2020 29th IEEE International Conference on
Robot and Human Interactive Communication (RO-MAN)
August 31 - September 4, 2020 - Virtual Conference

Hierarchical Motion Planning Framework for Manipulators in Human-Centered Dynamic Environments

Jonas Wittmann[1], Julius Jankowski[1,2], Daniel Wahrmann[2] and Daniel J. Rixen[1]

Abstract— **Collaborating robots face rising challenges with respect to autonomy and safety as they are deployed in flexible automation applications. The ability to perform the required tasks in the presence of humans and obstacles is key for the integration of these machines in industry. In this work we introduce a framework for motion planning of manipulators that builds upon the most promising existing approaches by combining them in an advantageous way. It includes a new *Obstacle-related Sampling Rejection Probabilistic Roadmap* planner (*ORSR-PRM*) that represents the free workspace in an efficient way. Using this representation, dynamic obstacles can be avoided in real-time using an attractor-based online trajectory generation. The resulting motions satisfy kinematic and dynamic joint limits, ensuring a safe human-robot interaction. We validate the functionality and performance of the presented framework in simulations and experiments.**

I. INTRODUCTION

As industries evolve into more flexible production processes, robots are expected to adapt and collaborate with humans in shared workspaces [1]. Thereby, the paradigm in robot control shifts from high accuracy to absolute safety while being able to quickly adapt to new tasks. However, existing motion planning frameworks for robot manipulators are still highly specialized for a particular application [2], [3], [4], [5], [6]. Their underlying algorithms can be divided into sampling-based, optimization-based and potential-field-based approaches with different drawbacks and advantages. Thereby, most of the proposed solutions are limited to a specific methodology which makes them less flexible. As motion planning algorithms are computationally very expensive, strong assumptions have to be made in order to deal with dynamic environments [7].

Currently, *MoveIt!* [2] is the state-of-the-art manipulator planning framework in research. It provides a simple interface to quickly setup basic motion planning functionalities. However, it uses the *Open Motion Planning Library (OMPL)* [3] which is limited to purely geometric sampling-based planners that don't consider the system's dynamics. Furthermore, it is limited to offline planning computations and does not allow real-time adaptions to the motion. Two prominent sampling-based methods are *Probabilistic Roadmaps (PRM)* [8] and *Rapidly-exploring Random Trees (RRT)* [9]. They differ in the way of addressing a motion planning problem. *PRM* constructs roadmaps offline for a static subsection of the configuration space and thus is well suited to be used multiple times for the same configuration space (*multiple-query*). *RRT* constructs roadmaps for each motion query on the fly, starting at the current configuration (*single-query*). Modifications and extensions have been presented that improve performance by reconnecting the graph during sampling or performing collision checks only during online queries and not during roadmap construction [10] [11]. Real-time modifications of a planned path can be done with neither of these sampling-based approaches. In [6] the *Covariant Hamiltonian Optimization for Motion Planning (CHOMP)* is introduced. The optimization-based approach is capable of finding feasible trajectories even in case of non-feasible initial straight line guesses in configuration space. The cost function includes obstacle avoidance for defined points on the robot. The formulation allows a decoupling of the path generation and its timing. However, the optimization-based approach is limited to offline planning. In [12] another decoupled approach for motion generation is presented, which makes the assumption that the behavior of the dynamic part of the environment is known a priori. The *Stochastic Trajectory Optimization for Motion Planning (STOMP)* [4] is another optimization-based technique that allows a non-differentiable optimization objective. It generates multiple trajectory candidates in proximity of the current solution and evaluates these candidates w.r.t. the optimization objective. In [5] the *Open Robotics and Animation Virtual Environment* is introduced which focuses on the automated manipulation problem. It is separated into a set of generic algorithms for controlling the robot and a construction process that adapts these algorithms to the application. The framework uses sampling-based planning methods and only focuses on reliability. Therefore, additional criteria such as optimality, dynamic constraints or post-processing for path smoothing are ignored as they would require the incorporation of velocities. For real-time adaptions, potential-field-based approaches are commonly used [13]. In [14] an attractor-based motion control scheme similar to the one in our work is presented. Repulsive and attracting forces are formulated and applied to a virtual particle that is a single mass point in Cartesian coordinates. That way, they have more control over the motion w.r.t. the instantaneous velocity of the particle. At one point in time, the designed second order system is simulated for multiple time steps; the next set point of the closed loop control can be computed from the simulated positions and the desired velocity of the particle. However, this method does not guarantee the validity of generated trajectories nor does it make use of joint dynamic limits other than Cartesian velocity for the end effector.

Jonas Wittmann and Julius Jankowski are co-first authors. Corresponding author: jonas.wittmann@tum.de
[1]Chair of Applied Mechanics, Technical University of Munich, Germany
[2]FRANKA EMIKA GmbH, Munich, Germany

The mentioned frameworks are either too slow to react to dynamic environments or make strong assumptions that are not always kinematically or dynamically reasonable. In this work, we present a motion planning framework that is able to generate collision free trajectories in partially dynamic environments while maintaining kinematic and dynamic constrains. By assuming partially static environments, motion roadmaps can be pre-computed such that modifications of planned motions can be made in real-time to deal with occasional dynamic changes or human interaction. Our approach, which directly incorporates external forces in the real-time trajectory generation, ensures a safe robot behavior in human-centered environments.

The remainder of this paper is organized as follows. In Sec. II, the general framework and the theory behind its main components, the offline sampling-based planner and the online trajectory generator are described. The developed concepts are evaluated in a Pick-and-Place application in Sec. III and the performance is presented. Conclusions follow in Sec. IV.

II. FRAMEWORK DESIGN

The motivation behind this work are shared workcells between humans and robots where the main components of the environment are static and a few of them (such as human workers) are dynamic. More formally, we form the following hypotheses:

- The robotic system is stationary, meaning that its base is rigidly fixed to the workspace.
- The main portion of the workspace is static, e.g. a Pick-and-Place application with human coworkers.
- The robot is equipped with force/torque sensors that allow the estimation of external forces and the implementation of compliant behavior.

Additionally, we assume that the task consists of reaching a fixed pose formulated in the robot's task space $\mathcal{W}$ with a feasible trajectory (i.e. a collision-free motion of the robot respecting its kinematic and dynamic limits) calculated in the robot's configuration space $\mathcal{C}$. Our framework splits the highly-complex task of generating safe motions into three distinct hierarchical phases, which are executed on different time horizons. First, a map is generated offline that efficiently represents the static part of the environment. In a second step, event-triggered online motion queries find a feasible and optimal path based on the weighted connections using A*-search. The path is forwarded to an online trajectory submodule that runs within the control cycle of the robot and appropriately adapts the motion on the fly depending on dynamic obstacles and external disturbances. In the following, we explain how these three different phases are implemented in order to solve the motion planning task for the defined set of problems.

A. Motion Planning Framework

The *Motion Planning Framework*, shown in Fig. 1, is integrated in a higher-level software framework that includes further modules like a *Vision System*, which is responsible for object detection or a *State Machine*, which is responsible for the overall task execution of the Pick-and-Place application. The communication to the other modules is implemented using the *Robot Operating System (ROS)* [15]. An overview of the included third party libraries is given in Sec. III-A. Three major modules, the *Planning Unit*, the *Context* and the *Low-Level Control*, calculate the joint control commands that are forwarded to the robot via the C++ *Robot Interface* in each control cycle (1 kHz).

The orange module *Context* serves as a central knowledge

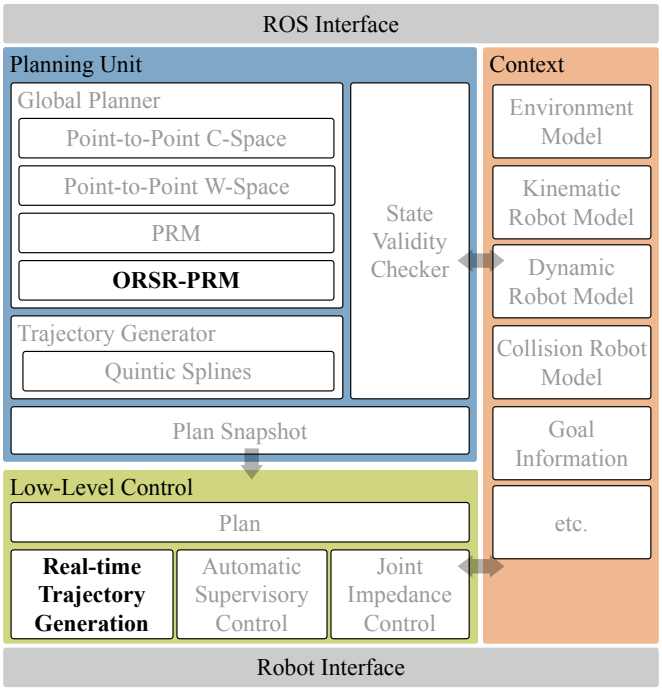


Fig. 1: Architecture of the *Motion Planning Framework* with its submodules required to navigate the robot in complex and human-centered environments.

database that keeps track of the current environment and goal objectives and implements the kinematic and dynamic robot models as well as a collision model of the robot that is based on simplified geometries.

The blue module *Planning Unit* is triggered by incoming motion commands specified by a desired goal state of the robot, e.g. Cartesian end effector pose, and is supposed to compute collision-free waypoints with a *Global Planner*. *Point-to-Point* motion planning in $\mathcal{W}$- and $\mathcal{C}$-space can be used as well as the *PRM* planners to compute a solution path for the particular motion task. An integrated offline *Trajectory Generator* can precompute a trajectory using *Quintic Splines*. The resulting path or trajectory is defined as a *Plan Snapshot*, which is passed to the green *Low-Level Control* module. Based on this *Plan*, the appropriate joint inputs are calculated using *Automatic Supervisory Control*, *Joint Impedance Control* or *Real-time Trajectory Generation*. In this paper we present an extended *PRM* planner which uses an *Obstacle-related Sample Rejection* strategy (*ORSR-PRM*) and a *Real-time Trajectory Generation* submodule.

These enable the framework to be used in complex industrial environments in which the robot moves in a mostly static environment where online adaptions due to human interaction and disturbance are occasionally required. The objective of the *ORSR-PRM*, presented in detail in Sec. II-B, is to shift computational effort to the offline calculation by generating an efficient environment representation within the map of the planner. The map consists of sampled robot configurations, i.e. joint angles, that are checked for collision with the environment using the *State Validity Checker*. In contrast to the classic *PRM* planner, *ORSR-PRM* adds a sample rejection step to the construction phase that ensures an efficient representation of the free static $\mathcal{W}$-space in $\mathcal{C}$-space. By precomputing via points using *ORSR-PRM*, we convert arbitrarily complex motions that are difficult to adapt in real-time into sections that are known to be less cluttered than the direct connections between the start and the end configuration. That way, the potential field based trajectory generation approach is less prone to getting stuck in local minima and ensures robust collision avoidance. The *Real-time Trajectory Generation* submodule, presented in detail in Sec. II-C, uses the computed path and turns it into a feasible trajectory in a reactive way, such that collisions with dynamic obstacles that are not represented in the map of the *ORSR-PRM* can be avoided. Furthermore, the *Real-time Trajectory Generation* submodule ensures the dynamic limits of the robot, i.e. joint velocity and acceleration.

B. Obstacle-related Sample Rejection PRM

The *Obstacle-related Sample Rejection PRM (ORSR-PRM)* planner is designed to accelerate the response to online motion queries. Therefore, we present an obstacle-related sampling strategy that generates a sparse map that has the same connectivity as a map generated from uniformly distributed samples. Due to the hierarchical design of our planning framework, the construction of the map is done offline. This is in accordance with the multiple-query approach of the classic *PRM* planner [8] that uses uniform sampling for the creation of the map. To avoid the resulting large maps - especially in high-dimensional $\mathcal{C}$-spaces - that are computationally costly to process during an online motion query, many approaches have been proposed to steer the distribution of the milestones (i.e. roadmap nodes) in free space [16][17][18]. The common idea is to control the density of milestones depending on the distance to obstacle boundaries in $\mathcal{C}$-space. Based on this idea, we propose a new strategy to reject certain configurations resulting from a uniform sampling process. Alg. 1 shows the underlying calculations.

For a newly sampled robot configuration $\mathbf{q}_s$ the minimum distance d_{min} to the closest environment objects and the Jacobian of translation $\mathbf{J}_{p,trans}$ of the closest point $\mathbf{p}$ of the robot are calculated. In a subsequent step, a distance metric $d_{clear,j}$ between the nearest neighbors $\mathbf{Q}_{nn}$ and the sampled configuration are obtained. Therefore, the Cartesian distance of the collision point $\mathbf{p}$ between the sampled configuration $\mathbf{q}_s$ and the neighbor $\mathbf{q}_j$ is approximated using the first-order

Algorithm 1 Algorithm for Map Construction

Input: Desired number of samples N, threshold parameter k_{clear}, threshold set $\mathcal{Q}_{box}$
Output: Map **G**(Vertices **V**, Edges **E**)
Initialization
while $n < N$ **do**
  $\mathbf{q}_s \leftarrow$ sampleCollisionFreeConfiguration()
  computeForwardKinematics($\mathbf{q}_s$)
  $d_{min} \leftarrow$ calculateMinimumDistance($\mathbf{q}_s$)
  $\mathbf{J}_{p,trans} \leftarrow$ calculateNearestPointJacobian($\mathbf{q}_s$)
  $\mathbf{Q}_{nn} \leftarrow$ getNearestNeighbors($\mathbf{q}_s$)
  sampleValid = true
  for $\mathbf{q}_j \in \mathbf{Q}_{nn}$ **do**
    $d_{clear,j} \leftarrow$ calculateClearance($\mathbf{q}_j$, $\mathbf{q}_s$, $\mathbf{J}_p$)
    if $d_{clear,j} \leq k_{clear} d_{min}$ **and** $\mathbf{q}_j - \mathbf{q}_s \in \mathcal{Q}_{box}$ **then**
      sampleValid = false
      break
    end if
  end for
  if sampleValid **then**
    G.addVertex($\mathbf{q}_s$)
    $\mathbf{E}_{sample} \leftarrow$ getCollisionFreeEdges($\mathbf{q}_s$, $\mathbf{Q}_{nn}$)
    G.addEdges($\mathbf{E}_{sample}$)
    $n \leftarrow n + 1$
  end if
end while

Taylor expansion

$$d_{clear,j} = ||\mathbf{p}(\mathbf{q}_j) - \mathbf{p}(\mathbf{q}_s)||_2 \quad (1)$$

$$\mathbf{p}(\mathbf{q}_j) - \mathbf{p}(\mathbf{q}_s) \approx \left.\frac{\partial \mathbf{p}}{\partial \mathbf{q}}\right|_{\mathbf{q}=\mathbf{q}_s} (\mathbf{q}_j - \mathbf{q}_s) \quad (2)$$

where $\left.\frac{\partial \mathbf{p}}{\partial \mathbf{q}}\right|_{\mathbf{q}=\mathbf{q}_s}$ corresponds to the calculated Jacobian of translation $\mathbf{J}_{p,trans}$. The Jacobian has to be computed only once for the distance computation with respect to every neighbor. Thus, we choose this distance approximation for the sake of computational efficiency. By using the Cartesian distance between the robot and the closest obstacle as a rejection criterion, the collision geometries in $\mathcal{W}$-space can be captured in $\mathcal{C}$-space, where obstacles do not have an explicit mathematical representation. This is realized by scaling the rejection threshold for the sample $\mathbf{q}_s$ by the Euclidean distance to the closest obstacle, such that the sample is rejected if there is a neighboring milestone $\mathbf{q}_j$ for which $d_{clear,j} \leq k_{clear} d_{min}$. The constant parameter $k_{clear} \in (0, 1)$ affects the resulting density of the milestones. In order to limit approximation errors inherent to Eq. 1 and to enable a better exploitation of the nullspace, we use a $\mathcal{C}$-space box $\mathcal{Q}_{box}$ as a second threshold

$$\mathcal{Q}_{box} = \{\mathbf{q} | \mathbf{q} = (q_1, .., q_i, .., q_n)^T \\ |q_i| \leq q_{box,i}, \forall i \in [1, n]\}$$

with the number of degrees of freedom n and the user-specified parameter $\mathbf{q}_{box}$. The intersection of the two convex

sets resulting from the distance thresholds can be interpreted as a local approximation of the visibility set of the newly sampled configuration. This is in turn used as a clearance set w.r.t. the milestones already added to the map such that it is more likely to reject a sample with a high visibility. Fig. 2 illustrates the relation of the $\mathcal{C}$-space clearance sets to the $\mathcal{W}$-space.

The created map of milestones is used as a bidirectional graph with the milestones as nodes and the straight-line connections as edges. A solution path to a motion query that consists of a start and end configuration is found by using a graph-search method. By applying A*-search, a cost function depending on the geometric path is optimized w.r.t. the map as a discretized representation of the continuous $\mathcal{C}$-space. When the motion goal is defined in $\mathcal{W}$-space, we use differential inverse kinematics to find a corresponding goal configuration by seeding the configuration randomly. If the robot is kinematically redundant w.r.t. the $\mathcal{W}$-space, we compute multiple solution paths for randomly seeded goal configurations and select the minimal-cost solution.

C. Real-time Trajectory Generation based on Limitation-aware Attractor Dynamics

In this section we introduce a new attractor-based trajectory generation approach that enables the reactive execution of multiple-via-point paths. In contrast to the approach in [14] the formulation is done in $\mathcal{C}$-space and enforces velocity and acceleration limits on joint level. Within the first step, a virtual command for the attractor is computed based on the superposition of multiple potential fields. In a subsequent step, the limitation-aware dynamics of the attractor are simulated in order to obtain a reference configuration $\mathbf{q}_{ref}^{k+1}$ and reference joint velocities $\dot{\mathbf{q}}_{ref}^{k+1}$ for the next control cycle of the robot controller. The input to the algorithm is composed of a path consisting of M via points $\mathbf{q}_{path} = [\mathbf{q}_{via,1}^T, ..., \mathbf{q}_{via,M}^T]^T$ and the time-invariant velocity and acceleration limits $\dot{\mathbf{q}}_{max}$, $\ddot{\mathbf{q}}_{max}$ defined in $\mathcal{C}$-space.

The virtual attractor command $\boldsymbol{\tau}_{virt}^k$ for time step k is the sum of an attracting torque $\boldsymbol{\tau}_{via}^k$ w.r.t. the next via point and a repelling torque $\boldsymbol{\tau}_{col}^k$ due to obstacles that are assumed to be ideally represented by the collision model. Further, the feedback of disturbance torques $\boldsymbol{\tau}_{ext}^k$ due to external forces as well as torques due to additional subtasks (e.g. joint limit avoidance) $\boldsymbol{\tau}_{sub}^k$ are included, such that

$$\boldsymbol{\tau}_{virt}^k = \boldsymbol{\tau}_{via}^k + \boldsymbol{\tau}_{col}^k + \boldsymbol{\tau}_{ext}^k + \boldsymbol{\tau}_{sub}^k \tag{3}$$

In order to realize the limitation-awareness of the attractor dynamics, the diagonal mass matrix $\mathbf{M}_{virt,k} = \mu_k \mathbf{M}_{min} \in \mathbf{R}^{n \times n}$ of the attractor is scaled by a factor μ_k such that the joint acceleration limits $\ddot{\mathbf{q}}_{max}$ of the robot are exploited by the attractor. In order to avoid high noise amplifications in case of small forces resulting from the superposition of the potential fields, a minimum virtual mass matrix $\mathbf{M}_{min}$ is defined, constraining the factor μ_k to a lower bound $\mu_k \geq 1 \ \forall k$. The resulting preliminary acceleration $\ddot{\mathbf{q}}_{pre}^k$ of the attractor is given by

$$\ddot{\mathbf{q}}_{pre}^k = \frac{1}{\mu_k} \mathbf{M}_{min}^{-1} \boldsymbol{\tau}_{virt}^k \tag{4}$$

The scalar factor μ_k is computed by solving the constrained optimization problem

$$\mu_k = \arg\max_{\mu \in \mathbf{R}} ||\ddot{\mathbf{q}}_{pre}^k(\mu)||_2 \tag{5}$$

$$\text{s.t.} \quad \mu \geq 1 \tag{6}$$

$$|\ddot{q}_{pre,i}^k| \leq \ddot{q}_{max,i} \ \forall i \in [1, n] \tag{7}$$

This optimization problem can be solved by computing the element-wise solutions. By selecting the maximum value for μ_k among those solutions, we guarantee that Eq. (7) is satisfied for all i.

Subsequently, a *reachable set* of joint velocities $\dot{\mathcal{Q}}_r^{k+1}$ containing all joint velocities that can be reached by the attractor at time step $k+1$ while respecting the acceleration limits $\ddot{\mathbf{q}}_{max}$ is calculated using the explicit Euler method:

$$\dot{\mathcal{Q}}_r^{k+1} = \{\dot{\mathbf{q}} \mid \dot{q}_{ref,i}^k - \Delta t \ddot{q}_{max,i} \leq \dot{q}_i \leq \dot{q}_{ref,i}^k + \Delta t \ddot{q}_{max,i}, \ \forall i \in [1, n]\}$$

If the preliminary acceleration is used to calculate the velocity $\dot{\mathbf{q}}_r^{k+1} = \dot{\mathbf{q}}_{ref}^k + \Delta t \ddot{\mathbf{q}}_{pre}^k$ at the next time step, it is trivial to show that $\dot{\mathbf{q}}_r^{k+1} \in \dot{\mathcal{Q}}_r^{k+1}$. In a next step, a *feasible set* of joint velocities $\dot{\mathcal{Q}}_f^{k+1}$ containing all velocities that are allowed to be reached at time step $k+1$ is calculated:

$$\dot{\mathcal{Q}}_f^{k+1} = \{\dot{\mathbf{q}} \mid |\dot{q}_i| \leq \xi_k \dot{\mathbf{q}}_{max,i}, \ \forall i \in [1, n]\}$$

with $\xi_k \in [0, 1]$ being a time-variant factor used to scale the attractor velocity. Details about the computation of ξ_k are presented later in this section. It is the goal of this algorithm to find a velocity that is an element of the intersection of $\dot{\mathcal{Q}}_f^{k+1}$ and $\dot{\mathcal{Q}}_r^{k+1}$ which corresponds to a velocity that satisfies the velocity and acceleration limits of the robot. The next step of the algorithm is to compute a feasible velocity $\dot{\mathbf{q}}_f^{k+1} = \lambda_k \dot{\mathbf{q}}_r^{k+1} \in \dot{\mathcal{Q}}_f^{k+1}$ by scaling down the previously computed reachable velocity as follows:

$$\lambda_k = \arg\min_{\lambda \in \mathbf{R}} ||\dot{\mathbf{q}}_f^{k+1}(\lambda) - \dot{\mathbf{q}}_r^{k+1}||_2 \tag{8}$$

$$\text{s.t.} \quad \dot{\mathbf{q}}_f^{k+1}(\lambda) \in \dot{\mathcal{Q}}_f^{k+1} \tag{9}$$

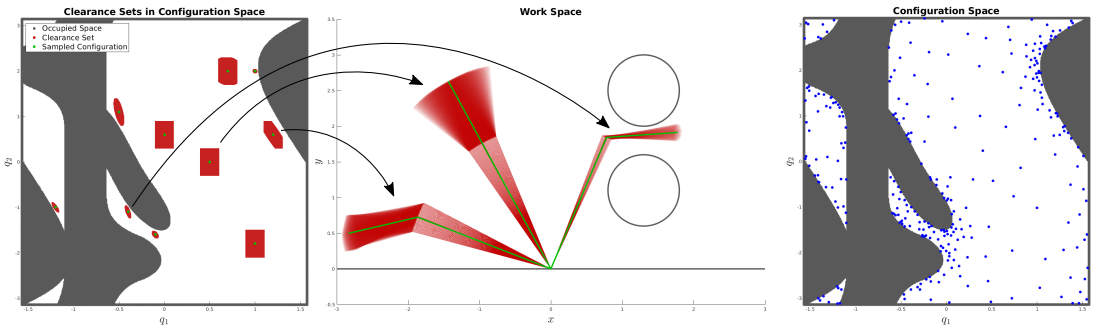
in order to preserve the originally commanded direction of motion. Because of the box shapes of $\dot{\mathcal{Q}}_f^{k+1}$ and $\dot{\mathcal{Q}}_r^{k+1}$, it is trivial to show that the closest reachable velocity to any feasible velocity is an element of the intersection set if the intersection set exists. We define the reference joint velocity accordingly:

$$\dot{\mathbf{q}}_{ref}^{k+1} = \arg\min_{\dot{\mathbf{q}}_{ref}} ||\dot{\mathbf{q}}_{ref} - \dot{\mathbf{q}}_f^{k+1}||_2 \tag{10}$$

$$\text{s.t.} \quad \dot{\mathbf{q}}_{ref} \in \dot{\mathcal{Q}}_r^{k+1} \tag{11}$$

The intersection set $\dot{\mathcal{Q}}_f^{k+1} \cap \dot{\mathcal{Q}}_r^{k+1}$ exists if $\dot{\mathcal{Q}}_f^k \cap \dot{\mathcal{Q}}_r^k \neq \emptyset$ and the scaled velocity limit satisfies the negative acceleration limit

$$\frac{\partial \xi_k \dot{\mathbf{q}}_{max,i}}{\partial t} \geq -\ddot{\mathbf{q}}_{max,i}, \ \forall i \in [1, n] \tag{12}$$



(a) (b)

Fig. 2: Illustration of the rejection process for a 2D minimal robot model situated in a cluttered environment. (a) illustrates the sets of configurations (red) that are within both thresholds of individual samples (green) and their corresponding forward kinematics in $\mathcal{W}$-space. (b) shows the resulting distribution of milestones after feeding uniformly distributed samples into the rejection algorithm. The parameters were set to $k_{clear} = 0.5$, $q_{box,1} = 0.2$ [rad] and $q_{box,2} = 0.6$ [rad].

This algorithm tries to maximize the instantaneous motion given the virtual attractor command and the system limitations. As a consequence, instead of employing high damping gains within the task-related controllers, we deploy a velocity shaping scheme that adapts the set of feasible velocities $\dot{\mathcal{Q}}_f^{k+1}$ w.r.t. the distance to the next via point by using the scaled velocity limit $\xi_k \dot{\mathbf{q}}_{max}$ with

$$\xi_k = \begin{cases} 1 & \text{if } e_k \geq c_1 \\ \frac{1}{2}\left(1 - cos\left(\pi \frac{e_k}{c_1}\right)\right) & \text{otherwise} \end{cases} \tag{13}$$

Hereby, $e_k = ||\mathbf{q}_{ref}^k - \mathbf{q}_{via,m}||_\infty$ is the distance between the current attractor position and the m-th via point. The tunable hyperparameter $c_1 \in \mathbf{R}^+$ represents the L_∞-distance in $\mathcal{C}$-space at which the system starts to slow down in order to stop at via point $\mathbf{q}_{via,m}$. By considering the worst case acceleration for a single joint by inserting Eq. 13 into Eq. 12, we derive a lower bound for c_1

$$c_1 \geq \frac{3\sqrt{3}}{16}\pi \frac{\dot{q}_{max,i}^2}{\ddot{q}_{max,i}}, \forall i \in [1, n] \tag{14}$$

Next, we enable the attractor to move along several via points by specifying another hyperparameter $c_2 \in \mathbf{R}^+$ that represents the L_∞-distance between the current attractor position and the next via point at which we switch to the succeeding via point. Due to the limitation-aware attractor dynamics, the algorithm is insensitive to jumps in the attractor command and thus to switching the set point of the attracting potential field. Fig. 3 shows the resulting trajectory for a 2D minimal example including collision avoidance.

III. SIMULATION AND EXPERIMENTS

A. System Overview

To navigate the robot in unstructured and human-centered environments that include interaction and disturbances, soft-robotics features like impedance controlled joints are essential. We use the *FRANKA EMIKA Panda* robot[19], which has torque sensors in each joint and allows safe operation in human-centered environments with a movable mass of only 12.8 kg. The redundant kinematic structure with seven links increases the flexibility and allows to consider further secondary objectives during motion planning (e.g. time, energy consumption). The software framework runs on a real-time capable robot control computer operated by 64-bit Ubuntu 16.04 with a preempted kernel version 4.16.18-rt12 running on a 12-core Intel i7-7800K CPU at 3.7 GHz. The computer is equipped with 32 GB RAM and an Intel I350 Gigabit network device.

For the implementation of the features of the *Motion Planning Framework* presented in Sec. II-A different libraries are used. *MoveIt!* is used for handling the kinematic robot model and the corresponding collision model. *OMPL* is used for the formulation of the robot's state space (e.g. distances, nearest neighbor determination) and the sampling of the configurations. The maps and A*-search are implemented using the *boost graph library*. The *Flexible Collision Library* [20] is used for the collisions checks and distance computations between the robot and the environment.

B. Static Scenarios

Fig. 4a shows the evaluation scenario for the *ORSR-PRM* presented in Sec. II-B. During the operation the robot has to navigate through narrow passages like the interior of the box or between two profiles (light barrier). Fig. 4b-4d show the corresponding generated maps. While the samples of the classic *PRM* in Fig. 4b are uniformly distributed in the robot's $\mathcal{W}$-space, the samples of the *ORSR-PRM* in Fig. 4c are located in the proximity of the obstacles resulting in more sampling points around the light barrier and inside the box. Fig. 4d shows that the *ORSR-PRM* generates samples with small distances to the environment in $\mathcal{W}$-space.

Table I shows the computation times for the map generation and path search of classic *PRM* and *ORSR-PRM*. For the A*-search, the connections within the maps are weighted with an optimization objective after all samples were generated. Two different objectives are evaluated. An objective

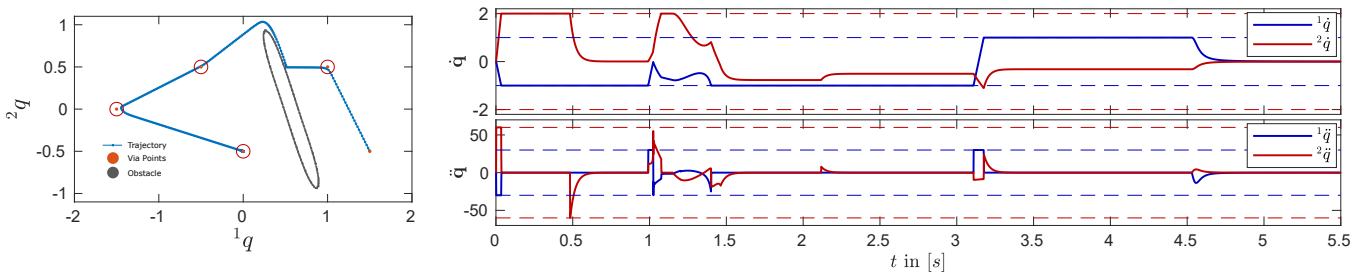
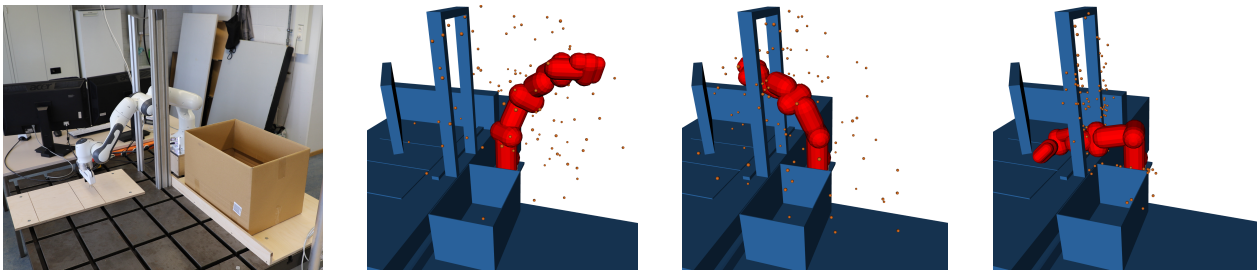


Fig. 3: Multiple-via-point path execution with on-the-fly trajectory adaption to avoid collisions with an obstacle using limitation-aware attractor dynamics applied to a 2D minimal model. The limits of the minimal model are $\dot{\mathbf{q}}_{max} = (1, 2)^T \frac{\text{rad}}{\text{s}}$ and $\ddot{\mathbf{q}}_{max} = (30, 60)^T \frac{\text{rad}}{\text{s}^2}$. These limits are illustrated by the dashed lines in the two lower plots. The hyperparameters of this example are $c_1 = c_2 = 0.08$ [rad].



(a) The robot picks objects out of a box and places them through a simulated light barrier onto a drop area.

(b) TCP positions of the robot configurations using classic *PRM*.

(c) TCP positions of the robot configurations using *ORSR-PRM*.

(d) Positions of the closest points on the robot to the obstacles using *ORSR-PRM*.

Fig. 4: The orange markers show the TCP position (b, c) and the position of the nearest points on the robot to the static obstacles (d) for the different configuration samples after calculating the forward kinematics. Each map consists of 200 milestones. For visualization purposes, $\mathcal{Q}_{box}$ of Alg. 1 is set to infinity.

L_{length}, which represents the distance of two samples in both $\mathcal{W}$- and $\mathcal{C}$-space and an objective $L_{gravity}$, which corresponds to the required joint torques of a configuration to counteract the gravitational forces due to the robot's self-weight. The formulas are given in the Appendix.

More than half of the free space samples are rejected by the post-processing rejection step stay the same, the smaller number of cost evaluations for *ORSR-PRM* predominate which results in less construction time for *ORSR-PRM* compared to classic *PRM*.

	Classic *PRM*	*ORSR-PRM*
Milestones / Connections	80 k / 1875 k	33 k / 678 k
Construction Time L_{length}	19.2 min	26.3 min
Construction Time $L_{gravity}$	35.1 min	33.1 min
Path Search Time L_{length}	9.5 $\pm$ 0.3 ms	2.1 $\pm$ 0.2 ms
Path Search Time $L_{gravity}$	438.6 $\pm$ 1.4 ms	158.1 $\pm$ 5.0 ms

TABLE I: For each map 80000 collision-free configurations are sampled. During cost evaluation, the connections between two samples are interpolated with a fixed stride of 0.01 rad.

described strategy in Sec. II-B. For L_{length}, the construction time of the roadmap for *ORSR-PRM* is higher compared to classic *PRM* as the computational costs due to the additional post-processing rejection step predominate the computational savings due to less samples in the roadmap. For $L_{gravity}$, the evaluation of the cost for a single configuration during the connection interpolations comes along with a higher computational effort as sin/cos-calculations are involved. However, as the computational costs due to the additional

It has been expected that the computation time for online motion queries decreases significantly by rejecting configurations that do not contribute to the roadmap quality. This expectation is met by the experiments in which the *ORSR-PRM* showed in average a nearly three-times faster computation time for L_{length} for a target configuration in 100 repetitive online motion queries and a nearly five-times faster computation time for $L_{gravity}$ as shown in Table I.

Fig. 5 shows a resulting robot motion calculated by *ORSR-PRM* for the described Pick-and-Place evaluation scenario.

C. Human-Centered Environment

Our framework is able to handle unexpected interactions in human-centered environments by incorporating both dynamic obstacles and external forces. Fig. 6 shows the effect of the attractor-based *Real-time Trajectory Generation* submodule in the presence of a dynamic obstacle. As the interface to the *Vision* module was not set up for dynamic obstacle tracking at the time of the experimental validation, the dynamic obstacle within the environment is simulated by an appearing orange sphere within the environment between

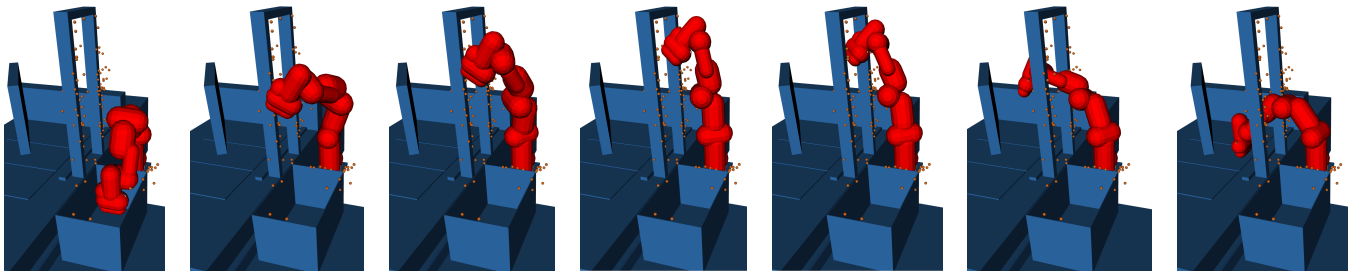


Fig. 5: Exemplary robot motion for the Pick-and-Place application using the map generated by *ORSR-PRM*.

the first and second frame. While the offline calculated trajectories would result in a collision with the previously unknown dynamic obstacle, the *Online Trajectory Generation* submodule successfully avoids the appearing obstacle despite the initially calculated infeasible path.

The incorporation of external forces due to human interaction is shown in Fig. 7. The estimated external forces acting on the robot's end effector are shown over time and applied as virtual torques to the system according to Eq. (3). The measurements are taken from the evaluation of the combined framework based on the *ORSR-PRM* and the *Real-time Trajectory Generation* submodules which is shown at: https://youtu.be/XuYyIna7-Eg.

During the experiments, we formulated an attractive potential field U_{via} to guide the robot to the next waypoint as well as repelling potential fields for obstacles U_{obs} and joint limits U_{limit}. The corresponding formulas are given in the Appendix.

IV. CONCLUSION

In this work, we presented a motion planning framework to allow the operation of a manipulator in human-centered environments. A new sampling-based planning approach, the *ORSR-PRM*, as well as an attractor-based online trajectory submodule were introduced. The considered class of problems has a finite, complex and mainly static workspace in which a robot is supposed to operate in a flexible, autonomous way. The efficient representation of the free $\mathcal{W}$-space in the generated roadmap of the *ORSR-PRM* results in up to five-times faster motion queries compared to the classic *PRM* in the experimental validation. The limitation-aware attractor dynamics ensure kinematically and dynamically feasible joint inputs. The repelling forces due to obstacles result in online adaptions of the motion plans to generate a collision-free motion even in the presence of dynamic obstacles.

It is still possible for the robot, due to the formulation based on potential fields, to get stuck in local minima. Therefore, in our future work we will include model predictive control approaches. Additionally, we plan to integrate nonlinear optimization techniques for global optimal trajectory generation in order to allow optimization objectives that depend on the robot's velocity or time.

APPENDIX

Optimization objectives representing the length are computed by

$$L_{length} = \sum_{m=1}^{M-1} ||\mathbf{q}_m - \mathbf{q}_{m+1}||_2 + ||\mathbf{x}_m - \mathbf{x}_{m+1}||_2$$

$\mathbf{x}_m$ *is the end effector position in* $\mathcal{W}$*-space corresponding to* $\mathbf{q}_m$*. Optimization objectives representing the gravity effort are computed by*

$$L_{gravity} = \sum_{m=1}^{M-1} (l_g(\mathbf{q}_m, \mathbf{q}_{m+1})||\mathbf{q}_m - \mathbf{q}_{m+1}||_2 + ||\mathbf{x}_m - \mathbf{x}_{m+1}||_2)$$

with

$$l_g(\mathbf{q}_m, \mathbf{q}_{m+1}) = \frac{1}{2}(\boldsymbol{\tau}_g(\mathbf{q}_m)^T \boldsymbol{\tau}_g(\mathbf{q}_m) + \boldsymbol{\tau}_g(\mathbf{q}_{m+1})^T \boldsymbol{\tau}_g(\mathbf{q}_{m+1})) + 1$$

such that the cost-to-go heuristic

$$L_{heuristic}(\mathbf{q}_m) = ||\mathbf{q}_m - \mathbf{q}_M||_2 + ||\mathbf{x}_m - \mathbf{x}_M||_2$$

for a milestone $\mathbf{q}_m$ *w.r.t. the goal configuration* $\mathbf{q}_M$ *is admissible.*

Formulation of the potential fields

$$U_{via}(\mathbf{q}) = \frac{1}{2}(\mathbf{q}_{via,m} - \mathbf{q})^T(\mathbf{q}_{via,m} - \mathbf{q})$$

$$U_{obs}(d_{obs}) = \begin{cases} \frac{1}{2}(d_{obs} - d_{active})^2 & \text{if } d_{obs} < d_{active} \\ 0 & \text{otherwise} \end{cases}$$

$$U_{limit}(q_i) = \begin{cases} \frac{1}{2}(q_i - q_{i,min,soft})^2 & \text{if } q_i < q_{i,min,soft} \\ \frac{1}{2}(q_i - q_{i,max,soft})^2 & \text{if } q_i > q_{i,max,soft} \\ 0 & \text{otherwise} \end{cases}$$

ACKNOWLEDGMENT

This work is supported by the *Bayerische Forschungsstiftung* (BFS, project number AZ-1318-17).

REFERENCES

[1] International Federation of Robotics (IFR), *World Robotics Industrial and Service Robots*. Frankfurt am Main: VDMA Verlag, 2019.

[2] D. Coleman, I. A. Sucan, S. Chitta, and N. Correll, "Reducing the Barrier to Entry of Complex Robotic Software: a MoveIt! Case Study," *CoRR*, vol. abs/1404.3, 2014. [Online]. Available: http://arxiv.org/abs/1404.3785

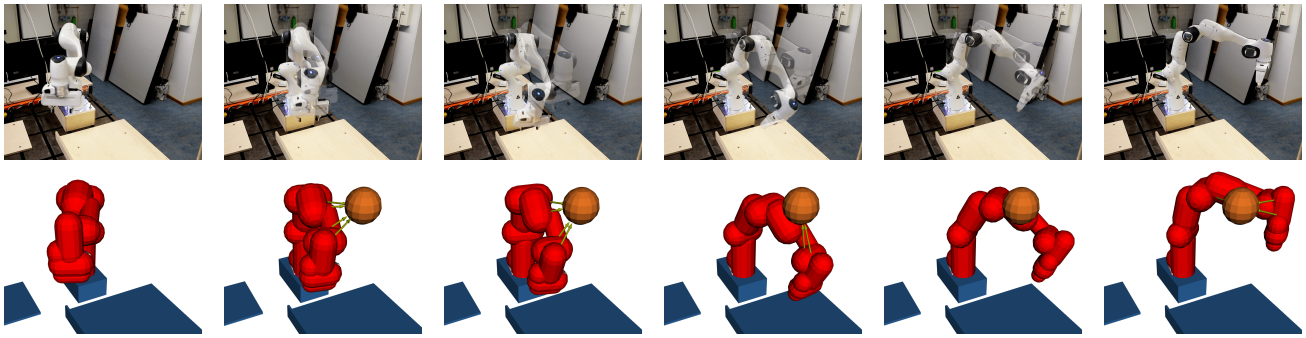


Fig. 6: Experimental validation of the *Real-time Trajectory Generation*. Each of the six frames shows a snapshots in time of a robot motion. The upper snapshots show the motion of the real robot avoiding a collision with a dynamic obstacle appearing during motion between the first and second frame. The light shaded robot in the upper snapshots show the corresponding robot motion without the collision avoidance feature. The lower snapshots show the simulated robot with the simplified environment model whose calculated joint velocity commands are forwarded to the real robot. The blue elements represent the static part of the environment. The orange sphere represents the dynamic part of the environment. The green arrows represent the distance calculations used for the potential field that are active within a defined threshold.

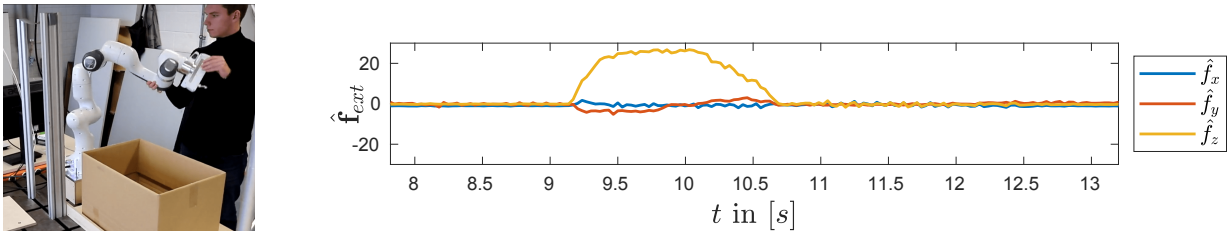


Fig. 7: Human interaction and corresponding external forces acting on the end effector over time. The forces are estimated based on proprioceptive sensor information and a dynamic robot model and are fed back to the *Real-time Trajectory Generation* submodule.

[3] "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, 2012.

[4] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: Stochastic trajectory optimization for motion planning," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 4569–4574.

[5] R. Diankov and J. Kuffner, "OpenRAVE : A Planning Architecture for Autonomous Robotics," *Robotics*, 2008.

[6] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "CHOMP: Gradient optimization techniques for efficient motion planning," in *2009 IEEE International Conference on Robotics and Automation*, 2009, pp. 489–494.

[7] D. Wahrmann, A.-C. Hildebrandt, C. Schuetz, R. Wittmann, and D. Rixen, "An Autonomous and Flexible Robotic Framework for Logistics Applications," *Journal of Intelligent & Robotic Systems*, vol. 93, no. 3, pp. 419–431, 2019. [Online]. Available: https://doi.org/10.1007/s10846-017-0746-8

[8] L. E. Kavraki, P. Svestka, J. . Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[9] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," Computer Science Dept., Iowa State University, Tech. Rep., 1998.

[10] K. Hauser, "Lazy collision checking in asymptotically-optimal motion planning," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 2951–2957.

[11] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," 2011.

[12] M. Cefalo and G. Oriolo, "A general framework for task-constrained motion planning with moving obstacles," *Robotica*, vol. 37, no. 3, p. 575–598, 2019.

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

[14] S. Haddadin, H. Urbanek, S. Parusel, D. Burschka, J. Roßmann, A. Albu-Schäffer, and G. Hirzinger, "Real-time reactive motion generation based on variable attractor dynamics and shaped velocities," in *IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems, IROS 2010 - Conference Proceedings*, 2010.

[15] Stanford Artificial Intelligence Laboratory et al., "Robotic operating system." [Online]. Available: https://www.ros.org

[16] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones, and D. Vallejo, "Obprm: An obstacle-based prm for 3d workspaces," 1998.

[17] Zheng Sun, D. Hsu, Tingting Jiang, H. Kurniawati, and J. H. Reif, "Narrow passage sampling for probabilistic roadmap planning," *IEEE Transactions on Robotics*, vol. 21, no. 6, pp. 1105–1115, 2005.

[18] V. Boor, M. H. Overmars, and A. F. van der Stappen, "The gaussian sampling strategy for probabilistic roadmap planners," in *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, vol. 2, 1999, pp. 1018–1023 vol.2.

[19] FRANKA EMIKA GmbH. (2020) Introducing the franka emika robot - soft-robot performance, smart and industry-ready. enabling automation for anyone, anywhere. [Online]. Available: https://www.franka.de/

[20] J. Pan, S. Chitta, and D. Manocha, "Fcl: A general purpose library for collision and proximity queries," in *2012 IEEE International Conference on Robotics and Automation*, 2012, pp. 3859–3866.