2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Macau, China, November 4-8, 2019

ARMCL: ARM Contact point Localization via Monte Carlo Localization

Adrian Zwiener, Richard Hanten, Cornelia Schulz and Andreas Zell

***Abstract*— Detecting and localizing contacts acting on a manipulator is a relevant problem for manipulation tasks like grasping, since contact information can be helpful for recovering from collisions or for improving the grasping performance itself. In this work, we present a solution for contact point localization, which is based on Monte Carlo Localization. Usually, an Articulated Robotic Manipulator (ARM) is not equipped with tactile skin, but with proprioceptive sensors, which we assume as an input for our method. In our experiments, we compare our method with a direct optimization method, machine learning approaches and another particle filter method, both on simulated and real world data from a Kinova Jaco2. While our proposed method clearly outperforms the other optimization approaches, it performs about equally well as Random Forest (RF) classifiers, although both methods have their strengths on different parts of the manipulator, and even achieves better results than multi-layer perceptrons (MLPs) on the links farthest from the manipulator base.**

I. Introduction

Mobile manipulation typically involves grasping, for which it is essential to avoid external objects as well as self-collisions. However, also unforeseen contacts, caused by accumulated errors, imprecise perception or planning or dynamic obstacles, need to be considered.

In such cases, joint force/torque sensors have been used to detect and recover from a collision [1], [2]. Further, it was discussed that the grasping performance can be improved by using tactile feedback [3], [4], [5]. In [5], the tactile sensor is similar to the human skin and located on the whole arm.

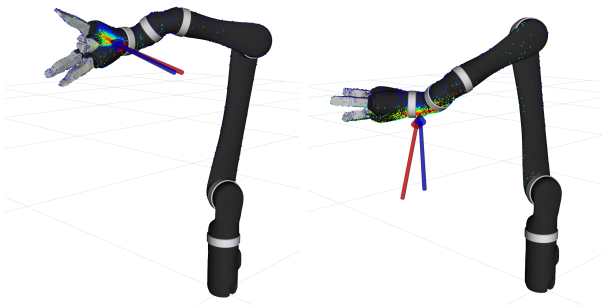
As most manipulators are not equipped with such skins or sensors, we investigate the problem of contact point localization based on proprioceptive sensors, i.e. joint positions, velocities and one-dimensional joint torques.

Whereas our preceding research [6] focused on different machine learning approaches, this work presents a solution based on a particle filter.

The contributions of our work are the following:

- We present a triangle mesh representation, which allows us to propagate the particles efficiently directly on the manipulator surface, as well as a suitable motion model.
- We introduce a measurement model based on the residual observer method [7] which outperforms a related approach [8] both in terms of runtime and contact localization results.
- We present exhaustive experimental results both on simulated and real world data using a Kinova Jaco2 manipulator, within which we compare our particle filter method with [8] and with the results we presented in [6], which includes machine learning approaches and a direct optimization method based on [9].
- Our implementation is open source and will be available on GitHub[1] and can easily be applied to further research with different manipulators or model functions.

A. Zwiener, R. Hanten, C. Schulz and A. Zell are with the Cognitive Systems Group at the Computer Science Department, University of Tübingen, Sand 1, 72076 Tübingen, Germany.
Contact: adrian.zwiener@uni-tuebingen.de



(a) (b)

Fig. 1. Simulated contact points. True and detected contacts are visualized as red and blue arrow, respectively. The coloured points represent the particles and their likelihood $p\left(\tilde{\mathbf{r}} \mid {}^{i}\boldsymbol{\rho}_{j}\right)$ between 0 (blue) and 1 (red).

Simulated contact localization examples of our particle filter approach are visualized in Fig. 1.

II. Related Work

The problem of detecting contacts has usually been done via thresholding [10], [11].

Regarding contact localization, the most convenient solutions are based on using tactile skins or proximity sensors [5], [12], [13]. In absence of such sensors, e.g. Magrini and De Luca [14] detected proximities between the manipulator and other objects using an RGB-D camera, which is a somewhat limited approach because of occlusions.

Other research is based on joint force/torque sensors [1], [2]. With our Kinova Jaco2, one problem is that only 1D joint torque sensors are available, with which no unique solution can be identified [1], [9].

In [7], several external torque observers are discussed. Further, a linear method is presented being able to localize contacts on the surface of a manipulator, which act normal to the surface. However, this method requires six nonzero components of the external torque vector. Otherwise, the linear problem is underdetermined. Thus, for the Jaco2 it will only work for the end effector. In this paper, we focus on methods which potentially work for every link. However, we found this method to be very accurate at the end effector.

[1] https://github.com/cogsys-tuebingen/muse_armcl

In our preceding work [6], we applied different machine learning approaches to the problem of contact point localization on a discrete set of possible contact locations on each link in comparison to the least squares optimization method of Likar et al. [9]. However, this discrete method does not consider the full continuous state space, and extensive training is necessary.

Similarly, Popov et al. [15] identify the link a contact is applied to via neural networks. The difference between soft and hard contacts is discussed and classified, and also used to detect if a contact is purposeful or accidental. This is an important feature for monitoring manipulation tasks.

In contrast to that, Manuelli and Tedrake [8] presented a particle filter based approach capable of detecting and localizing multiple contacts on the surface of a humanoid robot. However, they propagate the particles in full 3D space and then project them back on the surface of the robot. Further, they solve an optimization problem in each update step for each particle resulting in a noisy localization.

Therefore, we present a direct and much faster measurement model with which we achieve about 160 Hz with about ten times more particles. Further, we do not use 3D sampling and projection for particle propagation, but a mesh representation of the manipulator surface.

III. Contact Point Localization

In the following, we reduce the contact point localization problem to an optimization problem and introduce our particle filter solution which, unlike other work [8], operates directly on the manipulator surface.

A. Contact Point Optimization

As already explained in detail in our preceding work [6], finding a contact point and the corresponding force can be considered an optimization problem, which has been solved directly by Likar et al. [9] and Ralph and Pai [16].

Therefore, the external torques $\boldsymbol{\tau}_{ext}$ can be estimated using the *residual observer* method presented in Haddadin et al. [7], which approximates them by the residual vector

$$\mathbf{r}(t) = K\left(\mathbf{m} - \int_0^t \left(\boldsymbol{\tau} + C^T(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} - \mathbf{g}(\mathbf{q}) + \mathbf{r}\right) dt'\right), \quad (1)$$

with K being a large positive definite diagonal gain matrix, $\mathbf{m}$ being the generalized momentum, $\boldsymbol{\tau}$ being the measured torques, $C(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}}$ being centripetal and Coriolis forces and $\mathbf{g}(\mathbf{q})$ being the gravity vector. If K is large enough, $\mathbf{r} \approx \boldsymbol{\tau}_{ext}$. Although in simulation it can be shown that the accuracy of this approximation is reduced with higher accelerations, in reality this momentum observer leads to better results than using a dynamic model and very noisy joint accelerations.

Written as an optimization problem, the objective of contact point localization is to minimize $||\mathbf{r} - \boldsymbol{\tau}_{ext}||_2$ with respect to the relative contact location (s,ϕ) on the link, denoted as cylindrical model. Because of ambiguities, as discussed in [9], [6], we solve the normalized problem

$$\min_{s,\phi} ||\tilde{\mathbf{r}} - \tilde{\boldsymbol{\tau}}_{ext}(s,\phi)||_2, \quad (2)$$

with $\tilde{\mathbf{r}}$ and $\tilde{\boldsymbol{\tau}}_{ext}(s,\phi)$ being the normalized forms of $\mathbf{r}$ and $\boldsymbol{\tau}_{ext}(s,\phi)$, respectively. It is

$$\boldsymbol{\tau}_{ext}(s,\phi) = \mathbf{J}_i^T(\mathbf{q}) \begin{bmatrix} {}^cR(\phi) & -s\,{}^cR(\phi)\mathbf{e}_z\times \\ 0 & {}^cR(\phi) \end{bmatrix} \begin{bmatrix} 0 \\ ||\mathbf{f_c}||_2\mathbf{e}_x \end{bmatrix}, \quad (3)$$

with $\mathbf{e}_{x,z}$ being unit vectors, $\mathbf{f_c}$ being the contact force and $\mathbf{J}_i(\mathbf{q})$ being the Jacobian of joint i in the frame of joint i. Usually, the geometric Jacobian $\mathbf{J}(q)$ is used to derive the joint torques as $\boldsymbol{\tau}_{ext} = \mathbf{J}(q)^T(0,\mathbf{f}_c)$, where $(0,\mathbf{f}_c)$ has to be given w.r.t the base frame. Here, $\mathbf{J}_i(\mathbf{q})^T$ is applied in local joint coordinates of the associated joint j_i. The factorization of the contact Jacobian is discussed in [7].

B. Surface Mesh Representation

Polygonal meshes are well established in interactive 3D graphics applications which require an efficient implementation for surface models. Since this is also true for the localization of contact on the manipulator surface, we employ the library OpenMesh [17], [18]. This library implements a generic half-edge data structure for polygonal meshes. For the map representation of the manipulator surface it is sufficient to use triangular meshes implemented within OpenMesh. Triangular meshes consist of vertices $\{v_i\}$ containing information about 3D position $\mathbf{v}_i$ and 3D normal $\mathbf{n}_i$ of the associated surface point. Three vertices, connected by edges e, form a face f, which is a triangular approximation of the local surface. With this representation, each point $\mathbf{x}_{a,b}(s)$ between $\mathbf{v}_a$ and $\mathbf{v}_b$ with $0 \le s \le 1$ can be interpolated:

$$\mathbf{x}_{a,b}(s) = \mathbf{v}_a + s \cdot (\mathbf{v}_b - \mathbf{v}_a). \quad (4)$$

To model the complete manipulator, we build a tree of these meshes, and store time depending forward kinematic transformations ${}^jT_i$ between each link i and its parent j, as shown in Fig. 2.

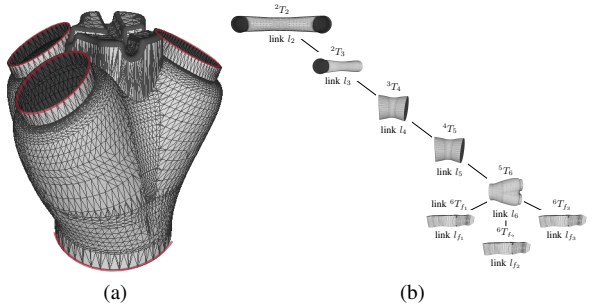


Fig. 2. (a) Exemplary mesh of the triangle mesh of link l_6 (hand) with boundaries in red. (b) Mesh map tree of the Kinova Jaco2 manipulator.

C. Motion Model

Similar to [8], we also use a random walk method to propagate the particles, but in contrast to their work, we directly propagate the hypotheses on the mesh surface instead of sampling from 3D Cartesian space and then projecting them onto the model.

In our random walk, we draw a momentum δ uniformly from $[0, \delta_{\max}]$ and then propagate each particle via randomly

selected neighbors to a final position, whereby we ensure that the distance to its original position grows monotonously. For particles corresponding to boundary vertices, we apply a probability $0 \leq \phi \leq 1$ of passing over to the adjacent link.

Thereby, each particle is represented as

$$ {}^{i}\boldsymbol{\rho}_j := \left\langle a_j, b_j, s_j, {}^{i}\mathbf{x}_{a_j,b_j}(s_j), {}^{i}\mathbf{n}_{a_j,b_j} \right\rangle, \tag{5} $$

where i denotes the index of the corresponding link, a and b are indices of the points defining the edge ${}^{i}\boldsymbol{\rho}_j$ is located on, s and ${}^{i}\mathbf{x}_{a,b}(s)$ are as in Eq. (4) and ${}^{i}\mathbf{n}_{a,b}$ is the normal of the edge.

To initialize the particle set uniformly distributed over the manipulator surface, we first estimate the number of particles we want to spawn per link. Therefore, we calculate for each link i the ratio m_i of the summed length of its edges to the summed length of the edges of all links.

Then, we uniformly draw $m_i \cdot N$ random vertices for each link i, where N is the desired total number of particles. The particles are initialized on the vertices with $s = 0$, since a motion update will be applied before the first measurement update. We chose to restrict the particles to the edges, since this results in a one-dimensional motion problem and feasible memory requirements.

D. Measurement Model

A measurement model is used to weight the particles according to a measure of how well they explain the present measurement. In case of contact point localization, this means how well the residual $\mathbf{r}$ can be explained by a force acting on a point on the surface, represented as particle ${}^{i}\boldsymbol{\rho}_j$.

Whereas Manuelli and Tedrake [8] use a "*polyhedral approximation to the friction cone*" to solve a quadratic program to find the optimal likelihood $p\left(\mathbf{r} \mid {}^{i}\boldsymbol{\rho}_j\right)$ for each particle, which results in massive computational costs, we present a direct method. Therefore, we need to assume that the forces act normal on the manipulator surface, which was done in many other works as well [7], [9], [10], [11], [16], [19].

For each particle ${}^{i}\boldsymbol{\rho}_j$, we model the expected normalized external torque $\tilde{\boldsymbol{\tau}}_{ext}({}^{i}\boldsymbol{\rho}_j)$ by using Eq. (3) with normal forces and coordinates relative to the coordinate frame of the corresponding joint:

$$ \boldsymbol{\tau}_{ext}({}^{i}\boldsymbol{\rho}_j) = \mathbf{J}_i^T(\mathbf{q}) \begin{pmatrix} \mathbb{1} & -{}^{i}\mathbf{x}_{a,b}(s)\times \\ 0 & \mathbb{1} \end{pmatrix} \begin{pmatrix} 0 \\ -{}^{i}\mathbf{n}_{a,b} \end{pmatrix}. \tag{6} $$

Then, we weight each particle ${}^{i}\boldsymbol{\rho}_j$ according to the likelihood $p\left(\tilde{\mathbf{r}} \mid {}^{i}\boldsymbol{\rho}_j\right)$ of the normalized torque residual $\tilde{\mathbf{r}}$, given particle ${}^{i}\boldsymbol{\rho}_j$, which is

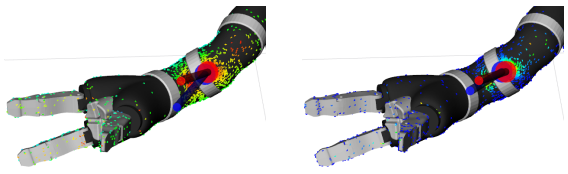
$$ p\left(\tilde{\mathbf{r}} \mid {}^{i}\boldsymbol{\rho}_j\right) \propto \exp\left(-\frac{1}{2}\left(\tilde{\mathbf{r}} - \tilde{\boldsymbol{\tau}}_{ext}({}^{i}\boldsymbol{\rho}_j)\right)^T \Sigma_{meas}^{-1} \left(\tilde{\mathbf{r}} - \tilde{\boldsymbol{\tau}}_{ext}({}^{i}\boldsymbol{\rho}_j)\right)\right), \tag{7} $$

with $\tilde{\boldsymbol{\tau}}_{ext}({}^{i}\boldsymbol{\rho}_j)$ being the normalized form of Eq. (6) and Σ_{meas} being the measurement model covariance. This directly corresponds with the optimization method presented in Section III-A, since a good solution ${}^{i}\boldsymbol{\rho}_j$ to the optimization problem will be assigned a high likelihood $p\left(\tilde{\mathbf{r}} \mid {}^{i}\boldsymbol{\rho}_j\right)$.

For the Kinova Jaco2, we empirically found that the information matrix

$$ \Sigma_{meas}^{-1} = \operatorname{diag}\left(\frac{1}{|l_2|}, \cdots, \frac{1}{|l_6|}\right). \tag{8} $$

with the inverse link lengths $|l_i|$ as diagonal entries significantly improves the results. Without this scaling term it is less likely for the particle filter to converge to solutions on the links with higher index (hand or fingers), because the torque residuals at these links are usually small. The impact of Σ_{meas}^{-1} can be observed in Fig. 3.



(a) (b)

Fig. 3. Particles, colored by their likelihood $p\left(\tilde{\mathbf{r}} \mid {}^{i}\boldsymbol{\rho}_j\right)$ (a) with Σ_{meas}^{-1} being the identity matrix and (b) with Σ_{meas}^{-1} as in Eq. (8). Using the inverse link lengths for the information matrix clearly leads to a higher convergence.

Our choice of Σ_{meas}^{-1} can also be justified by the derivation of the torques $\boldsymbol{\tau} = \mathbf{x} \times \mathbf{F}$, which is the cross product between force and distance to the joint axis. Hence, to calculate the norm of the force, we have to divide by this distance, which correlates with the l_i. Or in other words, the longer a lever, the higher the torque.

As for the assumption of the forces acting normal to the surface, it is not reasonable to consider non-normal forces, because this would lead to an infinite amount of solutions which can not be distinguished with the sensors available.

However, using a normalized external torque measurement has one disadvantage. The particle filter tries to find contacts which are caused by noise, since the measurements will never be exactly zero. This can be solved by thresholding. If the external torque measurement does not exceed a threshold, we do not perform a correction step and the particle distribution stays unchanged. A similar threshold for the null hypothesis is also used in [8] for reasons of computational efficiency.

E. Contact Force Amplitude

Since a normalized update model is used, the contact force amplitudes have to be derived after a contact location is found. This can be done very precisely by a *Singular Value Decomposition (SVD)* as long as we do not have more contacts than the n torque sensors.

For just one contact, this problem reduces to

$$ f_c = \frac{||\mathbf{r}||_2}{||\boldsymbol{\tau}_{ext}(\mathbf{p}, \mathbf{n})||_2}. \tag{9} $$

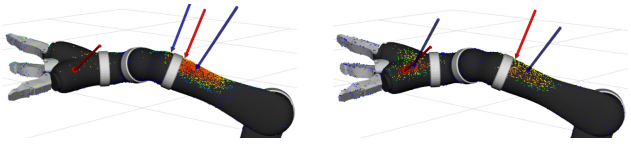
F. Multiple Contacts

To able to detect multiple contacts acting at the same time, in [8] additional particle sets are inserted until the external torque residual $\mathbf{r}$ is represented sufficiently well

by the particles. While this might be beneficial in case of a parallel tree of links, such as their walking robot, this presumably behaves differently in case of a serial tree.

With n joints and n 1D torque sensors, it is difficult to detect and localize multiple contacts, because the superposition of contact forces will likely lead to a single contact at a different location and with a different orientation. This single contact is still able to represent the external torque observation $\mathbf{r}$ well. Consequently, there are infinite solutions in such cases and it might not be beneficial to add more particle sets, at least with a serial kinematic chain and only 1D torque sensors available. For this reason, we focused on the detection of single contacts in this work. A combination with the proposed method of [8] might still be helpful and is left for future work.

However, if multiple contacts are detectable, the resampling step of the particle filter already provides multiple distinct distributions of particles, as displayed in Fig. 4. Obviously, the particle density depends on the force amplitudes, since small forces might be compensated by the serial structure and therefore might not be measured in several joint torque sensors, as for example force f_2 in Fig. 4a, which makes them hard to detect.



(a) (b)

Fig. 4. Multiple simulated contacts. In both examples, contacts on the same locations are applied, but with differing force amplitudes f_1 (link l_3) and f_2 (link l_6 (hand)). They are set to (a) $f_1 = -13.8\,\mathrm{N}$ and $f_2 = -1.4\,\mathrm{N}$ and (b) $f_1 = -13.8\,\mathrm{N}$ and $f_2 = -12.4\,\mathrm{N}$. In both examples, there are particles near both contacts, and their density correlates with the amplitude.

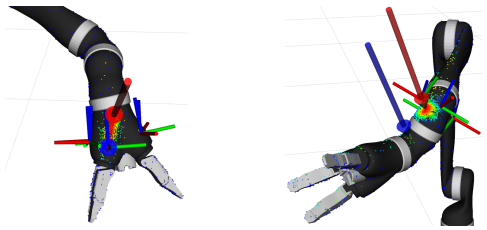
G. Density Estimation

The particle set itself is already an interesting output, because it gives us basically all solutions at once. This could for instance be used to match the particles with e.g. RGB-D data to find real contact points near the manipulator, similar to the approach of Magrini and De Luca [14]. On the other hand, RGB-D output could be used as a prior for the filter.

However, to get quantitative results, we need to apply clustering methods to find the acting contact point(s). Therefore, we implemented different such methods as plugins, which can be exchanged easily thanks to the abstract core framework we used as a basis for our implementation [20].

1) Binning (B): Since we want to compare our particle filter solution with the results of our preceding work [6], where we trained machine learning approaches to classify contacts of a discrete set $\{\Lambda_k\}$ of possible contacts, our first density function estimates the most likely of these contact points, or in this case also called *classes*.

For finding the best assignments, we optimize the likelihood of a particle ${}^i\boldsymbol{\rho}_j$ corresponding to the cluster h_{Λ_k} of contact point Λ_k and then assign the particle to the found histogram. Therefore, we transform position ${}^i\mathbf{x}_{a_j,b_j}(s_j)$ and



(a) (b)

Fig. 5. Real (red arrow) and detected (blue arrow) contact, using binning. The nearest classes of the discrete contact point set are displayed as small coordinate axes. In (a), the detected class is the nearest one, in (b), the detected class is on another link than the mean of the particle set, but the normal of the classes match. Note, that in both examples the true contact is not one of the discrete contact points of the set $\{\Lambda_k\}$.

normal ${}^i\mathbf{n}_{a_j,b_j}$ of the particle into a common frame and model the likelihood as a mixture of Gaussians

$$
\begin{aligned}
&p\left(h_{\Lambda_k} \leftarrow {}^i\boldsymbol{\rho}_j \mid {}^i\boldsymbol{\rho}_j, \Lambda_k\right) = \\
&\quad \phi_p \exp\left(-\frac{1}{2\Sigma_p}\left|\left|{}^0T_{i(k)}\,{}^{i(k)}\mathbf{x}_k - {}^0T_i\,{}^i\mathbf{x}_{a_j,b_j}(s_j)\right|\right|_2^2\right) \\
&+ \phi_n \exp\left(-\frac{1}{2\Sigma_n}\left|\left|{}^0T_{i(k)}\,{}^{i(k)}\mathbf{n}_k - {}^0T_i\,{}^i\mathbf{n}_{a_j,b_j}\right|\right|_2^2\right), \quad (10)
\end{aligned}
$$

which is maximal if normal and position match best. ϕ_p and ϕ_n are normalization factors, Σ_p and Σ_n are the covariances of position and normal in the particle set and ${}^0T_i$ is the transformation matrix from frame j_i to the common base frame j_0. Note, that each contact point Λ_k consists of a reference frame $j_{i(k)}$, a position ${}^{i(k)}\mathbf{x}_k$ and a normal ${}^{i(k)}\mathbf{n}_k$.

Finally, the histogram bins with the highest sum of likelihood values $p\left(\tilde{\mathbf{r}} \mid {}^i\boldsymbol{\rho}_j\right)$ of all assigned particles ${}^i\boldsymbol{\rho}_j$ are given as classification result. Examples are shown in Fig. 5.

The reason for optimizing in regard to both distance and orientation is, that the contact force direction can be determined unambiguously, while there may be multiple solutions for distance and force amplitude. Consequently, it is more important to find a contact point with matching normal.

2) Weighted Means (WM): In our second density function, we simply apply 3D voxel grid clustering with adaptive voxel grid size to our particle set. Then, we calculate the weighted mean of the particles of each cluster, whereby we use the likelihood values $p\left(\tilde{\mathbf{r}} \mid {}^i\boldsymbol{\rho}_j\right)$ as weights. Finally, for each cluster, we estimate the particle closest to the weighted mean, which we then take as the found contact point.

3) Nearest Neighbors (NN): In our third method, we first cluster the particles by their nearest mesh vertex. Then, we expand the clusters along the mesh edges and merge clusters, if necessary. Similar to before, the particles closest to the means of the clusters are given as output.

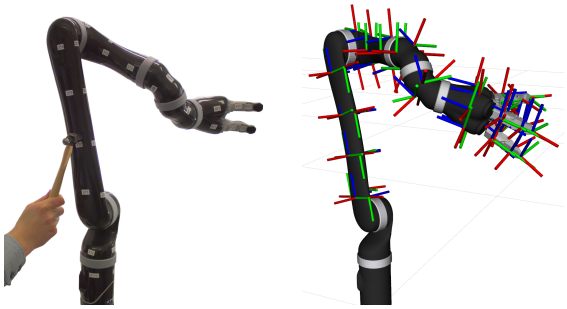
IV. Experiments

We conducted experiments to measure the accuracy of the particle filter approach on the same simulated data as in [6] and on real world data and evaluate the time needed for convergence as well as the general runtime of our models.

For importance sampling, we use *Kullback-Leibler distance* (KLD) [21]. We augment KLD to add a uniform sampled fraction of the removed particles after resampling. We empirically found that adding 2.5% uniform particles after resampling improves the overall results by approx. 2%.

A. Contact Point Classification

First, we present our simulated experiments with a subset of 20 random trajectories from the dataset used in [6]. Each trajectory was recorded with no contact and with a simulated normal contact force with uniformly sampled amplitude $f_c \in [2, 80]$ N applied to each of the 40 discrete contact points on the arm surface (see Fig. 6) individually. Further, random sampled noise with a standard deviation of $0.1\,\mathrm{N\,m}$ was applied to the torques. This resulted in over 455.000 samples.



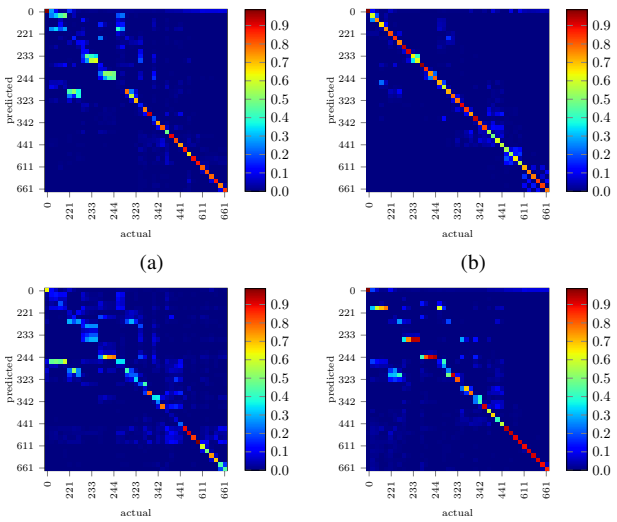
(a) (b)

Fig. 6. The set of possible contact points, displayed (a) as labels on the manipulator surface and (b) as small coordinate frames.

In Tab. I we present the classification results of different classifiers. The method *DR* is the optimization method presented in [6] and based on [9]. Random Forests (*RF*) and Multilayer Perceptrons (*MLP*) are presented in [6]. The last four entries represent particle filters. *PFWM* is a particle filter using *Weighted Means* as density estimation, where as *PFNN* uses *Nearest Neighbors* and *PFB* uses the presented *Binning* method. Finally, *FCB* is the particle filter using the polyhedral friction cone approximation disussed in [8]. ψ_{nc} is the rate of properly detected *no contact* events, ψ_c are the correctly classified contact points, ψ_{nd} are contacts which are not detected and classified as no contact. $\Delta\theta$ is the angle between the actual contact direction and the detected direction. Similarly, Δd is the Euclidean distance between true contact point and detected contact point.

The trajectories used in the set are unknown to the ML classifiers which are trained on 80 different trajectories.

Overall, the *MLP* achieves the highest rate of correctly classified contacts ψ_c, but fails to detect $\psi_{nd} = 14\%$ of the contacts. Besides, the distance error is minimal. The ARMCL method achieves overall good results with $3-7\%$ undetected contacts, and small errors in orientation $\approx 0.30\mathrm{rad}$ and distance $\approx 0.11\mathrm{m}$ which are equally good as the results of other methods. Since *WM* and *NN* clustering will result in different surface points, it cannot be expected tho have a high binary classification rate ψ_c. Besides, we see that *Binning*



(a) (b) (c) (d)

Fig. 7. Confusion matrices. (a) ARMCL and *Binning* density estimation. (b) MLP classifier. (c) DR optimizer. (d) friction cone update model.

TABLE I

CONTACT POINT CLASSIFICATION

method	ψ_{nc}	ψ_c	ψ_{nd}	$\Delta\theta$(rad)	Δd(m)	f(Hz)
DR	**99%**	35%	5%	0.30 ± 0.4	0.13 ± 0.11	30
RF	95%	52%	**0.4%**	0.40 ± 0.7	0.09 ± 0.11	$>$ **200**
MLP	97%	**73%**	14%	0.29 ± 0.7	$\mathbf{0.04 \pm 0.08}$	$>$ **200**
PFWM	**99%**	17%	3%	0.40 ± 0.4	0.11 ± 0.11	159
PFNN	**99%**	16%	7%	0.30 ± 0.4	0.11 ± 0.11	125
PFB	**99%**	53%	5%	0.30 ± 0.5	0.12 ± 0.13	71
FCB	98%	52%	4%	$\mathbf{0.25 \pm 0.5}$	0.11 ± 0.14	63

achieves a much higher classification rate ψ_c indicating the correctness of the likelihood model (10). Moreover, the *FC* update model has the lowest mean orientation error, but a similar distance error as the other methods. The standard deviation of the orientation error originates most likely in the density estimation (*Binning*).

All particle filters use a set size between 1500 and 5000 particles depending on KLD resampling. However, our update model achieves an update rate up to 159Hz, whereas the friction cone version only achieves 63Hz. The ML methods achieve more than 200Hz, the optimizer 30Hz.

Consequently, the particle filter is a very interesting method which is computationally more efficient and more accurate than the optimizer and has higher performance than a RF without any training. For density estimation, we use the *Binning* method.

The confusion matrices can be found in Fig. 7. For comparison, we show ARMCL *Binning* (Fig. 7a), the MLP (Fig. 7b), the optimization method (Fig. 7c) and ARMCL with the friction cone update model and *Binning* (Fig. 7d). We can see optimization (including ARMCL) works best close to the end effector (greatest labels), whereas ML methods achieve poorer results close to the end effector since any change in the joint angles results in a larger displacement of the end effector. Therefore, for ML we have to sample a lot of configurations. Besides, the friction cone update model has a slightly higher confusion rate. Clearly, the diagonal is more evident in both particle filter versions versus the optimization

method presented in [9].

B. Convergence Time

To measure the time the particle filter needs to converge, we apply forces shaped as box-functions to 10 different locations and two different configurations to the Jaco2 in simulation.

Thus, we define the convergence time Δt which is determined as follows: First, we estimate the error Δf between true force f_c and detected force f_d, c.f. Fig. 8 for a real world experiment at the location indicated by the yellow arrow in Fig 9a. Second, we derive the smoothed mean force error f_m and its standard deviation σ during the contact which is the region where $f_c \neq 0$ indicated in red in Fig. 8. The convergence time Δt is defined as the time difference between the time point t_{f_c} when the true force f_c first reached it's maximum and time point $t_{\Delta f}$ when Δf is in the margin with $\pm\frac{\sigma}{2}$ around f_m. Or in other words, the time when Δf is in between the blue dashed lines in Fig. 8, indicated by the black dotted line.

We obtain a mean convergence time $\Delta t = (0.38 \pm 0.13)$s and the mean force error is $\Delta f = (1.1 \pm 0.6)$N.

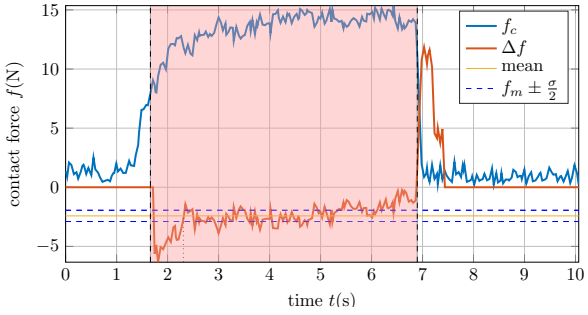


Fig. 8. Convergence experiment examples. f_c (blue) is the true force, Δf (red) the error, f_m (yellow) the mean error during the contact and σ the standard deviation.

C. Experiments on Real World Data

Similar to the convergence time experiments, we apply single contact forces manually to three points on the arm surface in real world experiments. Thereby, we use a force sensor to measure the ground truth forces. The contact points, detected and true forces over time as well as the linear and angular error between detected and true contacts (angle between detected and true direction) are displayed in Fig. 9.

The convergence time criteria is slightly modified since real sensors are used. The contact region highlighted in red in Fig. 8 is determined by the region where the true force $f_c(t)$ exceeds the standard deviation σ_{f_c} of the sequence.

Mean errors and convergence time until the filer reacts to the contact are shown in Tab. II. The convergence times are approximately twice the convergence times in the simulation. The reason is a slight delay between the force sensor and the response of the filter. In average we find a time offset $\Delta t_o \approx 0.2\,\mathrm{s}$ between a significant change of the force sensor and the detection output. Besides, we determined that the time difference between the maximum distance error or angular error and first local minimum of the signal is in order of the convergence time determined in simulation.

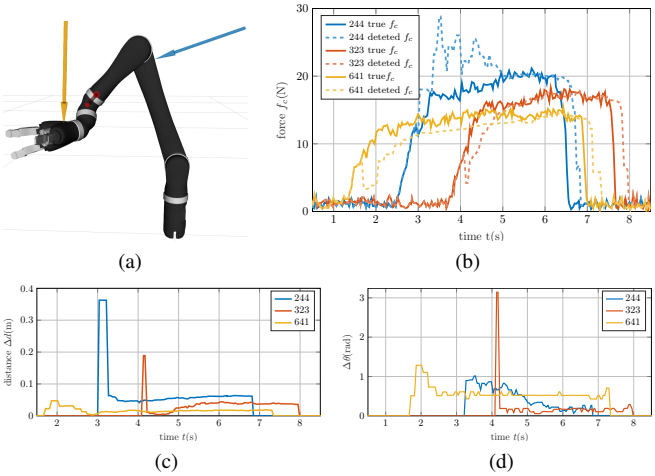


Fig. 9. Contact forces are applied to (a) three exemplary contacts. (b) True and detected force, (c) linear error and (d) angular error over time.

TABLE II

CONVERGENCE TIME AND ERROR FOR THE THREE CONTACT POINTS

contact	mean Δd(m)	mean $\Delta\theta$(rad)	mean Δf_c(N)	Δt(s)
244	0.070 ± 0.080	0.40 ± 0.30	3.0 ± 3.0	0.68
323	0.030 ± 0.027	0.20 ± 0.40	1.3 ± 1.6	0.67
641	0.017 ± 0.009	0.60 ± 0.20	1.8 ± 1.6	0.65

D. Runtime Evaluation

Finally, we want to give average runtimes for the experiments in Subsec. IV-A. These experiments were conducted using an Intel Xeon CPU E5-2623 v3 @3.00GHz. Tab. III displays the average runtimes of the filter without the overhead of the front end. The optimization required in the *FC* measurement model increases the runtime significantly w.r.t. the proposed normalized update model. In addition, the runtime prediction step is increased probably due to the fact that the *FC* particles require more memory (4 additional floats) and thus copying takes more time. The fastest density estimation method is *WM*, followed by *NN* and *Binning*.

V. DISCUSSION

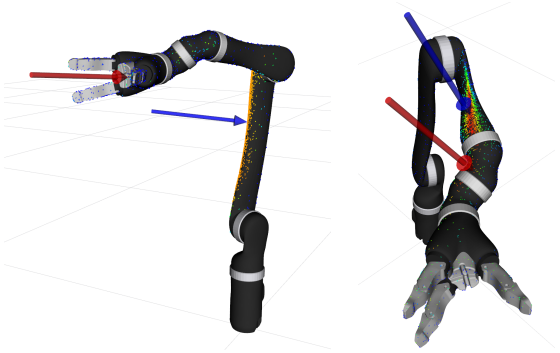
Contact localization via Monte Carlo optimization can be very accurate. However, there are some restrictions. First, the contact localization problem is ambiguous, thus the found location might differ from the actual one. For example in Fig. 10a all positions at the front facing side of link 2 are equally probable indicated by a similar update model value (particle color). Besides, as both examples in Fig 10 show, due to the geometry of the manipulator some contacts at

TABLE III

RUNTIME IN ms

method	complete	prediction	resampling	update
PFB	12.52	1.10	9.59	1.83
PFNN	10.42	1.64	6.52	2.26
PFWM	**7.43**	**1.08**	**4.44**	**1.91**
FCB	139.85	6.62	13.58	119.65
FCNN	138.23	6.10	6.63	125.50
FCWM	134.70	5.62	4.34	124.74

higher links might only lead to a significant external torque in lower joints. Consequently, depending on the number of vertices, locations at links closer to the base might be more likely. Still, the direction of the forces is detected correctly.



(a) (b)

Fig. 10. Examples of contacts leading only to significant external torques in lower joints. The red arrow is the true contact, the blue the detected one.

Finally, the detection is only as accurate as the dynamic model and the torque sensors. Not modeled effects such as crosstalk torques lead to a disturbance effecting the localization. Moreover, the used external torque observer is only an approximation. If the manipulator is highly accelerated, the accuracy is reduced. However, in most cases the observer still has a higher accuracy than using noisy joint accelerations.

VI. Conclusion

ARMCL is an efficient approach, as good as Random Forests, without training and superior to just direct optimization. Even for the MLP we might find regions in the configuration space where the detection is worse due to overfitting. However, if we only detect contacts for trained trajectories, the detection rate is between $82\,\%$ (real data) and $90\,\%$ (simulation).

Our measurement model is robust w.r.t to small disturbances introduced by random particles, the FC version is not. Besides, ARMCL offers a more continuous representation compared to ML. If we want to have a higher resolution, we can sub sample the graph to add more vertices.

In addition, ARMCL is able to sample the state space with a high density. The resulting particle cloud is an interesting feature since it shows the complete distribution of probable contacts. Thus, for future work this might have a high importance: First, we can further investigate multiple contacts. In [8], multiple particle sets are suggested to detect multiple contacts, however this might not be required for our larger particle sets and suitable density estimation methods. Second, with a 3D sensor sensing the environment, we can search for probable contacts within the environment, which is highly interesting for robust manipulation.

Finally, it can be very interesting to investigate the influence of tangent contact forces on the accuracy in simulation, especially during motion.

References

[1] A. D. Luca, A. Albu-Schaffer, S. Haddadin, and G. Hirzinger, "Collision Detection and Safe Reaction with the DLR-III Lightweight Manipulator Arm," in *Intelligent Robots and Systems (IROS), 2006 IEEE/RSJ International Conference on*, Oct 2006, pp. 1623–1630.

[2] S. Haddadin, A. Albu-Schaffer, A. D. Luca, and G. Hirzinger, "Collision detection and reaction: A contribution to safe physical Human-Robot Interaction," in *Intelligent Robots and Systems (IROS), 2008 IEEE/RSJ International Conference on*, Sept 2008, pp. 3356–3363.

[3] R. Platt, "Learning Grasp Strategies Composed of Contact Relative Motions," in *Humanoid Robots, 2007 7th IEEE-RAS International Conference on*. IEEE, 2007, pp. 49–56.

[4] P. Pastor, L. Righetti, M. Kalakrishnan, and S. Schaal, "Online movement adaptation based on previous sensor experiences," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. IEEE, 2011, pp. 365–371.

[5] A. Jain, M. D. Killpack, A. Edsinger, and C. C. Kemp, "Reaching in clutter with whole-arm tactile sensing," *The International Journal of Robotics Research*, vol. 32, no. 4, pp. 458–482, 2013.

[6] A. Zwiener, C. Geckeler, and A. Zell, "Contact Point Localization for Articulated Manipulators with Proprioceptive Sensors and Machine Learning," in *Robotics and Automation (ICRA), 2018 IEEE International Conference on*, May 2018, pp. 323–329.

[7] S. Haddadin, A. De 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.

[8] L. Manuelli and R. Tedrake, "Localizing External Contact Using Proprioceptive Sensors: The Contact Particle Filter," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE, 2016, pp. 5062–5069.

[9] N. Likar and L. Žlajpah, "External Joint Torque-based Estimation of Contact Information," *International Journal of Advanced Robotic Systems*, vol. 11, no. 7, p. 107, 2014.

[10] K. Schroeder, M. Pryor, and T. Harden, "On the Use of Joint Torque Sensors for Collision Detection in a Confined Environment," in *ANS 3rd International Joint Topical Meeting on Emergency Preparedness & Response and Robotics & Remote Systems*, Knoxville, TN, 2011, pp. 179–189.

[11] A. Zelenak, M. Pryor, and K. Schroeder, "An Extended Kalman Filter for Collision Detection During Manipulator Contact Tasks," in *ASME 2014 Dynamic Systems and Control Conference*. American Society of Mechanical Engineers, 2014, pp. V001T11A005–V001T11A005.

[12] A. Cirillo, F. Ficuciello, C. Natale, S. Pirozzi, and L. Villani, "A Conformable Force/Tactile Skin for Physical Human–Robot Interaction," *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 41–48, 2016.

[13] P. Piacenza, Y. Xiao, S. Park, I. Kymissis, and M. Ciocarlie, "Contact Localization through Spatially OverlappingPiezoresistive Signals," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE, 2016, pp. 195–201.

[14] E. Magrini, F. Flacco, and A. De Luca, "Estimation of Contact Forces using a Virtual Force Sensor," in *Intelligent Robots and Systems (IROS), 2014 IEEE/RSJ Int. Conf. on*. IEEE, 2014, pp. 2126–2133.

[15] D. Popov, A. Klimchik, and N. Mavridis, "Collision Detection, Localization & Classification for Industrial Robots with Joint Torque Sensors," in *2017 26th IEEE Int. Symposium on Robot and Human Interactive Communication (RO-MAN)*, Aug 2017, pp. 838–843.

[16] S. K. Ralph and D. K. Pai, "Detection and Localization of Unmodeled Manipulator Collisions," in *Intelligent Robots and Systems (IROS). 'Human Robot Interaction and Cooperative Robots', 1995 IEEE/RSJ International Conference on*, vol. 2, Aug 1995, pp. 504–509 vol.2.

[17] M. Botsch, S. Steinberg, S. Bischoff, and L. Kobbelt, "OpenMesh – a generic and efficient polygon mesh data structure," in *1st OpenSG Symposium*, 2002.

[18] RWTH Aachen University, "OpenMesh," https://www.openmesh.org/, 2019, [Online; accessed 7 July 2019].

[19] G. Buondonno and A. De Luca, "Combining Real and Virtual Sensors for Measuring Interaction Forces and Moments Acting on a Robot," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE, 2016, pp. 794–800.

[20] R. Hanten, C. Schulz, A. Zwiener, and A. Zell, "MuSe: Multi-Sensor Integration Strategies Applied to Sequential Monte Carlo Methods," in *Intelligent Robots and Systems (IROS), 2019 IEEE/RSJ International Conference on*, 2019, (accepted for publication).

[21] D. Fox, "Adapting the sample size in particle filters through kld-sampling." *I. J. Robotic Res.*, vol. 22, no. 12, pp. 985–1004, 2003.