Collision Detection and Isolation on a Robot using Joint Torque Sensing

Joao Bimbo[1], Claudio Pacchierotti[2], Nikos G. Tsagarakis[1], and Domenico Prattichizzo[1,3]

***Abstract*— As robotic systems become more flexible and intelligent, they must be able to move into environments with a high degree of uncertainty or clutter, such as our homes, workplaces, and the outdoors. In these unstructured scenarios, it is possible that the body of the robot collides with its surroundings. As such, it would be desirable to characterise these contacts in terms of their location and interaction forces. This paper addresses the problem of detecting and isolating collisions between a robotic manipulator and its environment, using only on-board joint torque and position sensing. The algorithm is based on a particle filter and, under some assumptions, is able to identify the contact location anywhere on the robot body. It requires the robot to perform small exploratory movements, progressively integrating the new sensing information through a Bayesian framework. The approach was tested and benchmarked in simulation, with respect to its accuracy and robustness. Validation using a robot with joint torque sensing in a real environment demonstrated the applicability of the method to real-world scenarios.**

I. INTRODUCTION

A. Motivation

Going beyond the factory floor, robotic systems are nowadays addressing environments which are more and more unstructured, unpredictable, and cluttered. In these situations, collisions between parts of the robot and the environment are likely to occur. Indeed, cluttered and unstructured environments present several challenges, and extremely advanced perception capabilities as well as safety measures are required. Detecting collisions between the robot and its environment and immediately take appropriate safety measures is of the utmost importance when the robot is in close proximity with humans. However, in other circumstances it might be useful to explore and characterise the collision. In teleoperation settings, informing the user about the location of a contact between the robot and its environment might help him choose an appropriate action [1]. Other fields which rely strongly on contacts between the robot and its environment, such as grasping and locomotion, could greatly benefit from an accurate estimate of the contact location anywhere on the robot body. These capabilities can be useful in other fields such as the agri-food industry and search and rescue robots, where robots operate in highly unstructured and unpredictable environments.

This research was funded by the European Union's Horizon 2020 Research and Innovation Programme, grant n. 645599 (SOMA).

[1] J. Bimbo, N. Tsagarakis and D. Prattichizzo are with the Istituto Italiano di Tecnologia, Italy.

[2] C. Pacchierotti is with the CNRS, Univ Rennes, Inria, IRISA, France.

[3] D. Prattichizzo is also with the University of Siena, Italy.

{joao.bimbo, nikos.tsagarakis, domenico.prattichizzo}@iit.it

claudio.pacchierotti@irisa.fr

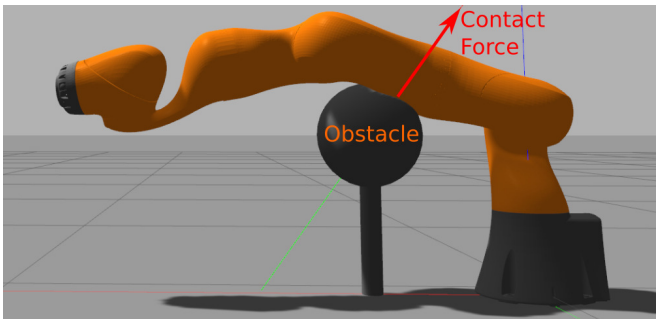


Fig. 1. Gazebo simulator environment with a 7-DoF Kuka iiwa robot and an object in collision.

This paper addresses an issue which is present in many autonomous and teleoperated systems, specially those working in cluttered environments: detecting the location of collisions happening between a robot and its environment at any point of the robot, using only sensors that are already available in many robot arms, i.e., position and torque. This problem is known in the literature as collision detection and isolation.

B. Related Work

The problem of characterising a collision between a robot and the environment has been addressed in multiple ways, most of which employ additional sensing. An immediate way of obtaining the contact location and magnitude of the external force is to cover the robot body with tactile sensors [2], [3], [4]. These can directly provide information about the collision location and the magnitude of the applied external force. This approach, besides costly, suffers from a number of drawbacks, such as the additional complexity, loosening, and the wear-and-tear of the tactile sensor surfaces. Another solution is to use vision systems to compute distances between the robot and its environment or between the robot and a user. External cameras can be used for collision avoidance [5], for safe Human-Robot cooperation [6], and to obtain the contact location [7], which is then used to compute and handle the contact forces exchanged with a user. A force-torque sensor mounted at the robot base also enables this computation [8] if a pure force is assumed.

Out of the approaches that rely solely on the robot's proprioception, the work of Alessandro De Luca stands out as the source for most of the major contributions to the field [5], [7], [8], [9], [10], [11], [12], [13]. Climent Bigas [14] and Manuelli [15], [16] proposed probabilistic methods to address this problem. These approaches are considered in the next section.

C. Problem Presentation

This paper presents a method to estimate the contact state of a serial robot manipulator with its environment using only the robot's proprioceptive sensors (joint torque and position). The problem of finding the contact location is referred to in the literature as "collision isolation" [13]. Without any additional external sensing this problem is not trivially solved, particularly when the contact happens near the base of the robot, i.e., when few joints are involved. In fact, Buondonno and De Luca [8] assert that a solution exists only if the contact is occurring after the sixth joint. This case allows the calculation of the external wrench, from which the collision point can be calculated by assuming an acting pure force, in a scheme similar to *intrinsic* contact sensing [17]. For what concerns detecting contacts before the sixth link, most of the existing literature [13] is limited to detecting *the link* in collision without estimating the exact contact point [9]. This is usually done by measuring the external joint torques and identifying the last joint to have a non-zero additional torque. However, this approach may still lead to errors, as a contact force may exert zero torque at the joint(s) immediately before its point of contact. This information is nevertheless sufficient in most cases where safety is the only concern, since the robot can "bounce" away from the collision [10] or continue, to some extent, performing the desired task [18]. Existing approaches that rely on probabilistic methods can overcome some of the limitations above. Climent Bigas [14] implemented a probabilistic function to find probable contact locations on a robot body using torque measurements. He found that the information is often insufficient to fully determine the collision point, and depends on the contact point and configuration of the robot. Manuelli and Tedrake [15] used a particle filter to isolate collisions on the body of a humanoid robot. They solved a quadratic problem to determine whether a force at particular contact point and inside a friction cone could produce the measured torque. The proposed method is fast and accurate, but suffers from the same drawback as [14]: it is not able to detect the correct contact point when several combinations of contact locations and forces could be responsible for the same observed external torque. This problem is referred to as the *identifiability* of the method. Contacts happening near the robot's base produce forces that generate torques on only a few of the robot's sensors. This means that there are often several possible contact locations that satisfy the (limited) available information. This prevents the isolation of the contact point on the links closer to the robot base.

This paper shows that this limitation derives from not fully exploiting the capabilities of Bayesian Filtering, namely with respect to the chosen motion model. We propose a particle filter approach that, under some assumptions, is able to identify the contact location anywhere in the robot body. It requires the robot to perform small exploratory movements, progressively integrating the new sensing information using a Bayesian framework.

II. METHODS

A. Particle Filtering

A Particle Filter is a non-parametric Bayesian Filter that approximates a distribution of states by a set of particles. Each particle represents a possible state of the system. Modelling how the measurements match the state of the system and how these states change over time given a control action allows us to recursively narrow down the possible states until a single solution is found. This technique relies on the Markov property of a system, i.e., the current state depends only on the previous state and the last action, making the problem computationally tractable.

The implementation of a Bayesian Filter requires the following models:

- A measurement model that describes the likelihood of a measurement $\boldsymbol{z}$ given a state $\boldsymbol{x}$,

$$p\left(\boldsymbol{z}(t) \mid \boldsymbol{x}(t)\right). \tag{1}$$

- A motion model (also known as update or actuation model), that describes the probability of transition to another state given the action performed $u(t)$,

$$p(\boldsymbol{x}(t) \mid u_t, \boldsymbol{x}(t-1)). \tag{2}$$

The posterior belief of each state given all the previous measurements and actions can then be obtained using the recursion formula:

$$\begin{aligned} \overline{\text{bel}}(\boldsymbol{x}_t) &= \int p(\boldsymbol{x}_t | u_t, \boldsymbol{x}_{t-1})\ \text{bel}(\boldsymbol{x}_{t-1}) d\boldsymbol{x}_{t-1} \\ \text{bel}(\boldsymbol{x}_t) &= \eta \cdot p(\boldsymbol{z}_t \mid \boldsymbol{x}_t)\ \overline{\text{bel}}(\boldsymbol{x}_t) \end{aligned} \tag{3}$$

where $\boldsymbol{x}_t = \boldsymbol{x}(t)$, and η is a constant which normalises the beliefs such that they integrate to 1.

The Particle Filter presents a discretised version of the Bayesian Filter. Additionally, this filtering technique has a resampling step, where the weight of each particle is updated according to its likelihood, and a new set of particles is generated by resampling the existing particles according to their weight. Step by step, this allows the survival of the most likely estimates at the expense of the ones less probable.

B. Contact estimation

The contact state $\boldsymbol{x}$ that we intend to characterise consists of the point of contact $\vec{p}$, the contact force $\vec{F}$, and the stiffness of the contact K. In this work, we assume that the object in collision is convex, that there is a single contact region, and that the friction forces are negligible. We start by retrieving the triangular mesh describing the geometry of each link and calculate the normals of each triangle. Besides, the points generating the mesh are used to build a k-d tree to allow fast nearest neighbour searches [19]. We randomly generate a high number of particles ($N = 10000$), where each particle defines a possible collision state. It contains the link number n, the index of the mesh triangle p, the force magnitude F, and the linear stiffness of the contact K.

1) Measurement Model: We are interested in estimating the *external torques* $\boldsymbol{\tau}_{ext}$, which result from the interaction of the robot with the collided object. Typically, these torques can be obtained from the manipulator equation:

$$\boldsymbol{\tau}_{ext} = \boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) - \boldsymbol{\tau}_m, \tag{4}$$

where $\boldsymbol{M}(\boldsymbol{q})$ is the inertia matrix, $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}$ is the centripetal and Coriolis vector, and $\boldsymbol{g}(\boldsymbol{q})$ is the gravity vector. Note that other methods of estimating the external torque $\boldsymbol{\tau}_{ext}$ exist. Most notably, the residual method in [10] uses a generalised momentum observer, which avoids the typically noisy estimation of joint acceleration $\ddot{q}$ and also works when $\dot{q} = 0$. In this work, since we are collecting measurements when the robot is in contact, we assume that $\ddot{q} \approx 0$. For further details, and other approaches to estimate $\boldsymbol{\tau}_{ext}$ see the survey in [13].

Given the array of measured external torques $\boldsymbol{\tau}_{ext}$, a likelihood function can be constructed from the difference between the measured torques and the torques $\hat{\boldsymbol{\tau}}$, estimated from the contact state $\boldsymbol{x}$,

$$p(\boldsymbol{z}_t \mid \boldsymbol{x}_t) = e^{-\alpha \|\hat{\boldsymbol{\tau}} - \boldsymbol{\tau}_{ext}\|}, \tag{5}$$

where α is a design parameter that relates with the accuracy of the torque sensing and also how "greedy" we want the search to be. The estimated torques $\hat{\boldsymbol{\tau}}$ can be obtained using the robot Jacobian, calculated for contact point $\vec{p}$ and current joint configuration of the robot as

$$\hat{\boldsymbol{\tau}} = \boldsymbol{J}_{\vec{p}}^T(\boldsymbol{q}) \cdot \boldsymbol{F_c}, \tag{6}$$

where $\boldsymbol{F_c} \in \mathbb{R}^{6\times 1}$ is the coordinate vector obtained from the contact force F and the normal direction $\hat{n}$,

$$F_c = \begin{bmatrix} F \cdot \hat{n} \\ \mathbf{0} \end{bmatrix}^T. \tag{7}$$

Another simpler way of calculating joint torques is through the cross product of vectors ${}^i\vec{p}$ and ${}^i\vec{F}$ in the coordinate frame of the i-th joint, projected onto the joint axis.

2) Motion Model: The motion or update model is the function that allows us to predict how the state will change given a control action, i.e., $p(\boldsymbol{x}(t) \mid u_t, \boldsymbol{x}(t-1))$.

Consider a contact state $x_t = \{n_t, \vec{p}_t, \vec{F}_t, K_t\}$, where n is the link in contact, $\vec{p}$ is the contact position, $\vec{F}$ is the contact force and K is the contact stiffness at time t. The control action is encoded by the kinematic postures of the robot at times $t-1$ and t. Here we take advantage of the assumptions that the surfaces in contact are locally convex and that the friction forces are negligible. Taking contact points p_{t-1} and p_t , we assume that there is a constant curvature along the plane that passes through the two points and contains $\vec{F}_{t-1}$. We sample a curvature radius r from a distribution uniform in the logarithmic space, and we obtain the centre of the curve c. p_t is obtained from a nearest neighbour search from point c, and the force is obtained from

$$F_t = F_{t-1} + (r - d_t) \cdot K_t, \tag{8}$$

where d_t is the distance $\|\vec{p_t} - \vec{c}\,\|$.

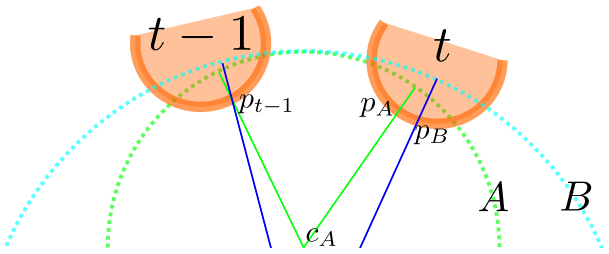


Fig. 2. Reasoning for the motion model. As the robot (in orange) moves between $t-1$ and t instants, the possible contact locations between the robot and surfaces A and B are limited if the surfaces are convex. The new contact locations p_A and p_B will depend on the shape's curvatures. Given enough particles and sampled curvatures, this enables the algorithm to narrow down the right point of collision at time t.

Figure 2 illustrates the method and shows the result for two example curvatures (A and B). For surface A, the centre of the curve is at point c_A and the contact point $p_t = p_A$. Surface B will have a different centre, and the contact point will change accordingly, i.e., $p_t = p_B$. Forces are calculated according to the distance between p_t and the curve, multiplied by K. Finally, additional Gaussian noise $w_i \sim \mathcal{N}(0, \sigma_i^2)$ was added to the particle's location, force, and stiffness.

The effectiveness of Particle Filters lies on a correct formulation of the motion model. Consider an example commonly used to explain Particle Filters [20], where the robot must locate itself in a map. If each measurement is considered individually, it is impossible to identify on which of the similar corridors the robot is currently at. Only through the correct updating of the particles, following a model of how the robot moves, all the possible robot locations can be narrowed down to a single point. Similarly, in this paper, only through a model of how the contact point and forces change when the robot performs small movements, the state can be correctly estimated.

Below, we compare the results of the proposed approach with two other motion models. One where the contact has zero velocity on the link frame, and another which assumes that the point of contact does not change with respect to an external frame.

III. RESULTS

A. System overview and implementation

The proposed method was implemented in C++ using the ROS platform [21]. Robot dynamics were calculated using the KDL library [22], and the geometric operations relied on the PCL (Point Cloud Library) [23]. The tested robot was a Kuka LBR iiwa, a 7 degree-of-freedom robot with joint torque sensing. Simulations were carried out using the Gazebo robot simulator [24].

B. Simulation

The simulated environment is shown in Fig. 1. The robot was made collide with a spherical object at different points. Ground truth values for contact location and force are obtained from the simulation allowing a quantitative evaluation of the algorithm.

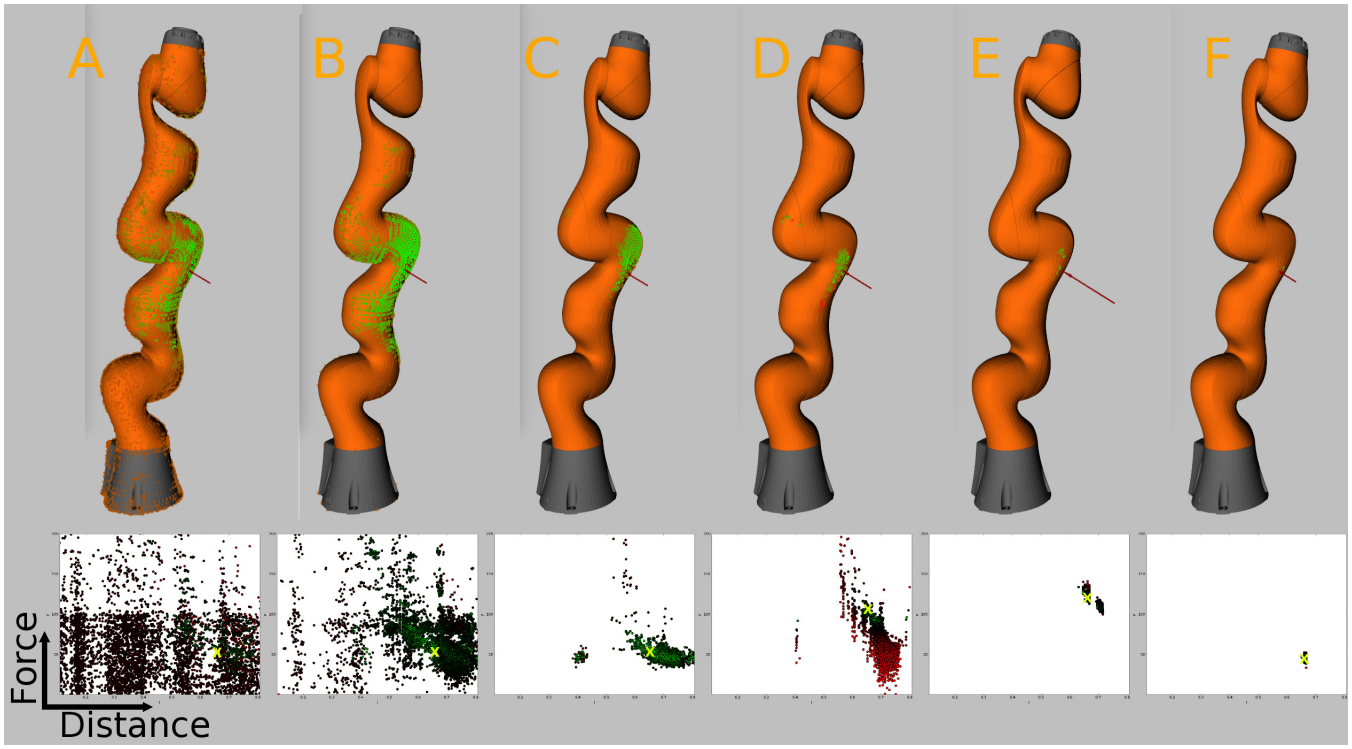


Fig. 3. Progress of the algorithm for a collision on the third link. Upper row: likely particles on the robot body. The red arrow shows the true contact point and force vector. Lower row: particle distribution, plot of contact force vs. distance to base. Higher likelihoods in green, lower in red. Ground truth is represented by the yellow cross.

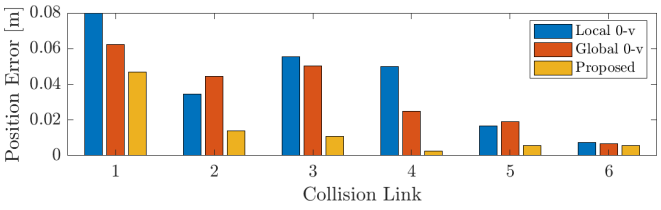


Fig. 4. Estimation accuracy of the contact location algorithm for different motion models. Mean position error for each link is plotted. Blue: zero-velocity on the link frame [15]. Red: zero-velocity on the global frame [15]. Yellow: method proposed in this paper.

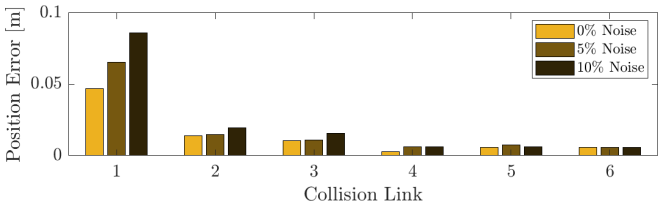


Fig. 5. Performance of the algorithm when adding noise to the torque measurements. Mean position error on each link for added Gaussian noises with standard deviation of 0%, 5%, and 10%. These results show a good robustness to noise.

As the robot contacted the object, the estimation process was started, setting a low value to α in (5) and a high standard deviation for the noise σ_i in order to cover the search space. As the robot starts moving, we increase α and decrease σ_i in order to narrow down the particles towards the correct solution. Figure 3 shows the progress of the algorithm for a contact happening at the third link of the robot. Initially, the particles with high likelihood (green dots) are scattered around links 2, 3, and also some on link 4. This is because there are possible forces at these locations that could generate the measured torque, requiring for instance a large force at link 2 or a lower force at link 4. The robot then makes small exploratory movements, narrowing down the possible contact locations. The bottom row of Fig. 3 shows the estimated force vs contact location. The small movements of the robots can be understood by the changes in the length of the true contact force (red arrow). See for example the decrease in contact force between the times E and F after a movement of the robot. These changes are correctly predicted by the particle filter, as can be seen from the shift of particles towards lower forces in last plot.

We compared the approach proposed in this paper with two alternative models described in [15]. One assumed that the contact location would remain approximately constant on the link frame, while the other one supposed that it would remain constant on a global coordinate frame.

We considered collisions at each robot link. As a metric, we evaluated the error between the estimated position of collision and the true one. Five trials were performed for each link and results are shown in Fig. 4. While there is no significant difference when collisions are happening on the sixth link, the advantages of the proposed approach become more and more visible as the contact takes place nearer to the robot base and less measures are available. For example,

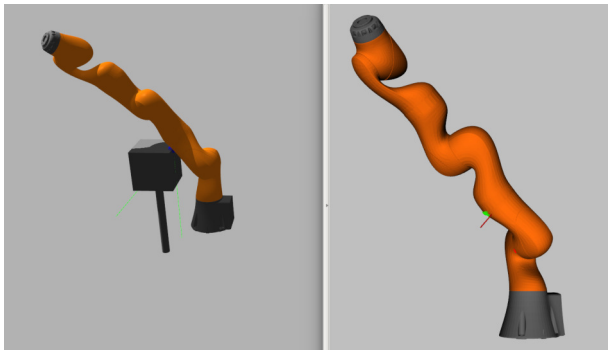


Fig. 6. Results in simulation when the robot is contacting a cubic object. Left: Simulated environment Right: Result of estimation (green dots show likely contact points).

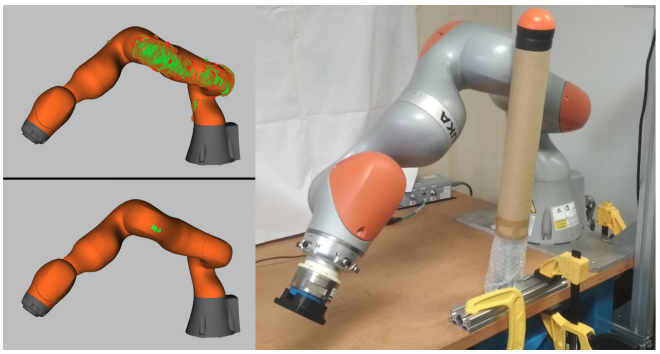


Fig. 7. Results in the real-world environment. Top-left: initial steps of the estimation before moving. Bottom-left: result of the estimation. Right: collision state (the robot's third link is colliding with a cylindrical obstacle).

for a contact at link 3, the error of the proposed algorithm is around 1 cm, while the other approaches' exceed 5 cm. For collisions that happen on links 5 and 6, the motion model becomes less important, since these may only require a single measurement to unambiguously compute the contact location and force.

With respect to robustness to noise, Fig. 5 shows the result when Gaussian error is added to the torque measurements. A slight decrease in accuracy is visible when the standard deviation of noise is 10% of the measured signal, particularly for link 1. The effect of noise is mitigated for contacts happening further down the kinematic chain, achieving mean errors below 2 cm for every link and noise level after link 1. This result is expected and in line with the known ability of particle filters to deal with noisy measurements. Another result that is worth highlighting is the ability of the algorithm to deal with shapes which are not strictly convex. Figure 6 shows an example of a cubic obstacle, in contact with the robot at its edge. The system is still able to locate the contact with remarkable accuracy.

C. Real System

The results in simulation indicate that this method is suitable to be applied on a real robotic system with joint torque sensing capabilities. While the correct convergence of the algorithm requires accurate measurements and a good knowledge of the robot's dynamical properties, the method is, to some extent, robust to uncertainties.

An experimental trial was performed using a real Kuka LBR iiwa robot, and results are shown in Fig. 7. The robot collides laterally with a cylindrical pole. Initially, many possible contact locations along that side of the robot have similar likelihood. The robot is then commanded to perform small exploratory movements within the robot's null space, while maintaining contact with the obstacle. Although quantitative evaluation was not possible, we can see that the algorithm converges to the correct location (green dots on the lower left side of the figure). Other objects with different shapes and materials were tested, achieving similar results.

IV. CONCLUSIONS

A. Discussion

Results show that Bayesian Filtering can be used to estimate the contact location between a robot and an object anywhere on the robot body, under a few assumptions. These results showcase the effectiveness of particle filtering when used to estimate a state that is not observable from a single measurement. Having a model, even an approximate one, of how the state evolves given control actions allows us to sequentially narrow down likely states until a solution is found. In this paper we take advantage of simple yet useful concepts to increase the likelihood of finding the right contact location, such as (i) moving against a collision force is likely to increase it; (ii) collisions are typically not point contacts but surfaces; (iii) if they are convex and the robot link moves to the left the contact will probably move to its right. In fact, the Contact Particle Filter reported in [15], [16] uses no such model/assumptions and performs poorly when the problem is not identifiable. Using the same Kuka robot, it reports errors between 10 cm and 30 cm for contacts in links 1, 2, and 3. An additional force-torque sensor was added to cope with these cases. Another remarkable strength of particle filters lies in its "forgiveness" to noisy measurements, errors in the model, and simplified assumptions. This effect can be seen in the method's ability to detect the contact location using a real robot. In our experiment with the real robot, we used an imperfect model and noise was obviously present in the torque data. Most of the assumptions also did not hold: the object was not strictly convex, its stiffness was not linear and there was some friction; however, the estimation converged to an approximately correct result.

Nevertheless, the proposed approach has a significant limitation. If the robot collides with a concave object, there is no way of predicting the behaviour of the contact point as well as that of the forces. Consider, for example that the robot arm is inside a narrow tube. Any small movement by the robot might lead to moving the contact to the opposite side of the robot. This might render the method unable to find the contact location when the problem is not identifiable from a single measurement.

B. Concluding Remarks

This paper presents a probabilistic method to determine the contact location between a serial manipulator and an object in its environment. It uses only the robot's joint position and torque sensing, and it uses small exploratory movements to recursively estimate the location, force, and stiffness of the contact. The method assumes negligible friction forces, convex contact surfaces, and linear contact stiffness. Compared to existing approaches, it allows this detection to be carried in almost all the surface of the robot's body. We tested the proposed approach both in simulation and in a real environment. Experiments in simulation showed that our approach outperformed two other methods that made simpler assumptions. Experiments in a real environment using a robot with joint torque sensors showed the applicability of the method to real world scenarios and its ability to cope with situations where the algorithm's assumptions did not hold.

C. Future Work

Future work will focus on the actions of the robot when contact is detected. While in this work the exploratory movements of the robot were carried out manually, either arbitrarily or in the robot's null space, we would like to explore the different possibilities in terms of what movements can or should be performed. While some movements will allow a faster identification of the contact location, they might be dangerous (e.g., moving against the contact force). This demands for a trade-off between informative and safe actions that is worth being investigated. Another possibility is to use a similar method to blindly create a map of the robot's environment, and then plan the robot's movement within that sensed map. Finally, a possible research direction is to improve the current method to include friction and also the joint stiffness of the robot.

References

[1] J. Bimbo, C. Pacchierotti, M. Aggravi, N. Tsagarakis, and D. Prattichizzo, "Teleoperation in cluttered environments using wearable haptic feedback," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2017.

[2] P. M. Grice, M. D. Killpack, A. Jain, S. Vaish, J. Hawke, and C. C. Kemp, "Whole-arm tactile sensing for beneficial and acceptable contact during robotic assistance," in *Proc. IEEE International Conference on Rehabilitation Robotics*, 2013, pp. 1–8.

[3] 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.

[4] 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.

[5] F. Flacco, T. Kröger, A. De Luca, and O. Khatib, "A depth space approach to human-robot collision avoidance," in *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2012, pp. 338–345.

[6] D. M. Ebert and D. D. Henrich, "Safe human-robot-cooperation: image-based collision detection for industrial robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 2, Sep. 2002, pp. 1826–1831 vol.2.

[7] E. Magrini, F. Flacco, and A. De Luca, "Estimation of contact forces using a virtual force sensor," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2014, pp. 2126–2133.

[8] G. Buondonno and A. De Luca, "Combining real and virtual sensors for measuring interaction forces and moments acting on a robot," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2016, pp. 794–800.

[9] A. De Luca and R. Mattone, "Sensorless robot collision detection and hybrid force/motion control," in *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2005, pp. 999–1004.

[10] A. De Luca, A. Albu-Schaffer, S. Haddadin, and G. Hirzinger, "Collision detection and safe reaction with the DLR-III lightweight manipulator arm," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2006, pp. 1623–1630.

[11] S. Haddadin, A. Albu-Schaffer, A. De Luca, and G. Hirzinger, "Collision detection and reaction: A contribution to safe physical human-robot interaction," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2008, pp. 3356–3363.

[12] A. De Santis, B. Siciliano, A. De Luca, and A. Bicchi, "An atlas of physical human–robot interaction," *Mechanism and Machine Theory*, vol. 43, no. 3, pp. 253–270, 2008.

[13] S. Haddadin, A. De Luca, and A. Albu-Schaffer, "Robot Collisions: A Survey on Detection, Isolation, and Identification," *IEEE Transactions on Robotics*, vol. 33, no. 6, pp. 1292–1312, dec 2017.

[14] A. Climent Bigas, "Proprioception for collision detection," Master's thesis, Universitat Politècnica de Catalunya, 2014.

[15] L. Manuelli and R. Tedrake, "Localizing external contact using proprioceptive sensors: The contact particle filter," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2016, pp. 5062–5069.

[16] L. Manuelli, "Localizing external contact using proprioceptive sensors: The contact particle filter," Master's thesis, MIT, 2018.

[17] A. Bicchi, J. K. Salisbury, and D. L. Brock, "Contact Sensing from Force Measurements," *The International Journal of Robotics Research*, vol. 12, no. 3, pp. 249–262, 1993.

[18] A. De Luca and L. Ferrajoli, "Exploiting robot redundancy in collision detection and reaction," *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pp. 3299–3305, 2008.

[19] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[20] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[21] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[22] R. Smits, "KDL: Kinematics and Dynamics Library," orocos.org/kdl.

[23] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2011.

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