# Machines That Think and Move: A Guide to the Future of Robotics

Jainson

First Printing, 2024

# Contents

# 1 Introduction

## 1.1 Motivation

> *„There certainly will be job disruption. Because what is going to happen is robots will be able to do everything better than us. I mean all of us."*
>
> Elon Musk

This statement about robotic application potential was given by Elon Musk during a press conference [18]. In general, a robot is defined as a machine, which is able to sense, think and act. Basically, robotics is a confluent technical development of sensors, advanced algorithms and actuators. Indeed, the sensors are needed to obtain information from environments. The advanced algorithms are adopted to make decisions autonomously, while the actuators are employed to exert forces resulting in robot motions in the environments [4].

In general, robotics has wide range applications, originated from classical manufacturing and industrial assembly facilities to other new domains [40]. For instance, a famous robotic Da Vinci system has been applied for medical surgery, which is able to allow two surgeons to operate together [89]. An ambitious DARPA program is another example, which aims to develop and revolutionize disaster response robots [35]. Overally, there are numerous unprecedented application potential such as in space exploration, automated driving and material handling [40, 49, 100]. In 2015, a global spending on robotics came about seventy one billion US dollars as shown in a statistics [106]. Due to their capability of replacing human labor and improving efficiency, robots have a huge market, which is expected to grow quickly in the next ten years [33]. However, there are great challenges in algorithms development. In fact, not only the number of robots grows but also more complex tasks have to be performed. Consequently, more advanced and efficient algorithms are required in order to solve these problems. Particularly, motion planning algorithm is the most important key, which enables a greater autonomy for robotic systems.

Motion planning is sometimes referred as trajectory generation, which is theoretically an extreme challenging problem. In trajectory generation, several factors must be taken into account. First, collisions with static obstacles as well as among robots must be avoided in order to guarantee safety. Moreover, trajectory performances are optimized for the purpose of utilizing robot capability and maximizing productivity. If the number of robots is large, cooperative trajectory planning is also a prerequisite. Therefore, coordination algorithms become essential. Motivated by these requirements and demands, this book is devoted to solve some optimal and distributed motion planning problems. In this book, several novel results and algorithms are presented.

## 1.2 Problem Description

As parts of production lines, material handling systems play an important role in industry. Conventionally, goods and wares are carried by one-dimensional linear conveyors. Possible paths are predefined for each item by installed equipments [87]. In this case, a continuous material transportation flow is generated. Moreover, collisions and deadlocks [1] are inherently prevented due to the continuous transportation flow. Nevertheless, technological and robotic advancements enable another type of material handling, which is discontinuous and more flexible. Indeed, each item can be individually transported in a two-dimensional space by a robot [29]. However, these discontinuous material handling systems induce a great demand for advanced algorithms, which are able to prevent collisions as well as resolve conflicts among robots. Consequently, several open research questions have been raised. Before making an excursus about these research questions, an overview about the discontinuous handling systems is provided and their characteristics will be specifically described.

**Figure 1.1:** Grid-flow with transport vehicles. Photograph of Amazon Kiva Systems [21]

Due to an industrial revolution 4.0 initiative, several modular flexible material handling systems have been recently developed and demonstrated in industrial shows. For interested readers, an exhaustive survey is conducted by Zäzilia Seibold [87]. Despite being known under different names such as plug and play, cellular and fluid logistics, etc., these systems can be categorized into two groups. The first group is a grid-flow with active transporting robots as shown in Figure 1.1. Such systems can be found in many warehouses, where pallets are placed in a grid pattern. Furthermore,

[1] *A set of processes is deadlock when every process in the set is waiting for a resource that must be released by another process in the set* [96].

the robots are able to translate underneath the pallets. Consequently, they are able to pick and deliver the pallets to any destination [21]. In such systems, each robot can decide its own movements. Moreover, conflicts are resolved by global central algorithms running on a master computer.

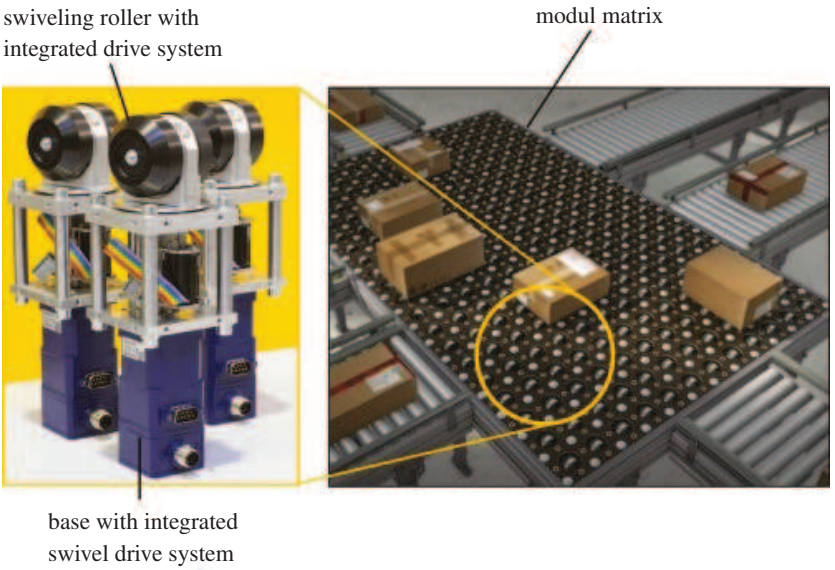


**Figure 1.2:** Photograph of a cognitive convey [50]. The omhniwheels are placed underneath items.

In contrast to the first group, robots are passive and driven by actuators placed underneath in the second group. The actuators are controlled in a decentralized manner. An example of such systems is depicted in Figure 1.2. In this example, the actuators are omniwheels. As opposed to the previous case, deadlocks and conflicts can be resolved by distributed algorithms deployed directly into the actuators. Nevertheless, very few results about the distributed algorithms are published, which are also not precisely described [87]. Moreover, these distributed algorithms can only be applied for primitive motions, e.g. four cardinal directional movements. In order to fully utilize the system flexibility and maximize the whole system performance, more complex and aggressive motions are desired, which results in the following open research problems.

**Problem 1: Time-optimal Trajectory Generation Problem**

Time-optimal trajectory generation is a fundamental optimal control problem, which is encountered in robotics. Given a list of obstacles, initial and goal states, computing time-optimal trajectories is a very complex problem. In fact, collision avoidance induces non-convex constraints.

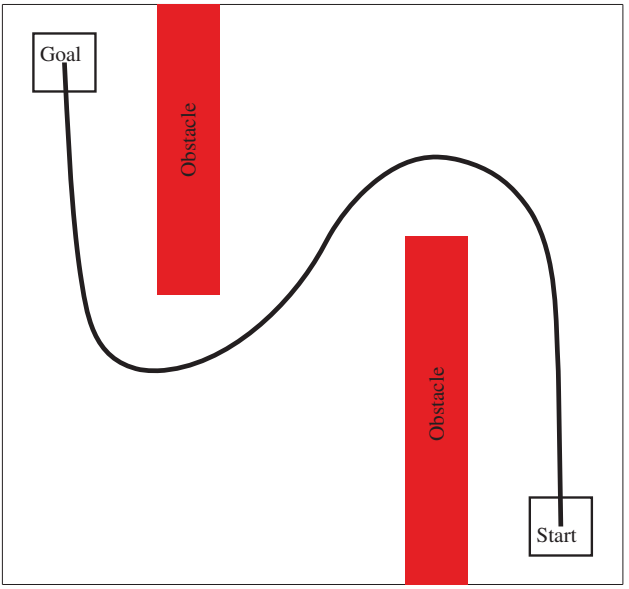


**Figure 1.3:** The time-optimal trajectory generation problem

Basically, the time-optimal control problem is formulated as follows

$$\min_{\mathbf{q}(t),T} \quad T \tag{1.1a}$$

$$\text{subject to} \quad \mathbf{q}(t) = \begin{bmatrix} q_x(t) \\ q_y(t) \end{bmatrix}, \quad \mathbf{q}(0) = \mathbf{q}_{\text{start}}, \qquad \mathbf{q}(T) = \mathbf{q}_{\text{goal}}, \tag{1.1b}$$

$$\dot{\mathbf{q}}(0) = \begin{bmatrix} v_{0,x} \\ v_{0,y} \end{bmatrix}, \quad \dot{\mathbf{q}}(T) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \qquad \ddot{\mathbf{q}}(0) = \begin{bmatrix} a_{0,x} \\ a_{0,y} \end{bmatrix}, \quad \ddot{\mathbf{q}}(T) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \tag{1.1c}$$

$$-\mathbf{v}_{\max} \leq \dot{\mathbf{q}}(t) \leq \mathbf{v}_{\max}, \qquad -\mathbf{a}_{\max} \leq \ddot{\mathbf{q}}(t) \leq \mathbf{a}_{\max}, \qquad -\mathbf{j}_{\max} \leq \dddot{\mathbf{q}}(t) \leq \mathbf{j}_{\max}, \tag{1.1d}$$

$$O(\mathbf{q}(t)) \cap O_i = \varnothing, \quad \forall\, i = 1, ..., N_{\text{o}}, \quad \forall\, t \in [0 \quad T], \tag{1.1e}$$

where

- $q_x(t)$ and $q_y(t)$ are respectively $x$ and $y$ coordinates. The start and goal positions are denoted as $\mathbf{q}_{\text{start}}$ and $\mathbf{q}_{\text{goal}}$. In this book, trajectory generation for two-dimensional robots is considered. However, the algorithm can be extended and applied to other robotic systems with higher dimensional configuration spaces.
- $\dot{\mathbf{q}}(0) = \begin{bmatrix} v_{0,x} & v_{0,y} \end{bmatrix}^T$ and $\ddot{\mathbf{q}}(0) = \begin{bmatrix} a_{0,x} & a_{0,y} \end{bmatrix}^T$ are the initial velocity and acceleration. Moreover, robots are required to be stopped at the goal postion, which leads to $\dot{\mathbf{q}}(T) = \mathbf{0}$ and $\ddot{\mathbf{q}}(T) = \mathbf{0}$.

- The velocity, acceleration and jerk are bounded by $\mathbf{v}_{\max}$, $\mathbf{a}_{\max}$ and $\mathbf{j}_{\max}$, which correspondingly determine kinematic constraints .
- $O(\mathbf{q}(t))$ is the robot occupied area at the position $\mathbf{q}(t)$ and $\mathbf{O} = \{O_1, ..., O_i, ..., O_{N_o}\}$ is a list of obstacles. Additionally, each obstacle $O_i$ is assumed to be a convex polygon. Nevertheless, this assumption is very universal, since general geometrical objects can be approximated by a set of convex polygons.

Solving this optimal control problem is extremely challenging, which usually requires intractable computational burden. However, an efficient algorithm solving the optimal control problem (1.1) has a huge industrial application potential. Therefore, there is a substantial motivation to develop a novel trajectory generation algorithm, which is able to outperform and mitigate some drawbacks of the state-of-the-art algorithms.

**Problem 2: Centralized coordination Problem**

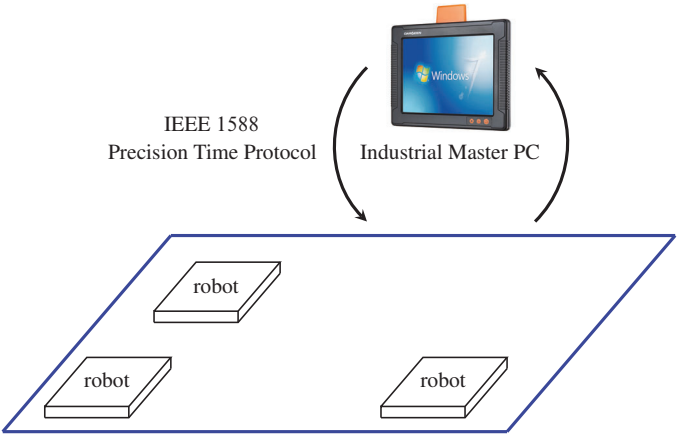


**Figure 1.4:** The centralized system architecture

Given a number of robots, our ultimate objective is to plan trajectories, which bring each robot from a start position to a goal postion. For several industrial applications, where the number of robots is small, all decisions can be made centrally by a master computer. As illustrated in Figure 1.4, the robots and the master computer clocks are communicated over a network, which is synchronized by standard technologies such as IEEE 1588 precision time protocol [2]. This system architecture is prevalent in industry and can be found in several products available in markets [8]. This system architecture and the objective lead to a second problem, how central control principles can be designed, which satisfies all subsequent requirements. The most important requirement is that collisions among robots as well as with static obstacles must be avoided. Additionally, deadlocks must be prevented such that each robot can reach its goal. These control principles, referred

a centralized motion coordination solution, are performed by the master computer. Furthermore, these control principles should consider the solutions of **Problem 1** as inputs and operate as a subsequent process. As a consequence, the previous development results can be utilized and a modular solution is achieved.

**Problem 3: Distributed Coordination Problem**

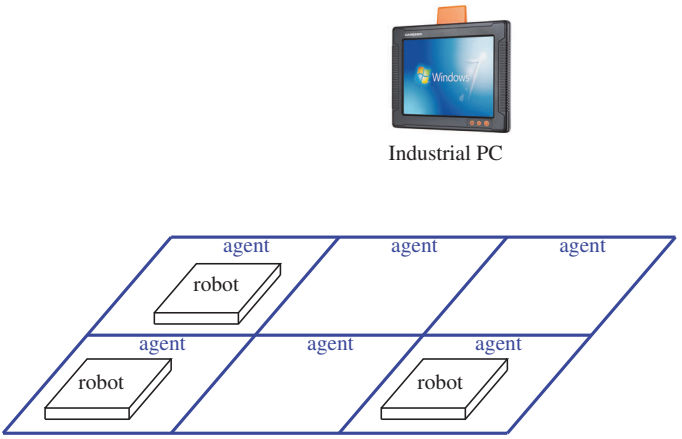


**Figure 1.5:** The distributed system architecture

In contrast to **Problem 2**, the system is divided into modules, namely agents. These agents are placed underneath passive robots, which actually transport wares as shown in Figure 1.5. The distributed architecture can model several industrial systems such as the aforementioned cognitive conveyor, etc. [28, 50, 87]. Moreover, decisions are not centrally taken by a master computer. Indeed, each agent decides movements for the passive robots placed on itself independently based on its local information. Due to a lack of global information and resources, the problem of designing a distributed coordination solution is more challenging. Nevertheless, applying the distributed coordination coordination solution will increase overall robustness, since the system can continue to operate in spite of the fact that a single agent is defected. Furthermore, one is able to scale up the system with the distributed coordination solution. Therefore, these articulated reasons inspire us to establish backgrounds and develop an initial solution to the distributed coordination problem. In order to design the distributed coordination solution, the following basic questions will be addressed in this work. Firstly, which information is required to be exchanged among agents? Secondly, how is the transferred information proceeded and fused together with the local information? Thirdly, how can mutual collision avoidance among the passive robots be guaranteed? In analogy to the centralized coordination solution, the solution of the **Problem 1** is employed as

inputs and the distributed coordination solution operates only as a post process in order to utilize the previous development results.

## 1.3 book Outline

The book is organized in seven chapters. The outline is sketched in Figure 1.6. After formulating the problems, a solution architecture is introduced in Chapter 2. Moreover, fundamental backgrounds as well as a literature review about the state-of-the-art algorithms are provided in this chapter, which are relevant for understanding the book. Additionally, the scientific contributions are highlighted.

Chapter 3 explains working principles of geometrical planners, which are essential elements of the trajectory optimization algorithm. Furthermore, computational efficiency and optimality performances among the distinct geometrical planners are compared. Moreover, advantages and shortcomings are analyzed in detail. In Chapter 4, the near time-optimal trajectory generation algorithm is presented as a solution to **Problem 1**. Remarkable properties and efficiency of the algorithm are illustrated by numerical examples.

In Chapter 5, the solution to the centralized coordination **Problem 2** is introduced. Ingredient algorithms for the centralized solution are disclosed. Moreover, a theoretical invariant safety guarantee proof is given. Chapter 6 proposes an asynchronous distributed collision avoidance algorithm, which is an important component to the solution of the distributed coordination **Problem 3**. Analogously, theoretical safety guarantee can be also proven. The applicability of the centralized and distributed coordination solutions is demonstrated and verified in simulations.

In Chapter 7, a summary of the book and an outlook on future works are given.

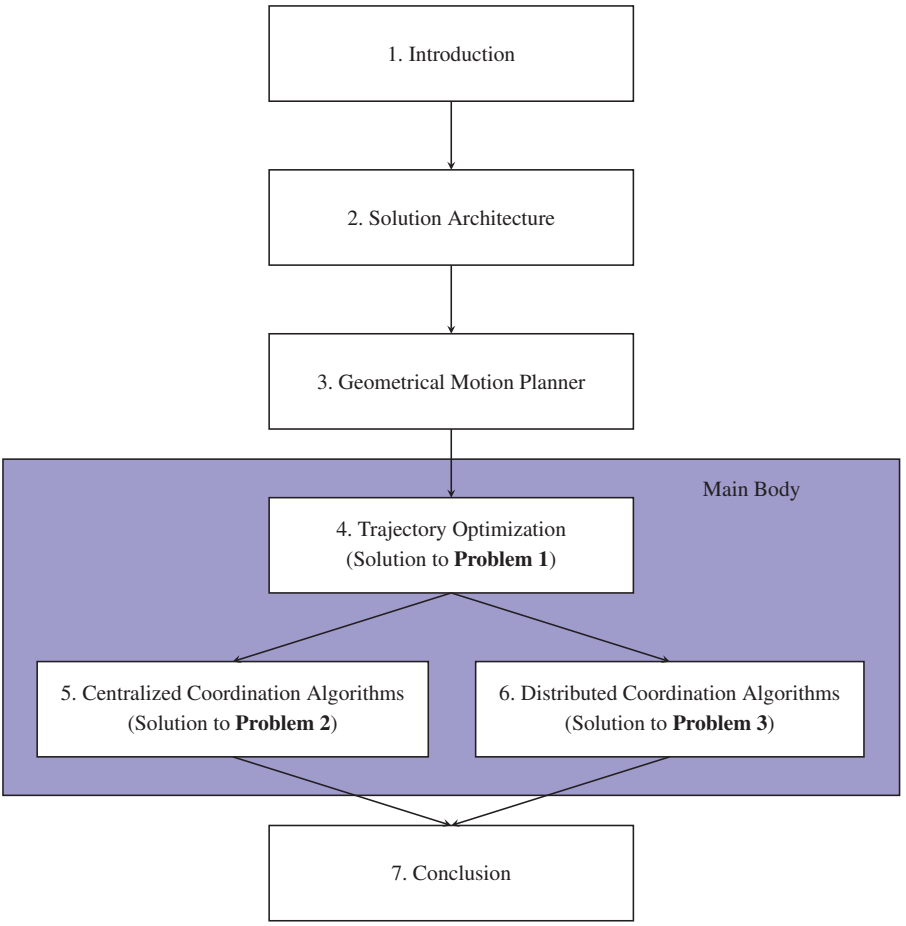


**Figure 1.6:** The book structure

# 2 Solution Architecture for Multiple Robots Motion Planning

*„If you procrastinate when faced with a big difficult problem, break the problem into parts, and handle one part at a time."*

Robert Collier

This statement is applied as an essential philosophy to tackle multirobot motion planning problems. The multirobot motion planning problems have been the most difficult since the first robot appearance [57]. Generally, the multirobot motion planning problems in two-dimensional configuration spaces can be briefly formulated in the following. Given two or more robots, an ultimate objective is bringing each robot from an initial position to a goal position simultaneously. Additionally, mutual collision avoidance must be ensured. Overall, performance optimality should be incorporated into these motion planning problems. An intrinsic approach is the global centralized planning, in which the robots are accumulated as a single composite robot. Therefore, a multirobot motion planning problem can be considered as a classical single robot motion planning problem in a composite joint configuration, which can be solved by applying planning algorithms for high-dimensional spaces. However, degrees of freedom of the single composite robot are not bounded and grow linearly with robot numbers. Consequently, these planning algorithms on the single composite robot has high computational burden [3, 57]. According to the best of our knowledge, no existing optimal algorithm is able to solve the multirobot motion planning problems in polynomial time. Therefore, optimality and completeness [1] are relaxed, which leads to decoupled planning approaches with remarkable low computational complexity. Based on the decoupled approaches and inspired by the aforementioned philosophy, a solution architecture for the multirobot motion planning problems is introduced in this chapter. Particularly, the solution architecture is divided into several modules, which have distinct tasks and functions. This chapter is structured as follows:

- **Background and Solution Architecture:** An overview on the decoupled planning approaches is given. Subsequently, the solution architecture is revealed.
- **Literature Review:** A literature review is provided, which gives an overview on the existing state-of-the art algorithms for each module as well as the latest development in the multirobot motion planning.
- book **Contribution:** The scientific contributions of this book are highlighted.

[1] *An algorithm is complete if it is guaranteed to find a solution when there is one [84, p. 80].*

## 2.1 Background and Solution Architecture

In order to reduce the computational complexity, the decoupled planning approaches are applied, which are presented in several publications [6, 36, 57, 70, 102]. In the decoupled planning approaches, trajectories are generated more independently. Consequently, these approaches induce low computational complexity in exchange for a loss of completeness. In other words, the approaches can fail to find an existing solution. Generally, the decoupled planning approaches can be classified into path coordination and prioritized planning [6, 36, 70, 102], which are briefly explained in the following.

Path coordination is the most prevalent decoupled planning approach [36, 70]. The idea of the path coordination is straight forward. Let's define a path for a robot $i$ as a mapping function

$$\mathcal{P}^i(u^i) : u^i \in \begin{bmatrix} 0 & 1 \end{bmatrix} \mapsto \mathbf{q}^i, \qquad \mathcal{P}^i(u=0) = \mathbf{q}^i_{\text{start}} \quad \text{and} \quad \mathcal{P}^i(u=1) = \mathbf{q}^i_{\text{goal}},$$

where $u^i$ is the path parameter and $\mathbf{q}^i$ is the robot $i$ position. In the path coordination, the path $\mathcal{P}^i(u^i)$ for the robot $i$ is firstly planned regardless of other robots. In order to obtain a trajectory, another mapping function from a time variable $t$ to the path parameter $u^i$

$$u^i(t) : t \in \begin{bmatrix} 0 & T^i_{\text{end}} \end{bmatrix} \mapsto u \in \begin{bmatrix} 0 & 1 \end{bmatrix}$$

is required, where $T^i_{\text{end}}$ is the total travel time of the robot $i$. Afterwards, scheduling algorithms are applied to avoid mutual collisions among robots [36, 70]. Nevertheless, the path coordination can lead to deadlocks. In fact, if a robot obstructs another robot from reaching its goal, the path coordination is not capable to resolve these conflicts. Therefore, another solution is derived, which is prioritized planning.

In the prioritized planning, a feasible priority sequence $\mathbf{P} = \{p_1, p_2, ..., p_{N_\text{r}}\}$ is searched and optimized [6, 102]. Let assume that a robot $i$ has a higher priority than a robot $j$, then the robot $j$ must consider the robot $i$ as a dynamical obstacle. Nevertheless, the main difficulty of the prioritized planning is that trajectories must be generated in a time-space configuration $\mathcal{CT}$, which is complex and usually requires high computational burden [31].

Based on both the prioritized planning and the path coordination, a solution architecture is developed, which is able to combine and utilize the advantages of the both approaches. Regardless of the centralized and distributed system architecture mentioned in Chapter 1, the solution architecture consists of three fundamental components. As shown in Figure 2.1, these components are subsequently explained in the following.

• **Trajectory Optimization Algorithm:**
In the solution architecture, the trajectory optimization algorithm is a key enabler, which generates

trajectories for each robot from an initial state to a goal state. In order to guarantee that the obtained trajectories are safe, the optimization algorithm must exchange information and cooperate with a mutual collision avoidance algorithm. Additionally, the trajectory optimization algorithm must collaborate with a conflict resolution algorithm for purpose of preventing deadlocks.

• **Mutual Collision Avoidance Algorithm:**
This algorithm is employed to avoid mutual collisions among robots. In the path coordination, the scheduling algorithm can be interpreted as a collision avoidance algorithm. As aforementioned, the collision avoidance algorithm must work with the trajectory optimization algorithm.

• **Conflict Resolution Algorithm**
Finally, a conflict resolution algorithm is required in order to avoid deadlocks. Based on priority assignments, a conflict resolution algorithm is presented in Chapter 5. In this conflict resolution algorithm, a robot can be commissioned to a temporary goal in order to reserve shared free working spaces for other robots. In order to give the temporary goal to the robot, the conflict resolution algorithm must consequently communicate with the trajectory optimization algorithm as shown in Figure 2.1.

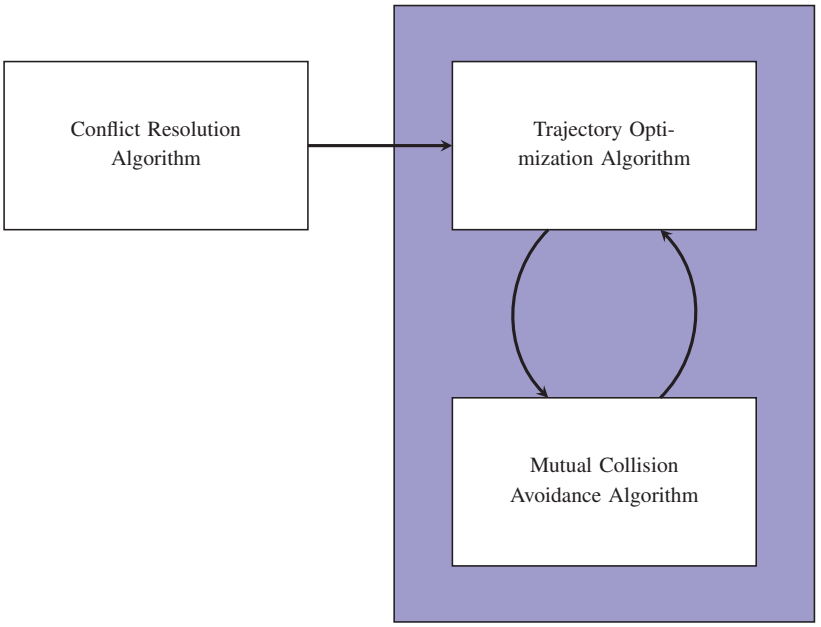


**Figure 2.1:** The solution architecture

## 2.2 Literature Review

In order to distinguish the scientific contributions, a literature review on existing algorithms for trajectory optimization, mutual collision avoidance as well as conflict resolution is given in the following. Subsequently, novel results are indicated, which are able to mitigate some disadvantages of the cutting edge algorithms.

### 2.2.1 Time-optimal Trajectory Generation Algorithm

As mentioned in Chapter 1, a time-optimal trajectory generation algorithm is a key to utilize robot capability. Nevertheless, time-optimal trajectory planning problems are the most challenging in robotics. Particularly, the problems become extremely complex if high-order time derivatives or kinematic constraints are considered [17, 23, 45, 46, 60, 73, 80]. Consequently, a trade-off between trajectory optimality and computational burden is introduced in practice. For solving the time-optimal trajectory planning problems, several approaches have been proposed by the control and computer science community in the last decades. In the following, these approaches as well as related works are revisited.

**Trajectory generation based on velocity planning**
The first approach employes geometrical path planners such as A$^*$, D$^*$, probabilistic roadmaps (PRMs) or rapid exploring random trees (RRTs). Afterwards, a list of way points is obtained. Trajectories are determined by iteratively planning velocity to a next way point. Moreover, the trajectories are smoothed by trying to delete a number of the way points during execution. Furthermore, static obstacle collision avoidance is guaranteed by incorporating *reactive methods* like Dynamic Window Approach (DWA) [30]. Due to its simplicity and easy implementation, this approach has gained a huge success. In fact, it is deployed in industrial mobile robots like Kiva System and applied by several teams in world robot soccer competitions [11, 12, 78, 110]. Nevertheless, trajectory generation based on velocity planning has two main drawbacks. Firstly, predicting ultimate smooth trajectories is difficult and overall performance such as total travel time is not optimized. Secondly, the velocity planning results in infinite jerks, which has a consequence of large tracking errors for high performance motion control systems [54].

**Optimal sampling-based motion planning on state spaces**
The second approach is to apply optimal sampling methods such as RRT$^*$ on state spaces [45, 46]. Nevertheless, the main difficulty associated with this approach is computational burden due to the following reasons. Firstly, dimensions of the state spaces are at least twice that of the configuration spaces. In the problem (1.1), the state space dimension is six, which is three times as many as the configuration space dimension. Moreover, searching on high-dimensional spaces is very time consuming. Eventually, RRT$^*$ requires optimal trajectory segments connecting any two

nodes in the state spaces, which is obtained by solving two-point boundary value problems (TBVPs). Under consideration of state constraints, solving a TBVP is very challenging and a high computational burden is inevitable. For instance, a variant algorithm of RRT* minimizing control effort and travel time is presented by Dustin Webb [108], which is not able to constrain velocities.

**Motion planning based on optimization and positive invariant sets**
A natural idea corresponding to model predictive control (MPC) is employing numerical optimization for trajectory generation. Motion planning algorithms based on receding horizon optimization are introduced by Tom Schouwenaars and Yoshiaki Kuwata [52, 85]. Nevertheless, incorporating collision avoidance constraints leads to a mixed integer linear programming problem (MILP). Moreover, solving a MILP requires high computational effort. An alternative algorithm is presented by Claus Danielson [20], in which state and input constraints are handled by designing a graph of local controllers using positive invariant sets. Afterwards, a graph search algorithm is applied to find a sequence of the local controllers. A main disadvantage of this approach is that no metric exists for weighting edges of the local controller graph in order to provide an overall good performance. Moreover, computing the graph of the local controllers is also time-consuming.

**The decompositional approach**
The trajectory generation algorithm presented in this book is categorized as a decompositional approach, which is the most favorable and effective in practice [39, 73, 91, 93, 111]. Particularly, the near time-optimal trajectory generation is decoupled into two separated phases. First, a feasible collision-free path is generated based on geometrical way points and spline function parametrization [39, 93, 111]. Afterwards, an optimal trajectory along the specified path is planned. An ultimate trajectory is obtained by optimizing path parameters. Nevertheless, a near time-optimal trajectory along a given path can be calculated by using forward and backward integrations along with analytically solving optimal switching points. Consequently, infinite jerk profiles are induced by these proposed approaches [7, 73, 88, 91]. Moreover, the time discretization effect is not considered by these approaches.

In this book, a near time-optimal trajectory generation algorithm is presented Chapter 4, which is able to handle the drawbacks of all aforementioned approaches. For high-performance precision motion control systems, considering the time discretization effect is the most decisive determinant [54]. The proposed algorithm is able to take the time discretization effect into account. In fact, the jerk $\dddot{q}_\xi(t)$, $\xi \in \{x, y\}$, is only discontinuous at time-stamps $t_\mathrm{d}$, which are integer numbers of the sampling time $T_\mathrm{s}$, i.e. $t_\mathrm{d} = nT_\mathrm{s}$ and $n \in \mathbb{N}$. Another novel algorithm is also submitted and accepted by the International Journal of Control, Automation and Systems [76], which is not reported in the book. Based on a clever geometrical construction, this algorithm utilizes some important properties of non-uniform B-splines to smooth geometrical paths. By employing an interior point optimizer (Ipopt) [15, 105], the algorithm is capable of exploiting kinematic constraints. Consequently, this algorithm is able to generate better trajectories within much smaller computational time amounts than the state of the art algorithm [45].

### 2.2.2 Mutual Collision Avoidance Algorithms

In general, mutual collision avoidance algorithms can be applied to not only robotics but also other fields such as air traffic control or studying biological flocks [82, 112]. Consequently, several research results on the mutual collision avoidance algorithms have been conducted [11, 12, 78, 86, 101]. Obviously, these algorithms must be compatible with the system architectures. In other words, an appropriate algorithm can be chosen based on a predefined system architecture. As described in Chapter 1, two system architectures are considered. Specifically, the mutual collision avoidance algorithms are implemented in either the central master computer or the distributed agents. Moreover, an invariant collision-free condition must be guaranteed all the time. Due to these requirements, improper approaches can be neglected. In the following, important approaches related to our centralized and distributed collision avoidance algorithms are reviewed. Moreover, the novelty of the proposed collision avoidance algorithms is explained.

**Centralized collision avoidance algorithms**
An instinctive centralized approach is employing the model predictive control idea. Particularly, the trajectory optimization and the mutual collision avoidance algorithms can be formulated together as a large mixed integer programming problem (MILP) [86]. This formulation yields a large non-convex optimization problem, which can consequently be applied for a small robot number [85]. Moreover, another disadvantage of this approach is non-modularity. In contrast to the MILP formulation, dynamical safety search (DSS) is able to operate modularly as a post process. Additionally, the DSS can be deployed to a larger robot number due to its polynomial computational complexity. Nevertheless, this approach can only be applied to lower-order trajectories [11], while the proposed trajectory optimization algorithm generates high-oder trajectories. Therefore, the DSS can not be incorporated into the trajectory optimization algorithm. Moreover, stochastic performance of the DSS is not appropriate in some industrial applications. Therefore, a novel collision avoidance algorithm will be proposed in Chapter 5, which can mitigate these disadvantages of the DSS as well as be integrated with the trajectory optimization algorithm.

**Distributed collision avoidance algorithms**
A well-known distributed approach is the reciprocal $n$-body collision avoidance algorithm presented by Jur van den Berg [101]. However, it is assumed that each robot is able to sense positions and velocities of other robots. Afterwards, optimal velocities can be planned by using linear programming. Due to the additional sensor requirement, this approach can be discarded. In analogy to the centralized approaches, the model predictive control idea can be applied for solving the distributed collision avoidance problems. Indeed, a distributed MILP formulation is introduced by Yoshiaki Kuwata, which has been applied to multiple unmanned aerial vehicles [52]. By exchanging planning trajectories, a smaller MILP can be formulated and deployed in a distributed manner. Furthermore, the collision-free condition can be invariantly guaranteed by incorporating additional constraints into the distributed MILP formulation. Alternatively, a distributed collision avoidance approach based on a reserved and requested concept was proposed by Oliver Purwin [78]. In analogy to the distributed MILP formulation, collision avoidance can be theoretically guaranteed by

exchanging plans and intentions among robots. Moreover, this approach is robust against stochastic communication delays. Nevertheless, the reserved and requested approach and the distributed MILP formulation can not be applied to solve our distributed collision avoidance problem due to the intrinsic distributed system architecture presented Chapter 1. Particularly, the algorithms are deployed into mobile robots instead of the static agents. Therefore, a novel distributed collision avoidance algorithm is developed, which is similarly based on the reserved and requested concept. Furthermore, the proposed distributed collision avoidance algorithm is able to operate in an asynchronous communication network. This distributed collision avoidance algorithm is presented in Chapter 6.

### 2.2.3 Conflict Resolution Algorithms

The collision avoidance algorithms guarantee safe movements among robots. Nevertheless, there are situations, where robots wait for others indefinitely. An example is that a robot at its goal position obstructs other robots from reaching their individual goals. These situations are defined as deadlocks, which can be resolved by deploying conflict resolution algorithms. In analogy to the collision avoidance algorithms, these conflict resolution algorithms can be categorized into centralized and distributed types, which will be explained in the following.

**Centralized conflict resolution algorithms**
In analogy to the centralized collision avoidance algorithms, a large MILP formulation can be employed to resolve deadlocks among multiple robots, which has intractable computational burden and consequently is impractical [52]. Contrarily, the prioritized planning is more appropriate, since it has lower computational burden than the MILP formulation. Nevertheless, the prioritized planning requires precise complex trajectory generation in a time-space configuration. In order to facilitate this trajectory generation, the time-space configuration is approximately split into a grid. Subsequently, an optimal priority sequence can be found by applying the A$^*$ and hill-climbing algorithms. In Chapter 5, this approach will be presented .

**Distributed conflict resolution algorithms**
In the distributed conflict resolution algorithms, considerable results are usually based on a set of rules [34, 95]. Particularly, a behavior-based model is proposed by Dali Sun [95]. Under an assumption that a robot is able to sense critical front areas, a set of traffic rules can be designed. This algorithm is robust and was demonstrated during an industrial fair in Stuttgart [95]. However, the proposed algorithm is not able to resolve conflicts involving more than two robots. Consequently, deadlock-free systems can not be guaranteed. Contrarily, a deadlock-free algorithm is proposed by Kevin Gue for a system, which consists of rectangular agents and is similar to the aforementioned distributed system architecture [34]. This algorithm can be briefly summarized in the following. Each agent has two possible states, which are empty and occupied. Based on these states and intentions of neighboring agents, a set of rules can be constructed, which guarantees a deadlock-free system. Nevertheless, robot start and goal locations are limited by this algorithm. Indeed, all

robots are assumed to start at one side and end at another side of conveyers. Consequently, the algorithm can not be deployed to any desired conveyor architecture. Furthermore, full flexibility as well as aggressive maneuvers for the robots can not be achieved by this algorithm, since only four cardinal primitive motions are allowed. Although the algorithm can be employed directly to our system, there is nonetheless a great demand for distributed conflict resolution algorithms in general cases.