Efficient Robot Motion Planning via Sampling and Optimization

Jessica Leu, Ge Zhang, Liting Sun, and Masayoshi Tomizuka

***Abstract*— Robot motion planning is one of the important elements in robotics. In environments full of obstacles, it is always challenging to find a collision-free and dynamically-feasible path between the robot's initial configuration and goal configuration. While many motion planning algorithms have been proposed in the past, each of them has its pros and cons. This work presents a benchmark which implements and compares existing planning algorithms on a variety of problems with extensive simulation. Based on that, we also propose a hybrid planning algorithm, RRT*-CFS, that combines the merits of sampling-based planning methods and optimization-based planning methods. The first layer, RRT*, quickly samples a semi-optimal path. The second layer, CFS, performs sequential convex optimization given the reference path from RRT*. The proposed RRT*-CFS has feasibility and convergence guarantees. Simulation results show that RRT*-CFS benefits from the hybrid structure and performs robustly in various scenarios including the narrow passage problems.**

I. INTRODUCTION

Motion planning is one of the key challenges in robotics. It refers to the problem of finding a collision-free and dynamically-feasible path between the initial configuration and the goal configuration in environments full of obstacles. Existing motion planning algorithms fall into two categories: planning-by-construction or planning-by-modification [1]. Searching-based planning and sampling-based planning are two typical plan-by-construction algorithms. Notable algorithms such as A* and D* search [2], [3] fall into the category of searching-based algorithms, whereas rapidly-exploring random tree (RRT) [4] and probabilistic roadmap (PRM) [5] belong to the category of sampling-based planning. Planning-by-modification refers to the algorithms that reshape a reference trajectory to obtain optimality regarding specific properties [6], [7], [8], [9], [10]. Optimization-based motion planning algorithms belong to this category.

Both types of motion planning algorithms have pros and cons. Planning-by-construction algorithms are guaranteed to generate collision-free trajectories and require short computational time. However, the constructed trajectories often do not consider dynamic constraints and are not smooth. Some modifications remedy these problems [11], [12], but require more computational resource. Optimization-based algorithms often start with an initial trajectory that links the initial configuration and the goal configuration [6], [7], [8]. Next, the algorithms iteratively improve the trajectory to satisfy the constraints (e.g., collision-free conditions) and minimize the cost. Note that the initial trajectory can be either feasible or infeasible depending on the requirements of the algorithms.

All authors are with the Department of Mechanical Engineering, University of California, Berkeley, CA 94720 USA jess.leu24, ge_zhang, litingsun, tomizuka@berkeley.edu

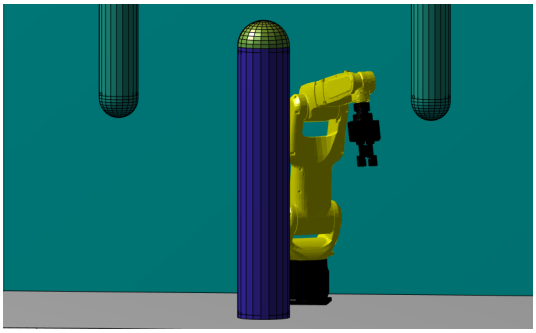


Fig. 1: A manipulator navigating through the obstacles.

The trajectories generated by optimization-based algorithms are usually smooth. However, optimization-based algorithms often fail to find a feasible solution if initial trajectories are naively chosen (e.g., a line segment from the initial to the goal in the configuration space). The reason behind is that most optimization-based algorithms rely on local gradient information, so the final trajectories and the computational time highly depend on the selection of the initial path.

In order to clearly see the effects of these pros and cons when solving motion planning problems, this paper presents a benchmark that tests motion planning algorithms from different categories. To the best knowledge of the authors, this is the first comprehensive benchmark that tests motion planning algorithms from different categories.

In addition to the benchmark, this work also presents a hybrid planning algorithm, RRT*-CFS, which combines the merits of planning-by-construction algorithms and planning-by-modification algorithms. The algorithm has two layers. We first use RRT* [13] to quickly generate a feasible and semi-optimal path. This path then serves as an initial trajectory for the optimization layer that uses the convex feasible set algorithm (CFS) [6] to quickly solve the non-convex motion planning problem. RRT*-CFS is able to find a globally-near-optimal solutions in complex environments, even in scenarios with narrow passages [14], and has good performance in terms of optimality and computational time. Our contributions are threefold as follows:

- A comprehensive benchmark that compares motion planning algorithms from different categories is presented.
- The proposed RRT*-CFS algorithm can solve planning problems that cannot be solved by many optimization-based algorithms alone, has short computational time, and has the lowest average-cost of all algorithms we compare in this work.

Manuscript 814 submitted to 2021 American Control Conference (ACC).
Received September 28, 2020.

- We implement RRT*-CFS and demonstrate its success with extensive simulation.

The remainder of the paper is organized as follows. Section 2 discusses the related works. Section 3 presents our proposed algorithm and the theoretical analysis. Section 4 presents the benchmark and the simulation results (video is publicly available at `jessicaleu24.github.io/ACC2021.html`). Finally, we conclude the work in Section 5.

II. RELATED WORKS

In this section, we first introduce our general formulation of the motion planning problem. We then present some key algorithms that we use or compare in our work.

A. Baseline Problem Formulation

In many scenarios, robot motion planning can be performed by solving an optimization problem with the following form:

$$\min_{\mathbf{x}\in\Gamma} f(\mathbf{x}), \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^n$ and Γ defines the feasible set:

$$\Gamma = \bigcap_j \Gamma_j = \bigcap_j \{\mathbf{x} : h_j(\mathbf{x}) \geq 0\}. \tag{2}$$

We assume that the constraint function $h_j(x)$ is a semi-convex function [6]. For example, $h_j(x)$ can be the distance between a robot and the jth obstacle. The cost function, $f : \mathbb{R}^n \to \mathbb{R}$, is strongly convex and smooth. Note that the motion planning problem is non-convex due to the existence of the obstacles and the non-linear robot dynamics.

B. Sampling-based Algorithms and RRT*

In general, sampling-based algorithms are fast [4]. There are many sampling-based motion planning algorithms, such as RRT*, variations of PRM [15], [16] and variations of RRT [13], [17], [18], [12], [19], [20]. These variations modify the original methods to obtain better planning performances, e.g., handling dynamic constraints, smoothing trajectories, short-cutting trajectories, etc. However, these modifications often require more computational time and resource. These computational requirements grow quickly as the robot system becomes more complex. As a result, these algorithms require longer computational time in complex systems. This weakens the computation advantage of sampling-based methods. Since our goal is to construct a good hybrid algorithm rather than a good stand-alone sampling-based algorithm, we value computational time highly and choose RRT*, which can provide a feasible and semi-optimal [13], [19] path with the shortest computational time.

C. Optimization-based Algorithms and CFS

There are also many optimization-based algorithms that can be used for motion planning, such as SQP [7], CHOMP [8], TrajOpt [21], and CFS [6]. SQP uses the Lagrangian of the original problem to formulate a transformed problem that solves for the Lagrange multipliers. The solution of these Lagrange multipliers is then used to update the decision variables of the original problem. CHOMP and TrajOpt both formulate an unconstrained problem with a cost function that penalizes the path's smoothness and proximity to the obstacles. However, the two have different approaches for collision detection. In addition, TrajOpt uses SQP to solve the problem; whereas CHOMP uses gradient descend.

Among the these algorithms, CFS is a fast optimization-based motion planning algorithm that can handle infeasible initialization under some assumptions. Here we give a brief review of the CFS algorithm. We can rewrite the non-convex optimization problem as follows:

$$\mathbf{x}^* = \arg\max_{\mathbf{x}\in\Gamma\subset\mathbb{R}^n} \quad f(\mathbf{x}). \tag{3}$$

CFS solves the non-convex problem iteratively. The following information is required:

- *Initialization*: An initial value of the state variable $\mathbf{x}^{(0)}$, which does not necessarily satisfy $\mathbf{x}^{(0)} \in \Gamma$.
- *Safety index and disjoint convex obstacles* : Similar to Eq. (2), where $h_j(\mathbf{x})$ is the safety index for the jth disjoint obstacle.
- *Convex feasible set*: The convex feasible set, $\chi^{(k)} := \chi(\mathbf{x}^{(k)}) \in \Gamma$, is constructed corresponding to previous states $\mathbf{x}^{(k)}$.

The convex feasible set in our case is as the following:

$$\begin{aligned}\chi^{(k)} &= \bigcap_{j=1}^{n} \chi_j^{(k)}, \\ &= \bigcap_{j=1}^{n} \{\mathbf{x} : h_j(\mathbf{x}^{(k)}) + \nabla^\top h_j(\mathbf{x}^{(k)})(\mathbf{x} - \mathbf{x}^{(k)}) \geq 0\}.\end{aligned} \tag{4}$$

With CFS, a convex sub-optimization problem is formulated and solved for the optimal value of $\mathbf{x}^{(k+1)}$:

$$\mathbf{x}^{(k+1)} = \arg\max_{\mathbf{x}\in\chi^{(k)}} \quad f(\mathbf{x}). \tag{5}$$

The algorithm solves the problem iteratively and results in a sequence of $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(k)}, \ldots$. It is guaranteed in [6] that this sequence will converge to a local optimal, $\mathbf{x}^*$.

Notice that SQP and TrajOpt rely on the Hessian information of the original problem, while CHOMP and CFS rely on the gradient information. As mentioned previously, optimization-based algorithms rely on local gradient information (or higher-order information); therefore the final trajectories and the computational time highly depend on the choice of the initial path.

D. Hybrid Algorithms

There have been several researches focusing on hybrid planners [1], [22], [21], [23], [24]. [1] used Lattice A* Search to generate initial trajectories for CFS and illustrated its performance on a mobile robot, a planar robot arm, and an areal robot. The sampling space of these experiments are either 2D or 3D. Since the number of points in the lattice grid grows exponentially as the dimension grows in the configuration space, the effectiveness of this work for high dimensional applications (e.g., 6-DoF manipulators) is highly doubtful.

Manuscript 814 submitted to 2021 American Control Conference (ACC).
Received September 28, 2020.

[22] adopted the Bidirectional Rapidly-exploring Random Tree (BiRRT) [17] to generate an initial feasible guess for the TrajOpt trajectory optimizer and demonstrated the success of their approach on the Atlas robot. However, only a limited number of testing scenarios were presented; the robots only needed to avoid no more than two obstacles. [23] presented a planning algorithm that combined a roadmap [25] and TrajOpt. Besides generating a collision-free and dynamically-feasible trajectory, they focused on avoiding singularities in redundant manipulators and meeting Cartesian constraints. However, the algorithm required long planning time. [24] combined a sparse roadmap with TrajOpt. However, their methods didn't consider complex environments.

A key challenge in motion planning problem is the narrow passage problem, which refers to planning problems that have a very narrow region in between the initial and the goal in the feasible configuration space. Motion planning algorithms often take too much time or even cannot find a solution when encountering a narrow passage problem even though the solution does exit [14], [26]. This type of problem is one of the target scenario in our simulation, which was not tested in [1], [22], [21], [23], [24].

III. THE PROPOSED ALGORITHM

In this section, we introduce the proposed RRT*-CFS algorithm as well as its feasibility and global convergence guarantees.

A. The RRT*-CFS Algorithm

The proposed RRT*-CFS inherits the merits and avoids the shortcomings of both algorithms. The RRT*-CFS algorithm solves the non-convex motion planning problem first by quickly finding a feasible and semi-optimal path, and then iteratively refining the solution using CFS. The RRT*-CFS has three main features.

- First, the RRT* layer can be implemented with multi-thread computation. This allows us to significantly reduce the computational time.
- Second, RRT*-CFS has stochasticity due to the random sampling process in RRT*. This helps RRT*-CFS to avoid bad local optima that optimization-based algorithms may get stuck in.
- Finally, RRT*-CFS inherits the properties of CFS so that feasibility, smoothness and convergence of the final solution are guaranteed.

Denote the configuration of a d-degree-of-freedom (d-DoF) robot as $\theta \in \mathbb{R}^d$, the initial configuration as θ_0, the goal configuration as θ_{goal}, the maximum number of samples in one RRT* thread as $n_{samples}$, and the obstacles as $\mathscr{O}$. The RRT*-CFS algorithm is summarized as in Algorithm 1.

As shown in Algorithm 1, given the inputs, $\theta_0, \theta_{goal}, n_{samples}$, and $\mathscr{O}$, each thread of the multi-thread RRT* starts to find a feasible path that connects the initial configuration and the goal configuration. If more than one thread find a path, we choose the shortest path and set it as θ^{RRT}. If no thread finds a solution, we repeat the multi-thread RRT* until we find a path. By setting up the $n_{samples}$ properly, we can find a solution in the first batch almost every time. We then generate the initial reference $\mathbf{x}^0$ for CFS using the sampled path $\theta^{\mathbf{RRT}}$. Then, the convex feasible set, $\chi^{(k)}$, is generated by linearizing the constraints at the reference point $\mathbf{x}^{(k)}$ for $k = 0, 1, \ldots$. The algorithm terminates when the change of the cost at each iteration is smaller than a threshold, i.e., $\|f(\mathbf{x}^{(k-1)}) - f(\mathbf{x}^{(k)}))\| \leq \epsilon$.

B. Theoretical Analysis

Both the feasibility and the global convergence of RRT*-CFS rely on the fact that the motion planning problem (Eq. (1) and (2)) satisfies the following assumption:

***Assumption** 1 (Problem formulation): The cost function $f(\mathbf{x})$ is strongly convex and smooth. The constraint function $h_j(\mathbf{x})$ is continuous, piece-wise smooth, and convex. The state constraint Γ is non-convex and its complement is a collection of disjoint convex sets, i.e., each of the obstacle-region is itself convex.*

Let $\mathbf{x}^r \in \mathbb{R}^n$ be a feasible reference point, i.e., $\mathbf{x}^r \in \Gamma$.

***Lemma** 1 (Feasibility): If $\mathbf{x}^r \in \Gamma$, then $\mathbf{x}^r \in \chi^r$ and $Int(\chi^r) \neq \emptyset$, where $Int(\chi^r)$ is the interior of the set χ^r.*

Proof: When $\mathbf{x}^r$ is feasible, $\mathbf{x}^r \in \chi^r_j$ for all j according to (4). Therefore, $\mathbf{x}^r \in \chi^r$. [6] proved that $\chi^{(0)}$ has nonempty interior if *Assumption 1* is satisfied. The problems studied in this work satisfy *Assumption 1*; therefore the proof holds true. ■

With *Lemma 1*, we obtain the first Theorem:

***Theorem** 1 (Feasibility of RRT*-CFS): Under Algorithm 1, the sequence $\{\mathbf{x}^{(k)}\}$ satisfies $\mathbf{x}^{(k)} \in \Gamma$ for $k = 0, 1, 2, ...$*

Proof: RRT* generates $\mathbf{x}^{(0)}$ such that $\mathbf{x}^{(0)} \in \Gamma$, i.e., feasibility holds when $k = 0$. According to *Lemma 1*, $\chi^{(0)}$ has nonempty interior, then $\mathbf{x}^{(1)} \in \Gamma$ can be attained by solving the convex optimization problem (5). By induction, we conclude that $\mathbf{x}^{(k)} \in \chi^{(k-1)} \subset \Gamma$ for $k = 1, 2, 3....$ ■

The following shows the convergence of RRT*-CFS. Given *Theorem 1*, the remainder of the proof is similar to that in [6], Theorem 4.1.

***Theorem** 2 (Global convergence of RRT*-CFS): Under Algorithm 1, the sequence $\{\mathbf{x}^{(k)}\}$ will always converge to some $\mathbf{x}^* \in \Gamma$. $\mathbf{x}^*$ is a strong local optimum of* (1) *if the limit is reached. $\mathbf{x}^*$ is at least a weak local optimum of* (1) *if the limit is not reached.*

Proof: By *Theorem 1*, we have $\mathbf{x}^{(k)} \in \Gamma$ for $k = 0, 1, 2, ...$ When iteratively solving the convex optimization (5), we have $f(\mathbf{x}^{(1)}) \geq f(\mathbf{x}^{(2)}) \geq ....$ This leads to two cases. The first one is that $f(\mathbf{x}^{(K)}) = f(\mathbf{x}^{(K+1)})$ for some K. The condition $f(\mathbf{x}^{(K)}) = f(\mathbf{x}^{(K+1)})$ is equivalent to $\mathbf{x}^{(K)} = \mathbf{x}^{(K+1)}$ according to strong descent lemma proved in [6]. By induction, the algorithm converges in the sense that $\mathbf{x}^{(K)} = \mathbf{x}^{(K+1)}$ and $\mathbf{x}^{(k)} = \mathbf{x}^{(k+1)}$ for all $k > K$. Moreover, as $\mathbf{x}^* := \mathbf{x}^{(K)}$ is a fixed point, it is a strong local optimum as shown in [6]. In the second case, the cost keeps decreasing strictly, i.e., $f(\mathbf{x}^{(1)}) > f(\mathbf{x}^{(2)}) > ...$, the sequence $\{\mathbf{x}^{(k)}\}$ converges to a weak local optimum $\mathbf{x}^*$ given the convergence of strictly descending sequence, which was proved in [6]. ■

Manuscript 814 submitted to 2021 American Control Conference (ACC).
Received September 28, 2020.

Algorithm 1 RRT*-CFS

procedure RRT*-CFS($\theta_0, \theta_{goal}, n_{samples}, \mathscr{O}$)
  while ! $\exists$ θ^{RRT} **do**
    $\theta^{RRT} \leftarrow$ Multi_thread_RRT*($\theta_0, \theta_{goal}, n_{samples}, \mathscr{O}$) ▷ Get the shortest RRT* path among all threads.
  $\mathbf{x}^0 \leftarrow$ generate_reference(θ^{RRT}) ▷ Generat the initial reference for CFS.
  while Stop criterion is not satisfied **do** ▷ $k = 0, 1, 2, \ldots$
    $\chi^{(k)} = \bigcap_{j=1}^{n} \{\mathbf{x} : h_j(\mathbf{x}^{(k)}) + \nabla^\top h_j(\mathbf{x}^{(k)})(\mathbf{x} - \mathbf{x}^{(k)}) \geq 0\}$
    $\mathbf{x}^{(k+1)} = \underset{x \in \chi^{(k)}}{\arg\min}\ f(x).$
  return $\mathbf{x}^{(k+1)}$

Fig. 2: A mobile robot (left) and a manipulator (right).

IV. SIMULATION SETUP AND RESULTS

A. Robot Models

We used two different robot platforms (Fig. 2) to test the RRT*-CFS and other algorithms for comparison.

1) Mobile robot: We use a simple kinematic model to model a mobile robot on a $2D$-plan. Denote the states of the mobile robot at time step t as $z_t = [x_t, y_t]^\top$, the input velocity as $u_t = [v_{x,t}, v_{y,t}]^\top$, and the robot configuration as $\theta = [x, y]^\top$. The linear kinematic model, is

$$\begin{bmatrix} x_{t+1} \\ y_{t+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_t \\ y_t \end{bmatrix} + \begin{bmatrix} T_s & 0 \\ 0 & T_s \end{bmatrix} \begin{bmatrix} v_{x,t} \\ v_{y,t} \end{bmatrix}, \tag{6}$$

where T_s is the sampling time.

2) Manipulator: We use a 5-degree-of-freedom manipulator. Denote the states of the manipulator as $z_t = [\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \omega_1, \omega_2, \omega_3, \omega_4, \omega_5]_t^\top$, where θ_i and ω_i, $i \in \{1, 2, 3, 4, 5\}$ are the angle position and the angular velocity of ith joint, respectively. The inputs are the angular acceleration at each joint, denoted as $u_t = [\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5]_t^\top$. The robot configuration is $\theta = [\theta_1, \theta_2, \theta_3, \theta_4, \theta_5]^\top$. The linear kinematic model is:

$$z_{t+1} = \mathbf{A} z_t + \mathbf{B} u_t, \tag{7}$$

where,

$$\mathbf{A} = \begin{bmatrix} \mathbf{I}_{5\times5} & T_s\mathbf{I}_{5\times5} \\ \mathbf{0}_{5\times5} & \mathbf{I}_{5\times5} \end{bmatrix},$$

and,

$$\mathbf{B} = \begin{bmatrix} 0.5T_s^2\mathbf{I}_{5\times5} \\ T_s\mathbf{I}_{5\times5} \end{bmatrix}.$$

B. The Motion Planning Problem

In this paper, the goal of the motion planning problem is to plan the command that brings the robot to the goal configuration while avoiding obstacles. We first solve for a path using the multi-thread-RRT* with the configuration, θ, defined previously. After getting the path θ^{RRT}, an optimization problem can be formulated. The decision variable for each time step is u_t, and the input vector that the problem optimizes is denoted as $\mathbf{u} := [u_0^\top, u_1^\top, \cdots, u_H^\top]^\top$, where H is the planning horizon. Similarly, the resulting state vector is $\mathbf{z} := [z_1^\top, z_2^\top, \cdots, z_{H+1}^\top]^\top$. Given the initial state, z_0, we obtain $\mathbf{z} = f_{ki}(z_0, \mathbf{u})$ by concatenating the kinematic function (Eq. (6) or (7)) throughout the planning horizon. For simplicity, denote the kinematic function as $f_{ki,z_0}(\mathbf{u}) := f_{ki}(z_0, \mathbf{u})$. In order to obtain the optimal solution $\mathbf{u}^*$ given the constrained feasible set Γ and the input constraint $\mathbf{u}_{max}$, the following optimization problem needs to be solved:

$$\begin{aligned} \mathbf{u}^* = \arg\min_{\mathbf{u}} \quad & f_{z_0}(\mathbf{u}), \\ s.t. \quad & f_{ki,z_0}(\mathbf{u}) \in \Gamma, \\ & -\mathbf{u}_{max} \leq \mathbf{u} \leq \mathbf{u}_{max}. \end{aligned} \tag{8}$$

The cost function is quadratic that has the form: $f_{z_0}(\mathbf{u}) = \|f_{ki,z_0}(\mathbf{u}) - \mathbf{z}_{goal}\|_2^2 + \lambda\|\mathbf{u}\|_2^2$, which is convex and regular. The first term penalizes the deviation from the goal and the second term penalizes the input.

C. Implementation

CFS updates $\mathbf{z}^{(k)} = f_{ki,z_0}(\mathbf{u}^{(k-1)})$ at iteration $k = 2, 3 \ldots$. Notice that $\mathbf{z}^{(1)}$ is determined by θ^{RRT} and $\mathbf{u}^{(0)}$ is initialized as a zero vector. The convex feasible set, $\chi^{(k)}$, is determined by $\mathbf{z}^{(k)}$. Given the feasible set $\Gamma = \bigcap_{j,t}\{\mathbf{z} : h_j(z_t) \geq 0\}$, where j numerates over obstacles, t numerates over time steps, the results of the previous iteration ($\mathbf{u}^{(k-1)}$ and $\mathbf{z}^{(k)}$), and the function $f_{ki,z_0}(\mathbf{u})$, we can construct the convex feasible set as the following:

$$\chi_{z_0}^{(k)} = \bigcap_{j,t}\{\mathbf{u} : h'_{j,t}(\mathbf{u}, z_t^{(k)}, \mathbf{u}^{(k-1)}) \geq 0\}, \tag{9}$$

where $h'_{j,t} = h_j(z_t^{(k)}) + \nabla^\top h_j(z_t^{(k)}) \nabla f_{ki,z_0,t}(\mathbf{u}^{(k-1)})(\mathbf{u} - \mathbf{u}^{(k-1)})$. Therefore, the iterative sub-problem is as the fol-

lowing:

$$\begin{aligned} \mathbf{u}^{*(k)} = \arg\min_{\mathbf{u}} \quad & f_{z_0}(\mathbf{u}), \\ s.t. \quad & \mathbf{u} \in \chi_{z_0}^{(k)}(\mathbf{u}^{(k-1)}, \mathbf{z}^{(k)}), \\ & -\mathbf{u}_{max} \leq \mathbf{u} \leq \mathbf{u}_{max}. \end{aligned} \tag{10}$$

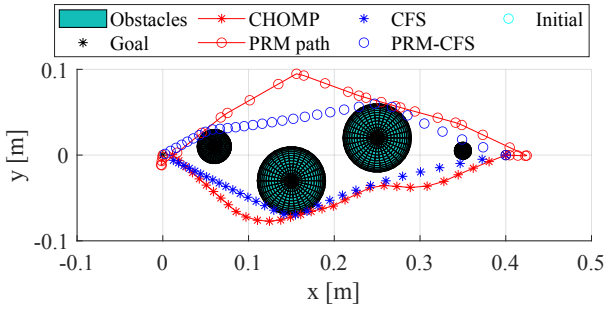
D. Simulation Setup

We show the motion planning simulation results in the following sections. The simulation is conducted in `Matlab R2020a` on a desktop with 3.2GHz Intel Core i7-8700 CPU. The stopping criteria for optimization-based methods (i.e., CFS, SQP, CHOMP, and the CFS layer of RRT*-CFS) are the same, which are (1) the algorithm reaches the maximum number of iterations and (2) the change of the cost is smaller than the threshold. The algorithms terminate if one of the criteria is met.

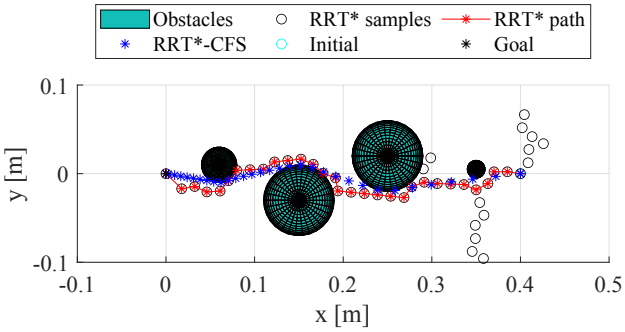
E. 2D Motion Planning Benchmark

Multiple Objects. One examplar simulation environment is shown in Fig. 3, where Fig. 3a shows the planning result of CHOMP (red-star-line), PRM-CFS (blue-circle), and CFS (blue-star-line) and Fig. 3b shows the planning result of RRT*-CFS (blue-line). By comparing the two figures, we see that all four trajectories are smooth and can successfully bring the robot to the goal (at $(0, 0)$). More importantly, RRT*-CFS converges to the global optimum, while others stuck in local optima. TABLE I shows the result averaging over 100 trials (obstacles are randomly located in each trials). Benefiting from the first layer, the proposed RRT*-CFS has a success rate of 100% and also requires less iterations to converge comparing to CFS, SQP, CHOMP, and PRM-CFS. RRT*-CFS also has the lowest average cost, which is contributed by the semi-optimal initial path given by RRT* and the optimization process by CFS. This also indicates that RRT*-CFS is more likely to converge to nearly-global optima. Comparing RRT*-CFS and RRT*-SQP, we see that RRT*-CFS requires less computational time. This is due to the face that CFS exploits the geometry of the motion planning problem. Also, RRT*-CFS is faster than PRM-CFS due to the face that RRT* is faster than PRM in these motion planning problems. Even though CFS is the fastest algorithm besides RRT* in both 4-obstacle and 1-obstacle cases, RRT*-CFS is faster than CFS in the 10-obstacle complex scenarios due to a better initialization and faster convergence rate. Also notice that CFS is more vulnerable to be trapped in bad local optima and cannot find a path to reach the goal in complex scenarios; whereas RRT*-CFS can always find a path (the failure cases indicate the resulting solutions do not bring the robot to the goal).

Narrow Passages. We also test RRT*-CFS in the narrow passage scenario. The goal of the robot is again to plan a path to the goal point. However the robot has to navigate through a narrow pathway in order to reach the goal point. Fig. 4 shows a planning result in one of these scenarios. RRT*-CFS successfully explores the narrow passage and plans a feasible trajectory (blue-circle-line). On the other hand, CFS, SQP, and CHOMP failed to find a solution. The CFS



(a) Planning result using CFS, CHOMP, and PRM-CFS.



(b) Planning result using RRT*-CFS.

Fig. 3: Simulation results of a 2D motion planning.

trajectory (blue-star-line) and the SQP trajectory (gray-star-line, under the CFS trajectory) stop in front of the wall while the CHOMP trajectory (red-star-line) directly penetrates the wall. These two failures are due to the lack of local gradient information, which is crucial for optimization-based methods that solve each iteration by calculating the gradient using the result of the previous iteration. The 99% success rate in TABLE I for CFS and CHOMP is due to the same reason, where the initial point and the goal point are both lying on the symmetric axis of the obstacle. Even though using local higher-order information can enable the algorithms to deal with some of these scenarios (e.g., SQP), solving the narrow passage problem is still hard for optimization-based methods alone; because sometimes there is no information contained in the higher-order terms in these scenarios (e.g., when facing the wall, the second order derivative of a line is zero). This observation again demonstrates the need of a sampling mechanism in the motion planning algorithm so that the algorithm can explore beyond the local gradient information. The simulation result of narrow passage environments is summarized in TABLE II. We see that RRT*-CFS improves the cost without sacrificing too much computational time. Although PRM-CFS also finds a solution and requires fewer iterations, its computational time is two magnitudes larger than RRT*-CFS.

F. 5D Motion Planning for a Manipulator

In 5D motion planning simulations, the goal of the manipulator is to reach the goal configuration from the initial configuration while avoiding obstacles. One of the simulation

TABLE I: Simulation comparison of 2D planning. (Average of 100 trials.)

Algorithm	# obstacles	Computational time [s]	# iterations	Cost	Success rate (%)
RRT*	1	**0.006**	N/A	1.266	**100**
PRM	1	1.1859	N/A	1.3547	**100**
CFS	1	0.038	7.40	1.007	99
SQP	1	1.73	7.40	1.006	**100**
CHOMP	1	0.053	137.2	1.100	99
PRM-CFS	1	1.261	7.22	1.009	**100**
RRT*-SQP	1	0.994	**7.01**	**1.001**	**100**
RRT*-CFS	1	0.038	**7.01**	**1.001**	**100**
RRT*	4	**0.03**	N/A	1.272	**100**
PRM	4	4.24	N/A	1.379	**100**
CFS	4	0.12	12.63	1.048	99
SQP	4	4.29	12.80	1.031	**100**
CHOMP	4	0.23	199.42	1.110	99
PRM-SQP	4	4.35	11.04	0.102	**100**
RRT*-SQP	4	2.63	**10.58**	**1.011**	**100**
RRT*-CFS	4	0.15	**10.58**	**1.011**	**100**
RRT*	10	**0.042**	N/A	1.266	**100**
CFS	10	0.695	14.46	1.107	87
RRT*-CFS	10	0.510	**12.69**	**1.061**	**100**

TABLE II: Simulation result comparison of 2D-narrow-passage planning. (Average of 20 trials.)

Algorithm	Computational time [s]	# iterations	Cost
RRT*	**0.27**	N/A	1.22
PRM	20.05	N/A	1.22
PRM-CFS	20.64	**6.6**	1.02
RRT*-CFS	0.43	8.26	**1.01**

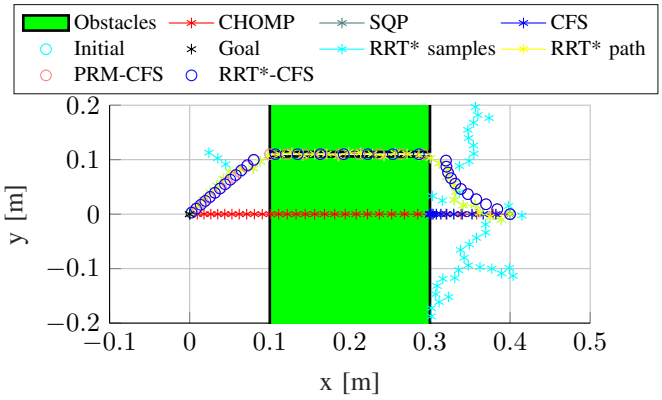


Fig. 4: Simulation results of motion planning in the narrow passage (close to $y = 0.1$) scenario.

environments is shown in Fig. 5a, where the manipulator moves to the goal configuration without colliding with the two obstacles. TABLE III shows the result averaging over 20 trials. Similar to the trend in 2D cases, RRT*-CFS has lower average cost comparing to CFS. In terms of computational time, CFS performs better in scenarios with one or two obstacles, while RRT*-CFS shows its strength in complex environment and performs better in scenarios with three or four obstacles. Benefiting from the first layer, the proposed RRT*-CFS has a success rate higher than that of CFS. RRT*-CFS also requires less iterations to converge comparing to CFS. Fig. 5b shows a RRT*-CFS planning result in a scenario with three obstacles. Notice that CFS fails to plan a trajectory that brings the manipulator to the goal configuration in this scenario. With the initial path from RRT*, RRT*-CFS plans a smoother trajectory that navigates through the complex environment comparing to the original RRT* trajectory.

V. CONCLUSION

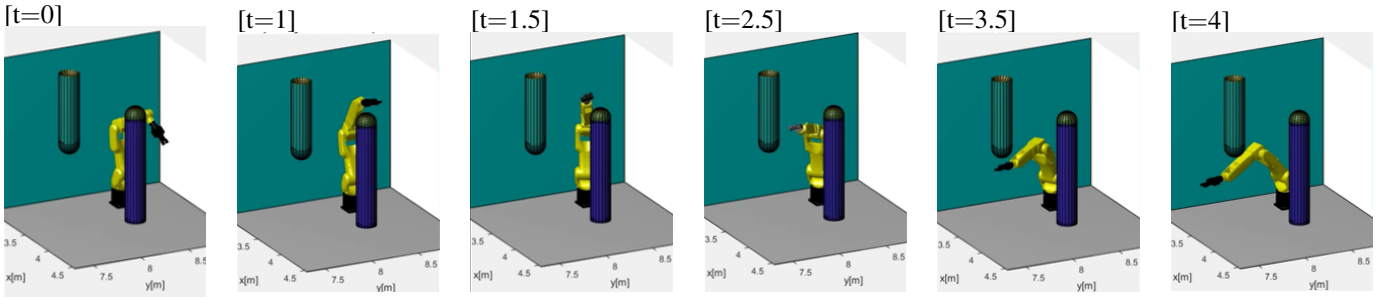
This paper presented a comprehensive benchmark that compares motion planning algorithms from different categories and introduced a fast motion planning algorithm, RRT*-CFS, that combined the merits of sampling-based planning methods and optimization-based planning methods. The RRT*-CFS quickly found a feasible and semi-optimal path using RRT* and iteritavly refined the solution using CFS. RRT*-CFS had feasibility and global convergence guarantees inherited from CFS in scenarios where obstacles could be represented by disjoint convex objects. Simulation results showed that RRT*-CFS benefited from the hybrid structure. Comparing to RRT*, PRM, CFS, SQP, CHOMP, and PRM-CFS, RRT*-CFS had the lowest cost and converged with less number of iterations. RRT*-CFS could also solve planning problems in complex scenarios such as the narrow passage problem, in which CHOMP, SQP, and CFS failed. Even though RRT*-CFS has two layers, the computational time is still competitive in simple scenarios and out performs other algorithms (except RRT* in terms of time) in complex scenarios. We concluded that the hybrid structure indeed brought strong performance. The future work is to improve both the sampling-based layer and the optimization-based layer as well as the connection between them.

ACKNOWLEDGEMENT

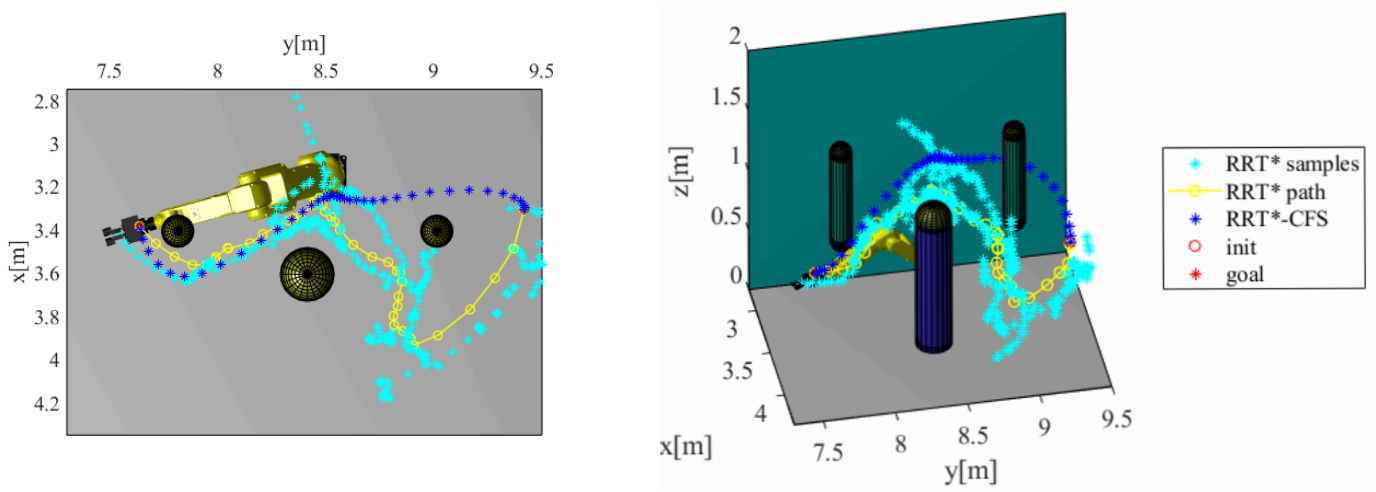
The authors thank Changliu Liu for helpful discussions. This work was supported by the National Science Foundation under Grant No.1734109. Any opinion, finding, and conclusion expressed in this paper are those of the authors and do not necessarily reflect those of the National Science Foundation.

TABLE III: Simulation comparison of 5D Motion planning. (Average of 20 trials.)

Algorithm	# obstacles	Computational time [s]	# iterations	Cost	Success rate (%)
RRT*	1 or 2	5.7	N/A	N/A	**100**
CFS	1 or 2	**4.74**	20	1.94	95
RRT*-CFS	1 or 2	12.42	**15.56**	**1.66**	99
RRT*	3 or 4	**8.45**	N/A	N/A	**100**
CFS	3 or 4	114.10	20	3.78	85
RRT*-CFS	3 or 4	59.675	**18.58**	**3.39**	89



(a) Motion planning for manipulator using RRT*-CFS.



(b) Motion planning result for manipulator using RRT*-CFS.

Fig. 5: Planning results using RRT*-CFS in 5D manipulation planning problems.

REFERENCES

[1] C. Liu, C. Lin, Y. Wang, and M. Tomizuka, "Convex feasible set algorithm for constrained trajectory smoothing," in *2017 American Control Conference (ACC)*, May 2017, pp. 4177–4182.

[2] 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.

[3] A. Stentz, "Optimal and efficient path planning for partially-known environments," in *IN PROCEEDINGS OF THE IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION*, 1994, pp. 3310–3317.

[4] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," Tech. Rep., 1998.

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

[6] C. Liu, C.-Y. Lin, and M. Tomizuka, "The convex feasible set algorithm for real time optimization in motion planning," *SIAM Journal on Control and Optimization*, vol. 56, no. 4, pp. 2712–2733, 2018.

[7] P. Spellucci, "A new technique for inconsistent qp problems in the sqp method," *Mathematical Methods of Operations Research*, vol. 47, pp. 355–400, 1998.

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

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

[10] C. Park, J. Pan, and D. Manocha, "Itomp: Incremental trajectory optimization for real-time replanning in dynamic environments," in *ICAPS*, 2012.

[11] J. Van Den Berg and M. Overmars, "Kinodynamic motion planning on roadmaps in dynamic environments," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2007, pp. 4253–4258.

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

[13] 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.

[14] S. Rodríguez, X. Tang, J. Lien, and N. Amato, "An obstacle-based rapidly-exploring random tree," *Proceedings 2006 IEEE International*

Conference on Robotics and Automation, 2006. ICRA 2006., pp. 895–900, 2006.

[15] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones, and D. Vallejo, "Obprm: An obstacle-based prm for 3d workspaces," in *Robotics: The Algorithmic Perspective: 1998 Workshop on the Algorithmic Foundations of Robotics*, 1998, pp. 155–168.

[16] 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. IEEE, 2000, pp. 521–528.

[17] 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, 2000, pp. 995–1001 vol.2.

[18] 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 *2012 IEEE International Conference on Robotics and Automation*, 2012, pp. 2537–2542.

[19] F. Islam, J. Nasir, U. Malik, Y. Ayaz, and O. Hasan, "Rrt-smart: Rapid convergence implementation of rrt towards optimal solution," in *2012 IEEE International Conference on Mechatronics and Automation*. IEEE, 2012, pp. 1651–1656.

[20] M. Otte and E. Frazzoli, "Rrtx: Asymptotically optimal single-query sampling-based motion planning with quick replanning," *The International Journal of Robotics Research*, vol. 35, no. 7, pp. 797–822, 2016.

[21] 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," *The International Journal of Robotics Research*, vol. 33, pp. 1251–1270, 08 2014.

[22] L. Li, X. Long, and M. A. Gennert, "Birrtopt: A combined sampling and optimizing motion planner for humanoid robots," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, 2016, pp. 469–476.

[23] C. Park, F. Rabe, S. Sharma, C. Scheurer, U. E. Zimmermann, and D. Manocha, "Parallel cartesian planning in dynamic environments using constrained trajectory planning," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, 2015, pp. 983–990.

[24] S. Dai, M. Orton, S. Schaffert, A. Hofmann, and B. Williams, "Improving trajectory optimization using a roadmap framework," 10 2018, pp. 8674–8681.

[25] L. Jaillet and T. Simeon, "Path deformation roadmaps: Compact graphs with useful cycles for motion planning," *The International Journal of Robotics Research*, vol. 27, no. 11-12, pp. 1175–1188, 2008. [Online]. Available: https://doi.org/10.1177/0278364908098411

[26] Liangjun Zhang and D. Manocha, "An efficient retraction-based rrt planner," in *2008 IEEE International Conference on Robotics and Automation*, 2008, pp. 3743–3750.