2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Closing the Loop: Real-Time Perception and Control for Robust Collision Avoidance with Occluded Obstacles

Andreea Tulbure Oussama Khatib

***Abstract*— Robots have been successfully used in well-structured and deterministic environments, but they are still unable to function in unstructured environments mainly because of missing reliable real-time systems that integrate perception and control. In this paper, we close the loop between perception and control for real-time obstacle avoidance by introducing a new robust perception algorithm and a new collision avoidance strategy, which combines local artificial potential fields with global elastic planning to maintain the convergence towards the goal. We evaluate our new approach in real-world experiments using a Franka Panda robot and show that it is able to robustly avoid dynamic or even partially occluded obstacles while performing position or path following tasks.**

I. Introduction

Enabling robots to interact with humans is a key challenge in robotics, as they will be required to safely work in unstructured and highly dynamic environments. One of the main aspects in reaching this goal is the robust avoidance of collisions, which relies on external environmental information and therefore, real-time perception is essential. However, although highly important, the integration of real-time perception in a control framework in order to close the loop between these two components is still an open problem, especially when working with high-dimensional robotic manipulators.

In this work, we introduce a real-time collision avoidance framework that is able to avoid partially occluded obstacles (Fig.1), robustly closing the loop between perception and control, without needing specialized hardware. Considering this, our contributions are: (1) a robust real-time perception algorithm that handles small occlusions and multiple obstacles and generates continuous artificial potential fields from point cloud data; (2) a new collision avoidance algorithm for a high-dimensional multi-link system combining local artificial potential fields with global elastic planning to maintain the convergence to the goal in the workspace.

A. Related Work

Due to its key importance, collision avoidance is one of the most studied fields in robotics. The seminal work in collision avoidance introduces virtual attractive and repulsive fields associated to the goal and obstacles respectively, called artificial potential fields (APF) [1]. As a result obstacles are avoided while the robot keeps moving to the goal. However, the APF has a local, reactive behavior and does not guarantee global convergence towards the goal.

The authors are with Stanford Robotics Lab, 353 Serra Mall, 94305, Stanford, CA, USA. This work is supported partially by Baden Wuerttemberg Stiftung e.V. and Toyota Research Institute (TRI).
{tulbure@mavt.ethz.ch, ok@cs.stanford.edu}

Fig. 1: Robot avoiding collision with a partially occluded obstacle

Warren et al. [2] and Ogren et al. [3] further develop the initial idea of artificial potential fields, while others introduce modified potential fields such as circular fields [4] or harmonic potential fields [5]. Khansari et al. [6] introduce an approach similar to harmonic potential fields, while Haddadin et al. [7] present an extension of the circular fields which is not subject to local minima.

The global convergence aspect was addressed also using elastic planning methods [8], [9], [10]. The originally computed collision-free path is updated in real-time to adapt to the dynamic environment, assuring collision-free paths at all times. A roadmap with collision-free edges and vertices ("milestones") is used to represent the planned trajectories in [11].

A large number of works in collision avoidance consider mobile robots [12, 1], not high-dimensional multi-link systems, such as robotic manipulators or humanoids. One work tackling dynamic collision avoidance for a multi-link industrial robot using artificial force fields is presented in [13]. However, this method is not suitable to be applied in a real-world scenario, as the CAD model of the obstacle is needed and a second robot holds the obstacle to get position information. Some authors use optimization-based controllers to tackle the collision avoidance problem for multi-link systems [14],[15]. Merkt et al. [15] introduce a continuous-time collision avoidance term which can be used as a cost or a constraint in a non-linear optimization problem. Static and dynamic obstacles can be handled with this approach. Fang et al. [14] tackles self-collision avoidance by formulating the it constraint as an inequality constraint for each task in a Stack-of-Task control framework [16].

However, most of the existing works assume that information about the environment is known, skipping perception, which is essential in real-world scenarios. Therefore, recent studies focus on integrating perception, collision avoidance and control for robot manipulators [17],[18],[19], [20]. Flacco et al. [17] [18] introduce a depth space distance computation method to generate repulsive forces. To compensate for noisy point clouds the authors average over all the

978-1-7281-6211-9/20/$31.00 ©2020 IEEE

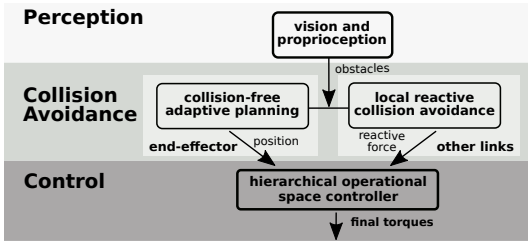

Fig. 2: Architecture of our fully integrated real-time collision avoidance framework.

obstacle points in a "region of surveillance" when computing the direction while using only the minimum distance value for the magnitude of the force. In [21] a GPU is used for voxel map generation from 3D point clouds and distance computation for collision avoidance. Chen et al. [19] propose an approach based on attractive and repulsive potential fields in which the robot can preserve the original end-effector task while avoiding collisions with the robots body by computing a compensation velocity at the joint level. Di Castro et al. [20] introduce a reconfigurable collision avoidance approach, which can avoid also self-collisions.

This work is related to the ones above, but has two significant differences: (1) our perception algorithm models the obstacles as convex meshes and therefore reduces the possibility of getting stuck in local minima and can handle small occlusions; (2) we introduce a new collision avoidance strategy combining global elastic planning with reactive APF for robotic manipulators to assure convergence to the goal.

B. System architecture

As a typical real-time collision avoidance framework, our system has 3 modules [17]: (1) Perception, (2) Collision avoidance and (3) Controller. The structure is presented in Fig. 2. Our perception module contains a depth camera and the proprioceptive sensors of the robot. Our collision avoidance module combines global elastic planning for creating a collision-free end-effector path with local reactive obstacle avoidance, i.e. APF, for the other links. Our controller is a hierarchical operational space controller.

II. Perception

The perception is the first module in a real-time collision avoidance framework and is necessary in order to enable the robots to act in real-world scenarios.

A. Obstacle detection and reconstruction

We are not interested in reconstructing the obstacles exactly, only to know which area should be avoided, therefore we model the obstacles as convex meshes with a fixed number of vertices. Before being able to reconstruct the obstacles, the raw point cloud is filtered and preprocessed. The points corresponding to the background and the robot are removed. The resulting point cloud is downsampled using a voxel grid filter which also reduces outliers. The filtered point cloud is used as input for the obstacle detection and reconstruction (Algorithm 1).

To increase robustness to noise we compare the current filtered point cloud to the previous one. If a certain difference threshold t is exceeded the new point cloud is processed and the obstacles are updated. The obstacles are detected using

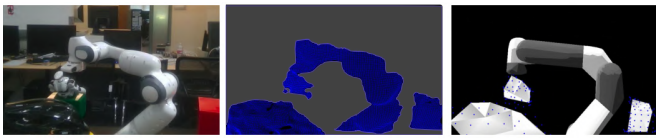

Fig. 3: Obstacle detection and modeling. $left$: original 2D image of the robot workspace; $middle$: filtered point cloud; $right$: reconstructed convex meshes of the obstacles.

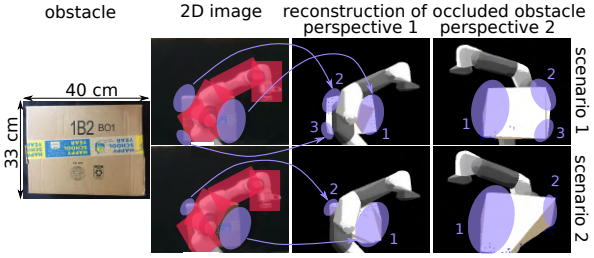

Fig. 4: Occluded obstacle reconstruction from point clouds. When the corners of an obstacle can be detected, the algorithm is able to reconstruct the partially occluded obstacle completely (scenario 1), otherwise just a part of the obstacle is reconstructed (scenario 2). As we model the robot using capsules (marked with red rectangles in the 2D pictures), we remove all the points inside those. This leads to the removal of some of the object corners which are visible in the 2D image and the initial point cloud, leading to an incomplete reconstruction, as it is the case in scenario 2. The point clusters and their corresponding part in the 2D images are marked with numbers and blue ovals.

Euclidean Clustering [22], such that every cluster represents an obstacle. To get the vertices of the convex mesh associated to the obstacle we compute the convex hull of the clusters. For efficiency purposes we limit the maximum number of vertices in a mesh to a certain number nr_v by clustering the convex hull points which are close using KMeans Clustering [23]. This results in a slightly lower obstacle shape accuracy, not leading to significant changes in the area that needs to be avoided and therefore in the overall performance. An example is shown in Fig. 3.

B. Handling occlusions

One main advantage of our detection and reconstruction algorithm is the fact that we can implicitly handle small occlusions. As long as the obstacle is not completely behind the robot, at least a part of it is reconstructed, as shown in Fig.4. This behavior is a result of adjusting the distance threshold parameter d_{ec} of the Euclidean Clustering [22]

Algorithm 1: Obstacle Detection and Reconstruction

Input: Filtered point cloud P ; threshold value t; maximum number of mesh vertices nr_v
Output: A vector $O = \{o_1 \dots o_n\}$ of obstacles

```
if difference(P, P_prev) > t then
    O ← ∅, P_prev ← P
    C = computeObstacleClusters()
    foreach cluster ∈ C do
        CH ← convexHullPoints(cluster)
        if CH.NumberPoints > nr_v then
            CH ← KMeansPointsOfCH()
        end
        O ← generateMeshesFromCHPoints()
    end
end
return O
```

algorithm, which defines the maximum distance between two points that belong to the same cluster. By setting a higher threshold, points further apart are considered to belong to the same cluster. This way small occlusions and even "holes" between two different obstacles which are close together and through which the robot would not fit can be handled. This means that traps, e.g. concave object settings through which the manipulator can not pass, can not occur, decreasing the probability of getting stuck in local minima.

III. Collision avoidance

We combine the elastic bands [8] global adaptive path planing algorithm for generating a collision-free end-effector path with local reactive reconfiguration using APF for the other links of the robot. This combination of local and global planning strategies is useful to avoid local minima along the way, as the elastic planning method keeps track of the global goal.

A. Reactive repulsive force computation for APF

All the obstacle vertices which are closer to the robot than a certain threshold d_0 are considered for reactive force computation and are said to be in the region of interest. To move the robot away from the obstacles we consider the center of mass (COM) of the links as control points and we compute one resulting repulsive force that acts on the COM of each link, as presented in Algorithm 2. The resulting direction of the repulsive force at each link is the sum of all the force directions acting on that link, while the resulting magnitude depends only on the minimum distance d_{min} between the obstacles and the link, according to:

$$F_{APF}(d_{min}) = \begin{cases} -\eta(\frac{1}{d_{min}} - \frac{1}{d_0})\frac{1}{d_{min}^2}, & d_{min} < d_0 \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

with $\eta = 0.35$. Otherwise, the force would be affected by the ratio of far and close points and the value of the closest distance would be mitigated.

1) Distance computation: As the APF approach depends on minimum distance values, smooth and continuous distances are required to generate potentials which push the robot smoothly away from the obstacles. To get such distances we filter out outliers by keeping track of the past distance and limiting the change at every time step using a specific "step" value (Algorithm 2,ln.28). As a result, if an outlier leads to a significant change in distance, then the value will change by this "step" value. As it is an outlier the change will not appear again. It is important to notice that this "step" value considers the maximum possible movement of an obstacle between two images and its role is to filter out possible outliers, not to limit the potential fields.

For efficient distance computation, we model the links of the robot as capsules, such that we compute point to line distances for every pair $[link, v]$ consisting of one robot's link and a reconstructed obstacle vertex (Algorithm 2,ln.6). However, when the obstacle is big the vertices are sparse. This can lead to bigger distances than the threshold d_0 between the robot link and its associated closest vertex on the obstacle v_{min}, even if the "true" distance to the obstacle d_{mesh} is smaller than d_0. More specifically, by "true" closest distance d_{mesh} between the link and obstacle o we denote the distance between the link and the triangle mesh m in the reconstructed obstacle o associated to the closest vertex v_{min}, which differs from the distance between the link and the closest vertex v_{min}. To overcome the presented problem, we keep track of the closest distance from every link to the associated closest obstacle vertex v_{min} even if the distance is greater than the threshold d_0 (Algorithm 2,ln.13). Finally, we compute the "true" closest distance for every link (Algorithm 2,ln.21) using the triangle mesh m associated to the closest vertex v_{min} for every robot link.

2) Force direction computation: Each repulsive force associated to an obstacle vertex acts on a point on a link associated to the minimum distance between the vertex and the link. The force is projected to the COM of the link where all repulsive forces coming from all vertices in a region of interest are summed up, resulting in one repulsive force per link. We use the closest distance d_{min} to compute the force magnitude (Algorithm 2,ln.29), while for the direction of the force we sum up all the force directions coming from points in the region of interest (including the direction coming from the "true" closest distance (d_{mesh})) (Algorithm 2,ln.17 and 23). Considering all points in a region of interest leads to a

Algorithm 2: Repulsive force computation

Input: A vector $O = \{o_1, o_2, \ldots, o_n\}$ containing the obstacles o with vertices v and triangle meshes m, robot model R and a distance threshold value d_0, a vector d_{prev} with previous distance values for each link

Output: A vector $F = \{f_1, \ldots f_n\}$ of forces that act on the COM of every link n; every force has a direction dir and magnitude mag component

```
1  F ← ∅
2  foreach link ∈ R do
3      initFlag ← true
4      foreach o ∈ O do
5          foreach v ∈ o do
6              d ← distance(link, v)
7              if initFlag then
8                  [v_min, d_min] ← update(d, v)
9                  initFlag ← false
10             end
11             else
12                 if d < d_min then
13                     [v_min, d_min] ← update(d, v)
14                 end
15             end
16             if d < d_0 then
17                 f.dir ← f.dir + direction(link_COM, v)
18             end
19         end
20     end
21     d_mesh ← distanceToMeshTriangleAt(v_min)
22     if d_mesh < d_0 then
23         f.dir ← f.dir + direction(link_COM, v_min)
24         if d_mesh < d_min then
25             d_min ← d_mesh
26         end
27     end
28     d_min ← limitDistance(d_prev[link])
29     f.mag ← F_APF(d_min)
30     F[link] ← [f.mag, f.dir]
31     d_prev[link] ← d_min
32 end
33 return F
```

smoother variation of the direction, being more robust against noise. Moreover, this way several obstacles can be handled as all the points contribute to the direction of the repulsive vector.

As the number of vertices in the obstacle is usually constant and limited to nr_v, the introduced algorithm's complexity depends only on the number of obstacles and the robot's links: $\mathcal{O}(olink)$.

B. Elastic bands

The elastic bands framework [8] addresses adaptive path modification under the influence of dynamic obstacles, which generate repulsive forces. The original end-effector path can be planned by a global offline path planner, which is then modified in real-time, under the exertion of external and internal forces, leading to similar properties as elastic bands. The external repulsive forces F_p^{ext} acting on the path at point p are necessary to keep the path collision-free. These are generated by:

$$F_p^{ext} = k_r \cdot (d_0 - d(p)) \cdot \frac{d(p)}{||d(p)||} \tag{2}$$

where $k_r = 0.6$ is the repulsive gain, d_0 a the threshold distance value and $d(p)$ is the closest distance from point p to the obstacle. This is computed as in the previous section, with p as control point.

The internal contraction forces are needed to shorten the path in case an obstacle recedes and are modeled as virtual links between two consecutive path setpoints p^i and p^{i+1}, defined by:

$$F_i^{int} = k_c \cdot (\frac{d^{i-1}}{d^{i-1} + d^i} \cdot (p^{i+1} - p^{i-1}) - (p^i - p^{i-1})) \tag{3}$$

where $k_c = 1.0$ is the contraction gain, $d^i = ||p^i - p^{i+1}||$ in the initial path and the scaling factor $\frac{d^{i-1}}{d^{i-1}+d^i}$ assures that the relative distance between two adjacent setpoints along the path is maintained.

By summing up the contraction and repulsive forces at each point p, the initial path is updated such that it is collision-free and with minimum possible deviation.

IV. Control strategy

We use the hierarchical operational space control formulation [24], which extends the operational space formulation to allow multiple tasks with different priorities. As depicted in Fig.5 the prioritized control hierarchy is structured into three different categories: constraint handling, operational and posture tasks. Inside these categories the hierarchy is kept accordingly, such that e.g. in the operational task category the position has a higher priority compared to the orientation.

First, we will briefly state the operational space formulation [25]. The equation of motion for a robot in free space is:

$$A(q)\ddot{q} + b(q, \dot{q}) + g(q) = \Gamma \tag{4}$$

where $\ddot{q}$ represents joint accelerations, $A(q)$ represents the robot Mass Matrix, $b(q, \dot{q})$ contains the effect of centrifugal and Coriolis forces, $g(q)$ is the gravity, and Γ is the commanded motor torques. For a task i the operational space equation of motion is obtained by multiplying the above equation by the transpose of the dynamically consistent inverse of the task Jacobian $\bar{J}_i = A^{-1}J_i^T\Lambda_i$ (with $\Lambda_i = (J_iA^{-1}J_i^T)^{-1}$ being the task inertia matrix) leading to:

$$\Lambda_i\ddot{x} + \mu_i + p_i = F_i \tag{5}$$

where F_i are the control forces for task i, μ_i are the Coriolis and centrifugal forces in the task space and p_i is the gravity projected on the task. Having perfect estimates $\hat{\Lambda}_i$, $\hat{\mu}_i$, $\hat{p}_i$ and choosing $F_t = \hat{\Lambda}_i F^* + \hat{\mu}_i + \hat{p}_i$, we get $I\ddot{x}_i = F^*$ where I is the identity matrix and F^* is the desired unit mass control force. As a result the task control torques can be written as $\Gamma_i = J_i^T F_i$.

To maintain task consistency in our controller we project the control torques of the lower priority task i into the nullspace of the higher priority task $i-1$ using the dynamically consistent nullspace matrix $N_{i-1} = I - \bar{J}_{i-1}J_{i-1}$. From now, the symbol $|$ is used to denote this consistency.

On the torque-level the control strategy is embodied using:

$$\Gamma = \Gamma_c + N_c^T(\Gamma_t + N_t^T\Gamma_p) \tag{6}$$

The torques Γ_c, Γ_t and Γ_p are associated with the constraints, operational task and posture torques, as shown in Fig.5. The hierarchy holds also inside the task category: e.g. for the operational task the torques are generated taking into account that the position task has a higher priority than the orientation task, as $\Gamma_t = \Gamma_{position} + N_{position}^T\Gamma_{orientation}$.

In the following, we will describe the constraint handling and switching strategy, but we will not detail the control strategy for the operational and posture tasks, as we use state-of-the-art control strategies. The operational task torques are generated considering unit mass forces at the end-effector computed in the nullspace of the higher priority tasks using the operational space formulation [24].

A. Constraint Handling

The collision avoidance is performed in the nullspace of joint limit avoidance, thus the latter has the highest priority.

1) Joint limit avoidance: We use artificial potential fields in our implementation, as introduced in [1].

To avoid the lower limit for joint i we create a positive virtual force:

$$\underline{\gamma_{q_i}} = \begin{cases} \eta_{jl} \cdot (\frac{1}{\underline{\rho}_i} - \frac{1}{\underline{\rho}_{i(0)}}) \cdot \frac{1}{\underline{\rho}_i^2}, & \text{if } \underline{\rho}_i < \underline{\rho}_{i(0)} \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

while to avoid the upper limit for joint i we create a negative virtual force:

$$\overline{\gamma_{q_i}} = \begin{cases} \eta_{jl} \cdot (\frac{1}{\overline{\rho}_i} - \frac{1}{\overline{\rho}_{i(0)}}) \cdot \frac{1}{\overline{\rho}_i^2}, & \text{if } \overline{\rho}_i > \overline{\rho}_{i(0)} \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

where $\eta_{jl} = 0.25$, $\underline{\rho}_i = q_i - \underline{q}_i$ and $\overline{\rho}_i = \overline{q}_i - q_i$, and $\overline{\rho}_i(0)$ and $\underline{\rho}_i(0)$ are the upper and lower joint limit threshold values, respectively. We project the resulting forces into the torque space using the associated Jacobians for the corresponding joints J_{q_i} and sum up the resulting torques $\Gamma_{jl} = \sum_{q_i} J_{q_i}\gamma_{qi}$.

2) Obstacle avoidance: The obstacle avoidance is partitioned into two components: global obstacle avoidance and local reactive obstacle avoidance. The former adapts the end-effector path, generating new goals for the position controller, while the local reactive obstacle avoidance is carried out in the nullspace of the joint limit avoidance. Having the

reactive repulsive forces at the COM of every link computed as explained in Sec.III-A and taking the higher priority of the joint limits avoidance constraint into consideration, the torques $\Gamma_{oa|jl}$ to avoid obstacles are generated by:

$$\Gamma_{oa|jl} = N_{jl}^T \sum (J_{oa_n}^T f_{n_{magnitude}}) \tag{9}$$

where $J_{oa_n} = f_{n_{direction}} J_n$ is the Jacobian of the COM of link n in the constrained direction and J_n is the Jacobian at the COM of link n.

B. Smooth switching between constraints

The constraints in the hierarchical controller can be active or inactive, depending on the current situation, e.g. if there are obstacles close to the robot or if the joint limits are reached. When a robot reaches a constraint the corresponding constraint turns active and pushes the robot away until the constraint is inactive again. As a result, the torques generated by lower-priority levels move the robot towards the constraint again. This fact leads to an undesired shaking behavior in the robot. To avoid the described behavior, it is necessary to have a smooth constraint activation function.

Therefore two activation parameters (α, β), which vary continuously and linearly between 0 and 1 as functions of the corresponding joint angle for joint limit avoidance and the corresponding closest distance for reactive obstacle avoidance are introduced. As a result, the constraints are guaranteed to be held when the activation parameters are equal to 1. Until then there is a trade-off between the constraints and the lower-priority tasks. Using α and β as weighting functions results in four boundary cases:

- $\alpha, \beta \neq 0$: $\Gamma_1 = \beta(\Gamma_{jl} + \alpha(\Gamma_{oa|jl} + \Gamma_{t|oa|jl} + \Gamma_{p|t|oa|jl}))$
- $\alpha, \beta = 0$: $\Gamma_2 = (1-\beta)(1-\alpha)(\Gamma_t + \Gamma_{p|t})$
- $\alpha = 0, \beta \neq 0$: $\Gamma_3 = \beta(\Gamma_{jl} + (1-\alpha)(\Gamma_{t|jl} + \Gamma_{p|t|jl}))$
- $\alpha \neq 0, \beta = 0$: $\Gamma_4 = (1-\beta)(\alpha(\Gamma_{oa} + \Gamma_{t|oa} + \Gamma_{p|t|oa}))$

For smooth constraint switching, linear combinations of the edge cases are added up $\Gamma = (\Gamma_1 + \Gamma_2 + \Gamma_3 + \Gamma_4 + \hat{b} + \hat{g})$, where $\hat{b}$ and $\hat{g}$ are compensating for Coriolis and centrifugal forces and gravity.

V. Experiments & Results

We evaluate our collision avoidance framework in a real world scenario using a Franka Panda. An Intel Realsense D415 depth camera is fixed with respect to the robot base

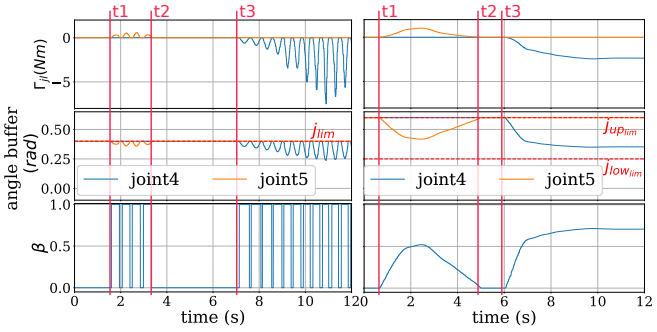

Fig. 6: The effect of smooth constraint activation for joint limit avoidance. On the left a hard constraint activation strategy, while on the right our smooth constraint activation strategy is used. t_1-joint limit constraint activates for $joint4$; t_2-constraint becomes inactive; t_3-constraint activates for $joint5$.

Obstacles	Links	Time [ms]
1	4	0.9
	16	3.7
	24	5.2
2	4	1.8
	16	6.9
	24	20.1
10	4	8.3
	16	32.8
	24	47.5

TABLE I: Distance computation algorithm - average computation times

Parameter	Value
k_{task_p}	50
k_{task_v}	20
$k_{posture_p}$	40
$k_{posture_v}$	20
$\alpha = 0$	$d = 0.2\,\text{m}$
$\alpha = 1$	$d = 0.12\,\text{m}$
$\beta = 0$	$\rho = 0.6\,\text{rad}$
$\beta = 1$	$\rho = 0.25\,\text{rad}$

TABLE II: Controller parameters

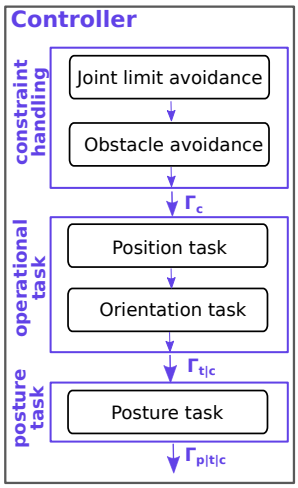

Fig. 5: Schematic of our hierarchical operational space controller

and monitors its workspace. The reconstruction and obstacle avoidance algorithms run on an Intel $i7-7600U$, $2.7-3.5GHz$, dual core 64-bit computer. The controller runs on a similar computer, with real-time Linux. As the robot's base is fixed to a specific location on a table, the first 3 links $(link_0, link_1, link_2)$ are not able to avoid obstacles. Therefore, for the experiments we consider links 3 to 6. The parameters for the controller and the activation buffers for the constraints shown in Table II are experimentally determined.

First, we analyze the performance and scalability of the repulsive force generation and point cloud processing algorithms. The experimental results presented in Table I are averaged over 10 experiments and carried out in simulation. The high-dimensional system in simulation was composed of several Franka Panda arms. The results show, that even for a system with 24 links and with 10 obstacles in the monitored workspace, the algorithm can run in real-time, at about $20\,Hz$. The point cloud processing on the Intel $i7-7600U$ PC takes on average $59\,ms$. With this setup this is currently the bottleneck of the framework and therefore, we chose a frame rate of 15 FPS during the real world experiments. The image resolution is 640x480.

We compare our results to the closest existing works regarding repulsive force computation. The authors consider a similar 7-DOF manipulator and a depth camera to monitor the workspace [21],[18](hybrid method). The first work [21] models the environment using 2.5D voxel maps, while Flacco et al. [18] use directly the 2.5D depth space to improve distance computation time. The comparison considers only one obstacle, as our point cloud processing algorithm generates only one obstacle using the scenarios presented in the papers. The results are presented in Table III and show that our method outperforms the existing ones in terms of repulsive force generation computation times. Furthermore, compared to the mentioned works, our method features the advantages stated in Sec. II-A resulting from modeling the obstacles as convex meshes.

Moreover, we evaluate our smooth constraint activation strategy introduced in Sec. IV-B in simulation by analyzing

Paper	Links	Time [ms]	Setup]
[21]	whole robot (6)	~5	GPU
[18]	1	~1	CPU
ours	**4**	**~1**	CPU

TABLE III: Comparison of repulsive force computation methods

the effect of parameter β on the computed torques for joint limit avoidance in comparison to the case of a hard switching strategy, where $\beta = 1$ or $\beta = 0$. For this, two end-effector poses for which the robot approaches joint limits for two different joints ($joint4$ and $joint5$) are commanded. The result is depicted in Fig.6. It can be noticed that using the smooth constraint activation strategy the generated torques do not oscillate and are smooth, leading the joint angles to converge to a certain value without oscillating.

Furthermore, we perform 6 experiments to validate our collision avoidance framework. These can be visualized in the attached video. Experiments 1 to 4 focus on closing the loop between perception and control using local collision avoidance in a position keeping task. Experiments 5 and 6 analyze the capabilities of our fully integrated collision avoidance framework in a path following scenario. In our experiments, we assume that direct path between the current end-effector position and the goal position is free, so the original path is considered to be a straight line. However, any global path planner can be used to generate the path.

In Experiment 1 a dynamic obstacle has to be avoided, while keeping the end-effector at a fixed position. The results in Fig.7 show that the overall position error increases to $\delta_{max} = 2.04\,cm$. This is explained by the fact that the collision avoidance constraint generates accelerations which affects the lower-priority position task. Therefore, the peak error is reached when higher torques are commanded (at approximately $t3$). The position error decreases when the obstacle stops or moves out of the region of interest. The former is proven in Experiment 2, while the latter is proven in this experiment starting with t_4.

In Experiment 2, a moving obstacle stops in the workspace at t_1. As stated before, it can be seen in Fig.8 that the position error decreases once the obstacle stops, as there are no significant accelerations generated by the collision avoidance constraint. Furthermore, it can be noticed that the links affected by repulsive forces do not move completely away from the obstacle (closest distance to obstacle $< d_0$). This is explained by the smooth constraint switching presented in Sec.IV-B, as $\alpha \neq 1$ and there is a trade-off between the lower and higher priority task. Moreover, when the obstacle stops, the robustness of the distance computation algorithm can be evaluated. The maximum variation in the computed closest distance during the time where the obstacle is stopped is $e_{max} = 1.96\,cm$.

Experiment 3 shows that the framework is can handle multiple dynamic obstacles, as presented in Fig.9. The behavior is similar to the one obstacle scenario in Experiment 1.

In Experiment 4, the robots capability of avoiding a partially occluded obstacle is analyzed. As both obstacle and robot move simultaneously it can not be guaranteed that all the corners of the obstacle are visible at all times, leading to incomplete reconstruction, as shown in II-B. Nevertheless, the obstacle is partially detected and generates repulsive forces, as depicted in Fig.10. Once the occluded part is

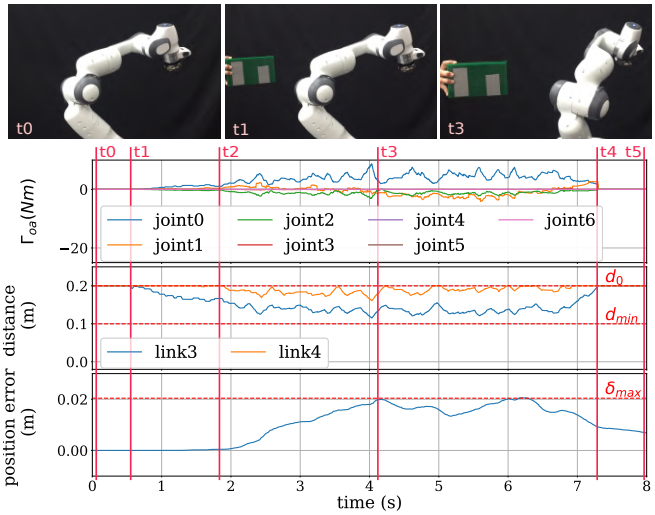

Fig. 7: Experiment 1. The images show the behavior of the robot during the experiment at the mentioned times. t_0-robot initial configuration; t_1-obstacle enters region of interest and generates repulsive forces on $link_3$; t_2-repulsive forces start acting also on $link_4$; t_3-robot avoids obstacle having maximum position error; t_4-obstacle moves out of the region of interest; t_5-robot moves to initial configuration. Γ_{oa} are collision avoidance torques.

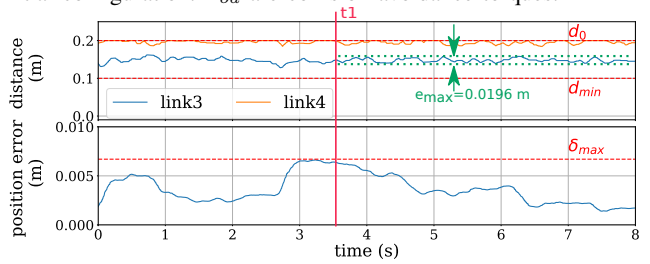

Fig. 8: Experiment 2. t_1-obstacle stops moving.

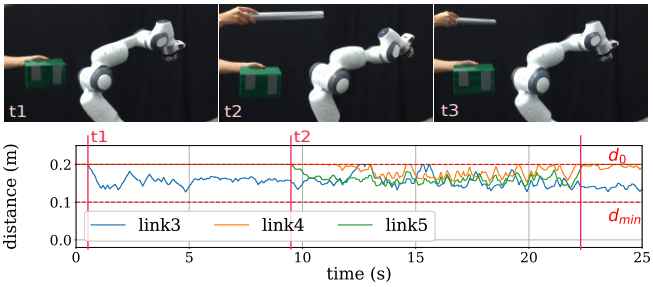

Fig. 9: Experiment 3. The images show the behavior of the robot during the experiment at the mentioned times. t_1-$obstacle1$ enters region of interest; t_2-$obstacle2$ enters region of interest; t_3-$obstacle2$ leaves region of interest

visible at t_2, the distance decreases significantly generating high forces that lead to an overshooting of the robot's posture at t_3. However, the distance to the obstacle stabilizes a few seconds after the overshoot at $t4$.

In Experiment 5, the robot end-effector follows a precomputed path while avoiding the obstacle as depicted in Fig.11. A dynamic obstacle acts on the robot links, but not on the elastic band, such that there is no path reconfiguration during the experiment. Again, it is noticed that the accelerations generated for collision avoidance affect the path following error. The accelerations generated by the joint limit avoidance constraint affect both collision avoidance and position task, leading to a higher variation in the closest distance and

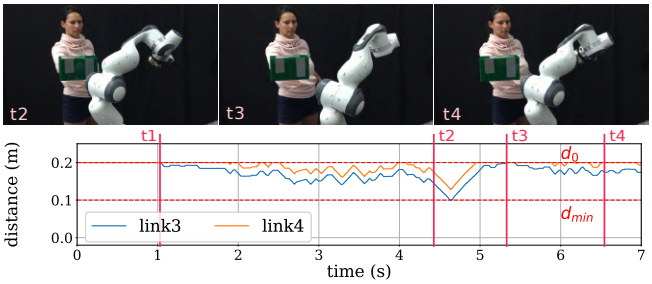

Fig. 10: Experiment 4. The images show the behavior of the robot during the experiment at the mentioned times. t_1-occluded obstacle enters region of interest; t_2-occlusion becomes visible; t_3-overshooting due to occlusion; t_4-distance stabilizes

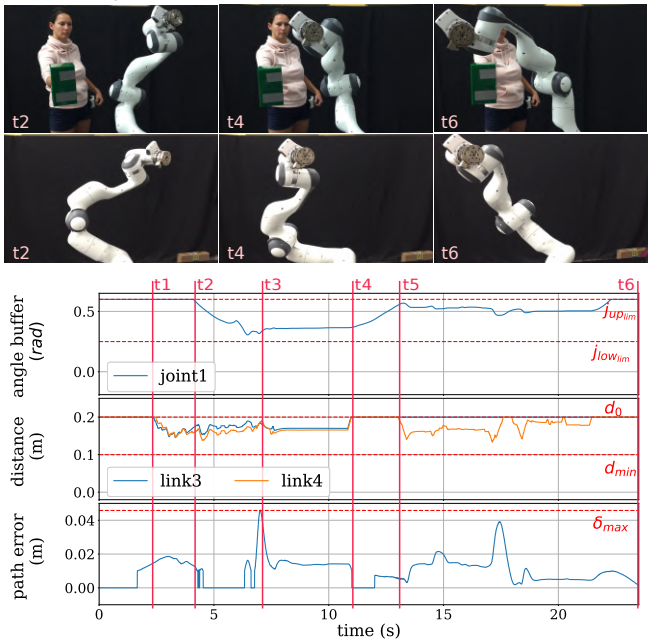

Fig. 11: Experiment 5. The images show the behavior of the robot during the experiment, while the lower images show the baseline (e.g. when there are no obstacles in the workspace). Both at the mentioned times. t_1-obstacle enters region of interest; t_2-joint limit avoidance turns active for $joint_1$; t_3-peak path following error; t_4-obstacle moves out of region of interest; t_5-obstacle enters region of interest; t_6-robot reaches goal

an increase in the path error. Both effects can be noticed in Fig.11 around t_3. This leads to a maximum path error of $\delta_{max} = 4.58\,cm$ and a total end-effector path length of $l = 1.08\,m$, instead of the original path length of $l_0 = 1\,m$.

The last experiment fully analyzes the capabilities of our collision avoidance framework, as depicted in Fig. 12. The elastic band generates collision-free paths for the end-effector online (e.g. during the execution of the preplanned path), while the local reactive collision avoidance strategy creates repulsive forces for the remaining links to avoid the obstacle. The original path length between the start and goal position is $l = 1\,m$, while the reconfigured path has a length of $l = 1.29\,m$. We stop the dynamic obstacle close to the goal at t_5, such that the robot can not reach its final goal position. The robot stops under the influence of repulsive forces (gray zone in Fig.12) at a certain distance to the obstacle. The goal is reached after the obstacle moves away.

The paths generated by the elastic bands at every time step depend on the movement of both the obstacle and robot. A

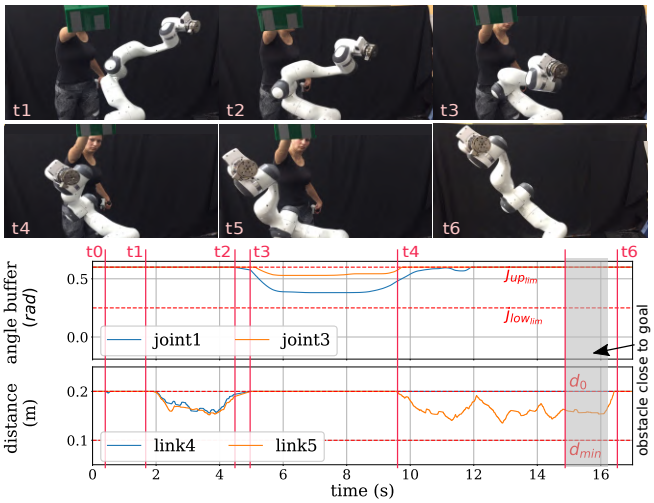

Fig. 12: Experiment 6. The images show the behavior of the robot during the experiment at the mentioned times. t_0-robot starts moving; t_1-obstacle acts on robot and elastic band; t_2-joint limit avoidance turns active; t_3-obstacle moves out of region of interest; t_4-obstacle enters region of interest; t_5-obstacle stops close to the goal position; t_6-robot reaches goal after obstacle moves away.

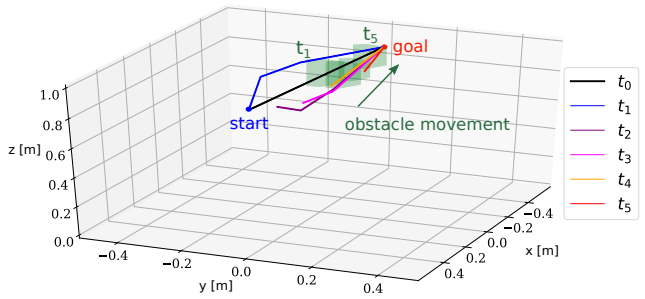

Fig. 13: Adapted paths using the elastic band algorithm under the action of the obstacle at different time steps selected according to the pictures in Fig.12. The obstacle positions at times t_1 to t_5 during which it acts on the elastic band are depicted in green.

selection of those for significant times can be seen in Fig.13. The length of the paths decrease as the time increases due to the fact that the path adaptation is done online, while the robot executes the path.

VI. Conclusions

We introduce a fully integrated real-time collision avoidance framework for high-dimensional multi-link systems which is able to robustly close the loop between perception and control. Our perception algorithm models the environment using only convex meshes, while our collision avoidance strategy combines local reactive APF with elastic path planning with in order to maintain the convergence towards the goal. We evaluate our framework on a Franka Panda, showing that it can robustly avoid collisions in different scenarios.

Having only convex meshes reduces the probability of getting stuck in local minima. In all experiments that were carried out, if the the goal could be reached, it was reached. However, there might be adversarial scenarios that can cause the robot to get stuck. Thorough investigations will be carried out in future work. Moreover, the jittery behavior of the robot will be addressed.

The point cloud processing time, which is currently our bottleneck, can be easily solved by using a GPU, as parallel computation dramatically reduces computation time [21]. Furthermore, also the repulsive force computation can be parallelized on a GPU, increasing the performance of the method overall. Obstacle occlusion only works for big obstacles when all the corners can be seen. This can be tackled by exploiting active perception. Moreover, an obstacle tracking algorithm or several cameras to monitor the workspace can be considered.

REFERENCES

[1] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, vol. 2, March 1985, pp. 500–505.

[2] C. W. Warren, "Global path planning using artificial potential fields," in *Proceedings, 1989 International Conference on Robotics and Automation*, May 1989, pp. 316–321 vol.1.

[3] P. Ogren, N. Egerstedt, and X. Hu, "Reactive mobile manipulation using dynamic trajectory tracking," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, vol. 4, April 2000, pp. 3473–3478 vol.4.

[4] L. Singh, H. Stephanou, and J. Wen, "Real-time robot motion control with circulatory fields," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 3, April 1996, pp. 2737–2742 vol.3.

[5] J. . Kim and P. K. Khosla, "Real-time obstacle avoidance using harmonic potential functions," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 3, pp. 338–349, June 1992.

[6] S. M. Khansari-Zadeh and A. Billard, "A dynamical system approach to realtime obstacle avoidance," *Autonomous Robots*, vol. 32, no. 4, pp. 433–454, 2012.

[7] S. Haddadin, R. Belder, and A. Albu-Schäffer, "Dynamic motion planning for robots in partially unknown environments*," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 6842 – 6850, 2011, 18th IFAC World Congress.

[8] S. Quinlan and O. Khatib, "Elastic bands: connecting path planning and control," in *[1993] Proceedings IEEE International Conference on Robotics and Automation*, May 1993, pp. 802–807 vol.2.

[9] O. Brock and O. Khatib, "Elastic strips: A framework for motion generation in human environments," *The International Journal of Robotics Research*, vol. 21, no. 12, pp. 1031–1052, 2002.

[10] S. R. Lindemann and S. M. LaValle, "Current issues in sampling-based motion planning," in *Robotics Research. The Eleventh International Symposium*, P. Dario and R. Chatila, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005.

[11] Y. Yang and O. Brock, "Elastic roadmaps—motion generation for autonomous mobile manipulation," *Autonomous Robots*, vol. 28, p. 113, Sep 2009.

[12] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics Automation Magazine*, vol. 4, no. 1, pp. 23–33, March 1997.

[13] A. Winkler and J. Such, "Dynamic collision avoidance of industrial cooperating robots using virtual force fields," *IFAC Proceedings Volumes*, vol. 45, no. 22, pp. 265 – 270, 2012, 10th IFAC Symposium on Robot Control.

[14] C. Fang *et al.*, "Efficient self-collision avoidance based on focus of interest for humanoid robots," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, Nov 2015, pp. 1060–1066.

[15] W. Merkt, V. Ivan, and S. Vijayakumar, "Continuous-time collision avoidance for trajectory optimization in dynamic environments," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov 2019, pp. 7248–7255.

[16] N. Mansard, O. Stasse, P. Evrard, and A. Kheddar, "A versatile generalized inverted kinematics implementation for collaborative working humanoid robots: The stack of tasks," in *2009 International Conference on Advanced Robotics*, June 2009, pp. 1–6.

[17] F. Flacco, T. Kröger, A. De Luca, and O. Khatib, "A depth space approach to human-robot collision avoidance," in *2012 IEEE International Conference on Robotics and Automation*, May 2012, pp. 338–345.

[18] ——, "A depth space approach for evaluating distance to objects," *Journal of Intelligent & Robotic Systems*, vol. 80, no. 1, pp. 7–22, Dec 2015.

[19] J.Chen and K. Song, "Collision-free motion planning for human-robot collaborative safety under cartesian constraint," 05 2018, pp. 1–7.

[20] M. Di Castro, D. B. Mulero, M. Ferre, and A. Masi, "A real-time reconfigurable collision avoidance system for robot manipulation," in *Proceedings of the 3rd International Conference on Mechatronics and Robotics Engineering*, ser. ICMRE 2017, 2017, p. 610.

[21] K. B. Kaldestad *et al.*, "Collision avoidance with potential fields based on parallel processing of 3d-point cloud data on the gpu," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 3250–3257.

[22] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[23] S. Lloyd, "Least squares quantization in pcm," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, March 1982.

[24] L. Sentis and O. Khatib, "A whole-body control framework for humanoids operating in human environments," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, May 2006, pp. 2641–2648.

[25] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *IEEE Journal on Robotics and Automation*, vol. 3, no. 1, pp. 43–53, February 1987.