Experience-based Torque Estimation for an Industrial Robot

Conference Paper · May 2016

DOI: 10.1109/ICRA.2016.7487127

5 authors, including:

Erik Berger
Technische Universität Bergakademie Freiberg
18 PUBLICATIONS **52** CITATIONS
SEE PROFILE

David Vogt
Technische Universität Bergakademie Freiberg
20 PUBLICATIONS **86** CITATIONS
SEE PROFILE

Heni Ben Amor
Arizona State University
56 PUBLICATIONS **434** CITATIONS
SEE PROFILE

Experience-based Torque Estimation for an Industrial Robot

Erik Berger[1], Steve Grehl[1], David Vogt[1], Bernhard Jung[1], Heni Ben Amor[2]

***Abstract*— Robotic manipulation tasks often require the control of forces and torques exerted on external objects. This paper presents a machine learning approach for estimating forces when no force sensors are present on the robot platform. In the training phase, the robot executes the desired manipulation tasks under controlled conditions with systematically varied parameter sets. All internal sensor data, in the presented case from more than 100 sensors, as well as the force exerted by the robot are recorded. Using Transfer Entropy, a statistical model is learned that identifies the subset of sensors relevant for torque estimation in the given task. At runtime, the model is used to accurately estimate the torques exerted during manipulations of the demonstrated kind. The feasibility of the approach is shown in a setting where a robotic manipulator operates a torque wrench to fasten a screw nut. Torque estimates with an accuracy of well below $\pm 1\,\mathrm{N\,m}$ are achieved. A strength of the presented model is that no prior knowledge of the robot's kinematics, mass distribution or sensor instrumentation is required.**

I. Introduction

Physical interaction between a robot and its environment requires accurate approaches for measuring and assessing forces applied by the robot to external objects and vice versa. For example, in human-robot interaction scenarios, exchanged forces may indicate unwanted collisions or result from intended human guidance. Accurate measurement of external forces is also important in manipulation tasks involving tool-usage. Humans often rely on sensory stimuli in order to estimate the state of a manipulation process, e.g., how hard a screw has been tightened.

This paper addresses the problem of measuring forces during a manipulation task. Force-torque (FT) sensors are often the first choice for measuring the forces exerted in interactions with external objects. However, these sensors are limited to a specific location. The spatial distribution and cost of these sensors therefore need to be carefully balanced. In dynamic tasks, such as manipulation tasks, it may also become challenging to distinguish between different kinds of forces such as, for instance, forces caused by the task, sensor noise, or external perturbations of the current behavior. Furthermore, modern robots, such as the UR5 robotic arm, come with built-in, purely software-based capabilities for force sensing utilizing a mass-acceleration model. However, such methods require detailed knowledge about the kinematics and mass distribution of the robot. A drawback of such a method is that it quickly deteriorates in estimation accuracy when the model is imprecise, e.g. due to additionally attached equipment. In particluar, when the robot is extended by a gripper or a tool, the mass distribution needs to be re-calibrated accurately. In case of the UR5 robot, the manufacturer specifies a force accuracy of $\pm 25\,\mathrm{N}$ for the robot's *tool center point* (TCP). Also, the manufacturer warns that such a kind of force estimation provides no protection against momentum.

Fig. 1. A 3-finger gripper mounted on an UR5 robotic arm is utilizing a usual torque wrench.

In contrast to the approaches mentioned above, and motivated by the ability of a trained mechanic to estimate a screw nut's tightening torque from prior experience, this paper proposes an experience-based approach that does not require prior information about the robot platform. Task models are learned during a training phase based on the available sensor data, without relying on any further knowledge, such as mass distribution or robot kinematics. During runtime, expected and measured sensor values are compared and detected discrepancies are turned into force estimates. A specific manipulation scenario is investigated, in which a UR5 learns to adjust a screw nut using a torque wrench. Figure 1 shows the principle setup which is elaborated below.

II. Related Work

Robots that engage in physical interactions with humans and objects need to regulate the forces exchanged with their

[1]Institute of Computer Science, Technical University Bergakademie Freiberg, Bernhard-von-Cotta-Str. 2, 09599 Freiberg, Germany

[2]School of Computing, Informatics and, Decision Systems Engineering, Arizona State University, 699 S Mill Ave, Tempe, AZ 85281, USA

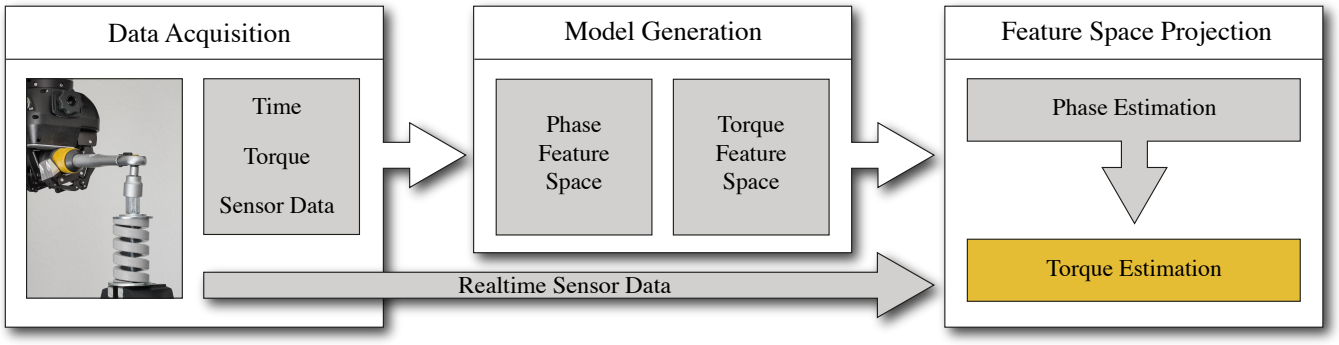

Fig. 2. An overview of the presented machine learning approach. During an offline training phase, sensor data together with information about the actual time and torque is recorded (section III-A) to generate two low-dimensional *Feature Spaces* (section III-B). In order to estimate the actual torque, realtime sensor data is projected into this spaces and compared with the training data (section III-C). The most similar matching is used as estimation of the actual torque.

environment. This requires methods for estimating the occurring forces as well as methods for adapting the robot behavior accordingly. Recent developments in compliant control have lead to the emergence of robots with joint torque sensing and feedback control [1]. For measuring external forces and perturbations, however, typically additional force sensors, e.g. force-torque sensors are used. Such sensors are often expensive, add weight to the robot, and are limited in their spatial resolution. Hence, various authors have suggested using algorithmic approaches for inferring applied forces. In [2], a depth camera is used in order to estimate applied forces. Using a depth camera allows for generic contact locations on the robot. In [3] machine learning methods are used to extract an inverse dynamics model for a cable-driven robot manipulator. Measuring the difference between predicted controls from the inverse dynamics model and the executed controls provides an estimate for external forces applied on the robot. A major challenge for such approaches however, is training an inverse dynamics model that is general enough to be applied to different situations. Using machine learning for force-based robot control has also been suggested a number of other publications. In [4] a robot learns to adapt its motion by anticipating human intentions from force measurements only. In [5], a method for learning force-based manipulation skills from demonstrations was presented. The approach generated variable-impedance control strategies thereby producing the necessary compliance for handling deformable objects.

The work presented in the remainder of our paper focuses on different aspect: how can a robot learn to estimate forces from experience? In contrast to earlier discussed papers, we learn behavior-specific models for force estimation, that are narrower in scope, but accurate in results.

III. Approach

An overview of the presented approach is shown in Figure 2, while the setup is further explained in Figure 3. The physical interaction considered is the tightening of a screw nut by using a wrench. Hence, the torque results in a preload force which is countered by a pressure spring. To secure a given tightening torque, a 3-fingered gripper mounted on a robotic arm is used to adjust the torque utilizing a custom torque wrench. The goal is to generate a behavior-specific model, which is able to estimate the torque from previous experience.

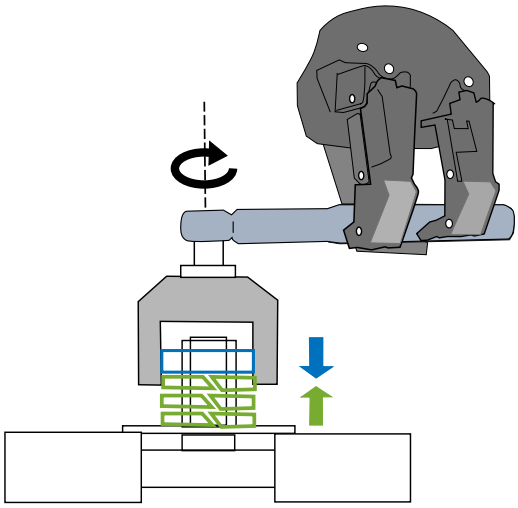

Fig. 3. A 3-finger gripper mounted on a UR5 robotic arm turns a usual torque wrench about $45°$. The applied torque (black) results in a preload force (blue) which is countered by the force conducted exerted by a pressure spring (green).

The first step of the presented approach is to record example data representing the evolution of sensor values for different torque values. In the training phase, the values of all sensors available on the robot together with the actually exerted torque and time stamps are recorded during a $45°$ tightening movement. This is performed multiple times for different preconfigured torques. The recorded data is used to generate a model consisting of two components, the *Phase-Feature Space* (P-FS), and the *Torque-Feature Space* (T-FS). These components are low-dimensional embeddings of the original high-dimensional data, that are extracted using *Transfer Entropy* [6] (TE) and Principal Component Analysis (PCA). During task execution, the P-FS is used to identify the phase of the behavior while the T-FS estimates the

applied torque based on the selected phase. To this end, the real-time sensor data is compared to sensor readings acquired during the training phase. Finally, the tightening torque is estimated, by determining the training data point with the highest similarity to the actual sensor readings. In the following, each step of the presented approach will be explained in more detail.

A. Data Acquisition

Similar to a skilled mechanic who is able to estimate a screw nuts torque from previous experience, the robot needs training data of the behavior. The realtime interface of the UR5 provides overall 105 different sensors, containing less useful information, e.g., the actual mainboard voltage and redundant data as control, target and actual joint values. However, the goal is to identify the sensors which are most significant for estimating the behavior phase and the exerted torque.

For this, the $m = 105$ sensor values $\mathbf{r} = (r_1, \ldots, r_m)$, are recorded during behavior execution. Additionally, the relative time w and the preconfigured torque v are recorded and stored in a $m + 2$ dimensional vector $\mathbf{s} = (\mathbf{r}, w, v)$. In contrast to the torque v, the relative time w increases during behavior execution and is reset after. Consequently, as the time for each training phase remains the same, w contains always the same data for each single recording. However, one behavior execution is recorded for two seconds with 125 Hz. The resulting training data $\mathbf{S} = (\mathbf{s_1}; \ldots; \mathbf{s_n})$ represents the training data for one possible torque v consisting of $n = 250$ equidistant samples.

To estimate the tightening torque, multiple behavior executions for varying configurations are recorded. First, the screw nut was tightened with $6.0\,\mathrm{N\,m}$ and equidistantly increased by $1.0\,\mathrm{N\,m}$ up to $15.0\,\mathrm{N\,m}$. In the following, the recorded training data $\mathbf{D} = (\mathbf{S_1}; \ldots; \mathbf{S_k})$ for $k = 10$ different torque configurations is used to estimate the torque. To achieve a higher level of accuracy without resulting in a time consuming training phase, virtual training sets are interpolated. For this, *Dynamic Mode Decomposition* (DMD) [7] is utilized . A detailed explanation of DMD and how it is used in the context of sensor data interpolation can be found in [8].

B. Model Generation

The recorded data $\mathbf{D}$ is investigated in order to extract relevant features. For this, two relevance values $\mathbf{o} = (o_1, \ldots, o_m)^T$ and $\mathbf{p} = (p_1, \ldots, p_m)^T$ are computed for each sensor. On the one hand $\mathbf{o}$ describes how strong the sensor is influenced by the preconfigured torque $\mathbf{v} = \mathbf{D}_{\cdot,m+2}$ and on the other hand $\mathbf{p}$ describes how strong a sensor is influenced by the relative time $\mathbf{w} = \mathbf{D}_{\cdot,m+1}$. The main idea is that sensor with a high TE are beneficial for estimating the corresponding value. In previous work [9], TE has been used to solve related tasks for perturbation detection during human-robot interaction. In the present work, TE is used as a measure of predictability and information flow between the relative time or torque and the evolution of sensor readings.

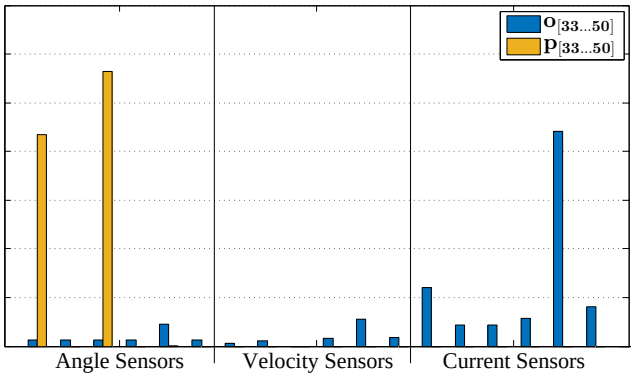

Fig. 4. The Transfer Entropy for the actual angle, velocity and current sensors is calculated for the torque applied to the screw nut (blue) and the relative time (orange).

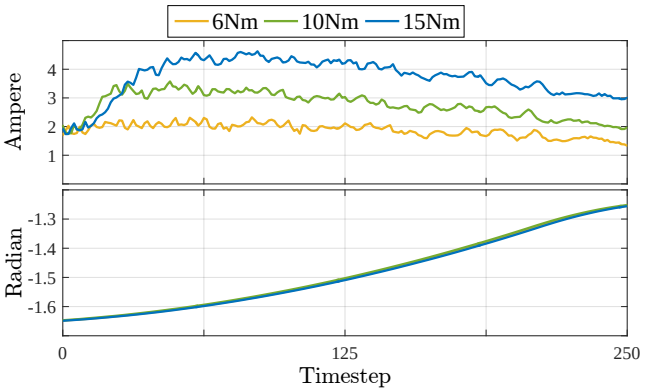

Fig. 5. The peak TE sensor streams. Top: Different torques result in varying sensor readings for the current sensor. Bottom: The angle sensor is not affected by the torque but increases over time.

The main idea is that sensors with a high TE w.r.t. the robot's behavior are deemed more influential and relevant. The formula for the TE between $\mathbf{j}$ and $\mathbf{i}$ is defined as

$$TE(\mathbf{j}, \mathbf{i}) = \sum_{t=1}^{|\mathbf{i}|-1} p(i_{t+1}, i_t, j_t) log_2 \frac{p(i_{t+1}|i_t, j_t)}{p(i_{t+1}|i_t)}, \quad (1)$$

where the function p describes the conditional probability. Utilizing Formula 1 the TE between the relative time and the sensors $\mathbf{p}(\mathbf{q})$ and for the torque and the sensor $\mathbf{o}(\mathbf{q})$ is calculated by

$$\begin{aligned} \mathbf{p}(q) &= TE(\mathbf{w}, \mathbf{D}_{\cdot,q}) \\ \mathbf{o}(q) &= TE(\mathbf{v}, \mathbf{D}_{\cdot,q}), \end{aligned} \quad (2)$$

where $q \in [1 \ldots m]$. Figure 4 shows the resulting TE for the actual angle, velocity and current sensors contained in $\mathbf{p_{33\ldots50}}$ and $\mathbf{o_{33\ldots50}}$. For $\mathbf{o}$ the highest TE values was found for the current sensors. The sensor with the highest TE $\mathbf{o_{49}}$ is visualized in the top of Figure 5. Obviously, the sensor stream differs for varying torques. The angle and velocity sensor share likewise less TE. Though, for $\mathbf{p}$ the TE is almost completely contained in two angle sensors. The sensor with the peak TE $\mathbf{o_{35}}$ is visualized in the bottom of Figure 5. As can be seen, the sensor stream is nearly the same for

different training data. Hence, independent from the applied torque, the sensor values are increasing over time. Therefore, the sensor is suitable for estimating the actual phase of the motor skill.

Next, the sensors with at least 90% of the overall TE are used to select a subset of sensors from $\mathbf{D}$ in order to build a feature space with

$$\omega \colon \mathbb{R}^m \rightarrow \mathbb{R}^{\hat{m}} \tag{3}$$

where $\hat{m} \leq m$. The threshold is determined empirically but can be changed in order to adapt the computational effort. However, sensors which are not influenced by the relative time are ignored during phase estimation while sensor not related to the torque are ignored during torque estimation.

Finally, the dimensionality of the feature spaces is reduced even further by applying well-known PCA. During the further procedure, the low dimensional embedding of $\omega(\mathbf{o})$ is used for the phase estimation and therefore is called P-FS. In contrast to that, the dimensional reduced $\omega(\mathbf{p})$ is used to predict the torque and is called T-FS.

C. Phase Estimation

Due to small time shifts, occurring between multiple executions of a motor skill, the relative time is not sufficient to estimate the current phase of a behavior. Because of that, in contrast to the training data, the realtime data does not contain information about the relative time. Instead, the current phase is estimated from the P-FS which among other sensors is strongly affected from joint positions.

For the estimation of the current phase of the behavior a time window with size t of the actual sensor stream is projected into the P-FS resulting in $\mathbf{X} = (\mathbf{x_1}, \ldots, \mathbf{x_t})$. In previous work [8], the *Subsequence Dynamic Time Warping technique* (SDTW) [10] was used for measuring the similarity between two sequences. In the case presented here, the low dimensional sequence $\mathbf{X}$ is compared with the low dimensional training data in the rows of the P-FS

$$\mathbf{Y} = \begin{bmatrix} \mathbf{y_{1,1}} & \cdots & \mathbf{y_{1,n}} \\ \vdots & \ddots & \vdots \\ \mathbf{y_{k,1}} & \cdots & \mathbf{y_{k,n}} \end{bmatrix} \tag{4}$$

where $n = 250$, $k = 10$, $n \geq t$ and $\mathbf{y}$ is the low dimensional projection of a single time step. Since the UR5 provides equidistant sensor readings, the optimization problem of the SDTW is reduced to finding the optimal path $\mathbf{p}^* = (b^*, \ldots, b^* + t - 1)$, where b^* is given by

$$b^* = \underset{b \in [0:(n-t)]}{\operatorname{argmin}} \sum_{i=1}^{t} c(\mathbf{x_i}, \mathbf{y_{b+i}}) \tag{5}$$

and c is a local distance measure, which in our case is the Euclidean distance $c = |\mathbf{x} - \mathbf{y}|$. In contrast to SDTW no accumulated cost matrix or warping need to be computed what results in significant less computational effort.

As illustrated in the bottom of Figure 5, an advantage of the presented phase estimation method is that the phases between the k recordings are just slightly shifted and therefore are almost the same. This effect is utilized in order to further decrease the computational effort. For this, the training data in the first row $k = 1$ of $\mathbf{Y}^*$ is compared with $\mathbf{X}$ by applying Formula 5. The resulting path $\mathbf{p_1^*}$ represents the current phase of the behavior and the last element in $\mathbf{p_1^*}$ the estimated actual state of the robot. Due to the similarity of the phases, the search space for all other rows in $\mathbf{Y}^*$ is reduced by applying a hill climbing approach. The starting point of the search space for the remaining training sets is at $b^* \in \mathbf{p_1^*}$. By applying Formula 5, the hill climbing method is searching the neighbors of b^* and stops when no reduced costs can be found. Since $\mathbf{p_1^*}$ contains the global minimum it is assumed that, due to their similarity, $\mathbf{p_2^*}, \ldots, \mathbf{p_k^*}$ only contain global minima. Finally, all phases are stored in $\mathbf{P}^* = (\mathbf{p_1^*}, \ldots, \mathbf{p_k^*})$.

D. Torque Estimation

In the following, the torque is estimated based on the previous phase estimation. By projecting the current time window into the T-FS resulting in $\hat{\mathbf{X}} = (\hat{\mathbf{x}}_\mathbf{1}, \ldots, \hat{\mathbf{x}}_\mathbf{t})$ the data can be compared to the low dimensional training data

$$\hat{\mathbf{Y}} = \begin{bmatrix} \hat{\mathbf{y}}_\mathbf{1,1} & \cdots & \hat{\mathbf{y}}_\mathbf{1,n} \\ \vdots & \ddots & \vdots \\ \hat{\mathbf{y}}_\mathbf{k,1} & \cdots & \hat{\mathbf{y}}_\mathbf{k,n} \end{bmatrix} \tag{6}$$

where $n = 250$, $k = 10$ and $n \geq t$ and $\hat{\mathbf{y}}$ is the low dimensional projection of a single timestep. Comparing the time window with the complete matrix $\hat{\mathbf{Y}}$ is time consuming. In order to reduce the computational effort the comparison is shrunk to the actual phases $\mathbf{P}^*$ calculated in the previous section. This reduces the number of computations from $(n - t) \cdot k$ to k. In order to estimate the torque, the training data l^* with the highest correspondence to the actual data $\hat{\mathbf{X}}$ is calculated by

$$l^* = \underset{l \in [1:k]}{\operatorname{argmin}} \sum_{i=1}^{t} c(\hat{x}_i, \hat{y}_{l,\mathbf{P}^*(k,i)}) \tag{7}$$

where $l^* \in [1 \ldots k]$. The preconfigured torque recorded during execution of the training data $\mathbf{v}_{l^*}$ is used as estimation of the actual one. Since, the robot performs the movement during the estimation phase, the torque continuous increases. This makes a realtime approach necessary. As mentioned above, the training data is interpolated which effectivley increases the number of datasets k. In order to further decrease the computational demands, the previous mentioned hill climbing method is utilized. Instead of computing k possible matches the algorithm has a maximal computation time in which it searches l^* from random start positions in l. This time is set to a value less than the data rate provided by the robot, which in the case of the UR5 is $125\,\mathrm{Hz}$. This ensures that the robot always know its actual state and is able to stop when a certain torque is reached. Obviously, the computational demands depend to the size of the time window t. However, in the following experiments $8\,\mathrm{ms}$ are sufficient for estimating the torque utilizing the introduced method.

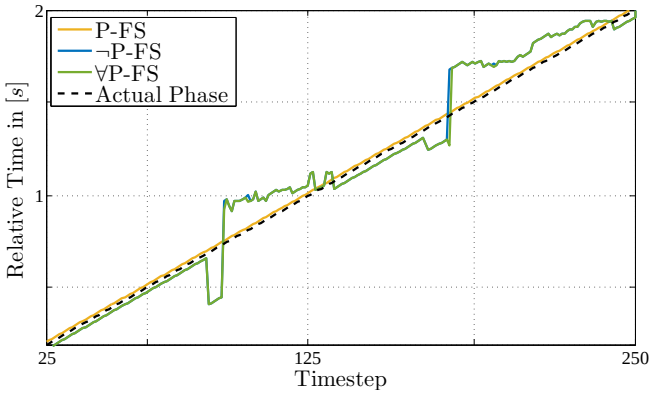

Fig. 6. Phase estimation during behavior execution utilizing different sensor selection schemes. P-FS is generated from the sensors with 90% of the overall Transfer Entropy (orange) while ¬P-FS is generated from the remaining sensors (blue) and ∀P-FS from all sensors (green). As can be seen, only P-FS is able to predict the actual phase (black) accurately.

IV. Experiments

To validate the proposed method different experiments are conducted. Therefore ten training data sets are recorded. For a more accurate torque estimation, the training data sets are further interpolated after feature extraction. The resulting data set contains 901 equidistant samples for torques in between $[6.0\,\mathrm{N\,m} \ldots 15.0\,\mathrm{N\,m}]$. The sensors selection process is evaluated and the reliability is proven by investigating the absolute error. Consequently, the accuracy of the presented results is $0.01\,\mathrm{N\,m}$.

A. Feature Selection Evaluation

The evaluation of the P-FS is done by comparing it to all sensors ∀P-FS and the negation of it ¬P-FS. The torque is randomly chosen and low-dimensional projections of the last 25 recordings is provided. Figure 6 shows the resulting phase estimation for the three different groups. As presumed, the original P-FS is able to estimate the current phase accurately while both other groups fail at certain time steps. Although, there is nearly no difference between the result of ¬P-FS and ∀P-FS. This is due to the fact that suitable sensors form a small subsets, identified by the introduced feature selection. This shows that the resulting feature space P-FS is suitable, to estimate the current phase of the behavior.

Next, the quality of the selected sensors for the torque estimation is evaluated. Therefore, the data set for a torque of $8.0\,\mathrm{N\,m}$ is selected from the training data. In order to prove the robustness of the T-FS it is disturbed with white noise of the following signal-to-noise ratio:

$$SNR = \frac{\sigma^2_{\hat{\mathbf{Y}}_{\mathbf{k,n}}}}{\sigma^2_{noise}}, \tag{8}$$

where σ is the standard deviation. As done previously ∀T-FS and ¬T-FS are generated in order to evaluate the selected sensors in the T-FS. Table I shows the resulting mean absolute errors for different levels of noise.

Similar to the phase estimation, the T-FS produces the best results while ¬T-FS and ∀T-FS result in incorrect estimations. That proofs that the selected sensors are a suitable choice for torque estimation.

TABLE I

The mean absolute errors in N m resulting from different feature spaces for different signal-to-noise ratios. The error signal was generated utilizing white noise.

SNR	**T-FS MAE**	**¬T-FS MAE**	**∀T-FS MAE**
20	0.0	1.05	0.53
10	0.0	1.34	0.89
5	0.0	1.42	1.03
2.5	0.01	1.74	1.18
1.0	0.07	2.51	1.75

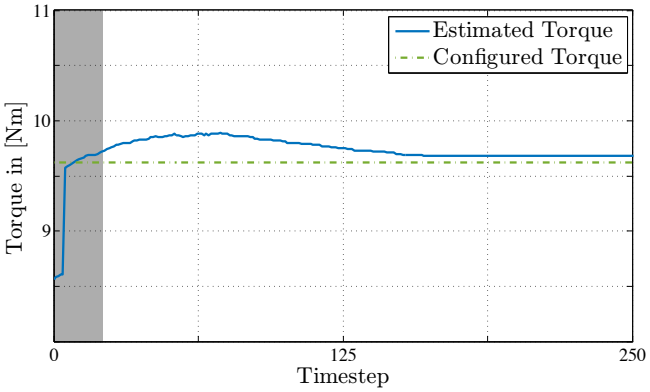

Fig. 7. The torque estimation during realtime behavior execution. The screw nut is preconfigured with $9.6\,\mathrm{N\,m}$ (green). The torque estimation (blue) fails at the beginning because of the small size of the captured realtime data. After $0.2\,\mathrm{s}$ (highlighted area) the estimation gets close to the correct value. The accuracy of the estimation directly depends to the size of the recorded realtime data and gets close to the real torque after $2\,\mathrm{s}$.

B. Realtime Torque Estimation

In the final experiment a torque of $9.6\,\mathrm{N\,m}$ was set. The goal is to identify this torque as fast as possible in order to avoid an additional tightening. As mentioned the realtime interface of the UR5 provides 105 different sensors 125 times per second. Each sample is first projected into the P-FS and appended to the previous ones. For the P-FS the size of this low-dimensional segment is set to a size of 25 values what, as illustrated in Figure 6, results in accurate phase estimation results. However, as illustrated in Figure 6 before the segment contains 25 elements, no phase estimation is solved. Instead, the relative time is used in order to allow at least a approximate torque estimation. Utilizing the actual phase, the T-FS is used to estimate the actual torque by comparing the realtime projection with the training data. As can be seen in Figure 7 the torque estimation fails at the beginning, due to the small size of the projected segment. In contrast to the estimation of the phase, the torque estimation accuracy is increased by using larger segments. In more detail, the size of the low-dimensional projection window is not restricted. As can be seen in Figure 7 after $0.2\,\mathrm{s}$ the estimated torque matches the preconfigured one. In order to

avoid an additional tightening of the screw nut, the robot may stop the behavior execution or perform a reverse motion to adjust the screw nut to the initial value. However, after $2\,\mathrm{s}$ the torque was estimated with sufficient accuracy.

This confirms the assumption, that the generated feature spaces can be applied in order to estimate the actual torque from previous experience. In the following section, the experimental results are discussed.

C. Discussion

As can be seen in Figure 6 the accuracy of the torque estimation is time dependent. This comes from the size and the actual values within the recorded time window. From time step 5 to about 60 the estimation overshoots the correct torque and converges slowly afterwards. This is due to the quality of the data recorded during the training phase. Taking into account Figure 5 top, one can see that the data model up to time step 40 contains overlapping sensor data. This leads to a less accurate model for this subsequence of the behavior. Past time step 40 the sensor data is clearly separated and therefore has a better estimation quality. Consequently, as the behavior continues, the phase shifts to a region with higher accuracy and improves the overall estimation results. The overall estimation converges slowly, since the time window still contains the less accurate data from the beginning of the estimation.

Taking a closer look at Figure 6, the offset after $0.5\,\mathrm{s}$ is $0.3\,\mathrm{N\,m}$ and after $2.0\,\mathrm{s}$ still about $0.1\,\mathrm{N\,m}$. The length of the torque wrench from the original tool center point is exactly $0.23\,\mathrm{m}$. This results in an accuracy of $1.3\,\mathrm{N} = \frac{0.3\,\mathrm{N\,m}}{0.23\,\mathrm{m}}$ after $0.5\,\mathrm{s}$ and $0.43\,\mathrm{N} = \frac{0.1\,\mathrm{N\,m}}{0.23\,\mathrm{m}}$ after $2.0\,\mathrm{s}$. Thus, the proposed model learning approach is more accurate than the mentioned force accuracy of $\pm 25\,\mathrm{N}$ specified by the manufacturer.

A limitation of the approach is that the robot needs to be configured exactly the same way between offline training and realtime estimation. However, in industrial settings this usually is the case.

V. Conclusion

In order to provide a robot with a sense of force, an approach for torque estimation from prior experience was presented. Instead of using dedicated force sensors, data from all sensors available on the robot is recorded, including e.g. angle, velocity and current sensors. Using Transfer Entropy, a model consisting of two feature spaces is learned from the recorded sensor data. During runtime, sensor data is captured and projected into these feature spaces. The first feature space is used to estimate the actual phase of the motor skill by comparing realtime and training data. The second feature space estimates the actual torque, again by comparing realtime with training data. By taking into account the previously calculated phase estimate, the computational effort for torque estimation is reduced.

As explained in the discussion, the quality of the extracted models varies over time. In some cases, this could lead to TE values that vary for different phases of the behavior. Thus, a phase-dependent selection of the optimal subset of samples could be a beneficial extension of the proposed approach in order to increase the overall accuracy of the torque estimate. Nonetheless, with the present implementation torques were accurately estimated, achieving a quality that compares to force estimation capabilities achieved by humans.

References

[1] R. Bischoff, J. Kurth, G. Schreiber, R. Koeppe, A. Albu-Schäffer, A. Beyer, O. Eiberger, S. Haddadin, A. Stemmer, G. Grunwald, and G. Hirzinger, "The KUKA-DLR lightweight robot arm - a new reference platform for robotics research and manufacturing," in *ISR/ROBOTIK 2010, Proceedings for the joint conference of ISR*. VDE Verlag, 2010, pp. 1–8.

[2] E. Magrini, F. Flacco, and A. De Luca, "Control of generalized contact motion and force in physical human-robot interaction," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, May 2015, pp. 2298–2304.

[3] A. Colome, D. Pardo, G. Alenya, and C. Torras, "External force estimation during compliant robot manipulation," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, May 2013, pp. 3535–3540.

[4] E. Gribovskaya, A. Kheddar, and A. Billard, "Motion learning and adaptive impedance for robot control during physical interaction with humans." in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2011, pp. 4326–4332.

[5] A. Lee, H. Lu, A. Gupta, S. Levine, and P. Abbeel, "Learning force-based manipulation of deformable objects from multiple demonstrations," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[6] T. Schreiber, "Measuring information transfer," *Physical Review Letters*, vol. 85, no. 2, pp. 461–464, 2000.

[7] M. R. Jovanovi, P. J. Schmid, and J. W. Nichols, "Sparsity-promoting dynamic mode decomposition," *Physics of Fluids (1994-present)*, vol. 26, no. 2, 2014.

[8] E. Berger, M. Sastuba, D. Vogt, B. Jung, and H. B. Amor, "Estimation of perturbations in robotic behavior using dynamic mode decomposition," *Advanced Robotics*, vol. 29, no. 5, pp. 331–343, 2015.

[9] E. Berger, D. Müller, D. Vogt, B. Jung, and H. Ben Amor, "Transfer entropy for feature extraction in physical human-robot interaction: Detecting perturbations from low-cost sensors," in *Humanoids'14*, 2014.

[10] H. Sakoe, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 26, pp. 43–49, 1978.