Circular Fields and Predictive Multi-Agents for Online Global Trajectory Planning

Marvin Becker, Torsten Lilge, Matthias A. Müller, and Sami Haddadin

***Abstract*—Safe and efficient trajectory planning for autonomous robots is becoming increasingly important in both industrial applications and everyday life. The demands on a robot which has to react quickly and precisely to changes in cluttered, unknown and dynamic environments are particularly high. Towards this end, based on the initial idea proposed in [27] we propose the Circular Field Predictions approach, which unifies reactive collision avoidance and global trajectory planning while providing smooth, fast and collision free trajectories for robotic motion planning reactive collision avoidance and global trajectory planning while providing smooth, fast and collision free trajectories for robotic motion planning. The proposed approach is inspired by electromagnetic fields, free of local minima and extended with artificial multi-agents to efficiently explore the environment. The algorithm is extensively analysed in complex simulation environments where it is shown to be able to generate smooth trajectories around arbitrarily shaped obstacles. Moreover, we experimentally verified the approach with a 7 Degree-of-Freedom (DoF) Franka Emika robot.**

***Index Terms*—Collision avoidance, motion and path planning, reactive and sensor-based planning.**

I. Introduction

With the recent shift to ever closer contact between humans and robots, the demands on motion planning are continuously increasing and classical motion planning approaches are reaching their limits. Sampling based planners received significant attraction in the field of motion planning over the last decades due to their effectiveness, good results, and straightforward implementation [1], [2]. Among those, the Rapidly-exploring Random Trees (RRT) [3], [4] and the Probabilistic Roadmap Method (PRM) [5] are probably the most widely used [6], [7]. In order to overcome the well-known disadvantages, in particular the suboptimality of the identified paths and the considerable post-processing, extensions like the well-known RRT* and PRM* methods were developed [6]. Furthermore, the extension to dynamic unknown environments was made possible with various attempts to improve the replanning efficiency [8]–[10]. However, even these efficient algorithms still suffer from a performance loss in environments with narrow passages [11].

On the other hand, reactive motion planning has drawn a lot of interest due to its simplicity and promising results. Khatib was among the first ones to develop the popular Artificial Potential Fields (APF) approach and apply it for multi-DoF manipulators [12]. Even though the algorithm only needs low computational resources, it suffers from local minima to which the robot converges instead of being able to reach the goal pose. Many approaches were developed to overcome this limitation, notably the Harmonic Functions [13] and the navigation functions [14]. Other approaches like [15]–[17] made use of the fast computation time of APF and combined the approach with sampling techniques. However, in addition to further limitations and a not negligible added complexity, those reactive approaches can mostly be used for local obstacle avoidance only and need to be combined with additional global planners. The authors of [18] developed another reactive planner derived from physics. Instead of using the analogy of electrostatic charges as in APF, they were inspired by the laws of electromagnetism and used Circular Fields (CF), which guide the robot around obstacles instead of simply repelling them. The virtual force does not induce any additional energy into the system as it acts always perpendicular to the robot's velocity and thus does not suffer from local minima either [18]. In an early similar approach the work [19] already showed the capabilities of CF by using them to evade dynamic obstacles in a simplified environment. The original algorithm was extended in [20] as it suffered from oscillations due to inconsistently defined artificial currents. Therefore, a rotation vector was introduced for each obstacle in order to define a consistent artificial current flow for each obstacle. However, full knowledge of the environment was needed in [20] since the geometric center of the obstacles had to be calculated. The approach was enhanced in [21] and the authors were able to mathematically prove obstacle avoidance of isolated CF for convex obstacles. Instead of a rotation vector, they used the projection of the robot's velocity vector on the obstacle to define a continuous artificial current, thus making it possible to use the algorithm in unknown environments. On the downside this method leads to a rather random choice of the avoidance strategy and can therefore result in trajectories that take a long detour or possibly are not even able

Manuscript received October 15, 2020; accepted February 6, 2021. Date of publication February 24, 2021; date of current version March 16, 2021. This work was supported in part by the Region Hannover in the project *roboterfabrik* and by the European Union's Horizon 2020 Research and Innovation programme as part of the projects ILIAD under Grant 732737, in part by the Lighthouse Initiative Geriatronics by StMWi Bayern Project X, under Grant 5140951, and in part by the LongLeif GaPa gGmbH Project Y, under Grant 5140953. *(Corresponding author: Marvin Becker.)*

Marvin Becker, Torsten Lilge, and Matthias A. Müller are with the Institute of Automatic Control, Leibniz University Hannover, Hannover 30167, Germany (e-mail: becker@irt.uni-hannover.de; lilge@irt.uni-hannover.de; mueller@irt.uni-hannover.de).

Sami Haddadin is with the Munich School of Robotics and Machine Intelligence and Chair of Robotics and System Intelligence, Technical University Munich, München 80333, Germany (e-mail: sami.haddadin@tum.de).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2021.3061997, provided by the authors.

Digital Object Identifier 10.1109/LRA.2021.3061997

to reach the goal at all. Further approaches like the Gyroscopic Force (GF) method [22] or motion planning using Maxwell's equations [23] are based on very similar principles. Due to its fast computation time GF was often applied in multi robot setups, where the single agents are resource-constrained like robotic sensor networks [24], unmanned aerial vehicle formations [25] or cooperative tasks [26]. Nevertheless, because of their limited exploration possibilities, CF and GF approaches can only serve as local planners and perform poorly for finding global optimal or even suboptimal solutions. The main contributions of the paper include a definition of a steering force by extending the CF approach with a Predictive multi-agent (CFP) framework which bridges the gap between global trajectory planning and reactive collision avoidance algorithms. The basic idea was introduced in [27] and implemented with a similar concept on a 7-DoF robot, however, no details were yet given. Even though CFs are inherently free of local minima, the resulting paths are in general globally suboptimal. Our extension with multi-agents solves this problem by exploring multiple paths around each obstacle. Hence, we obtain (approximately) globally optimal paths and can additionally use the multi-agents to tune the parameter settings depending on user-defined cost functions. Another contribution is the modification and enhancement of the original CF algorithm to ensure an improved interaction with the CFP and an additional attractive goal force which includes an extension such that non-convex obstacles can be handled and the CFP can be used more efficiently. Furthermore, we extensively compare our CFP against other global and local motion planners in 2D and 3D simulation environments and present a preliminary application on a 7-DoF robot.

II. Reactive Motion Planning Algorithm

The proposed virtual steering force has the form

$$\boldsymbol{F}_s = \boldsymbol{F}_{\mathrm{CF}} + k_{gr}\boldsymbol{F}_{\mathrm{VLC}} \tag{1}$$

and guides the robot to the goal pose. The definition of the obstacle avoidance force $\boldsymbol{F}_{\mathrm{CF}}$, the attractive force $\boldsymbol{F}_{\mathrm{VLC}}$, and the scaling k_{gr} are explained in detail in the next sections.

A. Circular Field Force

In order to efficiently avoid obstacles in the robot's environment, we adapt the Circular Fields (CF) algorithm, which was inspired from the physical laws in electromagnetism, specifically the force acting on a moving charged particle in an electromagnetic field.[1]

1) Current Vector: In the original approach in [18], each obstacle surface generates a virtual electromagnetic field, in which the robot behaves like a charged particle. Those electromagnetic fields are generated by virtual currents on each surface of the obstacles. The direction of the current vector $\boldsymbol{c}$ is crucial for the obstacle avoidance, as it also defines the direction of the CF force and thus the direction in which the robot is guided around an obstacle, as proven in [21]. As stated in [20], the original definition of the current vector from [18] is not sufficient as inconsistent current vectors on an obstacle lead to oscillations. In order to generate consistent current vectors we define a rotation vector ${}^j\boldsymbol{r}$ for each obstacle j similar to [20] which determines the direction of the current vectors uniformly over the entire obstacle (see below). The current vector for a surface i of an obstacle j can then be calculated by

$${}^j\boldsymbol{c}_i := {}^j\boldsymbol{n}_i \times {}^j\boldsymbol{r} \tag{2}$$

with ${}^j\boldsymbol{n}_i$ being the normal of the obstacle surface pointing outside of the obstacle. In contrast to previous approaches our definition of the current vector (2) and the rotation vector (3) does not depend on prior knowledge of the environment as in [20], leads to a continuous current direction over the surfaces of the obstacles (in contrast to [18]) and can be easily used to explore multiple trajectories to evade obstacles without depending on the current robot velocity (in contrast to [21]).

2) Rotation Vector: The rotation vector is a key element in our multi-agent framework as it is used to calculate the current vectors and therefore defines the direction in which the robot will pass an obstacle. The rotation vector only defines the orientation of the magnetic field and does not influence the orientation of the robot. Due to our predictive multi-agent approach multiple possibilities of evading an obstacle are evaluated in our framework. Therefore, we prioritized a computationally cheap and robust calculation of the first rotation vector instead of opting for more sophisticated solutions. The calculation of further rotation vectors is part of the multi-agent framework and therefore is described in detail in Section III. The rotation vector of an obstacle is calculated once when the obstacle appears for the first time in the vicinity of the robot by generating a normal $\boldsymbol{n}_{\mathrm{ref}}$ to the current normalized velocity vector $\bar{\boldsymbol{v}} = \frac{\dot{\boldsymbol{x}}}{\|\dot{\boldsymbol{x}}\|}$ of the robot. In $\mathbb{R}^3$ a method for calculating a normal to an arbitrary vector is to take the cross product of the vector with a basis vector of a reference coordinate frame. By picking the basis vector that yields in the smallest magnitude of the scalar product with the chosen vector a higher numerical stability is achieved.

$$\boldsymbol{n}_{\mathrm{ref}} := \begin{cases} \boldsymbol{e}_x \times \bar{\boldsymbol{v}} & \text{if } \boldsymbol{e}_x \cdot \bar{\boldsymbol{v}} = \min\{\boldsymbol{e}_x \cdot \bar{\boldsymbol{v}}, \boldsymbol{e}_y \cdot \bar{\boldsymbol{v}}, \boldsymbol{e}_z \cdot \bar{\boldsymbol{v}}\} \\ \boldsymbol{e}_y \times \bar{\boldsymbol{v}} & \text{if } \boldsymbol{e}_y \cdot \bar{\boldsymbol{v}} = \min\{\boldsymbol{e}_x \cdot \bar{\boldsymbol{v}}, \boldsymbol{e}_y \cdot \bar{\boldsymbol{v}}, \boldsymbol{e}_z \cdot \bar{\boldsymbol{v}}\} \\ \boldsymbol{e}_z \times \bar{\boldsymbol{v}} & \text{if } \boldsymbol{e}_z \cdot \bar{\boldsymbol{v}} = \min\{\boldsymbol{e}_x \cdot \bar{\boldsymbol{v}}, \boldsymbol{e}_y \cdot \bar{\boldsymbol{v}}, \boldsymbol{e}_z \cdot \bar{\boldsymbol{v}}\}, \end{cases}$$
$$\boldsymbol{r} := \bar{\boldsymbol{v}} \times \boldsymbol{n}_{\mathrm{ref}}, \tag{3}$$

where $\boldsymbol{e}_x, \boldsymbol{e}_y, \boldsymbol{e}_z$ are the basis vectors of the map coordinate frame. When the robot is not moving, e.g. in the initial pose, the rotation vector is instead calculated by replacing the normalized velocity vector with the vector $\boldsymbol{d}_g = \boldsymbol{x}_g - \boldsymbol{x}$ pointing from the robot position $\boldsymbol{x}$ to the goal position $\boldsymbol{x}_g$. Please note that in a 2D environment there are only two possible ways for the robot to bypass an obstacle, the left or the right side. Therefore, the rotation vector simply matches the z-Axis (or the negative z-Axis) as in 1.

3) Circular Field: Our modified version of the Biot-Savart law leads to the circular field resulting from surface i of

[1] According to the law of Biot-Savart the magnetic field at position $\boldsymbol{x}$ of a wire of length l carrying the current I is defined by $d\boldsymbol{B}(\boldsymbol{x}) = \frac{\mu_0}{4\pi}\frac{Idl\times\boldsymbol{x}}{\|\boldsymbol{x}\|^3}$ and will apply the Lorentz force $\boldsymbol{F} = q\dot{\boldsymbol{x}} \times \boldsymbol{B}$ on a particle charged with q and moving with velocity $\dot{\boldsymbol{x}}$.

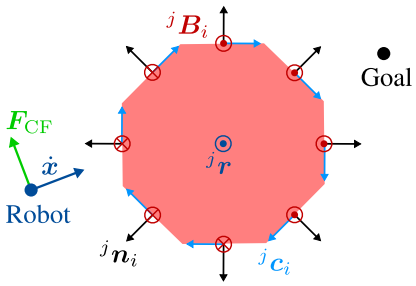


Fig. 1. Generation of circular field (dark red) and CF force (green) for an octagonal obstacle (light red) in 2D.

obstacle j

$$ {}^{j}\boldsymbol{B}_i := \frac{k_{\mathrm{CF}}}{\|{}^{j}\boldsymbol{d}_i\|} {}^{j}\boldsymbol{c}_i \times {}^{j}\dot{\boldsymbol{x}}_i, \tag{4} $$

where k_{CF} is a constant gain, ${}^{j}\dot{\boldsymbol{x}}_i$ is the relative velocity, between robot and obstacle surface and $\|{}^{j}\boldsymbol{d}_i\|$ is the minimal distance between robot and obstacle surface.

4) Artificial Force: In order to reduce disturbances from obstacle points which are not relevant for the motion planning the planner will only take those obstacle points into account that are inside of a range limit d_l around the robot and located on surfaces which face the robot. The first requirement is easily met by defining the CF force as follows

$$ {}^{j}\boldsymbol{F}_{\mathrm{CF},i} = \begin{cases} {}^{j}\dot{\boldsymbol{x}}_i \times {}^{j}\boldsymbol{B}_i & \text{if } \|{}^{j}\boldsymbol{d}_i\| \leq d_l \\ 0 & \text{if } \|{}^{j}\boldsymbol{d}_i\| > d_l. \end{cases} \tag{5} $$

The second requirement is met when the absolute value of the angle between the obstacle normal $\boldsymbol{n}$ at the obstacle point and the robot-obstacle distance vector $\boldsymbol{d}$ is greater than $90°$, i.e. ${}^{j}\boldsymbol{n}_i \cdot {}^{j}\boldsymbol{d}_i < 0$. The total CF force from n_o obstacles with m_j obstacle surfaces then results in

$$ \boldsymbol{F}_{\mathrm{CF}} = \sum_{j=0}^{n_o} \sum_{i=0}^{m_j} {}^{j}\boldsymbol{F}_{\mathrm{CF},i}. \tag{6} $$

In contrast to the APF approach our CF planner has multiple advantages. The force is perpendicular to the robot's velocity, thus it does not dissipate any energy from the system and consequently will not change the stability property of attractive fields when no collision with obstacles occurs [18]. Moreover, the planner does not suffer from local minima and when the robot is moving perpendicular to an obstacle surface the CF will not apply any force on the robot.

B. Attractive Goal Force

In order to guide the robot to its goal pose we extend the definition of the attractor dynamics in the classical potential field approach with the proposed Velocity Limiting Controller (VLC) from [12]. For this, we first need to define an artificial desired velocity from the current robot position $\boldsymbol{x} \in \mathbb{R}^3$ and its goal position $\boldsymbol{x}_g \in \mathbb{R}^3$ in the form $\dot{\boldsymbol{x}}_d = \frac{k_p}{k_v}(\boldsymbol{x}_g - \boldsymbol{x})$, where k_p is the position gain and k_v the velocity gain. The virtual force $\boldsymbol{F}_{\mathrm{VLC}}$ is then calculated from the difference of the current robot velocity $\dot{\boldsymbol{x}}$ and the artificial desired velocity $\dot{\boldsymbol{x}}_d$ with $\boldsymbol{F}_{\mathrm{VLC}} = -k_v (\dot{\boldsymbol{x}} - \nu \dot{\boldsymbol{x}}_d)$, where the factor ν leads to the limitation of the velocity magnitude and is defined as $\nu = \min(1, v_{\max}(\dot{\boldsymbol{x}}_d^T \dot{\boldsymbol{x}}_d)^{-\frac{1}{2}})$.

The resulting control law is better suited for longer distances between robot and goal pose since the generated virtual force vanishes when the robot travels with the maximum velocity in the direction of the goal pose. This leads to a constant velocity magnitude except during acceleration, deceleration and in the vicinity of obstacles when the robot is subject to further virtual forces. We use the same approach for the orientation of the robot by computing an artificial desired angular velocity from the orientation error $\boldsymbol{x}_{g,r} - \boldsymbol{x}_r$ by using the quaternion difference of the current orientation $\boldsymbol{q}$ of the robot and the goal orientation $\boldsymbol{q}_g$ as proposed in [28] with $\boldsymbol{x}_{g,r} - \boldsymbol{x}_r = q_0 \boldsymbol{q}_g - q_{g,0} \boldsymbol{q} - \boldsymbol{q} \times \boldsymbol{q}_g$, where q_0 and $q_{g,0}$ are the scalar parts of the quaternions describing the current orientation and the goal orientation of the robot. Please note that the orientation is not used for the collision avoidance.

C. Extensions and Application

This section focuses on further improvements and extensions of our algorithm. In contrast to previous CF approaches, the proposed algorithm should be able to work directly with point clouds to avoid the computationally intensive and error-prone segmentation of surfaces from the obstacles. Therefore, instead of obstacle surfaces, each of the point cloud points generates its own magnetic field. This also yields the advantage that the robot tends to be repelled stronger by larger obstacles and busier areas. The computational load can then be easily adjusted by downsampling the sensor point cloud. We assume that the sensor data originates from laser scanners or camera modules and make the assumption that the point cloud points are reasonably evenly distributed. The parameter k_{CF} can be scaled according to the point cloud resolution, i.e. a larger distance between the points in a cloud should result in a higher parameter k_{CF} to achieve similar trajectories. Moreover, as noted by [21], the combination of an APF and CF could lead to oscillations of the robot, or even goal convergence problems when large obstacles need to be avoided. We introduce similar scaling factors for avoiding this effect by reducing the attractive goal force when the robot is close to obstacles (factor w_1), when the goal is obscured by an obstacle (factor w_2) and when the distance to the goal grows large (factor w_2); for details see the goal relaxation term in [21]. In addition to these factors, we introduce another extension, which leads to significantly improved trajectories, in particular in environments with non-convex obstacles. Obstacle configurations which cause opposing VLC and CF forces can lead to a reduction of the robot's velocity as the goal force acts against the robot's velocity. In such a case the CF force should be dominating to guide the robot along the obstacles' boundaries until the obstacle is passed. This can be achieved by scaling the VLC force with the scalar product of the normalized VLC force and the current robot velocity in the form

$$ w_4 = \begin{cases} 1 + \frac{\dot{\boldsymbol{x}} \cdot \boldsymbol{F}_{\mathrm{VLC}}}{\|\dot{\boldsymbol{x}}\| \|\boldsymbol{F}_{\mathrm{VLC}}\|} & \text{if } \dot{\boldsymbol{x}} \cdot \boldsymbol{F}_{\mathrm{VLC}} < 0 \text{ and } \boldsymbol{F}_{\mathrm{CF}} \neq 0 \\ 1 & \text{otherwise.} \end{cases} $$

The overall factor is then calculated by $k_{gr} = w_1 w_2 w_3 w_4$.

III. Predictive Multi-Agent Framework

In order to overcome the limitations of traditional CF, achieve better global exploration and therefore find a more reasonable motion planning strategy, we extend our approach with predictive multi-agents, which are able to efficiently plan multiple trajectories to the goal pose. In our definition, a predictive agent simulates the robot in an environment snapshot and is guided by the same steering force as the real robot to reach the target state. The state space representation of an agent is defined as $\dot{\boldsymbol{z}} = f(\boldsymbol{z}, \boldsymbol{u}, \boldsymbol{P})$ with f, the robot dynamics under the influence of the CF and VLC forces, $\boldsymbol{z} = [\boldsymbol{x}\ \dot{\boldsymbol{x}}]^T$ the agent state, $\boldsymbol{x}, \dot{\boldsymbol{x}} \in \mathbb{R}^6$ the robot pose and velocity in the task space, $\boldsymbol{u} \in \mathbb{R}^6$ the interaction between agent and environment, and $\boldsymbol{P}$ the parameters of the agent. The parameter vector $\boldsymbol{P}$ is crucial in our approach and differs for each virtual agent allowing to efficiently explore the environment. It is defined as $\boldsymbol{P} = \{\boldsymbol{R}, k_{\mathrm{CF}}, d_l, d_s, k_p, k_v, v_{\max}\}$ and comprises the following parameters, where more parameters can easily be added:

- The Rotation vector matrix $\boldsymbol{R} = (\boldsymbol{r}_0, \ldots, \boldsymbol{r}_{\mathrm{n}-1})$ contains a rotation vector $\boldsymbol{r}_j$ for each obstacle j which was detected by the real robot and is therefore of order $n_o \times 3$ with n_o being the number of detected obstacles.
- The CF gain k_{CF} scales the magnitude of the CF. A higher value leads to a more obstacle sensitive robot.
- Within the region of interest d_l around the robot the planner evaluates obstacles for force generation with CF.
- The safety margin d_s limits the paths that agents can explore. If the passage between two obstacles is more narrow than twice this closing distance, the algorithm avoids to guide agents between them. An agent which violates the safety margin by coming closer to an obstacle than this distance is considered as collided.
- The VLC parameters $k_p, k_v, v_{\max}$ represent gain and damping factor of the VLC and the maximum velocity at which the agent is allowed to travel.

During a planning step, multiple predictive agents with different dynamical parameters are generated to explore the environment. Each agent is evaluated at equidistant sampling times, the agent with the lowest cost is selected and its parameters are copied by the real robot. While the predictive agents are being simulated, the real robot is moving under the influence of the CF and the VLC using the parameters of the current best agent instead of just following the trajectory of this agent. This approach ensures that the real robot can react quickly and robustly to events that could not be taken into account by the predictive agents, such as sudden changes in direction of dynamic obstacles or measurement errors. The procedure of the predictive agent simulation and planning can be described with the following steps:

1) Initialization: Set the start configurations of one or multiple agents at $\boldsymbol{z}_0$, which is the initial state of the robot. These agents are generated with different parameters $\boldsymbol{P}$. They share the knowledge of the current state of the environment with the number and pose of all obstacles in the field of view of the robot.

1) Exploration: Start the exploration of the agents. Each agent is attracted to the goal pose by the VLC and guided around obstacles by the CF.

1) New Agent Generation: In case an agent enters the limit distance of a new obstacle, n_a new agents with different parameter vectors are created if the maximum number of agents $n_{a,\max}$ has not yet been reached. These agents inherit state and parameter vector of the parent agent. Now, specific components of the child agents' parameter vectors, in particular the rotation vector of the new obstacle, are modified in order to explore other paths around the new obstacle. Please note that the rotation vectors for other known (possibly already passed) obstacles are not changed. Due to the importance of the rotation vector matrix $\boldsymbol{R}$ for the obstacle avoidance strategy, we focus on modifying the rotation vector and set $\boldsymbol{P} = \{\boldsymbol{R}\}$ for the rest of the paper. The generation of new agents in 2D environments is depicted in Fig. 2.

In this case, only one new agent with an opposing rotation vector is generated, while the rotation vectors for new agents in 3D environments are created by rotating the original rotation vector around the normal vector of the surface closest to the robot. In the unlikely case of a rotation vector exactly matching the normal vector, a new vector is created by adding a small positive constant to the normal, then the rotation vector is rotated around this new vector. The amount of the rotation angle $\theta_p = p\frac{2\pi}{n_a+1}$ for a new agent $p \in [1, 2, \ldots, n_a]$ is defined by the number of all new agents n_a. The rotation vector of a new agent p for an angle θ_p around the obstacles' normal $\boldsymbol{n}$ can be obtained by

$$\boldsymbol{r}_p = \boldsymbol{r}_0 \cos\theta_p + (\bar{\boldsymbol{n}} \times \boldsymbol{r}_0) \sin\theta_p + \bar{\boldsymbol{n}}\,(\bar{\boldsymbol{n}} \cdot \boldsymbol{r}_0)\,(1 - \cos\theta_p)$$

where $\bar{\boldsymbol{n}} = \frac{\boldsymbol{n}}{\|\boldsymbol{n}\|}$ is the normalized normal vector and $\boldsymbol{r}_0$ the original rotation vector [29]. Once all parameters of the new agents are defined, they are immediately used for further environment exploration. Other parameters of the robot can be modified and sampled accordingly to further refine the motion planning strategy.

1) Evaluation: The trajectories of all agents are evaluated after a defined time t_e or if all agents either reached the goal pose or are considered as collided. The cost function by which each agent p is evaluated includes the following criteria, weighted by the factors $k_{cl}, k_{cd}, k_c > 0$.

(1) With the position $\boldsymbol{x}_{a,p}(t)$ of agent p, the path length of the trajectory the agent travelled is

$$c_{\mathrm{length},p}(t_e) = k_{cl} \int_{t=0}^{t_e} \|\dot{\boldsymbol{x}}_{a,p}(t)\| dt.$$

(2) When the computation time is not sufficient for an agent to reach the goal, a simple heuristic for the distance to the goal is included in the form

$$c_{\mathrm{dist},p}(t_e) = k_{cd}\,(\boldsymbol{x}_g(t_e) - \boldsymbol{x}_{a,p}(t_e))$$

(3) In order to improve the safety and robustness of the motion planner the minimal distance

$$c_{\mathrm{obst},p}(t_e) = k_{co} \min_{\forall t \in [0,t_e], i \in [0,n_o]} (\|\boldsymbol{x}_{a,p}(t) - \boldsymbol{x}_{o,i}(t)\|)\,.$$

between an agent and all obstacles is included in the cost function. Thus, agents which pass obstacles with a greater distance are preferred.

The overall cost c_t can then be calculated by

$$c_t(t_e) = c_{\mathrm{length}}(t_e) + c_{\mathrm{dist}}(t_e) + c_{\mathrm{obst}}(t_e) \tag{7}$$

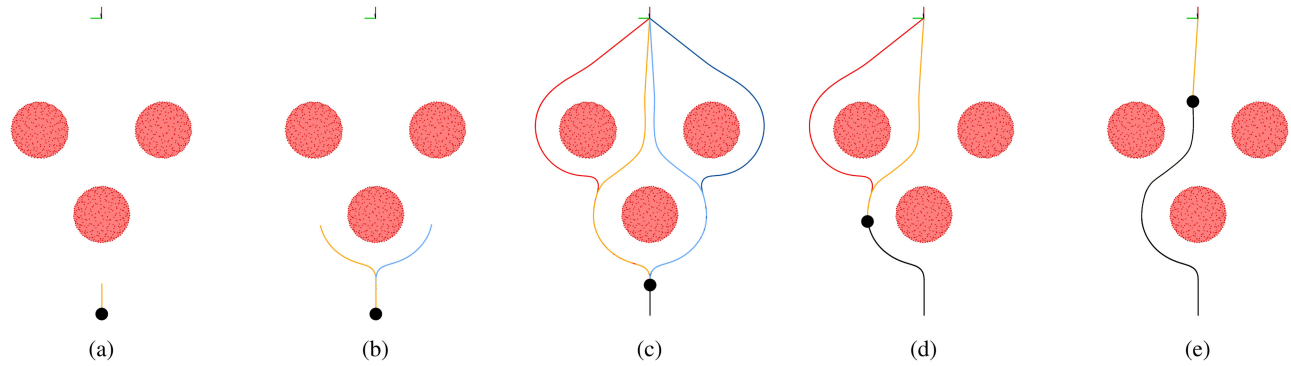


Fig. 2. Procedure of the agent creation in 2D environments. The first agent (orange) starts planning in Fig. 2(a). When it reaches the limit distances of the first sphere in Fig. 2(b), a new agent (light blue) is created. When the two agents meet the following obstacles in Fig. 2(c) two more agents (blue and red) are created. Meanwhile, the real robot depicted in black starts moving. When it reaches the first obstacle and chooses the trajectories of the red and orange agents, the other obsolete agents are deleted (Fig. 2(d)). The robot will use the last agent's parameters (orange) until it reaches the goal (Fig. 2(e)).

The agent with the lowest cost is selected and its parameter vector is copied to the real robot. Obviously, the cost function and the parameter vector can be easily extended to further evaluation criteria, like, e.g., complexity of motion on joint level, energy consumption, level of occlusions (and therefore unknown environment) on the trajectory of the agent, etc.

2) Removing Obsolete Agents: In order to save computational resources, agents which are considered as collided are deleted after the evaluation step. Additionally, when the real robot enters the limit distance of an obstacle, all agents which took a different path around this new obstacle than the current best agent are deleted, too. This also prevents the real robot from oscillating movements when the best predictive agent changes after the real robot has already started following a different path around an obstacle. The procedure of the algorithm is depicted in Fig. 2.

IV. Results and Analyses

In this section we compare our CFP algorithm with other motion planning approaches in challenging environments in order to evaluate the offline planning capabilities and the performance during online execution. The simulations implemented with C++ in the Robot Operating System (ROS) framework [30] on a computer with an Intel Core i9-9880H CPU, 2.30 GHz and 16 GB of memory. All results including the exploration of our CFP are shown in a video attachment.

A. Problem Statement

Given a holonomic robot with initial pose $\boldsymbol{x}_{r_0} \in \mathbb{R}^6$ in a 6D environment with cluttered static obstacle objects $\mathcal{O}$, and a goal location $\boldsymbol{x}_{\mathrm{g}} = const$, the motion planner has to find a viable trajectory in the task space along with command inputs (i.e. acceleration $\ddot{\boldsymbol{x}}_d$) for allowing the robot to navigate and avoid obstacles in the environment while obeying its physically based dynamics (i.e. $\|\dot{\boldsymbol{x}}_r\|, \|\ddot{\boldsymbol{x}}_r\| \leq v_{\max}, a_{\max}$). An obstacle object j is described by a finite set $\mathcal{O}_j \subseteq \mathbb{R}^3$ of i obstacle points $\boldsymbol{x}_{o_{ji}} \in \mathcal{O}_j$. We assume perfect knowledge of the completely static environment in the simulations, i.e., the position of each obstacle point is known and the robot can immediately see the entirety of an obstacle. Please note that complete knowledge of the position, shape and size of each obstacle is not necessary for our approach but naturally enhances the resulting final trajectories and is used to demonstrate the full capabilities of the predictive agent approach. The point-like robot has the simple dynamic model

$$m\ddot{\boldsymbol{x}}_d = \boldsymbol{F}_s \tag{8}$$

with m, the mass of the robot, and $\boldsymbol{F}_s$, the steering force from CFP and VLC exerted on the robot as defined in Eq. (1).

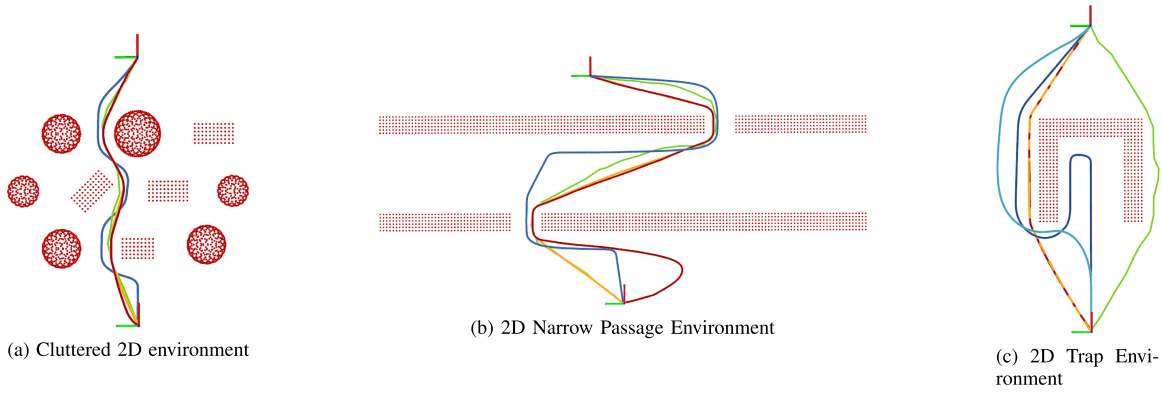
B. Simulations

1) Planning Evaluation: In order to evaluate the planning capabilities of the CFP planner, we compare the predicted trajectories with the popular sampling based RRT* approach from [6] and the extended Timed-Elastic-Bands (TEB) algorithm from [31]. Although the TEB is only available in 2D, it takes a similar approach to avoiding locally optimal solutions as the CFP by also generating candidate trajectories and is thus very well suited for comparison. The evaluation criteria include the average planning time and path length of the best trajectory and the first trajectory that was found to the goal pose. Moreover, we include the average amount of collision free trajectories to the goal per simulation. All approaches were able to find at least one collision free trajectory in each try. We used the ROS integration of the TEB planner [32] with the default temporal resolution of the trajectory 0.3s. In order to get comparable planning times, we stopped the TEB when the path length started to decrease by less than 0.1m in 1s. Similarly, the RRT* was terminated after 10 000 iterations and its discretization distance, i.e. the distance between two nodes of the tree, was set to 0.15m. The simulation of the predictive agents in our CFP planner was conducted with a temporal resolution of 0.2s and $n_a = 3$ new agents per obstacle with a maximum of 40 agents in 3D ($n_a = 1$ with no limit on the agents in 2D). All simulations were executed 100 times. As can be seen in Table I the CFP outperforms the RRT* in all environments and the TEB in all but the trap environment in terms of planning time. Even in the cluttered environment, where the algorithm generated 211 agents, the best trajectory was found on average in 52ms, which is faster than the average human reaction time (150ms to 300ms [33]). Although the TEB

TABLE I
PLANNING SIMULATION RESULTS

Environment	Dim	Approach	Best Trajectory Average		First Trajectory Average		N_R[3]
			L[1] [m]	T_c[2] [ms]	L[1] [m]	T_c[2] [ms]	
Cluttered	2D	CFP	8.28 ± 0.00	52 ± 12	12.78 ± 1.13	37 ± 14	211
		TEB	7.62 ± 0.00	246 ± 12	8.67 ± 0.00	44 ± 15	11
		RRT*	7.81 ± 0.11	417 ± 74	9.45 ± 1.30	67 ± 52	16.8
	3D	CFP	12.59 ± 0.00	125 ± 18	14.29 ± 0.86	67 ± 7	40
		RRT*	14.07 ± 0.97	1321 ± 939	15.08 ± 1.16	74 ± 51	4.3
Narrow Passage	2D	CFP	17.03 ± 0.00	144 ± 3	27.7 ± 0.0	64 ± 1	9
		TEB	15.10 ± 0.01	1731 ± 51	22.38 ± 0.00	60 ± 2	2
		RRT*	16.42 ± 0.49	851 ± 174	17.97 ± 0.64	137 ± 21	12.8
	3D	CFP	18.20 ± 0.00	189 ± 8	21.89 ± 5.05	188 ± 8	25
		RRT*	21.66 ± 3.25	1267 ± 1118	23.72 ± 3.24	974 ± 160	2.5
Trap	2D	CFP LR[4]	11.82 ± 0.00	44 ± 16	11.82 ± 0.00	44 ± 16	2
		TEB	10.04 ± 0.00	16 ± 10	10.04 ± 0.00	16 ± 10	2
		RRT*	10.67 ± 0.08	298 ± 47	12.99 ± 0.64	61 ± 21	31.1
	3D	CFP LR[4]	11.83 ± 0.00	86 ± 2	11.83 ± 0.00	86 ± 2	4
		RRT*	12.02 ± 0.63	1335 ± 385	15.69 ± 1.17	94 ± 29	10.7

1] L = Path length 2] T_c = Computation time 3] N_R = Average trajectories per run 4] LR = Long Range



(a) Cluttered 2D environment

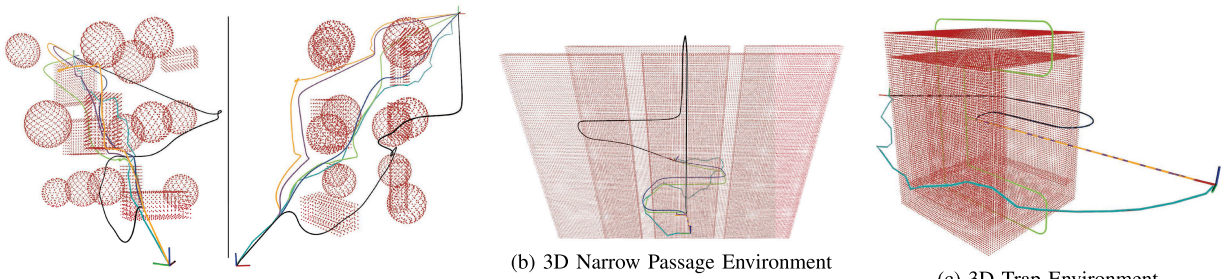
(b) 2D Narrow Passage Environment

(c) 2D Trap Environment

Fig. 3. Simulation environments with the final paths from the RRT* planner (green), the TEB planner during planning (yellow) and during execution (red) and the proposed CFP planner with an obstacle range of 0.8m in dark blue and with 2.0m in light blue. (a) Cluttered 2D environment. (b) 2D Narrow Passage Environment. (c) 2D Trap Environment.

and the RRT* were able to find shorter best trajectories, the chosen paths are all very similar as can be seen in Fig. 3 and the CFP was always able find the globally optimal direction around an obstacle. Please note that the TEB is designed as a local planner and has problems with big obstacles as, e.g., in the narrow passage environment. Therefore, we restricted the amount of distinctive trajectories for this environment to a maximum of two. Even with this constraint, the TEB takes a comparatively long time to find the direct route to the goal. In order to get a comparable path length for the trap environment, we performed the CFP simulations with a short range (where the robot is temporarily guided inside the obstacle) and a long range (which leads to an evasion of the trap), i.e. $d_l = 0.8m$ and $d_l = 2.0m$. In summary, the experiments show that our algorithm outperforms the RRT* in almost all aspects, while achieving comparable results to the TEB despite slightly longer path lengths.

2) Execution Evaluation: Next, we evaluate the online planning and execution performance of the CFP compared to the TEB in 2D and against reactive algorithms in additional 3D environments. In particular, we evaluate the classic APF method, the most recent Magnetic-Field-Inspired (MFI) approach [21], our CF without multi-agents (CF) and the GF from [26]. We modified the reactive approaches by combining them with the same VLC as in our approach to achieve a constant velocity profile and add a method for reaching the desired orientation. The 2D online simulations show that both approaches are able to find collision free trajectories to the goal pose but as depicted in Fig. 3 the TEB generates less optimal trajectories when no offline planning time is provided. This is particularly apparent in Fig. 3(b) where the TEB first follows the path around the outside of the obstacles before switching to the shorter path through the narrow passages. Additionally, the computation times for one execution step for the TEB are about two orders of magnitude higher compared to the reactive algorithms, see Table II. As expected, the APF has in general the fastest computation time whereas the other reactive algorithms are comparable. The CFP is actually faster on average than the CF without multi-agents, presumably since the force on the real robot is calculated in parallel to the predictive agents and does not need to perform the rather expensive calculations for generating the current vectors as those are only passed by the best agent. On the other hand, the CFP computation time has a relatively high standard deviation and a higher worst-case computation time, which probably originates from the longer time steps when agents are generated or removed and when the calculation of the predictive agents



(a) Cluttered 3D environment from two perspectives

(b) 3D Narrow Passage Environment

(c) 3D Trap Environment

Fig. 4. Simulation environments with the final paths from the APF planner (yellow), the MFI approach (green), the GF planner (purple), the CF without multi-agents (black), the RRT* (light blue) and the proposed CFP planner (blue). (a) Cluttered 3D environment from two perspectives. (b) 3D Narrow Passage Environment. (c) 3D Trap Environment.

TABLE II
EXECUTION SIMULATION RESULTS

Environment	Dim	Approach	L[1] [m]	T_g[2] [s]	T_c[3] [ms]
Cluttered	2D	CFP	8.31	24.81	0.45 ± 0.37
		TEB	7.53	18.84	59.17 ± 46.01
	3D	CFP	12.31	31.99	0.19 ± 0.45
		CF	20.82	52.51	0.30 ± 0.26
		APF	15.05	50.53	0.16 ± 0.07
		GF	13.84	46.85	0.27 ± 0.05
		MFI	13.61	37.31	0.19 ± 0.06
Narrow Passage	2D	CFP	17.04	44.73	0.31 ± 0.32
		TEB	18.55	42.60	35.84 ± 15.94
	3D	CFP	18.12	45.08	0.18 ± 0.49
		CF	35.37	99.39	0.43 ± 0.21
		APF	-	-	0.15 ± 0.06
		GF	-	-	0.44 ± 0.16
		MFI	18.15	51.77	0.11 ± 0.05
Trap	2D	CFP LR[5]	11.82	25.65	0.21 ± 0.31
		TEB	10.03	18.49	20.50 ± 14.61
	3D	CFP SR[4]	16.01	39.50	0.18 ± 0.52
		CF	15.97	37.90	0.33 ± 0.31
		APF	-	-	0.16 ± 0.08
		GF	-	-	0.41 ± 0.27
		MFI	-	-	0.50 ± 0.26

1] L = Path length 2] T_g = Time to goal 3] T_c = Step computation time 4] SR = Short Range 5] LR = Long Range

is blocking. The higher computation time of the 3D cluttered environment compared to the 2D equivalent stems from the different amount of generated agents. We limited the agents in the 3D case to 40 and did not set a maximum for the 2D simulation, where over 200 agents were generated. The first 3D simulation setting is a cluttered environment with 18 obstacles of different sizes and forms, see Fig. 4(a). All approaches were able to find a collision free trajectory and converge to the goal pose. The APF approach suffers only marginally from its typical oscillations, e.g., before the last obstacle in Fig. 4(a), has the fastest computation time, but leads to a long execution time and to the least smooth trajectory. The MFI performs well both in terms of path length and execution time and is only slightly outperformed by our CFP approach, see Table II. The CF generates the longest and slowest trajectory as obstacles are bypassed in rather random directions. For the second simulation, depicted in Fig. 4(b), our CFP and the MFI approach achieve the best results with similar path lengths. The MFI trajectory leads to higher execution times as the approach significantly decreases the robot's velocity when moving away from the goal due to the missing w_4 factor. Additionally, it has to be noted that the goal was chosen deliberately in the same direction as the passage between the obstacles to achieve a fair comparison against the MFI approach. If the goal is chosen such that the robot would start moving to the right, the MFI approach would guide the robot the long way around the obstacle. The APF approach was not able to reach the goal as it gets stuck in a local minimum. The GF approach uses a random perturbation when the goal and the obstacle distance vector are aligned, but this is not sufficient to overcome such large obstacles. In the last simulation environment we included a non-convex 3D-trap between the initial and the goal pose. The sensing range of the reactive approaches, i.e. the region of interest d_l was set such that the simulated robots were not able to detect the obstacle walls when entering the trap. The APF and the GF approach were again not able to reach the goal and we were also not able to find suitable parameters for achieving a goal convergence with the MFI approach as it was either trapped inside the obstacle or suffered from the caching effect of the magnetic fields which was also described in [20], see Fig. 4(c). On the contrary, the non-convexity of the obstacle does not pose problems to the CF and CFP approaches, which are able to guide the simulated robot to the goal pose. We also briefly tested our planner in dynamic environments. As can be seen in the video, the CFP is sufficiently fast to handle such environments even without any method for predicting the obstacle movements. A more detailed study of dynamic obstacles is subject of ongoing work.

C. Experiments

We also validated our algorithm by controlling the endeffector of the Franka Emika robot. The robot's endeffector had to move from start (red) to end pose (green) while evading two known obstacles (a desktop PC and a hanging electrical outlet) as shown in Fig. 5. The controlled endeffector frame was moved upwards along the endeffector's z-Axis into the Franka Emika's flange and a collision sphere of 0.12m was added to evade the obstacle (depicted by the spheres in Fig. 5). Using point mass dynamics (8), the steering forces from the CFP algorithm were transformed into velocity control signals which were passed to a Cartesian velocity motion generator and then applied by Franka Emika's internal Cartesian impedance controller. As can be seen

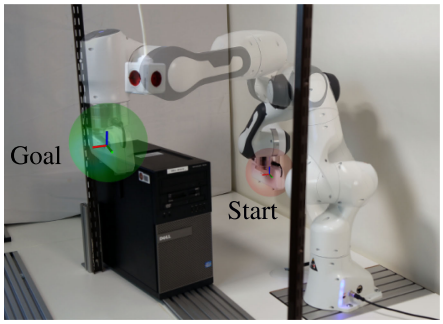


Fig. 5. Experiment on the collaborative Franka Emika robot.

in the video attachment, the CFP was able to guide the robot's endeffector safely and smoothly to the target pose while obeying the maximum velocity of 0.1m/s.

V. Limitations and Conclusion

In this letter, we proposed a motion planning approach which unifies local reactive collision avoidance and global trajectory planning for generating smooth trajectories around arbitrarily shaped obstacles in known and unknown environments. We significantly enhanced previous magnetic-field-inspired approaches and extended them with predictive multi-agents to efficiently explore the environment. Although the evaluation could show a fast computation time and demonstrated the general applicability in dynamic environments, the algorithm should still be extended for this use case, e.g, prediction models for the dynamic obstacles and more sophisticated methods for switching the best agent could significantly improve the planning quality. Furthermore, we only implemented simple point mass dynamics without considering more complex robot dynamics or full body obstacle avoidance for a robot. Another drawback of the planner is the missing optimality regarding the path length. Depending on the number of agents per obstacle and the obstacle form, none of the generated agents could lead to a satisfying trajectory. The planner could also be too slow to fully converge to the goal before reaching the first obstacle and then choose a suboptimal path. In this case, the planner is currently not able to detect that it chose a suboptimal path and therefore cannot switch to a more suitable agent to change the direction. Current research addresses these limitations and also aims for a theoretical proof of obstacle avoidance and goal convergence.

References

[1] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2006.
[2] D. Connell and H. M. La, "Extended rapidly exploring random tree-based dynamic path planning and replanning for mobile robots," *Int. J. Adv. Robot. Syst.*, vol. 15, no. 3, 2018.
[3] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning: Tech. Rep,"
[4] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2000, vol. 2, pp. 995–1001.
[5] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Autom.*, vol. 12, no. 4, pp. 566–580, Aug. 1996.
[6] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res*, no. 7, pp. 846–894, 2011.
[7] M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review," *IEEE Access*, vol. 2, pp. 56–77, 2014.
[8] J. Bruce and M. M. Veloso, "Real-time randomized path planning for robot navigation," in *Proc. Robot Soccer World Cup*, 2002, pp. 288–295.
[9] D. Ferguson, N. Kalra, and A. Stentz, "Replanning with RRTs," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2006, pp. 1243–1248.
[10] M. Otte and E. Frazzoli, "RRTX: Asymptotically optimal single-query sampling-based motion planning with quick replanning," *Int. J. Robot. Res.*, vol. 35, no. 7, pp. 797–822, 2016.
[11] V. Vasilopoulos *et al.*, "Reactive semantic planning in unexplored semantic environments using deep perceptual feedback," *IEEE Robot. Autom. Lett.*, vol. 5, no. 3, pp. 4455–4462, Jul. 2020.
[12] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Int. J. Robot. Res.*, vol. 5, no. 1, pp. 90–98, 1986.
[13] C. I. Connolly, J. B. Burns, and R. Weiss, "Path planning using laplace's equation," in Proc. IEEE Int. Conf. Robot. Autom., 1990, pp. 2102–2106.
[14] E. Rimon and D. E. Koditschek, "Exact robot navigation using artificial potential functions," *IEEE Trans. Robot. Autom.*, vol. 8, no. 5, pp. 501–518, Oct. 1992.
[15] J. Barraquand and J.-C. Latombe, "Robot motion planning: A distributed representation approach," *Int. J. Robot. Res.*, vol. 10, no. 6, pp. 628–649, 1991.
[16] K. Naderi, K. Taheri, H. Moradi, and M. N. Ahmadabadi, "An evolutionary artificial potential field algorithm for stable operation of a multi-robot system on domes," in *Proc. IEEE Int. Conf. Auton. Robot Syst*. Compet., 2014, pp. 92–97.
[17] A. H. Qureshi, K. F. Iqbal, S. M. Qamar, F. Islam, Y. Ayaz, and N. Muhammad, "Potential guided directional-RRT* for accelerated motion planning in cluttered environments," in *Proc. IEEE Int. Conf. Mechat. Autom.*, 2013, pp. 519–524.
[18] L. Singh, H. Stephanou, and J. Wen, "Real-time robot motion control with circulatory fields," in *Proc. IEEE Int. Conf. Robot. Autom.*, 1996, vol. 3, pp. 2737–2742.
[19] R. Iraji and M. T. M. Shalmani, "AMF: A novel reactive approach for motion planning of mobile robots in unknown dynamic environments," in *Proc. IEEE Int. Conf. Robot. Biomim.*, 2009, pp. 1698–1703.
[20] S. Haddadin, R. Belder, and A. Albu-Schäffer, "Dynamic motion planning for robots in partially unknown environments*," *IFAC Proc. Volumes*, vol. 44, no. 1, pp. 6842–6850, 2011.
[21] A. Ataka, H. K. Lam, and K. Althoefer, "Reactive magnetic-field-inspired navigation for non-holonomic mobile robots in unknown environments," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2018, pp. 6983–6988.
[22] G. Garimella, M. Sheckells, and M. Kobilarov, "A stabilizing gyroscopic obstacle avoidance controller for underactuated systems," in *Proc. IEEE Conf. Dec. and Contr.*, 2016, pp. 5010–5016.
[23] A. M. Hussein and A. Elnagar, "Motion planning using maxwell's equations," in Proc. IEEE/*RSJ Int. Conf. Intell. Robots Syst.*, vol. 3, 2002, pp. 2347–2352.
[24] Z. Mi, Y. Yang, and J. Y. Yang, "Restoring connectivity of mobile robotic sensor networks while avoiding obstacles," *IEEE Sensors J.*, vol. 15, no. 8, pp. 4640–4650, Aug. 2015.
[25] H. Min, F. Sun, and F. Niu, "Decentralized UAV formation tracking flight control using gyroscopic force," in *Proc. IEEE Int. Conf. Comp. Intel. Meas. Syst. Appl.*, 2009, pp. 91–96.
[26] L. Sabattini, C. Secchi, and C. Fantuzzi, "Collision avoidance for multiple lagrangian dynamical systems with gyroscopic forces," *Int. J. Adv. Robot. Syst.*, vol. 14, 2017, Art. no. 172988141668710.
[27] S. Haddadin, S. Parusel, R. Belder, and A. Albu-Schäffer, "It is (almost) all about human safety: A novel paradigm for robot design, control, and planning," in *Comput. Safety, Reliability, Secur.*, F. Bitsch, J. Guiochet, and M. Kaâniche, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 202–215.
[28] J. S. Yuan, "Closed-loop manipulator control using quaternion feedback," *IEEE J. Robot. Autom.*, vol. 4, no. 4, pp. 434–440, Aug. 1988.
[29] D. Koks, *Explorations in Mathematical Physics: The Concepts Behind an Elegant Language*. Berlin, Germany: Springer Science & Business Media, 2006.
[30] M. Quigley *et al.*, Eds., *ROS: An Open-Source Robot Operating System*, 2009.
[31] C. Rösmann, F. Hoffmann, and T. Bertram, "Integrated online trajectory planning and optimization in distinctive topologies," *Robot. Auton. Syst.*, vol. 88, pp. 142 – 153, 2017.
[32] "Ros Package Teb_local_planner," 2021. [Online]. Available: http://wiki.ros.org/teb_local_planner
[33] A. R. Jensen, *Clocking the Mind: Mental Chronometry and Individual Differences*. New York, NY, USA: Elsevier, 2006.