

Plan it Safe: A Risk-Driven Motion Planning Framework for Collaborative Robots

Elena Stracca¹, Alessandro Palleschi², Lucia Pallottino¹, Paolo Salaris¹

Abstract—This paper presents a novel trajectory planning method for collaborative robots, focusing on a risk-centric approach to enhance safety and performance. Drawing from risk management principles, encompassing factors such as operator safety, performance degradation, and internal robot-related issues, the proposed framework offers a comprehensive solution for offline trajectory planning and online reactive trajectory execution. The main contributions include developing and implementing a risk definition and assessment framework utilizing fuzzy inference systems, a mapping strategy leveraging fuzzy logic to correlate robot states with risk levels, and formulating a global measure for evaluating the overall risk. Additionally, the paper introduces a risk-driven motion planning algorithm aimed at minimizing trajectory risk. It also proposes a reactive trajectory adaptation method to respond dynamically to elevated risk levels during task execution. Validation through simulations and experiments with a 7 Degree of Freedom (DoF) robotic manipulator demonstrates the effectiveness of the proposed approach in generating risk-limited trajectories and adapting online to collision risk factors.

I. INTRODUCTION

In the past decades, industrial robots have proven to excel in executing repetitive tasks in structured and controlled environments, intending to maximize productivity by minimizing cycle times [1]. Structured and controlled environments reduce the complexity of the motion planning problem [2], [3]. Indeed, motion planning techniques primarily focused on optimizing performance and safety metrics such as task time completion and collision avoidance in structured, well-defined environments [1], [3].

In recent years, integrating collaborative robots into various industries has become increasingly prevalent due to greater flexibility and reduced installation costs [4]. Nowadays, however, collaborative robotics deployment still faces some barriers in small and medium-sized enterprises (SMEs) [5]. One of the main challenges is programming these robots to operate safely during task execution in shared and dynamic environments. To work safely around human operators, such

This work has received funding from the European Union's Horizon 2020 research and innovation program under agreement no. 101017274 (DARKO) and the Italian Ministry of Education and Research (MIUR) in the framework of the CrossLab FoReLab projects (Departments of Excellence). We acknowledge the support of the European Union by the Next Generation EU project ECS00000017 'Ecosistema dell'Innovazione' Tuscany Health Ecosystem (THE, PNRR, Spoke 4: Spoke 9: Robotics and Automation for Health).

¹Centro di Ricerca E. Piaggio and Dipartimento di Ingegneria dell'Informazione, Università di Pisa, Largo Lucio Lazzarino 1, Pisa, Italy, {elena.stracca@phd.unipi.it, {lucia.pallottino, paolo.salaris}@unipi.it

²Floating Robotics AG, Zurich, Switzerland, a.palleschi@floatingrobotics.com
Corresponding author: Elena Stracca

robots require risk assessment procedures in all their life cycle, from hardware projection and limits identification [6], installation [7], and deployment [8] phases.

The collaborative robot safety specifications are presented in the ISO/TS 15066 [9]. The purpose of these normative specifications for the motion planning phase is to guarantee human safety by imposing velocity constraints. In particular, for collaborative autonomous manipulators, there are two methods imposed as alternatives: Speed and Separation Monitoring (SSM) and Power and Force Limiting (PFL). The first imposes that the maximum speed of the robot should be anytime less of the distance needed to completely stop the robot at that velocity. With PFL, instead, non-zero speed collisions are possible, but the robot can only impart limited static and dynamic forces, resulting in harmless impacts. In this case, the limits do not depend on the distance with the human (and so on the collision probability), but on the severity of a possible collision.

These methods described in the ISO normatives of industrial robots output hard constraints on the speed along the robot trajectory. On the other hand, approaches in the scientific literature are more focused on the problem of quantifying the risk of collisions during robot motion.

The risk-aware motion planning state of the art also follows two distinct lines of research. Within the first, researchers try to formally evaluate the probability of collisions along a robot trajectory reasoning about uncertainty and different risk metrics (i.e. Value at Risk, Conditional Value at Risk) [10], [11]. On the other side, the goal is to minimize the severity of the eventual collision, using impact databases [12], soft materials [13], post-collision strategies [14].

However, in settings like intralogistics or smart factories, more risk factors are concurrently present, not limited to the safety of operators or potential collisions. Additional risks involve the possibility of delays in meeting established deadlines, such as fetching an item from a conveyor belt. Moreover, inherent risks are associated with the self-safety of robots, including issues like overheating and self-collisions.

Hence, the concept of risk is much broader than just the one related to human safety and collision avoidance such as risks related to the accomplishment of the task or its quality of execution that are beyond ISO normative. According to risk management literature [15], risk is the likelihood of any detrimental events occurring to the task or the overall project and it can be expressed as a combination of likelihood and severity [16]. The main idea of this work is to apply the combined likelihood and severity risk definition to serial manipulators in both the design of the robot motion and its

execution taking into account a broader set of risk factors. There are numerous strategies in the literature for quantifying the level of risk during human-robot collaboration, but again, mostly focus on the safety risk for the human operator [8].

Thus, in this work, we propose broadening the risk concept to include a larger class of detrimental events that may prevent the task's success. The aim is to employ this expanded risk space by optimizing risk levels during the offline motion planning phase and the online execution of the planned trajectory. Based on a fuzzy definition of risk levels, the proposed method outlines a general way to characterize risk for autonomous agents working in unstructured and partially unknown environments. The proposed framework is also extremely flexible and scalable to the introduction of new risk factors.

To the best of the authors' knowledge, frameworks capable of simultaneously managing all these different risks to plan robot motion are not yet present in the literature.

Multi-objective trajectory planning is a vast field of study in robotics [17], [18]. No works, however, consider the optimization objective as a combination of heterogeneous risk factors.

The main contributions of the paper are:

- 1) The development and implementation of a risk space definition and risk level assessment framework using a fuzzy inference system.
- 2) Definition of a mapping (based on fuzzy logic) between the robot's state and a global risk index that evaluates the overall risk associated with a specific state, considering all possible risk factors.
- 3) Formulation and presentation of an offline risk-driven motion planning algorithm capable of reducing the risk level of the planned trajectory;
- 4) Proposal of a reactive trajectory adaptation method designed to cope with unpredicted elevated risk levels during task execution.

The framework is finally validated with both simulations and experiments with a 7 Degree of Freedom (DoF) robotic manipulator, the Franka Emika Panda [19], showing that the proposed approach is able to generate low-risk trajectories. We compare trajectories obtained offline with the ones generated by RRT-Connect [20] and STOMP [21] algorithms.

II. PROBLEM DEFINITION

Let us consider a robot whose working space is denoted as \mathcal{C} . Let us also assume that the robot has to move from an initial configuration $\mathbf{q}(t_0) = \mathbf{q}_0 \in \mathcal{C}$ at time t_0 to a final one $\mathbf{q}(t_f) = \mathbf{q}_f \in \mathcal{C}$ at time t_f in a risky environment due to the lack of information about the surrounding (e. g. about the exact configuration of obstacles) and/or to the unpredictability of possible events (e. g. the planned trajectory is perceived as non-smooth).

Let us hence define a space $\mathcal{S} \subseteq \mathbb{R}^m$ ($\mathcal{C} \subset \mathcal{S}$), where m are all of the variables (e. g. robot and environment state), that characterize a risky scenario and influence risk levels (e. g. joints variables, accelerations and jerks, obstacle positions and velocities, environment and motor temperature).

Let us collect in \mathbf{s} the above-mentioned variables in \mathcal{S} . In this paper, every undesired event that may prevent a satisfactory task completion is named *Risk Factor* and referred to as ρ . Each risk factor, in turns, represents a dimension of a multidimensional space, hereafter named *Risk Space* and referred to as \mathcal{R} (see Section III). A fuzzy tree inference method \mathcal{F} is then used (see Section IV) to map from the set \mathcal{S} to the risk level of each risk factor ρ , based on the severity and likelihood of undesired outcomes. Finally, all the possible detrimental events that may happen due to the motion execution are combined in a way that highlights the riskier ones. A global risk measure, hereafter called Λ , to combine the heterogeneous risk factors, is provided (see Section IV-C). We will then use Λ in a function (see Section V-A) to map from \mathcal{R} to \mathbb{R}^+ and finally define our risk metric as

$$J_{\text{Risk}} : \mathcal{S} \xrightarrow{\mathcal{F}} \mathcal{R} \xrightarrow{f(\Lambda)} \mathbb{R}^+, \quad (1)$$

that associates a certain level of risk to \mathbf{s} . The goal of this paper is to find a trajectory $\tau(t) \in \mathcal{T}$, $t \in [t_0, t_f]$ and \mathcal{T} being the set of all the feasible trajectories, that brings the robot from $\tau(t_0) = \mathbf{q}_0$ to $\tau(t_f) = \mathbf{q}_f$ while minimizing the overall level of risk. The values in \mathbf{s} depend on the chosen trajectory, and we will denote this as $\mathbf{s}(\tau(t))$.

We will then formulate this search over all the possible trajectories \mathcal{T} as an optimization problem, where the optimal trajectory τ^* is obtained as:

$$\tau^* = \arg \min_{\tau \in \mathcal{T}} J_{\text{Risk}}(\mathbf{s}(\tau(t))). \quad (2)$$

To show the effectiveness of our methodology, we will consider throughout the whole paper the particular application of a fixed manipulator working in a dynamic environment, possibly shared with humans. It is however important to point out that the presented methodology can be applied to any application and robotic system as long as the pertinent risk factors are suitably identified and assessed.

III. RISK SPACE DEFINITION

This section presents a formal definition of our *Risk Space* as a combination of *Risk Factors*, with specific example related to our case study.

In literature, risk level is measured using two parameters: *risk likelihood* (\mathcal{L}) and *risk severity* (Σ) [22]. Risk likelihood, or probability, indicates a chance of the risk event to occur, while risk severity represents the outcome generated from the risk event if it occurs. Note that both the likelihood and the severity are not, in general, constant values. They depend on $\mathbf{s}(\tau(t))$ and therefore, each risk factor is in general time-dependent. In the following, for simplicity, we will omit the time dependence in each risk factors.

We thus formally define a *risk factor* ρ at time t as a tuple:

$$\rho(t) = \langle T, \mathcal{L}(t), \Sigma(t) \rangle, \quad (3)$$

where T is the risk type, $\mathcal{L}(t)$ is the risk likelihood/probability at time t , and $\Sigma(t)$ is the risk severity at time t as described in the next paragraphs.

TABLE I: Fuzzy rules to map probability and severity to the risk level value.

Risk Level						
0	1	2	3	4	5	6
Almost Impossible - Minor	Almost Impossible - Moderate Low - Minor	Almost Impossible - Severe Low - Moderate Medium - Minor	Almost Impossible - Catastrophic Low - Severe Medium - Moderate High - Minor	Low - Catastrophic Medium - Severe High - Moderate Almost Certain - Minor	Medium - Catastrophic High - Severe Almost Certain - Moderate	High - Catastrophic Almost Certain - Severe/Catastrophic

1) *Risks Type*: We propose to classify the most relevant risk factors within the following three families:

- **Performance risks**: Risks related to the overall performance of the robot movement (e. g. , an excessive power consumption, a delay in the task execution);
- **External risks**: Risks concerning the damage that the movement of the robot can cause to the external environment (e. g. , obstacles, people);
- **Internal risks**: Risks related to undesired events that the robot's movement can cause to the robot itself (e. g. , the robot blocks due to singularities, self-collisions).

2) *Risks Likelihood*: Risk likelihood (\mathcal{L}) aims to assess the probability of occurrence of the risk factor. The happening of an undesired event also depends on variables that are not directly observable, such as an error in the measurement of a sensor or a delay in sending a command to an actuator. In this work, with risk likelihood, we will measure how closely the observable state aligns with the potential occurrence of a detrimental event. We consider five likelihood levels: *Almost Impossible, Low, Medium, High, Almost certain*.

3) *Risks Severity*: The severity (Σ) will depend on the impact of each risk on the system. It represents the gravity of the outcome in the case that the undesired event occurs in that configuration of the robot state. We classify the degree of severity of a risk as:

- **Minor**: Minor damage, resulting in no long-term problems.
- **Moderate**: Damage that has a major immediate impact, like task failure, but does not result in long-term effects.
- **Severe**: Damage that has a major impact that goes to permanent damages (for the robot or the environment) or involves a mild collision with the human operator.
- **Catastrophic**: Damage that may have medical significance to the human operator.

4) *Risk Space*: Based on the above definition of risk factor, a multidimensional risk space \mathcal{R} can be defined as a finite collection of risk factors ρ_i , each one representing a risky direction in this multidimensional space. The identified risk space is both system- and task-dependent and can be easily extended if new risk factors are identified or removed depending on the particular scenario (e. g. the internal risk of self collision, which is likely for manipulators, is unlikely for mobile vehicles).

IV. GLOBAL RISK INDEX

This section presents how each state $s \in \mathcal{S}$ is mapped to the different risk factor levels and from the latter to the final global risk metric that we will use to define our optimal risk-aware motion planning problem.

A. Risk Level

The risk level of each risk factor can be expressed quantitatively by combining the probability that the hazardous event occurs and its severity [16], [22]. This can be achieved using, e. g. probability and impact grids, representing risk events on a grid consisting of probability on one axis and severity on the other. Instead of a grid-based approach, this work uses a Takagi-Sugeno-Kang fuzzy inference system [23], with trapezoidal membership functions, to smoothly map likelihood and severity to a value that quantifies the risk level. For each risk factor ρ , the risk level $\lambda_\rho(s)$ ranges between 0 and 6, resulting in the fuzzy rules shown in TABLE I.

B. Mapping the State Space to the Risk Space

To complete the mapping between the state and the risk space we need to map between the system variables in s to the values of \mathcal{L} and Σ to be used as an input to the risk level inference system described in previous Section IV-A, thus creating a fuzzy tree \mathcal{F} . This map is defined using another fuzzy inference system that maps specific robot-related and task-related variables into appropriate severity and probability values for each risk.

TABLE II shows the variables we used as inputs to our fuzzy inference system and the possible severity levels associated with each risk factor for the particular case study of a manipulator. Some risk factors (*Delay, Energy Consumption, Target Point not Reached*) require a user-defined coefficient between 0 and 1 to indicate how critical the current robot task is regarding the risk factor. For example, if the robot needs to pick an object from a conveyor belt, the task would be more delay-critical than a static case.

For now, state variables from which each risk factor should depend, the associated trapezoidal membership functions parameters, and fuzzy rules are expert-defined through manual specification. Tuning the fuzzy inference systems with a data-driven approach is not an easy task, as it would require a dataset for each risk factor, and it is beyond the purpose of this paper. We hence left this problem to future works.

Note that the proposed framework does not depend on the specific method used to compute the severity and likelihood associated with each risk factor. Indeed, if a more accurate metric to quantify the likelihood or severity of a specific risk is available, the obtained value can be used directly in the risk level inference system.

C. Global risk measure

In this paper, we propose to measure the level of the overall risk as the LogSumExp (LSE) of the concurrent risk

TABLE II: Classification of the identified risk factors, fuzzy variables to infer likelihood and severity, and possible severity levels associated with each risk factor for the case study of a manipulator where both joint and End-Effector (EE) variables are involved.

Performance Risks					External Risks
	Target Point not reached	Delay	Non-smooth motions	High Energy Consumption	Collisions with Obstacles
\mathcal{L}	Distance from the goal EE velocity	Remaining time	EE jerk EE acceleration	EE acceleration EE trajectory length $\frac{\text{Start/Goal Euclidean Distance}}{\text{Task}}$	Relative Position Approach Velocity
Σ	Task	Task	EE jerk High jerk duration	EE trajectory length $\frac{\text{Start/Goal Euclidean Distance}}{\text{Task}}$	EE Velocity
Σ Levels	<i>Minor, Moderate</i>	<i>Minor, Moderate</i>	<i>Minor, Moderate</i>	<i>Minor, Moderate</i>	<i>Minor, Moderate</i> <i>Severe</i>

External Risks		Internal Risks			
	Human Safety	Self Collisions	Singularities	Overheating	Exceed Limits
\mathcal{L}	Relative Position Approach Velocity	Robot configuration	Manipulability index	Room temperature Accelerations	Joint & EE Velocities Accelerations and Jerks
Σ	EE Velocity Impact Point	EE Velocity	Manipulability index	Motor temperature Time in Overheating	-
Σ Levels	<i>Moderate, Severe</i> <i>Catastrophic</i>	<i>Minor, Moderate</i> <i>Severe</i>	<i>Minor, Moderate</i>	<i>Minor, Moderate</i> <i>Severe</i>	<i>Moderate</i>

factors' levels

$$\Lambda(\tau(t)) = \log \left(\sum_{\rho} e^{\lambda_{\rho}(s)} \right). \quad (4)$$

We use the LSE of the risk levels to better detect, and thus avoid, high-risk states. Indeed, a simple sum of the risk levels would not be informative enough and hence would fail to distinguish between a mild risk state and a higher risk state. For example, let us assume that a risk space comprises two risk factors ρ_1 and ρ_2 . If the robot is in a state s_1 where both single risk levels $\lambda(\rho_i)$ are equal to 2, so classified as mild risk, by simply summing the levels would give us a value of 4. Similarly, in a different state s_2 , a naive sum yields the same value 4 if one risk factor is at level 4 (high risk) and the other at level 0. Using the global risk measure in (4), instead, we can distinguish between the two cases obtaining $\Lambda_1 \approx 1.17$ and $\Lambda_2 \approx 1.75$.

Using an infinite norm (i. e. selecting the maximum) of the risk levels would still not be informative enough. Indeed, a state $s_1 \in \mathcal{S}$ where the risk factors ρ_i had levels ${}^1\lambda_{\rho} = (0, 1, 0, 2, 0, 0)$ and a different state $s_2 \in \mathcal{S}$ with risk levels ${}^2\lambda_{\rho} = (2, 2, 2, 2, 2, 2)$, would result in the same global risk values. However, the second state is clearly riskier than the first. For this case, our global risk factor with ${}^1\lambda_{\rho}$ results in $\Lambda_1 \approx 1.15$ for s_1 , while with ${}^1\lambda_{\rho}$ results in $\Lambda_2 \approx 1.65$ for s_2 , hence reflecting the higher risk level of s_2 . Since risk levels $\lambda_{\rho}(s)$ range from 0 to 6, there will be no underflow or overflow issues.

V. RISK-BASED TRAJECTORY PLANNING

In this section, we show how $\Lambda(\tau)$ is used to generate optimal trajectories that minimize our risk measure.

We define our risk metric

$$J_{\text{Risk}}(\tau) = \frac{1}{T} \int_0^T \Lambda(\tau(t)) dt + \max_{\tau} (\Lambda(\tau(t))), \quad (5)$$

where $T = t_f - t_0$ is the total time (to be optimized) to go from the initial to the final robot's configuration.

As we have defined the global risk measure (4), the cost function $J_{\text{Risk}}(\tau)$ (5) strongly penalizes reaching high values in any single risk factor. This approach allows the problem to be solved without imposing strict constraints on the risk levels, thus reducing the problem's complexity.

We propose to solve the problem in (2) using a Genetic Algorithm (GA) offline. GAs are well suited for handling non-differentiable and multi-modal functions, such as the mapping between the robot trajectory and risk metric defined in this paper. However, in executing a task, especially in a human-robot collaboration situation, some risks can only be thoroughly evaluated online, e. g. the presence of human operators or offline unknown obstacles. For this reason, it is necessary to implement also a strategy to monitor the risks in real-time and strategies to bring the risk back to an acceptable level if suitable indicators exceed the guard level.

Replanning online the whole trajectory with the same approach used offline, but with an updated obstacle position takes too much time to ensure a real-time execution. Therefore, in Section V-B we will also propose a reactive strategy to keep the collision risk factor under a certain threshold.

Note that, even if in the case study of this paper, for simplicity, the online strategy only considers the risk of collision, all risks are monitored in real-time using fuzzy trees and could be included in the reactive strategies.

The following sections refer explicitly to the fixed manipulator case study considered in this paper.

A. Offline Risks Optimization

To find the offline trajectory that minimizes J_{Risk} , we define an optimization problem to be solved with a GA having J_{Risk} as the fitness function. We plan in the configuration space, using cubic splines to interpolate joint positions along the trajectory.

We take as decision variables N intermediate configurations $(\mathbf{q}(t_1), \mathbf{q}(t_2), \dots, \mathbf{q}(t_N)) = (\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_N)$, with $t_1, t_2, \dots, t_N \in]t_0, t_f[$ that the robot will reach between the starting \mathbf{q}_0 , and the final \mathbf{q}_f configuration.

We consider as inequality constraints the manipulator's position joint limits ($\mathbf{q}, \bar{\mathbf{q}}$), and that time instances should be non-decreasing. These constraints are intended to confine the range within which the GA seeks the optimal solution. However, they alone do not suffice to ensure compliance with the kinematic constraints within the joint and the task space. Adherence to these constraints is assessed through the risk factor *Exceed limits*, which penalizes solutions that are close to or do not adhere to the manipulator's limits.

The completed optimization problem to be solved offline can be written as:

$$\begin{aligned} & \min_{\mathbf{q}_1, \dots, \mathbf{q}_n, t_1, \dots, t_N, T} J_{\text{Risk}} \\ \text{s.t. } & \underline{\mathbf{q}} < \mathbf{q}_i < \bar{\mathbf{q}}, \quad i = 1, \dots, N \\ & 0 = t_0 \leq t_1 \leq t_2 \leq \dots \leq t_N \leq t_f = T \end{aligned} \quad (6)$$

Algorithm 1 outlines the process for evaluating our metric J_{Risk} . Initially, $\mathbf{q}_0, \mathbf{q}_1, \dots, \mathbf{q}_f$ are interpolated along with their respective time points using cubic splines. These splines are then differentiated with respect to time to yield $\dot{\mathbf{q}}_s, \ddot{\mathbf{q}}_s, \ddot{\mathbf{q}}_s$ (line 2). Subsequently, the total time T is partitioned into n_t time instants, and all the splines are evaluated for those time instants.

Using manipulator direct kinematics, end-effector, and link positions $\mathbf{x}_i(t)$ in the Cartesian space are determined (see line 5 in Algorithm 1). $\mathbf{x}(t), \dot{\mathbf{x}}(t), \ddot{\mathbf{x}}(t), \ddot{\mathbf{x}}(t)$ represent the robot end-effector position, velocity, acceleration, and jerk in the Cartesian space. $d_{\text{self}}(t)$ represents the minimum distance between the end-effector and the other joint Cartesian positions $\mathbf{x}_i(t)$. In line 9 of Algorithm 1, the distance between the robot and the nearest obstacle ($d_{\text{obs}}(t)$) and the robot's approach velocity in the nearest obstacle direction v_{rel} are computed. $man(t)$ is the Yoshikawa manipulability index [24] needed to compute the singularities risk factor. At each time instant, the necessary variables for computing risk factors are evaluated, ultimately leading to the acquisition of the offline global risk index.

The population in the GA is fixed at 200 individuals, with the initial generation having random admissible values. One individual is initialized to provide a favorable starting solution. To compute this first solution, the final time is set at 70% of the maximum fixed time, with interpolation times at regular intervals between 0 and T . Joint variables are linearly interpolated between the initial and final known configurations. The presence of obstacles in the environment is not considered in generating the initial guess.

B. Risk Monitoring and Reactive Response

To adapt the offline computed trajectory online, we propose two concurrent strategies working in a receding horizon fashion: a scaling factor that reduces the velocity along the planned trajectory to maintain the collision risk (with obstacles/humans) below level 3, and a horizon receding Quadratic Programming (QP) activated when the obstacle is too close to the end-effector reference Cartesian path. The goal is to track, at best, the reference trajectory while complying with a set of hard constraints and optimizing other behaviors.

Algorithm 1: Fitness Function

Input: variables, risks fuzzy trees, obstacles, $\mathbf{q}_0, \mathbf{q}_f$

Output: $J_{\text{Risk}}(\tau)$

- 1: Interpolate $\{\mathbf{q}_0, \mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_f\}, \{0, t_1, t_2, \dots, T\}$ with cubic splines
 - 2: $\ddot{\mathbf{q}}_s, \ddot{\mathbf{q}}_s, \dot{\mathbf{q}}_s, \frac{d}{dt} \mathbf{q}_s$
 - 3: **for** $t \leftarrow [0, \frac{T}{n_t}, \frac{2T}{n_t}, \dots, T]$ **do**
 - 4: Evaluate $\mathbf{q}(t), \dot{\mathbf{q}}(t), \ddot{\mathbf{q}}(t), \ddot{\mathbf{q}}(t)$
 - 5: $\mathbf{x}_i(t), \mathbf{x}(t) \stackrel{\text{fkine}}{\leftarrow} \mathbf{q}(t)$
 - 6: $\dot{\mathbf{x}}(t) \stackrel{k}{\leftarrow} \dot{\mathbf{q}}(t)$
 - 7: $\ddot{\mathbf{x}}(t), \ddot{\mathbf{x}}(t) \stackrel{d}{\leftarrow} \ddot{\mathbf{x}}(t)$
 - 8: **for all** Obstacle **in** Known Obstacles **do**
 - 9: Compute $d_{\text{obs}}(t), v_{\text{rel}}(t)$
 - 10: **end for**
 - 11: Compute $man(t), d_{\text{self}}(t)$
 - 12: Evaluate $\Lambda(t)$ using the risk factors fuzzy trees
 - 13: **end for**
 - 14: $J_{\text{Risk}} = \frac{1}{T} \sum_{t=0}^T (\Lambda(t) \frac{T}{n_t}) + \max_{t \in \{0, \dots, T\}} (\Lambda(t))$
-

Algorithm 2 reports the procedure to find the correct scaling factor to maintain a safe trajectory in the considered time window. The aim is to find the scaling factor σ_k to be used to scale the velocity at the next time step, starting from the current reference trajectory ($\mathbf{q}_{\text{ref}}, \dot{\mathbf{q}}_{\text{ref}}$), and the scaling factor σ_p used to scale the velocity at the current time step.

For each $k \in [1, 2, \dots, K]$ corresponding to the time window $[t, t + t_s K]$, with t the current time and t_s the sampling time, the collision risk factor is computed using its fuzzy tree having as input the relative distance (d_{min}), the approach velocity (v_{rel}) and the module of robot end-effector velocity ($\dot{\mathbf{x}}$) (see lines 2 \rightarrow 4 in Algorithm 2). If the maximum value of the collision risk factor (λ_{coll}) is above 3, the maximum scaling factor (σ_{max}) to be reached in the time window is increased until the collision risk is below the limit for all the times in the interval. To be less conservative and to decrease velocity smoothly, the scaling factor is chosen as σ_{max} at $k = k_{\text{max}}$ and $\sigma_p + \frac{k}{k_{\text{max}}} (\sigma_{\text{max}} - \sigma_p)$ for all the steps between $k = 1$ and k_{max} , the latter being the step in which the maximum risk is reached. On the other hand, when the collision risk is below the threshold for all the time window and the scaling factor is greater than 1, the scaling factor is incrementally decreased as long the safety limit is respected (lines 15 \rightarrow 22 in Algorithm 2). After selecting the scaling factor for the next step, the system computes the new reference trajectory by scaling the original offline trajectory with the newly determined scaling factor. This is done by defining a reference time t_{ref} that increases at every iteration as $t_{\text{ref}} = t_{\text{ref}} + \frac{t_s}{\sigma_k}$, which is in turn used to evaluate the offline splines trajectories \mathbf{q}_{off} as follows: $\mathbf{q}_{\text{ref}}[t, t + t_s K] = \mathbf{q}_{\text{off}}[t_{\text{ref}}, t_{\text{ref}} + \frac{t_s K}{\sigma_k}]$, $\dot{\mathbf{q}}_{\text{ref}}[t, t + t_s K] = \dot{\mathbf{q}}_{\text{off}}[t_{\text{ref}}, t_{\text{ref}} + \frac{t_s K}{\sigma_k}] / \sigma_k$.

The scaling factor alone produces a sub-optimal trajectory, as the robot will have to slow down considerably (and possibly stop if there is an obstacle in the way), thus increasing the risk of delay. The idea, therefore, is to combine

Algorithm 2: Scaling Factor

Input: $k, \mathbf{q}_{\text{ref}}[t, t + t_s K], \dot{\mathbf{q}}_{\text{ref}}[t, t + t_s K], \sigma_p, \text{obs}$
Output: σ_k

- 1: $\sigma_k = \sigma_p$
- 2: **for** $k \leftarrow [1, 2, \dots, K]$ **do**
- 3: $\lambda_{\text{coll}}(k) \stackrel{\text{fuzzy}}{\leftarrow} d_{\text{min}}(t + t_s k), v_{\text{rel}}(t + t_s k), \dot{x}(t + t_s k)$
- 4: **end for**
- 5: $[\text{coll}_{\text{max}}, k_{\text{max}}] \leftarrow \max\{\lambda_{\text{coll}}[t, t + t_s K]\}$
- 6: **if** $\text{coll}_{\text{max}} > 3$ **then**
- 7: $\sigma_{\text{max}} = \sigma_k$
- 8: **while** $\text{coll}_{\text{max}} > 3$ **do**
- 9: $\sigma_{\text{max}} + = 0.1$
- 10: $r_{\sigma}(k : 1 \rightarrow k_{\text{max}}) = \frac{\sigma_p}{\sigma_p + \frac{k}{k_{\text{max}}}(\sigma_{\text{max}} - \sigma_p)}$
- 11: $\lambda_{\text{coll}}(k : 1 \rightarrow k_{\text{max}}) \stackrel{\text{fuzzy}}{\leftarrow} d_{\text{min}}(t + t_s k), v_{\text{rel}}(t + t_s k)r_{\sigma}(k), \dot{x}(t + t_s k)r_{\sigma}(k)$
- 12: $[\text{coll}_{\text{max}}, k_{\text{max}}] \leftarrow \max\{\lambda_{\text{coll}}(k)\}$
- 13: **end while**
- 14: $\sigma_k = \sigma_p + \frac{1}{k_{\text{max}}}(\sigma_{\text{max}} - \sigma_p)$
- 15: **else**
- 16: **while** $\text{coll}_{\text{max}} < 3$ **and** $\sigma_k \geq 0.9$ **do**
- 17: $\sigma_k - = 0.1$
- 18: $r_{\sigma} = \frac{\sigma_p}{\sigma_k}$
- 19: $\lambda_{\text{coll}}(k : 1 \rightarrow K) \stackrel{\text{fuzzy}}{\leftarrow} d_{\text{min}}(t + t_s k), v_{\text{rel}}(t + t_s k)r_{\sigma}, \dot{x}(t + t_s k)r_{\sigma}$
- 20: $[\text{coll}_{\text{max}}, k_{\text{max}}] \leftarrow \max\{\lambda_{\text{coll}}(k)\}$
- 21: **end while**
- 22: $\sigma_k + = 0.1$
- 23: **end if**
- 24: $\sigma_k = \max\{\sigma_k, 1\}$
- 25: $\sigma_p = \sigma_k$

the scaling factor with a QP problem that locally modifies the trajectory path to increase the distance from obstacles. By denoting \mathbf{q}^k the configurations at step k , i. e. $\mathbf{q}^k = \mathbf{q}(t + t_s k)$, the QP problem is stated as,

$$\begin{aligned} & \min_{\{\ddot{\mathbf{q}}, \delta_{\text{coll}}, \delta_{\text{lim}}\}} \sum_{k=1}^K (c_1(\mathbf{q}_{\text{ref}}^k - \mathbf{q}^k)^2 + c_2(\dot{\mathbf{q}}_{\text{ref}}^k - \dot{\mathbf{q}}^k)^2 \\ & c_3(\delta_{\text{coll}}^k)^2 + c_4(\delta_{\text{lim}_{\text{inf}}}^k)^2 + c_5(\delta_{\text{lim}_{\text{sup}}}^k)^2 + c_6(\ddot{\mathbf{q}}_i^k - \ddot{\mathbf{q}}^{k-1})) \\ \text{s.t. } & \underline{\mathbf{q}} \leq \mathbf{q}^k + \delta_{\text{lim}} \leq \bar{\mathbf{q}} \\ & \underline{\dot{\mathbf{q}}} \leq \dot{\mathbf{q}}^k \leq \bar{\dot{\mathbf{q}}} \\ & \underline{\ddot{\mathbf{q}}} \leq \ddot{\mathbf{q}}^k \leq \bar{\ddot{\mathbf{q}}} \\ & \underline{\dot{\mathbf{x}}} \leq \dot{\mathbf{x}}^k \leq \bar{\dot{\mathbf{x}}} \\ & \underline{\ddot{\mathbf{x}}} \leq \ddot{\mathbf{x}}^k \leq \bar{\ddot{\mathbf{x}}} \\ & d_{\text{min}} \leq d^k + \delta_{\text{coll}} \\ & d_{\text{min}} \leq d^1 \\ & \underline{\mathbf{q}} \leq \mathbf{q}^1 \leq \bar{\mathbf{q}}. \end{aligned} \quad (7)$$

Within the time horizon $[t + t_s, t + K t_s]$, the goal of the

QP problem is to minimize the difference between the real $(\mathbf{q}^k, \dot{\mathbf{q}}^k)$ and the reference trajectory positions and velocities $(\mathbf{q}_{\text{ref}}^k, \dot{\mathbf{q}}_{\text{ref}}^k)$ along with input (joints acceleration) variation $(\ddot{\mathbf{q}}_i^k - \ddot{\mathbf{q}}^{k-1})$ and soft constraints regarding minimum Manhattan distance with obstacles threshold δ_{coll} and manipulator joint limits ($\delta_{\text{lim}_{\text{inf}}}$ and $\delta_{\text{lim}_{\text{sup}}}$). However, at time step $k = 1$, the constraints on the minimum distance with the obstacle (d^1) and the robot position joints limits (\mathbf{q}^1) are considered as hard constraint (see the last inequalities in (7)). $c_i, i = 1, \dots, 6$ are user-defined coefficients.

Current joint positions and velocities are denoted as $(\mathbf{q}^0, \dot{\mathbf{q}}^0)$. Moreover, joint positions, velocities, accelerations, jerks minimum and maximum limit values are denoted as $(\underline{\mathbf{q}}, \bar{\mathbf{q}}), (\underline{\dot{\mathbf{q}}}, \bar{\dot{\mathbf{q}}}), (\underline{\ddot{\mathbf{q}}}, \bar{\ddot{\mathbf{q}}}), (\underline{\dot{\mathbf{x}}}, \bar{\dot{\mathbf{x}}}), (\underline{\ddot{\mathbf{x}}}, \bar{\ddot{\mathbf{x}}})$ are the Cartesian velocity and acceleration minimum and maximum limit values.

At each time instant, the QP is solved, computing the $\ddot{\mathbf{q}}$ for the next K steps ($\ddot{\mathbf{q}}^{1 \rightarrow K}$). From $\ddot{\mathbf{q}}^{1 \rightarrow K}$ we compute $\mathbf{q}^{1 \rightarrow K}$ and $\dot{\mathbf{q}}^{1 \rightarrow K}$ as $\dot{\mathbf{q}}^k = \dot{\mathbf{q}}^{k-1} + \ddot{\mathbf{q}}^k t_s$ and $\mathbf{q}^k = \mathbf{q}^{k-1} + \dot{\mathbf{q}}^{k-1} t_s + \frac{1}{2} \ddot{\mathbf{q}}^k t_s^2$. Then, we use the Algorithm 2 to compute the scaling factor, setting $\mathbf{q}_{\text{ref}}[t, t + t_s K] = \mathbf{q}^{1 \rightarrow K}$ and $\dot{\mathbf{q}}_{\text{ref}}[t, t + t_s K] = \dot{\mathbf{q}}^{1 \rightarrow K}$. This means that we calculate σ_k using as input not the offline computed trajectory, but the one obtained as output of the QP problem. The resulting scaling factor is then used to scale both the reference trajectory at the next iteration and the tuple $(\mathbf{q}^1, \dot{\mathbf{q}}^1, \ddot{\mathbf{q}}^1)$ in output from the QP problem. Defining $\sigma_r = \max(1, \sigma_k / \sigma_p)$, the new tuple to be given as input to the robot, $(\mathbf{q}_u, \dot{\mathbf{q}}_u, \ddot{\mathbf{q}}_u)$, is computed as $\dot{\mathbf{q}}_u = \frac{\dot{\mathbf{q}}^1}{\sigma_r}$, $\ddot{\mathbf{q}}_u = \frac{\ddot{\mathbf{q}}^1 - \ddot{\mathbf{q}}^0}{t_s}$, $\mathbf{q}_u = \mathbf{q}^0 + \dot{\mathbf{q}}^0 t_s + \frac{1}{2} \ddot{\mathbf{q}}_u t_s^2$.

The new scaled $\mathbf{q}_u, \dot{\mathbf{q}}_u, \ddot{\mathbf{q}}_u$ are given the manipulator to be executed, and the QP is iteratively solved again for the time window $[t + 2t_s, t + t_s + t_s K]$, until the goal is reached. If the QP problem becomes infeasible, the robot stops immediately.

VI. VALIDATION

We tested the proposed framework in simulation on a Franka Emika Panda manipulator and executed the obtained trajectories on the real robot. The manipulator limits are reported in [25]. The framework has been implemented in MATLAB, and the QP problem is solved using the qpOASES solver in CasADi [26]. We consider a scenario where the robot has to move from $\mathbf{q}_0 = [-0.7, 0, -0.8, -2.44, -0.27, 2.22, -0.08]$ to $\mathbf{q}_f = [0, -0.67, 1.14, -2.55, 1.15, 2.06, 1.72]$. The maximum time for executing the trajectory is 6 s.

Obstacles are approximated as spheres as commonly done in literature [27]. However, we assume that due to a not precise knowledge of the environment and the presence of unmodeled uncertainty, the spheres are not guaranteed to bind the obstacles. In the case of perfect knowledge of the robot and the world state, we wouldn't have collision risk as an impact event would deterministically happen or not.

We tested the trajectories obtained in simulation on the real robot. We control the robot using the Joint Trajectory Controller available in ROS Noetic, providing at each times-

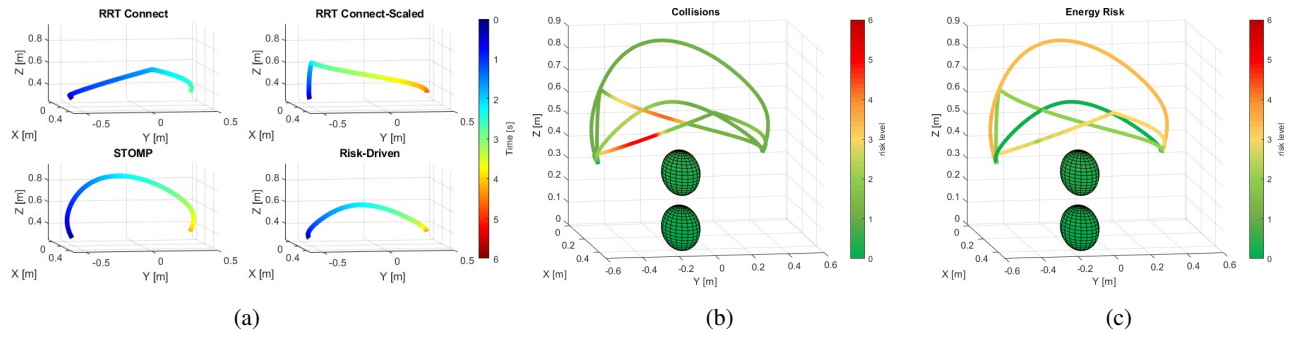


Fig. 1: Comparison of the three obtained trajectories: **a)** Time profiles, **b)** Collision risk factor, **c)** Energy risk factor.

TABLE III: Results - Offline Trajectories. J_{Risk} is expressed as the sum of its two components ($\frac{1}{T} \int_0^T \Lambda dt$ and $\max \Lambda$).

	J_{Risk}		$T[s]$	$L_r[m]$	$\bar{a}_{ee} [m/s^2]$	$\bar{j}_{ee} [m/s^3]$	Delay	Target Point	Non smooth	Self Collisions	Singularities	Energy	Collision				Limits
	$\frac{1}{T} \int_0^T \Lambda dt$	$\max \Lambda$											Risk	d	v_{rel}	v_{cc}	
Risk-Driven	2.96	3.18	4.68	1.35	0.38	0.72	0.94	1.23	0.06	0.06	1.50	0.22	2.49	0.23	0.11	0.42	1
RRT-Connect	3.96	5.43	3.07	1.48	1.71	11.82	0.75	1.88	2.07	0.45	1.24	2.89	5.08	0.12	0.52	1.04	1
sRRT-Connect	3.36	4.46	5.04	1.57	0.57	1.57	0.99	1.67	0.56	0.21	1.34	2.04	4.19	0.15	0.45	0.61	1
STOMP	3.86	4.05	4.24	1.73	0.65	2.49	0.83	1.67	0.76	0.24	1.09	3.43	1.15	0.46	-0.11	0.59	1

tamp joint positions, velocities and accelerations. Videos of the performed experiments are available at this link. ¹

A. Offline Risk Planner

We will show in the following the paths resulting in the presence of an obstacle approximated as two spheres at positions $x_{obs1} = [0.4, -0.15, 0.1]$ m, $x_{obs2} = [0.4, -0.15, 0.35]$ m and radius 0.1 m. We optimize on 3 intermediate points (q_1, q_2, q_3), so the GA will have 25 variables. For the genetic operators used to produce the next generation after the fitness function evaluation (reproduction, crossover, and mutation) we use MATLAB default options. The maximum number of generations is set to 50. *Time, Precision, and Energy Critical* task coefficients are all set to 0.8. n_t is set to 20. The Overheating and Human Safety risk factors were omitted from the offline Λ computation. These risk factors require data (human positions, motor temperatures) to be acquired online, conflicting with the offline computation of the GA. A response to the overheating risk factor will be investigated in future works.

In TABLE III, we compare the performances of our risk-driven trajectory to two trajectories obtained with MoveIt RRT-Connect planner [28], and the trajectory obtained using STOMP [21], a stochastic optimization algorithm, with the default MoveIt parameters, in the same scenario. The first of the RRT-Connect trajectories is obtained using the robot's nominal joint limits, the latter has been computed by imposing lower kinematic limits to obtain a slower trajectory with an execution time comparable to the one obtained by the risk-driven optimization algorithm. The table reports some performance indicators (time, EE trajectory length, medium acceleration and jerk at the EE) and the assessed risk values. For some risk factors, we report the average value along the trajectory (denoted as $\bar{\cdot}$), and for others the maximum ($\hat{\cdot}$), depending on which one is more significant. For the collision

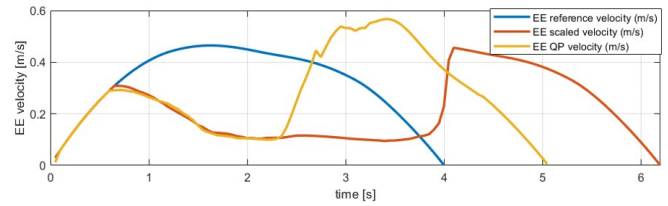


Fig. 2: EE velocity profiles for the reference trajectory, and the three online responses.

risk, we report its maximum and the associated variable values. The four resulting EE trajectories are shown along with their time profile, collision, and energy risk factors in Fig. 1.

B. Reactive Trajectory Modification

Starting from the same q_0 and q_f as in previous case, the reference trajectory is now obtained by solving the GA without knowing obstacles in the scene. An unknown obstacle is however present in the workspace and approximated as two spheres of center $[0.45, -0.2, 0.25]$ m, $[0.45, -0.1, 0.3]$ m and radius 0.05 m. The parameters used for the scaling factor and the QP problem algorithms are: $K=15$, $t_s = 0.05$ s, $c_1 = 1e2$, $c_2 = 10$, $c_3 = 1e6$, $c_4 = 1e5$, $c_5 = 1e5$, $c_6 = 1$, $d_{min} = 0.2$ m.

We compare the collision risk factor for the reference trajectory (Fig. 3a), the trajectory obtained when the scaling factor algorithm is active (and the QP disabled) (Fig. 3b), and the trajectory obtained when both the scaling factor and the QP problem are active (Fig. 3c). The EE velocity profiles for the three scenarios are reported in Fig. 2. When only the scaling factor algorithm is activated, the output trajectory is safe but it becomes very slow (exceeding the maximum time of 6 s). When the QP problem is also active (see Fig. 2), but when the minimum distance from the obstacle is reached the evasion

¹<https://drive.google.com/drive/folders/1Q9Tgv132B0a94rENapq43Khrsv2Fk6Sf?usp=sharing>

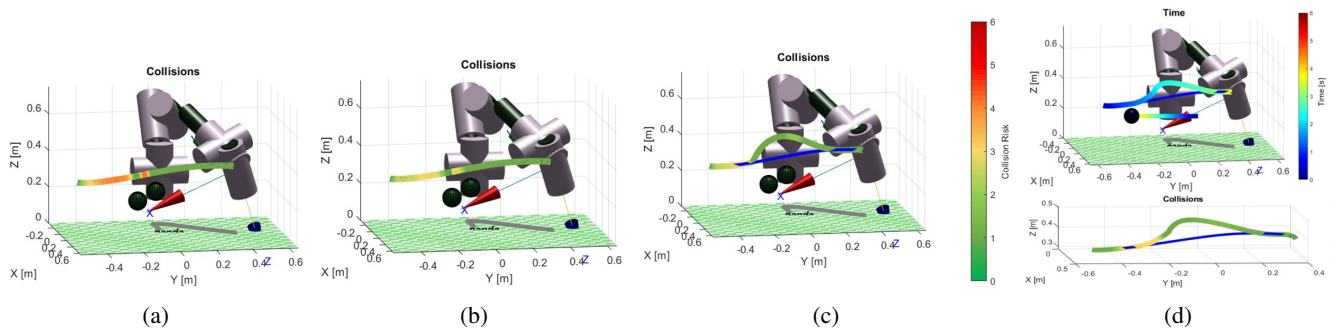


Fig. 3: Collision risk in case of unknown offline obstacle: **a)** Reference trajectory, without employing any strategy, **b)** Scaling factor activated, **c)** Scaling factor and QP activated, **d)** Moving obstacle: Scaling Factor and QP activated.

maneuver begins, resulting in a reduction of the overall time needed for accomplishing the task. Fig. 3d shows a scenario where an obstacle moves with constant velocity $v_{\text{obs}} = [-0.02, -0.1, 0]$ m/s, starting at time 0 from $x_{\text{obs}} = [0.5, 0.1, 0.28]$ m.

VII. CONCLUSIONS

In this paper, we have extended the concept of risk to manipulator motion planning by including a variety of performance, external, and internal risk factors. We have proposed a metric to quantify the level of risk along a trajectory and used a GA to compute offline the trajectories with minimum risk. Trajectories are then monitored online, and a reactive strategy that uses a scaling factor and a receding horizon QP problem has been implemented to maintain a low risk level. The membership functions parameters and fuzzy rules to map from the system state space \mathcal{S} to the probability and severity levels are defined by an expert. Future works will be to learn and automatically tune these parameters using available data. Furthermore, we also aim to include additional reactive strategies to address online heterogeneous risk factors.

REFERENCES

- [1] A. Paleschi *et al.*, "Time-optimal path tracking for jerk controlled robots," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3932–3939, 2019.
- [2] T. Lozano-Perez, "A simple motion-planning algorithm for general robot manipulators," *IEEE Journal on Robotics and Automation*, vol. 3, no. 3, pp. 224–238, 1987.
- [3] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006, available at <http://planning.cs.uiuc.edu/>.
- [4] S. Proia *et al.*, "Control techniques for safe, ergonomic, and efficient human-robot collaboration in the digital industry: A survey," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 3, pp. 1798–1819, 2021.
- [5] A. Grau *et al.*, "Robots in industry: The past, present, and future of a growing collaboration with humans," *IEEE Industrial Electronics Magazine*, vol. 15, no. 1, pp. 50–61, 2020.
- [6] I. O. for Standardization, "Safety of machinery - general principles for design - risk assessment and risk reduction," Geneva, CH, Standard ISO 12100:2010, 2010.
- [7] L. Poot *et al.*, "Supporting risk assessment of human-robot collaborative production layouts: a proposed design automation framework," *Procedia Manufacturing*, vol. 25, pp. 543–548, 2018.
- [8] Z. Liu *et al.*, "Dynamic risk assessment and active response strategy for industrial human-robot collaboration," *Computers & Industrial Engineering*, vol. 141, p. 106302, 2020.
- [9] I. O. for Standardization, "Robots and robotic devices – collaborative robots," Geneva, CH, Standard ISO/TS 15066:2016, 2016.
- [10] A. Hakobyan *et al.*, "Risk-aware motion planning and control using cvar-constrained optimization," *IEEE Robotics and Automation letters*, vol. 4, no. 4, pp. 3924–3931, 2019.
- [11] A. Dixit *et al.*, "Risk-sensitive motion planning using entropic value-at-risk," in *2021 European Control Conference (ECC)*. IEEE, 2021, pp. 1726–1732.
- [12] S. Haddadin *et al.*, "On making robots understand safety: Embedding injury knowledge into control," *The International Journal of Robotics Research*, vol. 31, no. 13, pp. 1578–1602, 2012.
- [13] M. Hu, "Research on safety design and optimization of collaborative robots," *International Journal of Intelligent Robotics and Applications*, vol. 7, no. 4, pp. 795–809, 2023.
- [14] A. De Luca *et al.*, "Collision detection and safe reaction with the dlr-iii lightweight manipulator arm," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2006, pp. 1623–1630.
- [15] D. Baloi *et al.*, "Modelling global risk factors affecting construction cost performance," *International journal of project management*, vol. 21, no. 4, pp. 261–269, 2003.
- [16] A. Nieto-Morote *et al.*, "A fuzzy approach to construction project risk assessment," *Elsevier*, vol. 29, pp. 220–231, feb 2010.
- [17] S. S. Chiddarwar *et al.*, "Optimal trajectory planning for industrial robot along a specified path with payload constraint using trigonometric splines," *International Journal of Automation and Control*, vol. 6, no. 1, pp. 39–65, 2012.
- [18] J. Lan *et al.*, "A multi-objective trajectory planning method for collaborative robot," *Electronics*, vol. 9, no. 5, p. 859, 2020.
- [19] S. Haddadin *et al.*, "The franka emika robot: A reference platform for robotics research and education," *IEEE Robotics & Automation Magazine*, vol. 29, no. 2, pp. 46–64, 2022.
- [20] J. Kuffner *et al.*, "Rrt-connect: An efficient approach to single-query path planning," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, vol. 2, 2000, pp. 995–1001 vol.2.
- [21] M. Kalakrishnan *et al.*, "Stomp: Stochastic trajectory optimization for motion planning," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 4569–4574.
- [22] A. Ahmed *et al.*, "A review of techniques for risk management in projects," *Benchmarking: An International Journal*, 2007.
- [23] M. Sugeno, "An introductory survey of fuzzy control," *Information sciences*, vol. 36, no. 1-2, pp. 59–83, 1985.
- [24] T. Yoshikawa, "Manipulability and redundancy control of robotic mechanisms," in *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, vol. 2. IEEE, 1985, pp. 1004–1009.
- [25] Franka Emika, "Franka emika documentation," <https://frankaemika.github.io/docs/>.
- [26] J. A. E. Andersson *et al.*, "CasADi – A software framework for non-linear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.
- [27] L. Balan *et al.*, "Real-time 3d collision avoidance method for safe human and robot coexistence," in *2006 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2006, pp. 276–282.
- [28] S. Chitta *et al.*, "Moveit![ros topics]," *IEEE Robot. Autom. Mag.*, vol. 19, pp. 18–19, 03 2012.