DOI: 10.21857/m8vqrt3p19

Pregledni članak / Review paper
Prihvaćeno / Accepted: 15. veljače 2024.

MOTION PLANNING FOR ROBOTIC MANIPULATION

Ivan Petrović, Filip Marić, Luka Petrović, Ivan Marković

Summary

Robotic manipulators are a class of robots specialized for tasks that involve picking up, placing and manipulating objects, motion coordination, and signaling. Motion planning is a fundamental problem in robotics that involves computing feasible joint space trajectories for a robot to execute, connecting multiple states defined by task constraints. Due to the wide variability of robot structures and environments, there are a variety of methods for solving this problem. These methods are often complementary and interchangeable, making it difficult to categorize and formulate them with common terminology. In this paper, we list methods and approaches, and place them in the context of the broader problem of motion planning. We begin with an overview of sampling and graph-based methods used in path planning, where a path is defined by a discrete set of configurations. We then continue with an overview of methods for inverse kinematics and trajectory optimization. We discuss the challenges in these methods and approaches to solving them, and review some of the recent advances in the field.

Keywords: robotic manipulation; motion planning; inverse kinematics; trajectory optimization.

1. INTRODUCTION

Robotic manipulation refers to the use of robots, also known as manipulators, to interact with the physical world by grasping, moving and manipulating objects [1]. This is typically achieved through the use of robotic arms, such as those shown in Fig. 1, which are equipped with joints and end-effectors that allow them to move and manipulate objects in a variety of ways. In order to perform these tasks, robots must be able to perceive their environment, plan their actions, and execute them with precision. This requires a combination of sensors, algorithms, and control systems that allow the robot to understand its surroundings and make decisions about how to move and interact with objects. Robotic manipulation has many applications in fields, such as manufacturing, logistics,

and healthcare, where robots can assist humans with tasks that are repetitive, dangerous, or require a high level of precision. Increasingly, algorithms for motion optimization and computation have enabled manipulators to perform many of the tasks in these domains *autonomously*.

Motion planning algorithms are a crucial component in modern autonomous robotic systems and may be categorized into several different techniques, such as path planning, inverse kinematics and trajectory optimization, which can be interchangeable or complementary depending on the scenario. Path planning techniques use sampling and graph search algorithms to efficiently explore the configuration space, finding a sequence of configurations connecting a start and goal state, and avoiding collisions and infeasible states in the process. Inverse kinematics deals with the problem of computing joint angles that place the robot's end-effector (e.g., a gripper) in a specific pose, or joint velocities that trace a defined end-effector path. This is often a crucial step in motion planning, as it allows the motion planner to determine the target configuration of the robot's joints. Trajectory optimization, on the other hand, focuses on computing smooth, feasible motions for the robot's joints and end effectors, optimizing them in terms of energy consumption, smoothness, or other criteria. This is particularly important in the context of robotic manipulation, where the goal is to compute optimal joint trajectories that allow the robot to interact with its environment effectively.

One of the major challenges in motion planning is the computational complexity of the problem [2]. In order to compute a feasible path for a robot to follow, motion planners must take into account a wide range of constraints, including the robot's kinematic limitations, as well as any obstacles in the environment. This can require a significant amount of computation, especially for robots with many degrees of freedom or for tasks that involve complex. Another major challenge in motion planning is the ability to handle the environment's uncertainty and stochasticity. In many real-world scenarios, the robot's environment and the objects within it may be uncertain or subject to random variations. This can make it difficult for motion planners to compute reliable and robust paths for the robot to follow. In the context of robotic manipulation, motion planning presents additional challenges, such as the need to compute feasible grasps and contact points for the robot's end effectors. This can require the motion planner to take into account the geometry and kinematics of the robot, as well as the physical properties of the objects that the robot is manipulating. Overall, motion planning is a challenging problem, where an effective solution requires sophisticated algorithms and computational techniques.

Combined, these techniques enable robots to move efficiently and accurately in complex environments, performing a wide range of tasks, such as grasping objects, navigat-

ing through obstacles, and executing precise movements. Motivated by the implications that these advances have on the projected capabilities of robotic manipulators in the near future, this paper gives a broad overview of related methods and approaches, discussing their key features, applications, and challenges. While these techniques may be used interchangeably (or not used at all) and the boundary between them is often unclear, this paper makes a distinction between path planning, inverse kinematics and trajectory optimization. In the following text, we first offer a brief overview of the main challenges and basic definitions in robotic manipulation in Section 2. The following three sections contain a review of the state-of-the-art algorithms used for path planning, inverse kinematics and trajectory optimization in the context of the challenges outlined in Section 2. The paper concludes with a summary of the advances and their potential impact in the future.

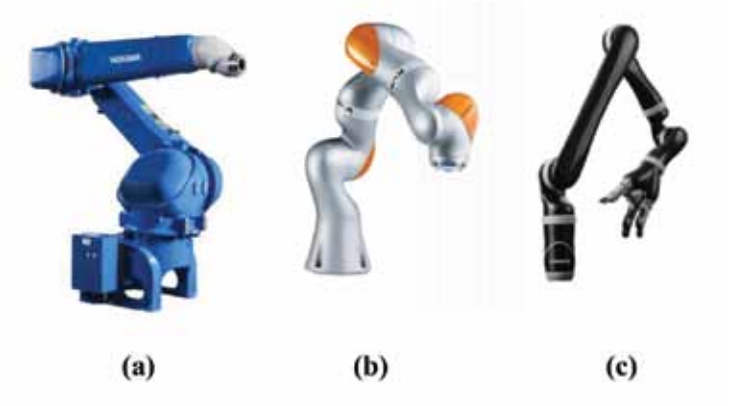

Figure 1. Some examples of robotic manipulators: a) Yasakawa Motoman industrial robotic arm, b) KUKA IIWA robotic arm, c) Kinova Jaco robotic arm.
Slika 1. Neki primjeri robotskih manipulatora: a) industrijska robotska ruka Yasakawa Motoman, b) robotska ruka KUKA IIWA, c) robotska ruka Kinova Jaco.

2. FUNDAMENTALS AND CHALLENGES

Robotic manipulators were initially utilized in controlled industrial settings, where they performed simple, repetitive tasks by following predetermined joint trajectories [3]. These manipulators have been widely employed in the automotive industry, replacing human workers in tasks, such as spray painting, welding, and parts transport. One advantage of industrial manipulators over traditional heavy machinery is their ability to be reprogrammed to adapt to changes in task requirements, providing a more cost-

effective solution compared to bespoke machines. Operators can manually reprogram these large robots by altering key joint configurations and connecting them with an appropriate joint velocity or acceleration trajectory computed offline [4]. However, the fixed nature of these trajectories can present additional safety considerations for human workers, leading to the use of protective enclosures, such as cages in the workplace. As a result, research and engineering efforts in the field of robotic manipulation have focused on expanding the adaptation capabilities of these robots to enable safe collaboration with humans, with the ultimate goal of achieving full autonomy.

2.1. Basic Definitions and Notation

Most robotic manipulators are modelled as kinematic chains comprised of revolute joints connected by rigid links. The joint angles are arranged in a vector $\theta \in \mathcal{C}$, where $\mathcal{C} \subseteq R^n$ is known as the *configuration space*. The minimum number of coordinates required to represent the configuration of the robot is equal to its *degrees of freedom* (DoF). Knowledge of the current configuration allows for the computation of the position and orientation of arbitrary coordinate frames attached to the links of the robot, such as those denoted by F in Fig. 2, using *forward kinematics*. Given that robots are constructed using a large variety of different joint and link types specialized for particular tasks, the geometry of their configuration spaces can vary significantly, and it therefore requires careful analysis.

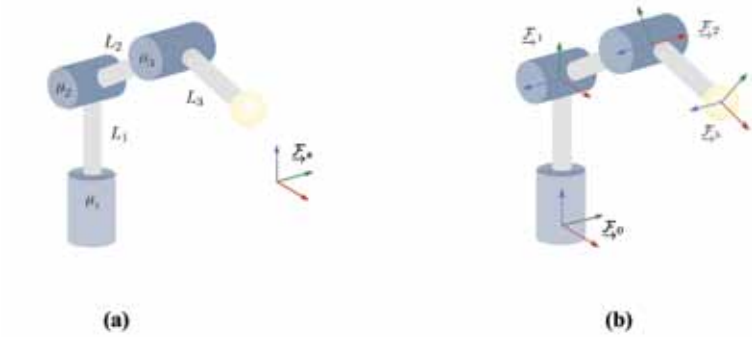

Figure 2. a) Simplified sketch of a robotic manipulator, with links denoted by L and joints denoted by θ, b) sketch of the coordinate systems commonly defined and used in computations related to manipulation tasks. The yellow sphere represents an end-effector (such as a tool or a gripper).
Slika 2. a) Pojednostavljena skica robotskog manipulatora, s vezama označenim s L i spojevima označenim s θ, b) skica koordinatnih sustava koji se obično definiraju i koriste u proračunima povezanim sa zadacima manipulacije. Žuta kugla predstavlja krajnjeg izvršitelja (kao što je alat ili hvataljka).

A class of tasks performed by the robot is generally parameterized using coordinates τ_i arranged into a vector $\boldsymbol{\tau} \in \mathcal{T}$, where $\mathcal{T}$ is the *task space* defined by this parameterization. The task space is specified by the user and generally coincides with the pose or position of the end-effector (e.g., a gripper used to manipulate objects), but may also be learned (e.g., from demonstration data). A point in the task space may be reachable by multiple distinct robot configurations. Moreover, if the number of degrees of freedom of a robot exceeds the dimension of the task space (i.e., $m = \dim(\mathcal{C}) - \dim(\mathcal{T}) > 0$), any τ may be reachable by an m-dimensional set of configurations. Such sets are often more accurately defined as algebraic varieties – sets of solutions to systems of polynomial equations. For example, robots with five degrees of freedom may reach any end-effector position with an infinite number of configurations on a two-dimensional variety, while robots with seven degrees of freedom can reach any end-effector pose with an one-dimensional variety of configurations. These robots are often said to be *redundant* in the context of a given task space.

2.2. Challenges in Robotic Manipulation

Motion planning algorithms generally require solutions to three distinct subproblems. First, tasks are defined through a set of spatial and joint constraints on one or multiple key states. For example, the starting state is generally fixed as the current robot configuration, while the final state may be described by constraints that place the robot's end-effector in a pose where an object can be easily grasped. In order to compute configurations that adhere to these constraints, inverse kinematics [5; 6; 7] techniques employ local optimization approaches that search the joint space for one of many locally optimal solutions. Second, a path that connects these key states is found using path planning techniques [8]. Path planners employ sampling and graph search algorithms [9; 10] to find a discrete sequence of configurations that connects the starting and final state. Notably, path planning needs to account for task constraints, such as obstacle avoidance and joint limits. Third, the resulting sequence of states needs to be transformed into a time-parameterized trajectory. This step is performed using trajectory optimization techniques [11; 12; 13], which use local optimization over the space of parameterized functions to generate a trajectory that is smooth and dynamically feasible. These techniques also admit additional performance criteria that enforce preferable joint motions when optimized.

In summary, there are several challenges that ought to be overcome in order to achieve successful motion planning for autonomous robotic manipulation. These challenges include:

- *High-dimensional configuration space*. The configuration space of a robot manipulator can be very high-dimensional, which can make it difficult to find a feasible path between two configurations. This can be especially challenging in cluttered environments, where the number of obstacles and other constraints can increase the dimensionality of the space.
- *Complex constraints*. Robotic manipulation often involves satisfying a wide range of constraints, such as avoiding collisions with obstacles, minimizing energy consumption, or following a desired trajectory. Incorporating these constraints into the motion planning process can be challenging and require the use of sophisticated algorithms and optimization techniques.
- *Uncertainty and noise*. Real-world environments can be unpredictable and noisy, which can make it difficult for a robot to accurately perceive its surroundings and plan its actions. This uncertainty and noise can introduce errors into the motion planning process, which can lead to suboptimal or infeasible paths.
- *Physical limits and constraints*. Robots have physical limitations, such as maximum joint velocities, maximum joint torques, and maximum payloads that ought to be taken into account during motion planning. These limits and constraints can further complicate the search for a feasible path and require the use of specialized algorithms and optimization techniques.

These challenges require the development of advanced algorithms and techniques that can handle high-dimensional configuration spaces, complex constraints, uncertainty and noise, and physical limitations.

3. PATH PLANNING IN CONFIGURATION SPACE

Path planning is a computational approach to finding a feasible path for a robotic manipulator to move from a starting configuration to a goal configuration while avoiding obstacles and satisfying any other constraints. Path planning can be subject to a large variety of constraints specific to a particular problem type. However, we propose the following broad formal definition for clarity.

Note that Definition 1 covers only some of the general properties of a successfully defined path, while many other constraints ensuring task-specific feasibility (e.g., stability for legged robots) exist. Most conventional approaches involve either representing the configuration space as a discrete grid of uniformly distributed configurations (grid-based methods) or randomly generating samples within the search space and using them to construct a representation of the free space (sampling-based methods). The pertaining configuration space representation is then searched to find a path to the goal.

Definition 1 (Path Planing) *Let $\boldsymbol{\theta_0} \in \mathcal{C}$ and $\boldsymbol{\theta_T} \in \mathcal{C}$ be a starting and goal configuration, respectively. Find a sequence of configurations $\Theta = \{\theta_0, \theta_1, \cdots, \theta_t, \cdots, \theta_T\}$ such that:*

- *For each pair of consecutive configurations θ_t, θ_{t+1} it holds that $g(\theta_t, \theta_{t+1}) \leq L_d$ for some metric g and $L_d \in R$.*
- *For every configuration θ_t the robot is not in collision.*
- *For every configuration $\boldsymbol{\theta_t}$ it holds that $\theta_l \leq \theta_t \leq \theta_u$, where $(\theta_u, \theta_l) \in \mathcal{C}$ are the upper and lower joint limits, respectively.*

The resulting sequence is then used to define a continuous feasible path between the starting and goal configurations $\theta: t \in [0,1] \mapsto \theta(t)$.

3.1. Grid-based methods

The main two benefits of grid-based methods are that they find optimal paths to the goal and that they are complete, which is the ability to always find a solution if one exists, or report failure otherwise. While naive application of grid-based methods is typically unsuitable for robotic manipulation planning since number of points on the grid grows exponentially in the configuration space dimension, current research has made advances in applying grid search to high-dimensional problems. One example is [14], where the authors presented a framework for efficiently updating the A* search while smoothly reducing heuristic inflation, allowing resolution complete search in an anytime fashion on a broader variety of problems than previously computable. The A*-Connect method [15] introduced a fast approximation of the front-to-front heuristic to lead the forward and backward searches towards each other, while retaining theoretical guarantees on completeness and demonstrating its suitability for manipulation planning. To achieve a better balance between computational efficiency and the ability to find collision-free paths, a multi-resolution A* algorithm has been proposed [16], using a fine discretization in areas with obstacles, and a coarse discretization in free areas of the configuration space, demonstrating competitive performance in manipulation planning for a 7 DoF robot arm. Moreover, recently there have been attempts to parallelize the grid search [17; 18], providing a speed-up that is almost linear in the number of threads used for parallelization. While grid-based methods offer guarantees on completeness, their main drawback is computational complexity, as even the state-of-the-art methods often suffer from the curse of dimensionality and become computationally intractable on commodity hardware for very high DoF systems, such as humanoid robots. Despite the recent improvements in search heuristics, exploitation of multiple resolution grids and leveraging parallelization, grid-based methods are unable to ensure efficient and reliable high-dimensional robot operation, especially in dynamic environments.

3.2. Sampling-based methods

Sampling-based methods have become popular in the domain of path planning for robotic manipulators due to their ability to handle high-dimensional configuration spaces and complex constraints. This makes them well-suited to a wide range of applications, such as robot navigation in cluttered environments or collision-free motion planning for mechanical systems [2]. In practice, sampling-based path planning algorithms typically iteratively refine a roadmap or graph structure that encodes the connections between free samples [8]. This graph is used to guide the search for a path, and can be updated as the search progresses to incorporate new samples and improve the quality of the path. Sampling-based path planners are understood to be probabilistically complete [19], a weaker notion of completeness that ensures a solution will be provided, if one exists, given sufficient runtime of the algorithm. Some popular sampling-based path planning algorithms include probabilistic roadmap methods (PRMs) [20], rapidly-exploring random trees (RRTs) [10], and their variants [2]. These algorithms differ in the specific strategy used to generate samples and construct the graph, but share the core idea of using random sampling to represent and search the free space. In the following sections, we explore the RRTs and the PRMs in more depth, as they are the two most widely adopted methods for path planning for robot manipulation.

3.2.1. Rapidly exploring random trees

Rapidly-exploring random trees (RRTs) are a class of sampling-based path planning algorithms that can be used to efficiently search for a feasible path between a starting configuration and a goal configuration of a robotic manipulator in a high-dimensional space. The basic idea of the RRTs is to incrementally build a tree structure that encodes the connectivity between configurations in the space. At each iteration, a new configuration is randomly sampled from the space, and the algorithm finds the nearest configuration in the tree to the new configuration. The new configuration is then added to the tree as a child of the nearest configuration, and the process is repeated until a path from the starting configuration to the goal configuration is found. More formally, the RRT algorithm can be described as follows:

1. Initialize the tree with the starting configuration as the root node.
2. Repeat the following steps until a path is found or the maximum number of iterations reached:
   (a) Sample a new configuration q_{rand} from the configuration space.
   (b) Find the nearest configuration q_{near} in the tree to q_{rand}.

(c) Compute a new configuration q_{new} that is a step along the line between q_{near} and q_{rand}.

(d) If q_{new} is not in collision, add it to the tree as a child of q_{near}.

3. If a path from the starting configuration to the goal configuration has been found in the tree, return the path as the solution to the motion planning problem. Otherwise, return failure.

Overall, the RRT algorithm uses a simple and efficient search strategy to find a feasible path in a high dimensional configuration space, making it a popular choice for motion planning in robotics and other applications. The algorithm then iteratively grows the tree from the starting configuration towards the goal, using local steering functions to guide the tree towards the goal and avoid obstacles. An example path found with the RRT algorithm in a two-dimensional environment is shown in Fig. 3.

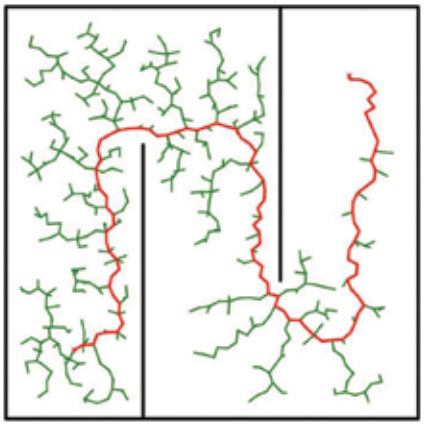

Figure 3. An example path found with the RRT algorithm in a two-dimensional environment. The green lines depict the tree structure, while the red line represents the found solution path.
Slika 3. Primjer putanje pronađene RRT algoritmom u dvodimenzionalnom okruženju. Zelene linije prikazuju strukturu stabla, dok crvena linija predstavlja putanju pronađenog rješenja.

The RRTs have several key features that make them well-suited to a wide range of motion planning problems. First, the tree structure allows the algorithm to quickly explore large areas of the configuration space and rapidly converge on a feasible path. Second, the use of random sampling enables RRTs to handle high-dimensional configuration spaces and complex constraints. Third, the local steering functions allow the RRTs to smoothly follow the natural motion of the system, resulting in paths that are more likely to be feasible in the real world. There are several variants of the RRTs that have been pro-

posed in the literature, including the RRT-Connect [21], the RRT* [22] that guarantees asymptotical optimality, and the RRT-X [23]. More recently, there have been developed extensions [24] that try to improve RRT's performance by relying on heuristics, which *inform* the search [25] or consider kinematic and dynamic constraints [26; 27] to facilitate its applications for complex real-world robots. These variations differ in the specific strategy used to grow the tree and guide the search, but all share the core idea of using random sampling and a tree structure to efficiently search the configuration space.

3.2.2. Probabilistic roadmaps

Probabilistic roadmaps (PRMs) are a class of sampling-based path planning algorithms that can be used to find a feasible path between a starting configuration and a goal configuration in a high-dimensional space while avoiding obstacles. The PRMs work by randomly sampling the configuration space and using these samples to construct a roadmap or graph that encodes the connectivity between configurations in free space. Each sampled configuration is connected to the nearest existing configurations in the roadmap, subject to a given collision-checking constraint. Once the roadmap is constructed, a path from the starting configuration to the goal configuration can be found using a graph search algorithm, such as A [28] or Dijkstra's [29] algorithm. One key advantage of the PRMs is that they can handle high-dimensional configuration spaces and complex constraints, making them well-suited to a wide range of applications, such as robot navigation in cluttered environments or collision-free motion planning for mechanical systems. In practice, the PRMs typically iteratively refine the roadmap by adding new samples and connecting them to the existing roadmap in a way that maintains the connectivity of the free space. This process enables the gradual construction of a more accurate and comprehensive representation of the free space, which in turn improves the quality of the paths found by the algorithm. There are several variants of the PRMs that have been proposed in the literature, including single-query PRMs [9; 30], multi-query PRMs [31; 32], and lazy PRMs [33; 34; 35]. Even though sampling-based methods do not explicitly optimize a cost function, authors in [22] proposed the PRM* algorithm, which is an adaptation of the PRM that was proven to be asymptotically optimal, alleviating one of the main downsides of the original PRM method. These variations differ in the specific strategy used to construct and update the roadmap, but all share the core idea of using random sampling to represent free configuration space with a graph structure that can be queried for the shortest path with a graph search algorithm.

More formally, the PRM algorithm can be described as follows:

1. Initialize the roadmap with the starting and goal configurations.
2. Repeat the following steps until the roadmap is sufficiently dense or the maximum number of iterations reached:

(a) Sample a new configuration q_{rand} from the configuration space.
(b) Find the nearest k configurations $q_{near}1$, $q_{near}2$, ..., $q_{near}k$ in the roadmap to q_{rand}.
(c) Compute a new configuration q_{new} that is a step along the line between q_{rand} and the nearest configuration.
(d) If q_{new} is not in collision, add it to the roadmap and connect it to the nearest configurations.

3. Use a graph search algorithm to find a path in the roadmap from the starting configuration to the goal configuration.
4. If a path from the starting configuration to the goal configuration has been found, return the path as the solution to the motion planning problem. Otherwise, return failure.

An example path found with Dijkstra's algorithm searching over a probabilistic roadmap in a two-dimensional environment is shown in Fig. 4. Overall, the PRM algorithm uses a randomized search strategy to construct a roadmap of the configuration space, and then uses a graph search algorithm to find a feasible path in the roadmap. This can provide efficient and robust solutions to motion planning problems in high-dimensional spaces, which makes the algorithm suitable not only for robotic manipulation, but also for a variety of other applications, such as robot navigation, mechanism design, and computer graphics.

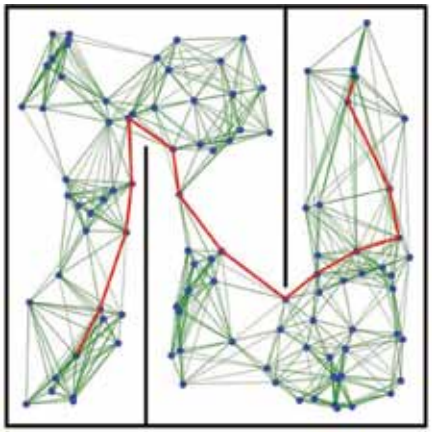

Figure 4. An example path found with Dijkstra's algorithm searching over a probabilistic roadmap in a two-dimensional environment. The blue dots represent collision-free samples from which a roadmap was constructed, the green lines depict edges of the roadmap, whereas the red line represents the found collision-free robot's path.

Slika 4. Primjer puta pronađen Dijkstrinim algoritmom koji pretražuje vjerojatnosnu kartu putanja u dvodimenzionalnom okruženju. Plave točke predstavljaju uzorke bez sudara iz kojih je konstruirana karta putanja, zelene linije prikazuju rubove karte putanja, a crvena linija predstavlja pronađenu putanju robota bez sudara

3.2.3. Limitations of Sampling-based Methods

As the sampling-based planners became increasingly well understood in the recent years, it was suggested that randomization may not, by itself, account for their efficiency [36]. It was shown that quasi-random sampling sequences could accomplish similar or better performance than their randomized counterparts [37]. Nevertheless, many sampling-based methods are computationally inefficient for challenging high-dimensional problems, especially if the environment is dynamic and replanning needed. They may suffer from the curse of dimensionality and frequently spend sizable computational effort for sampling parts of the configuration space that might be irrelevant for the task. Furthermore, sampling-based methods rarely take into account trajectory smoothness; consequently, the obtained paths necessitate post-processing to avoid jerky or redundant motion. In the task space, redundant motion induced by sampling-based paths may manifest itself in relatively large and potentially dangerous swings by the manipulator, just to avoid a relatively small obstacle. The post-processing step that improves smoothness may lead back to infeasible paths and the failure to plan for and execute a desired task. To mitigate the mentioned deficiencies, methods combining potential fields with sampling based planning [38; 39] have been introduced. However, they are unable to optimize secondary costs (e.g. motor torque minimization); for high-dimensional spaces, they either rely on human demonstrations or still require significant computational effort.

4. INVERSE KINEMATICS

Most applications of robotic manipulation in the real world, such as grasping objects or interacting with the environment, are difficult to define solely in terms of joint angles (i.e., in the configuration space). Instead, most tasks are defined in terms of end-effector poses (or other task space coordinates) relative to a static coordinate frame. The problem of computing the end-effector pose corresponding to a joint configuration, known as *forward kinematics*, is solved for common robotic manipulators using constructive geometric techniques. However, the problem of mapping a vector of task space coordinates to a robot's joint configuration, known as *inverse kinematics* (IK), is comparatively more difficult. In the context of the challenges described in Section 2, inverse kinematics encompasses a wide set of problems pertaining to finding configurations that adhere to a set of constraints. Some examples of these constraints include obstacle and self-collision avoidance, remaining within joint limits and keeping elements of the robot within a desired workspace area. This problem may frequently involve optimizing for certain criteria, such as manipulability, stability and proximity of a home configuration. This

means that the solutions to the inverse kinematics problem must not only satisfy the constraints, but it also ought to be designed so as to maximize factors related to the ease of manipulation. Solving the inverse kinematics problem in this way can be difficult, but it is essential for achieving high-performance and reliable control of robotic systems. On account of its widespread use in robotics [40] and computer graphics [7], the IK remains an active research area with an abundance of relevant literature.

Prior to the emergence of robotics, in the 19th century, Sir Isaac Newton and other scientists developed the concept of kinematics, which is the study of motion without considering the forces that cause it. In the 20th century, researchers started to develop methods for solving the inverse kinematics problem, including the use of algebraic equations, geometric techniques, and optimization methods [41]. For robots with six or fewer DoF, that can reach any feasible end-effector pose with up to sixteen configurations, geometric identities can be used to find a set of closed-form solutions by hand [42; 43] or automatically [44]. However, many robots of interest featured a solution set that could not be defined in closed-form [1]. Moreover, an analytical approach generally lacks an elegant way of defining additional constraints and optimality criteria, consequently requiring analysis of each provided solution. In the 1980s and 1990s, there was a surge of interest in inverse kinematics with the advent of modern robotics and computer animation, which led to the development of more sophisticated algorithms that could solve complex inverse kinematics problems. Notably, optimization and heuristic-based techniques started being used for robots with redundant degrees of freedom. Such approaches allowed the inclusion of a variety of optimization criteria, leading to a particular locally optimal solution dependent on the initialization used.

The inverse kinematics problem can be formally defined as

Definition 2 (Inverse Kinematics) *The mapping* $IK: \mathcal{T} \to \mathcal{C}$ *of a task space coordinates* $\boldsymbol{\tau} \in \mathcal{T}$ *to a set of joint configurations* $IK(\boldsymbol{\tau}) = \{\boldsymbol{\theta} \in \mathcal{C} \mid \mathrm{FK}(\boldsymbol{\theta}) = \boldsymbol{\tau}\}$ *is known as the inverse kinematics mapping. The procedure for fully or partially computing this mapping is known as a robot's inverse kinematics.*

The above definition states that, unlike forward kinematics, the IK is generally not unique. In other words, forward kinematics is not injective, and therefore multiple feasible configurations exist for a single set of task coordinates τ. Consequently, there is no single best approach to solving the inverse kinematics problem, and many solution approaches have been developed for particular applications. However, it is important to note that the difficulty of the IK problem varies depending on the underlying assumptions imposed by a given application (e.g., mechanism, task space, etc.) and that these assumptions are often not explicitly stated. To avoid this ambiguity, this review focuses on and stresses a distinction between two instances of the IK problem common to robotics applications: *differential inverse kinematics* and *pose inverse kinematics*.

4.1. Differential Inverse Kinematics

Differential inverse kinematics is the problem of finding joint velocities that generate a desired twist[1] of the end-effector. This instance of the IK also occurs in task space control, where the goal end-effector pose is defined in terms of an incremental motion of the end-effector and carries the assumption of the solution being close to the starting configuration.

When $\dim(\mathcal{C}) = \dim(\mathcal{T})$, joint velocities can be computed by simply inverting the linear Jacobian identity

$$\boldsymbol{J}\dot{\boldsymbol{\theta}} = \dot{\boldsymbol{\xi}}, \tag{1}$$

where $\dot{\xi}$ is the task space velocity, $\dot{\theta}$ is the configuration space velocity, whereas J is the Jacobian matrix. However, this does not generally hold for $\mathcal{T} \triangleq \mathrm{SE}(3)$, since many commercial manipulators have $\dim(\mathcal{C}) > 6$. In his pioneering work [45], Whitney proposed solving Eq. (1) for $\dot{\theta}$ using

$$\dot{\theta} = J^{\dagger}\dot{\xi}, \tag{2}$$

where $\boldsymbol{J}^{\dagger} = \boldsymbol{J}^{T}(\boldsymbol{J}\boldsymbol{J}^{T})^{-1}$ is the Moore-Penrose pseudoinverse. Note that Eq. (2) is the *least-squares* solution to the optimization problem

$$\begin{aligned} &\min_{\dot{\boldsymbol{\theta}}} \tfrac{1}{2}\,\dot{\boldsymbol{\theta}}^{T}\boldsymbol{W}\dot{\boldsymbol{\theta}} + \boldsymbol{c}^{\mathrm{T}}\dot{\boldsymbol{\theta}} \\ &\text{s.t.}\,\mathbf{A}\dot{\boldsymbol{\theta}} = \mathbf{b} \end{aligned} \tag{3}$$

where $\mathbf{W} = \mathbf{I}$, $\mathbf{c} = \mathbf{0}$, $\mathbf{A} = \mathbf{J}$ and $\mathbf{b} = \dot{\xi}$. Crucially, a solution to any such quadratic program [46] reduces to solving a linear system. By applying Eq. (3) iteratively, it is also possible to obtain solutions to the more conventional pose IK problem [47]. Unfortunately, these methods cannot ensure the repeatability of joint trajectories for an end-effector trajectory that is followed [5].

4.1.1. Numerical stability

Regardless of the application, choosing an appropriate method for solving Eq. (3) requires careful consideration. The formulation in Eq. (2) may often become numerically unstable near kinematic singularities [48], and a variety of methods have been developed to address this problem. For example, the pseudoinverse can be replaced with a simple transpose operation [47]

$$\dot{\boldsymbol{\theta}} = \boldsymbol{J}^{T}\dot{\boldsymbol{\xi}}, \tag{4}$$

[1] or linear and angular velocity.

ensuring high numerical robustness at the cost of accuracy [49]. The *damped least squares* (DLS) method[2] [50] adds a damping constant to the Jacobian pseudoinverse

$$\dot{\boldsymbol{\theta}} = \boldsymbol{J}^T(\boldsymbol{J}\boldsymbol{J}^T + \lambda \mathbf{I})^{-1}\,\dot{\boldsymbol{\xi}}, \tag{5}$$

to increase numerical stability. The pseudoinverse damped least squares (PDLS) method uses an SVD decomposition

$$\boldsymbol{J}^T(\boldsymbol{J}\boldsymbol{J}^T + \lambda \mathbf{I})^{-1} = \sum_{i=0}^{n} \frac{\sigma_i}{\sigma_i^2 + \lambda} \boldsymbol{u}_i \boldsymbol{v}_i^T \tag{6}$$

to compute the DLS formulation in Eq. (5) in a way that does not affect performance far from singular configurations [51], where σ_i are singular values of the Jacobian matrix $\boldsymbol{J}$, $\boldsymbol{u}_i$ while and $\boldsymbol{v}_i$ are left and right singular vectors, respectively. Finally, the *selectively damped least squares* (SDLS) method only adds a damping factor to singular values that are close to zero [52].

4.1.2. Redundancy Resolution

Redundant degrees of freedom may be used to track reference velocities in a secondary task space by setting the cost function of Eq. (3) to $\left\|\boldsymbol{J}_s^{\dagger}\dot{\boldsymbol{\tau}}_s - \dot{\boldsymbol{\theta}}\right\|^2$, where $\boldsymbol{J}_s$ and $\dot{\boldsymbol{\tau}}_s$ are the Jacobian and target velocity of the secondary task. The closed-form solution then becomes

$$\dot{\boldsymbol{\theta}} = \boldsymbol{J}^{\dagger}\dot{\boldsymbol{\xi}} + (\boldsymbol{J}^{\dagger}\boldsymbol{J} - \mathbf{I})\,\boldsymbol{J}_s^{\dagger}\dot{\boldsymbol{\tau}}_s\,, \tag{7}$$

These *redundancy resolution* techniques [53; 54] demonstrate the utility of redundant degrees of freedom, for they make it possible to choose solutions that fit a certain criteria. With some modifications, this framework can be extended to multi-level task hierarchies [55] and to non-Euclidean [56] or learned [57] task spaces.

4.2. Pose Inverse Kinematics

Unlike differential IK, pose IK involves finding a configuration that allows some segment of the robot (usually the end-effector) to reach a desired pose. The problem of pose IK exactly fits Definition 2 insofar that no assumptions exist regarding the proximity of the solution manifold to the initial configuration, making solutions difficult to find using the first-order approximations [1]. Moreover, infinitely many solutions may exist to a given problem instance, making pose IK a more difficult problem overall.

[2] also known as the Levenberg-Marquardt algorithm.

4.2.1. Closed-Loop IK

A robot manipulator can be guided towards a desired pose by controlling the angular velocity of its joints. Analogously, the pose IK problem may be solved by solving a sequence of differential IK problems, incrementally guiding the end-effector to the target by following a reference pose change. Some literature refers to this first-order approach as *closed-loop IK* (CLIK) [47], as it emulates a feedback control problem [58]. Major advantages of CLIK methods include their ease of implementation relative to other nonlinear optimization algorithms. Due to its connection with control literature, there also exist a variety of extensions providing numerical robustness [52] and efficient incorporation of secondary objectives through redundancy resolution [54]. However, alongside the convergence issues commonly encountered with first-order local optimization, CLIK methods will often trade numerical stability for accuracy [59].

Assuming a continuous-time model, the convergence of closed-loop methods can be shown using the Lyapunov theory [60]. However, real-time computations have a fixed time step and are limited by the processing power of a computer, so convergence cannot always be guaranteed. Another key problem is the one of determining the magnitude of pose changes in each increment. When no additional constraints are added, the iteration of the Gauss-Newton method is exactly equal to Eq. (2) with $\Delta\boldsymbol{\theta} = \alpha\dot{\boldsymbol{\theta}}$, where the step size α is chosen using a line search algorithm. In [61], an upper bound on the gain α that ensures convergence is found, but this result only holds for a restricted operational space, where the Jacobian has full rank and bounded singular values. Therefore, the result is not applicable in general.

4.2.2. IK as a Nonlinear Program

The pose IK problem can be formulated as a nonlinear program of the form

$$\begin{aligned} \boldsymbol{\theta}^* &= \min_{\boldsymbol{\theta}} f(\boldsymbol{\theta}) \\ \text{s.t.} \quad & h_i(\boldsymbol{\theta}) = 0, \ i = 1, \dots, N_h, \\ & g_j(\boldsymbol{\theta}) \leq 0, \ j = 1, \dots, N_g, \end{aligned} \tag{8}$$

where an error $e(\boldsymbol{\theta}, \mathbf{T}_{goal})$ between the current and goal end-effector poses, or its norm, is encoded as an objective f or as the (in)equality constraints h_i, g_j. While this formulation admits a more general set of constraints and optimality criteria, it usually requires the use of optimization approaches with more computational overhead.

It has been shown that variants of Eq. (8) may be solved with a higher success rate using a variety of unconstrained or bounded nonlinear programming methods, such as the

L-BFGS-B [62] or the SQP [63]. This framework also supports alternative approaches that are based on optimizing over distances and points instead of joint angles. These methods have robust theoretical underpinnings [46] and can approximately support a wide range of constraints through the addition of penalties to the cost function [64]. However, the highly nonconvex nature of the problem makes them susceptible to local minima, often requiring multiple initial guesses before returning a global minimum, if at all.

5. TRAJECTORY OPTIMIZATION

Trajectory optimization techniques carry out the task of computing a continuous, time-parameterized joint trajectory that adheres to a defined set of criteria. Trajectories are generated by optimizing a continuous function representing a set of robot states in order to minimize or maximize a user-defined cost function [47], staying within the defined constraints. In the context of the challenges listed in Section 2, trajectory optimization addresses the problem of keeping the joint trajectories within the physical capabilities of the robot, minimizing joint velocities and torques in order to reduce wear on components and energy expenditure. The overall length of the trajectory in the configuration space is often kept to a minimum as well, since redundant motion of the manipulator is generally undesirable due to safety and maintenance concerns. Additional optimization objectives, such as singularity avoidance [48; 65] and collision avoidance [63], are often included as well.

Trajectory optimization is closely related to the problem of path planning in robotics. While path planning algorithms focus on computing a set of feasible configurations connecting an initial to a final state, trajectory optimization algorithms focus on connecting these configurations with a smooth trajectory representation that minimizes a given cost function. In other words, trajectory optimization can be seen as a step within the broader process of motion planning, where the goal is to compute optimal joint trajectories that allow the robot to successfully execute a given motion plan. Trajectory optimization methods may be used in two different ways to obtain feasible and optimal motion plans. One is to first compute a feasible path using path planning methods presented in Section 3, and then use trajectory optimization methods as a post-processing step that results in a smooth time-parameterized trajectory that can be executed on a real robot. The other is to use trajectory optimization to compute feasible motion plans from scratch by starting from a naive, possibly in-collision initialization, such as a straight-line path in configuration space. While in this case, there are no guarantees that trajectory optimization method will find a feasible motion plan, especially in particularly cluttered environments or with complex task-space constraints, recently proposed methods have showcased success rate in common practical problems [13]. The problem of trajectory optimization is often formalized as

Definition 3 (Trajectory optimization) *Let* $[\boldsymbol{\theta}(t)]$ *be a family of smooth time-parameterized functions (i.e., trajectory representations), where* $\boldsymbol{\theta}(t) \in \mathcal{C}$ and $t \in [0, T]$. *The optimization problem*

$$\begin{aligned} [\boldsymbol{\theta}(t)]^* = \min_{[\boldsymbol{\theta}(t)]} \mathcal{F}[\boldsymbol{\theta}(t)] \\ \text{s.t.} \quad \mathcal{H}_i[\boldsymbol{\theta}(t)] = 0, \ i = 1, \dots, N_h, \\ \mathcal{G}_j[\boldsymbol{\theta}(t)] \leq 0, \ j = 1, \dots, N_g, \end{aligned} \tag{9}$$

where $\mathcal{F}, \mathcal{H}, \mathcal{G}$ *are the cost, equality and inequality constraints, is known as trajectory optimization.*

By solving the problem in Definition 3 using local or stochastic optimization, trajectory optimization algorithms yield smooth, feasible joint motions that minimize a given cost function. An illustration of a trajectory optimization procedure for a planar omnidirectional robot operating in a two-dimensional environment with three obstacles is shown in Fig. 5.

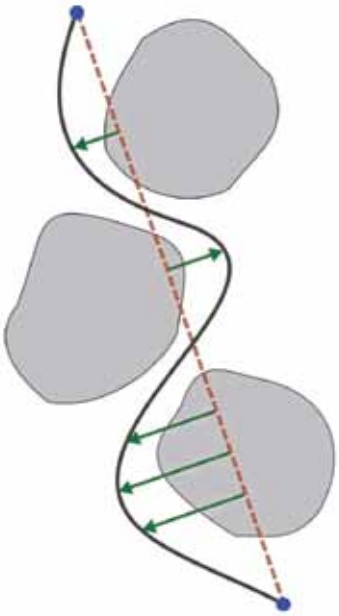

Figure 5. An illustration of the trajectory optimization procedure in a two-dimensional environment. Blue dots represent start and goal states, while the red dotted line depicts the naive straight-line representation that is in collision with the environment. Green arrows represent gradient values at given points, guiding the optimization procedure towards an optimal trajectory that is collision-free. The black line represents the path of the optimal trajectory that is free of collisions.
Slika 5. Ilustracija postupka optimizacije putanje u dvodimenzionalnom okruženju. Plave točke predstavljaju početna i ciljna stanja, dok crvena točkasta linija prikazuje naivni pravocrtni prikaz koji je u sudaru s okolinom. Zelene strelice predstavljaju vrijednosti gradijenta u danim točkama, usmjeravajući postupak optimizacije prema optimalnoj putanji koja je bez sudara. Crna linija predstavlja optimalnu putanju bez sudara.

The history of trajectory optimization can be traced back to the early days of robotics, when researchers first began to develop algorithms for computing smooth, feasible motions for robots. The earliest and most straightforward approach involved optimizing a set of discrete configurations, which could then be linearly interpolated to form a piecewise-smooth trajectory [47] between each pair. The cubic spline interpolation algorithm [66] was introduced in the 1980s [67] and allowed robots to follow smooth, continuous paths between a set of configurations. Since then, a variety of other trajectory representation were explored and applied to trajectory estimation and filtering, based on B-splines [68] and hierarchical wavelets [69]. Representations based on the kernel Hilbert space (RKHS) [70] and probabilistic distributions [13] have proven to be particularly well-suited for trajectory optimization, as they optimize cost functionals in the space of continuous functions.

5.1. Gradient-based Optimization

The elastic strip framework introduced in [11] enables the execution of a previously planned motion in a dynamic environment for robots with many degrees of freedom. It combines real-time obstacle avoidance with desired posture behavior, allowing the modification of a motion in a task-consistent manner, without affecting task execution. The first representative of modern gradient-based trajectory optimization methods is the CHOMP algorithm proposed by Zucker et al. [71], which uses functional gradient techniques to iteratively improve the quality of an initial trajectory, optimizing a cost function that balances smoothness and obstacle avoidance. They define the smoothness functional in terms of a metric in the space of trajectories. The obstacle functional is developed as a line integral of a scalar cost field – a precomputed signed distance field, defined so that it is invariant to retiming. Regardless of how fast or slow the arm moves through the field, it will accumulate the exact same cost. Several augmentations of CHOMP have been proposed to improve it in specific areas, such as performance under constraints [72] and inclusion of time-dependent cost functionals [73]. An important benefit of trajectory optimization approaches is their computational efficiency, which fostered the development of the ITOMP [74] algorithm, which demonstrated real-time replanning in dynamic environments by taking into account moving obstacles.

The Trajopt algorithm [63] can provide feasible motion plans from scratch by optimizing naive straight line trajectories that are in collision using a sequential convex optimization procedure, which penalizes collisions with a hinge loss. The ability to add new constraints and costs to the optimization problem allows Trajopt to tackle a larger range of motion planning problems, including planning for underactuated, non-holonomic sys-

tems. In [13], Mukadam et al. introduce Gaussian Process Motion Planner (GPMP), a novel application of probabilistic inference that uses sparse Gaussian process (GP) models to represent smooth continuous-time trajectories. The gradient based GPMP algorithm exploits the sparsity and the GP interpolation of these models, enabling trajectory optimization to be formulated as probabilistic inference on a factor graph, leading to the development of the highly efficient GPMP2 algorithm. Finally, the authors extend the GPMP2 to an incremental algorithm, the iGPMP2, which is able to efficiently replan in the case of changing conditions [75]. A useful property of the GPMP2 is its extensibility and applicability for a wide range of problems. For example, combined learning from demonstration and motion planning [76] presented an efficient approach to skill learning and generalizable skill reproduction, while the GPMP-GRAPH algorithm [77] enabled planning over multiple homotopy classes.

5.2. Stochastic Optimization

The gradient-based optimization methods mentioned in the previous section rely on first order information about the cost functionals in order to carry out the optimization. In some use cases, the use of gradient-based optimization might not be the best way to solve the problem. Notably, in cluttered environments, there are many local minima to which gradient-based optimization methods may converge, leading to infeasible trajectories and making gradient-based methods unsuitable for reliable motion planning. Furthermore, depending on the task, we might potentially have non-differentiable constraints or cost functions that do not have available derivations.

The STOMP [12] algorithm randomly samples a small perturbation to the current trajectory and uses this perturbation to update the trajectory at each iteration. The updated trajectory is then passed through a smoothing process to ensure that it is feasible and accepted if it results in a lower overall cost than the current trajectory. Due to its capabilities and relatively easy implementation, STOMP was extended for multi-criteria optimization [78] and for supporting Cartesian path constraints [79]. A particle swarm filter method for trajectory optimization [80] displayed being less prone to local minima than gradient-based methods, but it did not showcase its performance in highly cluttered environments. To successfully find multiple solution trajectories, the stochastic multimodal trajectory optimization algorithm (SMTO) [81] utilized variational Bayesian expectation maximization, but it is orders of magnitude computationally slower than STOMP. Mixtures of Gaussian processes were coupled with a STOMP-like optimization technique in [82], which led to good performance in multi-criteria motion planning. Another group of methods uses cross-entropy optimization [83; 84]. In [83], the authors rely

on drawing trajectories from a set of feasible paths, which are then used to optimize a trajectory smoothness objective. The proposed framework generates smooth trajectories; however, sampling whole collision-free trajectories can become infeasible in complex environments. In [84], the authors represented trajectories as samples from a heteroscedastic GP, and the method displayed good performance in cluttered environments with the underlying cross-entropy optimization.

6. CONCLUSION

Robotic manipulation has made significant strides in the recent years, thanks to advances in artificial intelligence and machine learning. These technologies have enabled robots to learn from experience and adapt to new situations, making them more versatile and capable of handling a wider range of tasks. Additionally, the development of new materials and manufacturing techniques has led to the creation of more agile and sophisticated robotic arms, with a greater range of motion and improved tactile sensing capabilities. This has opened up new possibilities for the use of robots in a variety of settings, from factories and warehouses to hospitals and homes. As these technologies continue to evolve, we can expect to see even more impressive feats of robotic manipulation in the future.

In this paper, we presented the challenges of motion planning for robotic arms and several different approaches to tackle them. We first presented approaches for path planning in configuration space, that seek to find a feasible connection between start and goal state robot configurations. We covered the two most commonly used path planning methods – the RRT and the PRM – in more depth. We then presented the inverse kinematics solvers that tackle the problem of finding a configuration that achieves a given end-effector pose in the task space. Since the robot is often given task-space Cartesian goals, our coverage of inverse kinematics provides a connection between task-space goals and configuration space, in which path planning occurs. Subsequently, we presented trajectory optimization approaches as a way of post-processing feasible paths or computing feasible motion plans from scratch. The trajectory optimization methods result in time-parameterized trajectories that can be executed on the robot. When coupled together, path planning, inverse kinematic and trajectory optimization form a robust motion planning framework that is able to efficiently produce feasible manipulation motion in complex environments and with different motion constraints.

REFERENCES

[1] K. M. Lynch and F. C. Park, *Modern Robotics*. Cambridge University Press, 2017.

[2] S. M. LaValle, *Planning algorithms*. Cambridge University Press, 2006.

[3] E. Mandfield, "The diffusion of industrial robots in Japan and the United States," *Research Policy*, vol. 18, no. 4, pp. 183–192, 1989.

[4] L. Sciavicco and B. Siciliano, *Modelling and Control of Robot Manipulators*. Advanced Textbooks in Control and Signal Processing, Springer, 2012.

[5] C. A. Klein and C.-H. Huang, "Review of pseudoinverse control for use with kinematically redundant manipulators," *IEEE Transactions on Systems, Man, and Cybernetics*, no. 2, pp. 245–250, 1983.

[6] S. Chiaverini, B. Siciliano, and O. Egeland, "Review of the damped least-squares inverse kinematics with experiments on an industrial robot manipulator," *IEEE Transactions on control systems technology*, vol. 2, no. 2, pp. 123–134, 1994.

[7] A. Aristidou, J. Lasenby, Y. Chrysanthou, and A. Shamir, "Inverse kinematics techniques in computer graphics: A survey," in *Computer graphics forum*, vol. 37, pp. 35–58, Wiley Online Library, 2018.

[8] M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review," *IEEE Access*, vol. 2, pp. 56–77, 2014.

[9] L. E. Kavraki, M. N. Kolountzakis, and J.-C. Latombe, "Analysis of probabilistic roadmaps for path planning," *IEEE Transactions on Robotics and automation*, vol. 14, no. 1, pp. 166–171, 1998.

[10] S. M. LaValle *et al.*, "Rapidly-exploring random trees: A new tool for path planning," 1998.

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

[12] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: Stochastic trajectory optimization for motion planning," in *2011 IEEE international conference on robotics and automation*, pp. 4569–4574, IEEE, 2011.

[13] M. Mukadam, J. Dong, X. Yan, F. Dellaert, and B. Boots, "Continuous-time Gaussian process motion planning via probabilistic inference," *The International Journal of Robotics Research*, vol. 37, no. 11, pp. 1319–1340, 2018.

[14] M. Likhachev, G. J. Gordon, and S. Thrun, "Ara*: Anytime a* with provable bounds on suboptimality," in *Neural Information Processing Systems (NIPS)*, pp. 767–774, 2004.

[15] F. Islam, V. Narayanan, and M. Likhachev, "A*-connect: Bounded suboptimal bidirectional heuristic search," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2752–2758, IEEE, 2016.

[16] W. Du, F. Islam, and M. Likhachev, "Multi-resolution a," in *Annual Symposium on Combinatorial Search*, pp. 29–37, 2020.

[17] S. Mukherjee, S. Aine, and M. Likhachev, "Mplp: Massively parallelized lazy planning," *IEEE Robotics and Automation Letters*, 2022.

[18] S. Mukherjee, S. Aine, and M. Likhachev, "epa* se: Edge-based parallel a* for slow evaluations," *arXiv preprint arXiv:2203.01369*, 2022.

[19] S. M. LaValle, J. J. Kuffner, B. Donald, *et al.*, "Rapidly-exploring random trees: Progress and prospects," *Algorithmic and computational robotics: new directions*, vol. 5, pp. 293–308, 2001.

[20] J. Barraquand, L. Kavraki, J.-C. Latombe, R. Motwani, T.-Y. Li, and P. Raghavan, "A random sampling scheme for path planning," *The International Journal of Robotics Research*, vol. 16, no. 6, pp. 759–774, 1997.

[21] J. J. Kuffner and S. M. LaValle, "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, pp. 995–1001, IEEE, 2000.

[22] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[23] M. Otte and E. Frazzoli, "RRT^X: Real-time motion planning/replanning for environments with unpredictable obstacles," in *Algorithmic Foundations of Robotics XI*, pp. 461–478, Springer, 2015.

[24] I. Noreen, A. Khan, and Z. Habib, "Optimal path planning using RRT* based approaches: a survey and future directions," *International Journal of Advanced Computer Science and Applications*, vol. 7, no. 11, 2016.

[25] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2997–3004, IEEE, 2014.

[26] A. Perez, R. Platt, G. Konidaris, L. Kaelbling, and T. Lozano-Perez, "LQR-RRT*: Optimal sampling-based motion planning with automatically derived extension heuristics," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2537–2542, IEEE, 2012.

[27] D. J. Webb and J. Van Den Berg, "Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5054–5061, IEEE, 2013.

[28] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[29] D. B. Johnson, "A note on Dijkstra's shortest path algorithm," *Journal of the ACM (JACM)*, vol. 20, no. 3, pp. 385–388, 1973.

[30] D. Hsu, J.-C. Latombe, and H. Kurniawati, "On the probabilistic foun- dations of probabilistic roadmap planning," *The International Journal of Robotics Research*, vol. 25, no. 7, pp. 627–643, 2006.

[31] K. E. Bekris, B. Y. Chen, A. M. Ladd, E. Plaku, and L. E. Kavraki, "Multiple query probabilistic roadmap planning using single query planning primitives," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, vol. 1, pp. 656–661, IEEE, 2003.

[32] M. Akinc, K. E. Bekris, B. Y. Chen, A. M. Ladd, E. Plaku, and L. E. Kavraki, "Probabilistic roadmaps of trees for parallel computation of multiple query roadmaps," in *Robotics Research. The Eleventh International Symposium*, pp. 80–89, Springer, 2005.

[33] G. Sánchez and J.-C. Latombe, "A single-query bidirectional probabilistic roadmap planner with lazy collision checking," in *Robotics research*, pp. 403–417, Springer, 2003.

[34] R. Bohlin and L. E. Kavraki, "Path planning using lazy PRM," in *Proceedings 2000 ICRA. Millennium conference. IEEE international conference on robotics and automation. Symposia proceedings (Cat. No. 00CH37065)*, vol. 1, pp. 521–528, IEEE, 2000.

[35] J. Denny, K. Shi, and N. M. Amato, "Lazy toggle PRM: A single-query approach to motion planning," in *2013 IEEE International Conference on Robotics and Automation*, pp. 2407–2414, IEEE, 2013.

[36] S. M. LaValle, M. S. Branicky, and S. R. Lindemann, "On the relationship between classical grid search and probabilistic roadmaps," *The International Journal of Robotics Research*, vol. 23, no. 7-8, pp. 673–692, 2004.

[37] M. S. Branicky, S. M. LaValle, K. Olson, and L. Yang, "Quasi-randomized path planning," in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, vol. 2, pp. 1481–1487, IEEE, 2001.

[38] Z. Tahir, A. H. Qureshi, Y. Ayaz, and R. Nawaz, "Potentially guided bidirectionalized RRT* for fast optimal path planning in cluttered environments," *Robotics and Autonomous Systems*, vol. 108, pp. 13–27, 2018.

[39] N. García, J. Rosell, and R. Suárez, "Motion planning by demonstration with human-likeness evaluation for dual-arm robots," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 11, pp. 2298–2307, 2017.

[40] J. Angeles, G. Hommel, and P. Kovács, *Computational Kinematics*, vol. 28. Springer Science & Business Media, 2013.

[41] J. M. Selig, *Geometric fundamentals of robotics*, vol. 128. Springer, 2005.

[42] D. Manocha and J. F. Canny, "Efficient inverse kinematics for general 6r manipulators," *IEEE Transactions on Robotics*, vol. 10, no. 5, pp. 648–657, 1994.

[43] M. Husty, M. Pfurner, and H. Schröcker, "A new and efficient algorithm for the inverse kinematics of a general serial 6r manipulator," *Mech. Mach. Theory*, vol. 42, no. 1, pp. 66–81, 2007.

[44] H.-Y. Lee and C.-G. Liang, "A new vector theory for the analysis of spatial mechanisms," *Mech. Mach. Theory*, vol. 23, no. 3, pp. 209–217, 1988.

[45] D. E. Whitney, "Resolved motion rate control of manipulators and human prostheses," *IEEE Transactions on man-machine systems*, vol. 10, no. 2, pp. 47–53, 1969.

[46] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[47] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*. Springer Science & Business Media, 2010.

[48] T. Yoshikawa, "Manipulability of robotic mechanisms," *The International Journal of Robotics Research*, vol. 4, no. 2, pp. 3–9, 1985.

[49] L. Unzueta, M. Peinado, R. Boulic, and Á. Suescun, "Full-body performance animation with sequential inverse kinematics," *Graphical models*, vol. 70, no. 5, pp. 87–104, 2008.

[50] C. W. Wampler, "Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 16, no. 1, pp. 93–101, 1986.

[51] A. A. Maciejewski, "Dealing with the ill-conditioned equations of motion for articulated figures," *IEEE Computer Graphics and Applications*, vol. 10, no. 3, pp. 63–71, 1990.

[52] S. R. Buss and J.-S. Kim, "Selectively damped least squares for inverse kinematics," *J. Graphics Tools*, vol. 10, pp. 37–49, 2005.

[53] H. Hanafusa, T. Yoshikawa, and Y. Nakamura, "Analysis and control of articulated robot arms with redundancy," *IFAC Proceedings Volumes*, vol. 14, no. 2, pp. 1927–1932, 1981.

[54] Y. Nakamura, H. Hanafusa, and T. Yoshikawa, "Task-priority based redundancy control of robot manipulators," *Int. J. Rob. Res.*, vol. 6, no. 2, pp. 3–15, 1987.

[55] B. Siciliano and J.-J. Slotine, "A general framework for managing multiple tasks in highly redundant robotic systems," *Advanced Robotics*, pp. 1211–1216, 1991.

[56] N. Jaquier, L. Rozo, D. G. Caldwell, and S. Calinon, "Geometry-aware manipulability learning, tracking, and transfer," *The International Journal of Robotics Research*, 2020.

[57] C.-A. Cheng, M. Mukadam, J. Issac, S. Birchfield, D. Fox, B. Boots, and N. Ratliff, "Rmpflow: A computational graph for automatic motion policy generation," in *International Workshop on the Algorithmic Foundations of Robotics*, pp. 441–457, Springer, 2018.

[58] L. Sciavicco and B. Siciliano, "Coordinate transformation: A solution algorithm for one class of robots," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 16, no. 4, pp. 550–559, 1986.

[59] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and Control*. John Wiley & Sons, 2005.

[60] G. Antonelli, "Stability analysis for prioritized closed-loop inverse kinematic algorithms for redundant robotic systems," *IEEE Transactions on Robotics*, vol. 25, no. 5, pp. 985–994, 2009.

[61] P. Falco and C. Natale, "On the stability of closed-loop inverse kinematics algorithms for redundant robots," *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 780–784, 2011.

[62] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal, "Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization," *ACM Trans. Math. Softw.*, vol. 23, pp. 550–560, Dec. 1997.

[63] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *Int. J. Rob. Res.*, 2014.

[64] P. Beeson and B. Ames, "TRAC-IK: An open-source library for improved solving of generic inverse kinematics," in *Proc. IEEE Int. Conf. Humanoid Robots (Humanoids)*, pp. 928–935, 2015.

[65] L. Petrović, F. Marić, I. Marković, J. Kelly, and I. Petrović, "Trajectory optimization with geometry-aware singularity avoidance for robot motion planning," in *2021 21st International Conference on Control, Automation and Systems (ICCAS)*, pp. 1760–1765, IEEE, 2021.

[66] S. McKinley and M. Levine, "Cubic spline interpolation," *College of the Redwoods*, vol. 45, no. 1, pp. 1049–1060, 1998.

[67] S. E. Thompson and R. V. Patel, "Formulation of joint trajectories for industrial robots using b-splines," *IEEE Transactions on industrial Electronics*, no. 2, pp. 192–199, 1987.

[68] M. Elbanhawi, M. Simic, and R. N. Jazar, "Continuous path smoothing for car-like robots using b-spline curves," *Journal of Intelligent & Robotic Systems*, vol. 80, no. 1, pp. 23–56, 2015.

[69] S. Anderson, F. Dellaert, and T. D. Barfoot, "A hierarchical wavelet decom- position for continuous-time slam," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 373–380, IEEE, 2014.

[70] Z. Marinho, A. Dragan, A. Byravan, B. Boots, S. Srinivasa, and G. Gordon, "Functional gradient motion planning in reproducing kernel Hilbert spaces," *arXiv preprint arXiv:1601.03648*, 2016.

[71] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "Chomp: Covariant Hamiltonian optimization for motion planning," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1164–1193, 2013.

[72] K. He, E. Martin, and M. Zucker, "Multigrid chomp with local smoothing," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pp. 315–322, IEEE, 2013.

[73] A. Byravan, B. Boots, S. S. Srinivasa, and D. Fox, "Space-time functional gradient optimization for motion planning," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6499–6506, IEEE, 2014.

[74] C. Park, J. Pan, and D. Manocha, "ITOMP: Incremental trajectory optimization for real-time replanning in dynamic environments," in *Twenty-Second International Conference on Automated Planning and Scheduling*, 2012.

[75] J. Dong, M. Mukadam, F. Dellaert, and B. Boots, "Motion planning as probabilistic inference using Gaussian processes and factor graphs," in *Robotics: Science and Systems (RSS)*, vol. 12, 2016.

[76] M. A. Rana, M. Mukadam, S. R. Ahmadzadeh, S. Chernova, and B. Boots, "Towards robust skill generalization: Unifying learning from demonstration and motion planning," in *Conference on Robot Learning (CoRL)*, pp. 109–118, 2017.

[77] E. Huang, M. Mukadam, Z. Liu, and B. Boots, "Motion planning with graph-based trajectories and Gaussian process inference," in *IEEE Inter- national Conference on Robotics and Automation (ICRA)*, pp. 5591–5598, 2017.

[78] D. Pavlichenko and S. Behnke, "Efficient stochastic multicriteria arm trajectory optimization," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4018–4025, IEEE, 2017.

[79] M. Dobiš, M. Dekan, A. Sojka, P. Beňo, and F. Duchoň, "Cartesian constrained stochastic trajectory optimization for motion planning," *Applied Sciences*, vol. 11, no. 24, p. 11712, 2021.

[80] J.-J. Kim and J.-J. Lee, "Trajectory optimization with particle swarm optimization for manipulator motion planning," *IEEE Transactions on Industrial Informatics*, vol. 11, no. 3, pp. 620–631, 2015.

[81] T. Osa, "Multimodal trajectory optimization for motion planning," *The International Journal of Robotics Research*, vol. 39, no. 8, pp. 983–1001, 2020.

[82] L. Petrović, I. Marković, and I. Petrović, "Mixtures of Gaussian processes for robot motion planning using stochastic trajectory optimization," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2022.

[83] M. Kobilarov, "Cross-entropy motion planning," *The International Journal of Robotics Research*, vol. 31, no. 7, pp. 855–871, 2012.

[84] L. Petrović, J. Peršić, M. Seder, and I. Marković, "Cross-entropy based stochastic optimization of robot trajectories using heteroscedastic continuous-time Gaussian processes," *Robotics and Autonomous Systems*, vol. 133, p. 103618, 2020.

PLANIRANJE GIBANJA ZA ROBOTSKU MANIPULACIJU

Sažetak

Robotski manipulatori su klasa robota specijaliziranih za zadatke koji uključuju podizanje, postavljanje i manipuliranje objektima, koordinaciju pokreta i signaliziranje. Planiranje gibanja temeljni je problem u robotici koji uključuje izračunavanje izvedivih trajektorija u prostoru zglobova koje robot treba slijediti, povezujući višestruka stanja definirana ograničenjima zadatka. Zbog velike raznolikosti robotskih struktura i radnih okruženja, postoji niz metoda za rješavanje ovoga problema. Te su metode često komplementarne i međusobno zamjenjive, što otežava njihovu kategorizaciju i formuliranje zajedničkom terminologijom. U ovome radu navodimo metode i pristupe i smještamo ih u kontekst širega problema planiranja gibanja. Počinjemo s pregledom metoda zasnovanih na uzorkovanju i metoda zasnovanih na grafovima koje se koriste u planiranju putanja, gdje je putanja definirana diskretnim skupom konfiguracija, a zatim nastavljamo s pregledom metoda za inverznu kinematiku i optimizaciju putanje. Raspravljamo o izazovima u ovim metodama i pristupima njihovom rješavanju te dajemo pregled nekih recentnih postignuća u području.

Ključne riječi: robotska manipulacija; planiranje gibanja; inverzna kinematika; optimizacija trajektorije.

Ivan Petrović
University of Zagreb
Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
e-mail: ivan.petrovic@fer.hr

Luka Petrović
University of Zagreb
Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
e-mail: luka.petrovic@fer.hr

Filip Marić
University of Zagreb
Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
e-mail: filip.maric@fer.hr

Ivan Marković
University of Zagreb
Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
e-mail: ivan.markovic@fer.hr