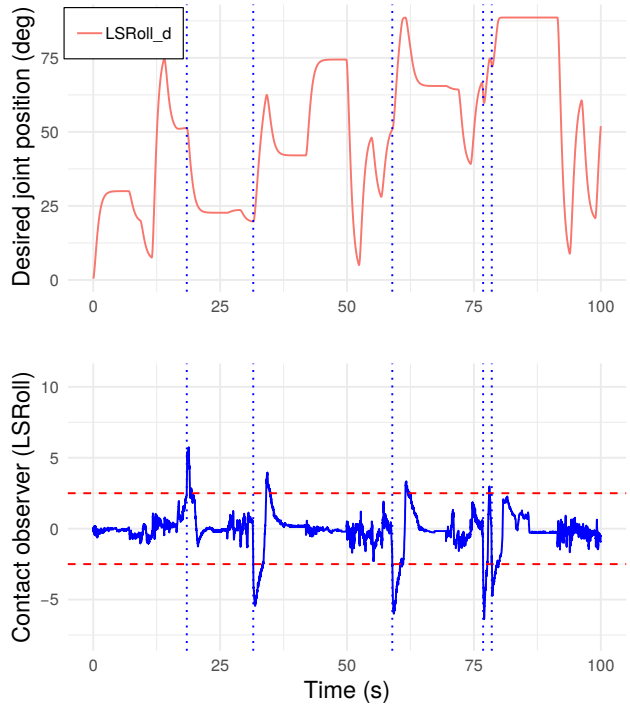

Fig. 6: Robot moves joints to random set-points. When contact is detected, joint position is set according to Eq. 1.

VI. Conclusion

We have presented the integration of a joint position tracking discrepancy based contact observer as a feedback signal into the compliant motion controller. The theoretical and implementation details of contact observer robustification and compliance controller design have been described. Finally, we demonstrated the performance of the proposed method in pHRI experiments with humanoid robot Pepper.

Future work will focus on enhancing the overall contact observer and compliance behaviors to the entire robot body including also an omnidirectional mobile base or a floating base of a humanoid and provide it as an additional functionality to the commercial package of Pepper.

References

[1] S. Haddadin, A. D. Luca, and A. Albu-Schäffer, "Robot collisions: A survey on detection, isolation, and identification," *IEEE Transactions on Robotics*, vol. 33, no. 6, pp. 1292–1312, 2017.

[2] F. Flacco, A. Paolillo, and A. Kheddar, "Residual-based contacts estimation for humanoid robots," in *IEEE-RAS International Conference on Humanoid Robots*, pp. 409–415, 2016.

[3] F. Flacco and A. Kheddar, "Contact detection and physical interaction for low cost personal robots," in *IEEE International Conference on Robot and Human Interactive Communication*, pp. 495–501, 2017.

[4] A. Bolotnikova, S. Courtois, and A. Kheddar, "Contact observer for humanoid robot pepper based on tracking joint position discrepancies," in *IEEE International Conference on Robot and Human Interactive Communication*, pp. 29–34, 2018.

[5] A. Kheddar and A. Billard, "A tactile matrix for whole-body humanoid haptic sensing and safe interaction," in *IEEE International Conference on Robotics and Biomimetics*, (Phuket, Thailand), pp. 1433–1438, 7-11 December 2011.

[6] T. Mattioli and M. Vendittelli, "Interaction force reconstruction for humanoid robots," *IEEE Robotics and Automation Letters*, vol. 2, no. 1, pp. 282–289, 2017.

[7] J. Vorndamme, M. Schappler, and S. Haddadin, "Collision detection, isolation and identification for humanoids," in *IEEE International Conference on Robotics and Automation*, pp. 4754–4761, 2017.

[8] A. D. Luca and R. Mattone, "Sensorless robot collision detection and hybrid force/motion control," in *IEEE International Conference on Robotics and Automation*, pp. 999–1004, 2005.

[9] S. Haddadin, A. Albu-Schaffer, A. D. Luca, and G. Hirzinger, "Collision detection and reaction: A contribution to safe physical human-robot interaction," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3356–3363, 2008.

[10] A. D. Luca and L. Ferrajoli, "Exploiting robot redundancy in collision detection and reaction," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3299–3305, 2008.

[11] S. Parusel, S. Haddadin, and A. Albu-Schäffer, "Modular state-based behavior control for safe human-robot interaction: A lightweight control architecture for a lightweight robot," in *IEEE International Conference on Robotics and Automation*, pp. 4298–4305, 2011.

[12] S.-H. Hyon, J. G. Hale, and G. Cheng, "Full-body compliant human–humanoid interaction: balancing in the presence of unknown external forces," *IEEE Transactions on Robotics*, vol. 23, no. 5, pp. 884–898, 2007.

[13] C. Ott, B. Henze, and D. Lee, "Kinesthetic teaching of humanoid motion based on whole-body compliance control with interaction-aware balancing," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4615–4621, 2013.

[14] M. Kim, J. H. Kim, S. Kim, J. Sim, , and J. Park, "Disturbance observer based linear feedback controller for compliant motion of humanoid robot," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 403–410, 2018.

[15] K. Bouyarmane, K. Chappellet, J. Vaillant, and A. Kheddar, "Multi-robot and task-space force control with quadratic programming," *IEEE Transactions on Robotics*, "to appear".

[16] T. G. Dietterich, "Ensemble methods in machine learning," in *International Workshop on Multiple Classifier Systems*, pp. 1–15, 2000.

[17] H. Drucker and C. Cortes, "Boosting decision trees," in *Advances in neural information processing systems*, pp. 479–485, 1996.

[18] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *International conference on knowledge discovery and data mining*, pp. 785–794, 2016.