2021 International Conference on Unmanned Aircraft Systems (ICUAS)
Athens, Greece. June 15-18, 2021

MK-RRT*: Multi-Robot Kinodynamic RRT* Trajectory Planning

Brennan Cain[1], Michail Kalaitzakis[2], and Nikolaos Vitzilaios[2]

2021 International Conference on Unmanned Aircraft Systems (ICUAS) | 978-1-6654-1535-4/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICUAS51884.2021.9476688

Abstract— **This paper introduces MK-RRT*: a Multi-robot Kinodynamic RRT*-based framework for trajectory planning of multiple dynamically-modeled robots. The framework includes both tightly-coupled and loosely-coupled methods for planning. The simultaneous, tightly-coupled, method provides an asymptotically optimal solution to the multi-robot trajectory planning problem. A sequential, loosely-coupled, method is also developed to compare costs of the generated trajectories and computational complexity of both methods. An application of this framework to multi-UAV trajectory planning is presented and experimentally evaluated using a team of Ryze Tello EDU UAVs.**

I. Introduction

Research on path planning, the problem of finding a path from a starting state to a final state, has seen much progress over the past decades. Rapidly-exploring Random Trees (RRTs) [1] and Probabilistic Roadmaps (PRMs) [2] provided early solutions to path planning problems in the high-dimensional space by using sampling to provide fast solutions to the PSPACE-hard problem [3]. These methods gave way to PRM* and RRT* [4] which provide asymptotic optimality guarantees, assuring that the probability of finding the optimal solution approaches 1 as the number of samples approach infinity. These methods, however, are limited to systems with limited dynamic constraints, as they assume that any two states can be connected using straight lines. The Kinodynamic RRT* algorithm (KRRT*) [5] was created to support systems with differential constraints by extending and rewiring the $Tree$ nodes with optimal kinodynamic connections. Since the trajectories that KRRT* produces are time-dependent, they may be referred to as trajectories.

KRRT* is not alone among dynamically constrained RRT*s. Poli-RRT* [6] has been proposed as an extension of KRRT* to include differentially-flat robots with constrained dynamics. The main difference between these two methods is that Poli-RRT* is valid for any constrained, linearisable system. Poli-RRT* is evaluated through simulation of a unicycle with actuation constraints [6].

Path planning grows more difficult with the introduction of additional robots. Graph based methods [7] have been introduced to solve this problem with exponential complexity and without kinematic constraints. A near-optimal approach for robots with kinematic constraints has also been proposed in [8], although the solution remains PSPACE-hard.

Two general approaches to the problem of path planning with multiple robots have been proposed [9]. Centralized methods attempt to find a globally optimal solution by allowing for simultaneous planning in the workspace defined as the Cartesian product of the workspaces of the individual robots [10]. On the other hand, in distributed or decoupled approaches, each robot's path is initially independently calculated, and then the interactions between the robots are considered [11], [12].

Multi-agent RRT* [13] provided the earliest adaptation of RRT* to work with multiple agents, where paths for multiple autonomous aircraft, modeled on a four-connected graph, are generated. No dynamic constraints are given and all planning is constrained to a discrete grid. Moreover, [14] extended Poli-RRT* to enable kinodynamic motion planning with multiple agents. The proposed solution was to sequentially solve each robot's individual trajectory one at a time, while avoiding the previous robots' trajectories. Simulations with unicycle-like robots were used to validate the proposed solution.

In this work, we extend the Kinodynamic RRT* approach and introduce the Multi-robot Kinodynamic RRT* (MK-RRT*) trajectory planner for linearised robotic models, both tightly-coupled\simultaneous and loosely-coupled\sequential. Unlike previous works that have dealt with simultaneous path planning in graphs, this method couples the controllers and removes the need for additional operations to negotiate traffic while also considering the kinodynamic constraints of the robots.

Furthermore, we present an application of this methodology to the trajectory planning problem for multiple Unmanned Aerial Vehicles (UAVs). The developed planners are tested in real world experiments to demonstrate the validity of the generated trajectories and offer a comparison between the planned trajectory costs with the actual cost to follow these trajectories. Finally, we analyze the key differences between the generated trajectories and explain the advantages of each method. In summary, the main contributions of this paper are:

1) a centralized, globally asymptotically optimal solution to the linear, dynamically-constrained multi-robot trajectory planning problem,
2) a comparison of this simultaneous optimal method to a sequential, faster method for trajectory generation for small UAVs, and
3) experimental results demonstrating the validity of the trajectories generated by these trajectory planners.

*This work was partially supported by the University of South Carolina (UofSC) Office of Undergraduate Research [Magellan Award 80003125].

[1]Brennan Cain is with the Department of Computer Science and Engineering, University of South Carolina, Columbia, SC 29208, USA bscain@email.sc.edu

[2]Michail Kalaitzakis and Nikolaos Vitzilaios are with the Department of Mechanical Engineering, University of South Carolina, Columbia, SC 29208, USA michailk@email.sc.edu, vitzilaios@sc.edu

The rest of this paper is organized as follows: Section II defines the trajectory planning problem in the (a) single robot, (b) multi-robot sequential, and (c) multi-robot simultaneous cases. Section III discusses the original KRRT* algorithm and proposes extensions to allow for simultaneous and sequential solutions for multiple agents. Section IV defines our experimental setup and outlines the scenarios in which we test our trajectory planners. Section V provides experimental results and gives a brief discussion on the data collected during the trajectory generations and flights. Section VI discusses our results in detail and provides interesting insights into new problems which arise from our experiments on physical robots. Finally, Section VII concludes this work and discusses areas of interest which may be explored in the future.

II. Problem Definitions

In this section, we define the trajectory generation problem for three separate cases: single robot, multi-robot sequential, and multi-robot simultaneous planning.

A. Single Robot

First, we define the problem in the case of a single robot. We assume that a robot can be represented as a linear, time-invariant system. The state and input spaces are defined as $\mathcal{X} = \mathbb{R}^n$ and $\mathcal{U} = \mathbb{R}^m$. Therefore, the dynamics of the system are defined by:

$$\dot{\mathbf{x}}[t] = A\mathbf{x}[t] + B\mathbf{u}[t] \tag{1}$$

where $\mathbf{x}[t] \in \mathcal{X}$ is the state at time t, $\mathbf{u}[t] \in \mathcal{U}$ is the input at time t, $A \in \mathbb{R}^{n\times n}$ the state matrix, and $B \in \mathbb{R}^{n\times m}$ the input matrix. The system is assumed to be both controllable and observable.

Let $\mathcal{X}_{free} \subseteq \mathcal{X}$ define the free state space of the robot, $\mathcal{X}_{dyn}[t] \subseteq \mathcal{X}$ define the state space of the robot which is occupied by dynamic obstacles at time t, and $\mathcal{U}_{free} \subseteq \mathcal{U}$ be the free input state of the robot. That is the sets of states and inputs which are collision free and possible to achieve.

We define trajectories as the ordered set $\pi = (\mathbf{x}[], \mathbf{u}[], \tau)$, where $\mathbf{x}[t] \in \{\mathcal{X}_{free} \cap \mathcal{X}^C_{dyn}[t]\}$ for $t \in [0, \tau]$ is the state along the trajectory, $\mathbf{u}[t] \in \mathcal{U}_{free}$ for $t \in [0, \tau]$ is the input along the trajectory, and finally τ is the duration of the trajectory.

The optimal trajectory planning problem may now be defined: given a starting state $\mathbf{x}_{start}$ and goal state $\mathbf{x}_{goal}$ find the trajectory in the time-determined free space $\{\mathcal{X}_{free} \cap \mathcal{X}^C_{dyn}[t]\}$ with inputs in $\mathcal{U}_{free}$ which minimizes the cost function:

$$\begin{aligned} \pi^*_{free} = &\min_{\pi} \quad \int_0^{\tau} (\alpha + u[t]^T R u[t]) dt \\ &\text{s.t.} \quad \mathbf{x}[0] = \mathbf{x}_{start}, \\ &\qquad \mathbf{x}[\tau] = \mathbf{x}_{goal}, \\ &\qquad \mathbf{x}[t] \in \{\mathcal{X}_{free} \cap \mathcal{X}^C_{dyn}[t]\}, \forall t \in [0, \tau], \\ &\qquad \mathbf{u}[t] \in \mathcal{U}_{free}, \forall t \in [0, \tau], \\ &\qquad \dot{\mathbf{x}}[t] = A\mathbf{x}[t] + B\mathbf{u}[t]. \end{aligned} \tag{2}$$

where $\alpha \in \mathbb{R}^+$ is a scalar to weigh the cost of the time of the trajectory, $R \in \mathbb{R}^{m\times m}$ is a positive-definite, constant matrix to weigh the input costs and $\mathbf{x}[0]$ is the initial state of the linear system.

The cost function in Eq. 2 obeys the optimal substructure property. An optimal trajectory from $\mathbf{x}_{start}$ to $\mathbf{x}_{goal}$ is a concatenation of successive optimal sub-trajectories.

B. Multi-robot Loosely-Coupled, Sequential System

The sequential trajectory planning algorithm builds on the single robot problem and is defined as a hierarchical trajectory planning problem. First, a trajectory for the first robot in the queue is found subject to the cost minimization function in Eq. 2. Next, the space of dynamic obstacles $\mathcal{X}_{dyn}[]$ is updated using the generated robot's trajectory. By including each previous robot's trajectory in $\mathcal{X}_{dyn}[]$, the minimization function remains the same for each subsequent robot. Therefore, the problem is defined as follows: for each robot, minimize Eq. 2, where $\mathcal{X}_{dyn}[]$ includes the trajectories of all previous robots.

C. Multi-robot Tightly-Coupled, Simultaneous System

Using the problem statement from section II-A, we may now specify a new problem statement for a multi-robot system. First, let the free state space be the Cartesian product of the individual workspaces. Next, we redefine the free input space of the multi-robot problem using the Cartesian product of the free input spaces of each robot. We define the state of the multi-robot system as the vertical concatenation of the states of the individual robots:

$$\mathbf{x}[t] = \begin{bmatrix} \mathbf{x}_1[t]^T & \mathbf{x}_2[t]^T & \dots & \mathbf{x}_N[t]^T \end{bmatrix}^T \tag{3}$$

The inputs of the individual robots may be concatenated vertically as well:

$$\mathbf{u}[t] = \begin{bmatrix} \mathbf{u}_1[t]^T & \mathbf{u}_2[t]^T & \dots & \mathbf{u}_N[t]^T \end{bmatrix}^T \tag{4}$$

The state transition matrix A can be defined as the block diagonal matrix of the individual robots' state transition matrices:

$$A = \text{diag}(\begin{bmatrix} A_1 & A_2 & \dots & A_N \end{bmatrix}) \tag{5}$$

The input matrix may also be defined as the block diagonal matrix of the individual robot's input matrices:

$$B = \text{diag}(\begin{bmatrix} B_1 & B_2 & \dots & B_N \end{bmatrix}) \tag{6}$$

Finally, the cost matrix R must also be defined as the block diagonal matrix of the individual robot's input cost matrices:

$$R = \text{diag}(\begin{bmatrix} R_1 & R_2 & \dots & R_N \end{bmatrix}) \tag{7}$$

This combined model may now be used in place of the single robot model in the definition of the trajectory planning problem.

III. MK-RRT*: MULTI-ROBOT KINODYNAMIC RRT*

Kinodynamic RRT* was proposed to extend RRT* to systems which exhibit dynamic constraints, as shown in [5]. RRT* is unable to plan trajectories for kinodynamic systems which may not be able to be driven in a point-to-point fashion. The major innovation of Kinodynamic RRT* is the use of the weighted controllability Gramian for finding the optimal control and arrival time of each driving step.

A. Optimal Control

The derivation of an optimal control solution from an initial state to a final state is provided in [5]. First, a fixed final time is used to find an optimal solution to the fixed final state and final time problem. The weighted controllability Gramian is used to determine the inputs required to drive an initial state to a final state in fixed time using the general solution to the state-space differential equation with no inputs. Next, an optimal arrival time is found by minimizing the arrival time. Finally, a closed form solution is derived to find the optimal arrival time and inputs for a given start and end state.

This solution is unique if and only if the A matrix is nilpotent. In the original work, a constant bias vector was also included, however, our model identification did not show any bias. The bias may easily be included by solving the original differential equation including the bias vector, as shown in [5].

B. Single Robot Algorithm

The single robot Algorithm 1 is the base of the multi-robot algorithms and closely follows that of the original paper. At the beginning of the algorithm, there is a check if the optimal connection from the initial to final state is collision free, and if so, immediately connects them and returns.

The CollisionFree method, shown in Algorithm 2, checks the states in the path at regular intervals. At each step, the state is compared against given obstacles to determine whether the robot is intersecting the obstacle. Robots are modeled as spheres of radius r for collision checking.

C. Simultaneous MK-RRT* Algorithm

The multi-robot simultaneous variant of Kinodynamic RRT* is similar to the single robot variant, with the difference being that the controllers of the individual robots are coupled as shown in section II.B. This allows states to be sampled and controlled simultaneously.

The CollisionFree method is adapted to compare the state of robots both against obstacles, as well as against each others' states, as shown in Algorithm 3. Because this algorithm matches the original RRT* algorithm, the computational complexity of the two methods are of the same order. As shown in [4], this complexity was found to be $O(n\log n)$.

D. Sequential MK-RRT* Algorithm

The multi-robot sequential variant of Kinodynamic RRT* finds the paths of each robot in a sequential fashion, as shown in Algorithm 4. This leads to an inherent hierarchy, where the first robot has the highest priority and the ability to find a completely optimal path, given infinite iterations. Once the path of the first robot is found, the states are saved as a dynamic obstacle where time maps to the position of the obstacle.

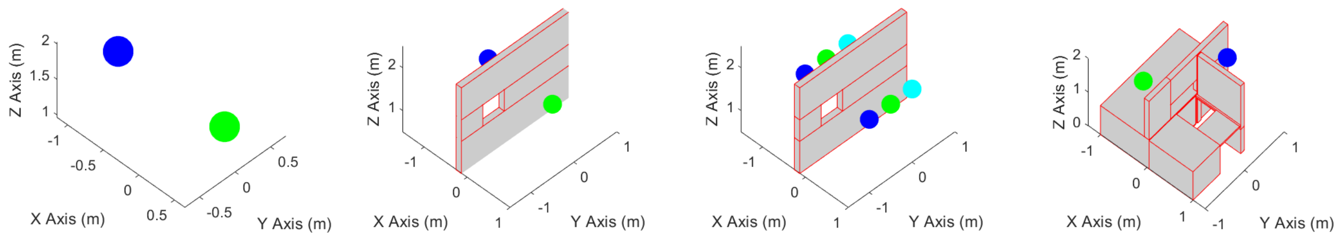
The CollisionFree function is modified to take these dynamic obstacles and, at each time step, find the position of all previous robots to check for collisions. In the case of dynamic obstacles, the safety margin is doubled to allow both robots adequate space, as each one needs one radius to move safely.

Algorithm 1: Kinodynamic RRT* Algorithm

```
Given: $\mathbf{x}_{start} \in \{\mathcal{X}_{free} \cap \mathcal{X}_{dyn}^C[0]\}$, $\mathbf{x}_{goal} \in \mathcal{X}_{free}$,
  and $i_{limit} \in \mathbb{Z}^+$
$\mathcal{T} \leftarrow \mathbf{x}_{start}$
if CollisionFree[$\pi^*[\mathbf{x}_{start}, \mathbf{x}_{goal}]$] then
    return [$\mathbf{x}_{start}$,$\mathbf{x}_{goal}$]
else
    for $i \in [1, i_{limit}]$ do
        Sample $\mathbf{x}_i \in \mathcal{X}_{free}$
        $\mathbf{x} \leftarrow$ argmin$\{\mathbf{x} \in \mathcal{T} | c^*[\mathbf{x}, \mathbf{x}_i] < r \wedge$
          CollisionFree$[\pi^*[\mathbf{x}, \mathbf{x}_i]]\}$(cost$[\mathbf{x}] + c^*[\mathbf{x}, \mathbf{x}_i]$)
        parent $[\mathbf{x}_i] \leftarrow \mathbf{x}$
        cost $[\mathbf{x}_i] =$ (cost$[\mathbf{x}] + c^*[\mathbf{x}, \mathbf{x}_i]$)
        for $\mathbf{x} \in \mathcal{T} \cup \{\mathbf{x}_{goal}\} | c^*[\mathbf{x}_i, \mathbf{x}] < r \wedge$
          cost$[\mathbf{x}_i] + c^*[\mathbf{x}_i, \mathbf{x}] <$cost$[\mathbf{x}]$
          CollisionFree$[\pi^*[\mathbf{x}_i, \mathbf{x}]]\}$ do
            cost$[\mathbf{x}] \leftarrow$cost$[\mathbf{x}_i] + c^*[\mathbf{x}_i, \mathbf{x}]$
            parent$[\mathbf{x}] \leftarrow \mathbf{x}_i$
        end
        $\mathcal{T} \leftarrow \mathcal{T} \cup \{\mathbf{x}_i\}$
    end
    if $\mathbf{x}_{goal} \in \mathcal{T}$ then
        Traj = [$\mathbf{x}_{goal}$]
        while parent(Path.end) $\neq \mathbf{x}_{start}$ do
            Traj.push_front(parent(Traj(end)))
        end
        return [$\mathbf{x}_{start}$,Traj]
    else
        return Ø
    end
end
```

Algorithm 2: Single Robot CollisionFree

```
Given: $\mathbf{x}[]$, $T = [t_0, t_1]$, and $\{\mathcal{X}_{free} \cap \mathcal{X}_{dyn}^C[]\}$
for $t \in T$ do
    if $x[t] \notin \{\mathcal{X}_{free} \cap \mathcal{X}_{dyn}^C[t]\}$ then
        return False
    end
end
return True
```



(a) ***Obstacle-Free Swap*** (b) ***Through Window Swap*** (c) ***Three UAVs through Window*** (d) ***Obstacle Course Swap***

Fig. 1: Scenarios used to demonstrate the Tello UAV path planning. For scenarios a, b, and d, the start position for Tello 1 is shown as blue sphere, while the start position for Tello 2 is shown as green sphere. For scenario c, the Tellos begin on the top-left side of the wall (Tello 1-blue, Tello 2-green, Tello 3-cyan) and must move to the same colors on the bottom right side.

Algorithm 3: Simultaneous CollisionFree

```
Given: $\mathbf{x}[]$, $T = [t_0, t_1]$, $\mathcal{X}_{free}$, $\mathcal{X}_{dyn}[]$, Radius,
  RobotCount
for $t \in T$ do
    if $x[t] \notin \{\mathcal{X}_{free} \cap \mathcal{X}_{dyn}^C[t]\}$ then
        return False
    end
    for $i = 1; i <$RobotCount$; i++$ do
        for $k = i+1; k <$RobotCount$; k++$ do
            if dist($\mathbf{x}_i[t], \mathbf{x}_k[t]$) $< 2 * r$ then
                return False
            end
        end
    end
end
return True
```

Algorithm 4: Sequential MK-RRT* Algorithm

```
Given: $\{\mathbf{x}_{start,1}, ..., \mathbf{x}_{start,N}\} \in \{\mathcal{X}_{free}^N \cap \mathcal{X}'^N_{dyn}[0]\}$,
  $\{\mathbf{x}_{goal,1}, ..., \mathbf{x}_{goal,N}\} \in \mathcal{X}_{free}^N$, and RobotEnum
paths = {}
for r in RobotEnum do
    path$_r$[] = KRRT*($\mathbf{x}_{start,r}, \mathbf{x}_{goal,r}, \mathcal{X}_{dyn}[]$)
    $\mathcal{X}_{dyn}[] = \{\mathcal{X}_{dyn}[] \cup$ dilate(path$_r$[])$\}$
    paths = paths $\cup$ path$_r$[]
end
return paths
```

IV. Experiments

MK-RRT* is a solution to the problem of dynamic trajectory planning in higher dimensions. To demonstrate an application of our solution to this problem, we perform a variety of experiments with multiple UAVs. Figure 1 shows the four application scenarios of increasing difficulty that are considered in the testing of the proposed framework:

I ***Obstacle-Free Swap***. In this scenario, two UAVs start at an altitude of $1.5m$ and at a distance of $1.5m$ in an obstacle-free environment, as shown in Figure 1a. The target point for each UAV is the start point of the other. The workspace is $3m \times 5m \times 3m$. This scenario supplies a baseline in an obstacle free state space.

II ***Through Window Swap***. In this scenario, two UAVs are placed $1.5m$ apart on either side of a wall, with a single, $0.5m \times 0.5m$ window. The workspace is $3m \times 5m \times 3m$, shown in Figure 1b. This scenario increases the difficulty of trajectory planning by forcing the UAVs to pass through a small opening.

III ***Three UAVs through Window***. Three UAVs are placed side-by-side on the same side of the same wall described above, with their goal points on the opposite side of the wall in the same configuration. The workspace is $2.5m \times 2.5m \times 3m$, shown in Figure 1c. This scenario increases the number of UAVs to show a typical bottleneck for multiple UAVs.

IV ***Obstacle Course Swap***. In this scenario, an obstacle course is constructed to force the MK-RRT* to path plan through a vertical window, a horizontal window, and another vertical window, perpendicular to the first. A UAV is placed at each end of the obstacle course and their goal is to swap positions. The workspace is $2m \times 2m \times 2m$, shown in Figure 1d. This scenario forces the UAVs to pass through successive narrow openings, a task which indoor UAVs may be subject to.

Generally, both planners are limited to a maximum of 10000 iterations to produce a valid trajectory. However, in the case of the ***Obstacle-Free Swap*** this limit is 5000 iterations. These iteration numbers are sufficient to find solutions, and in most cases, find several increasingly good solutions. Note that the number of iterations is the number of nodes in each tree. For the simultaneous method, the *Tree* will consist of 10000 nodes. For the sequential case, each independent *Tree* is allowed 10000 nodes. These numbers of nodes are chosen to allow each algorithm to have the same number of states embedded in their nodes. A node for the simultaneous method contains a state for all of the robots. Each node in the sequential method only contains the state of a single robot.

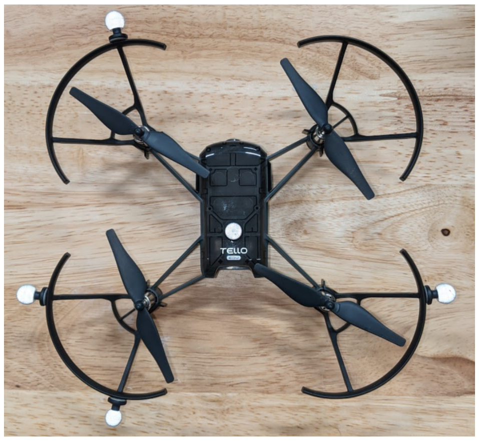


Fig. 2: Ryze Tello EDU UAV with OptiTrack markers connected to provide localization.

TABLE I: Tello model parameters

Axis	$a_{\dot{p},\dot{p}}$	$b_{\dot{p},\dot{p}_r}$
X	-0.6	1.1
Y	-0.6	1.1
Z	-2.1	2.0

For the experimental validation of both MK-RRT* variants, the palm-sized Ryze Tello EDU UAV, shown in Figure 2, was used. For the planners, the Tellos are approximated as spheres with a radius of 15cm, allowing for a small safety region around each platform. An OptiTrack Motion Capture (MoCap) system was used to provide real-time, high-precision localization, while a Kalman filter was used as a full state observer. Finally, a Model Predictive Controller (MPC) was used to drive the Tellos to the generated trajectories.

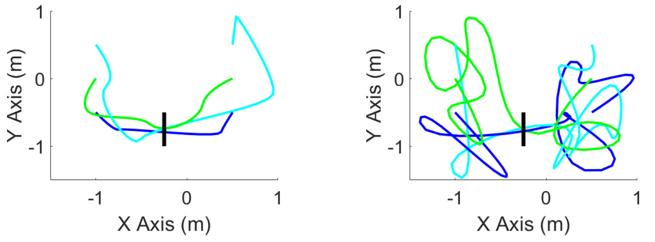
The Tello allows reference velocity inputs on all three axes for its control. A linear model that describes the internal system dynamics of the Tello must be derived, since it is needed for both the planners and the controller. To derive such a model, flight data were captured using the MoCap system and used in a least-squares based model identification algorithm. The state-space model for a single Tello is:

$$\begin{bmatrix}\dot{p}\\ \ddot{p}\end{bmatrix} = \begin{bmatrix}0 & 1\\ 0 & a_{\dot{p},\dot{p}}\end{bmatrix}\begin{bmatrix}p\\ \dot{p}\end{bmatrix} + \begin{bmatrix}0\\ b_{\dot{p},\dot{p}_r}\end{bmatrix}u \tag{8}$$

where p is the position and the input u is the reference velocity on each of the x, y, z axes. The parameters for the three axes are presented in Table I. To satisfy the KRRT* requirement that the state matrix needs to be nilpotent, the parameter $\alpha_{\dot{z},\dot{z}}$ was changed to -1 for the planning part and the z velocity range was limited to ensure a feasible trajectory.

TABLE II: Duration of planned trajectories in each scenario

Scenario	Sequential	Simultaneous
Obstacle-Free Swap	3.4	5.9
Through Window Swap	9.5	18.3
Three UAVs through Window	11.85	29.05
Obstacle Course Swap	12.1	16.85



(a) Sequential (b) Simultaneous

Fig. 3: Planned trajectories of the sequential and simultaneous planners in the ***Three UAVs through Window*** scenario. Dark blue represents Tello 1, green represents Tello 2, cyan represents Tello 3. Note that Tellos fly from the left side to the right.

V. Results

A. Planner Comparison

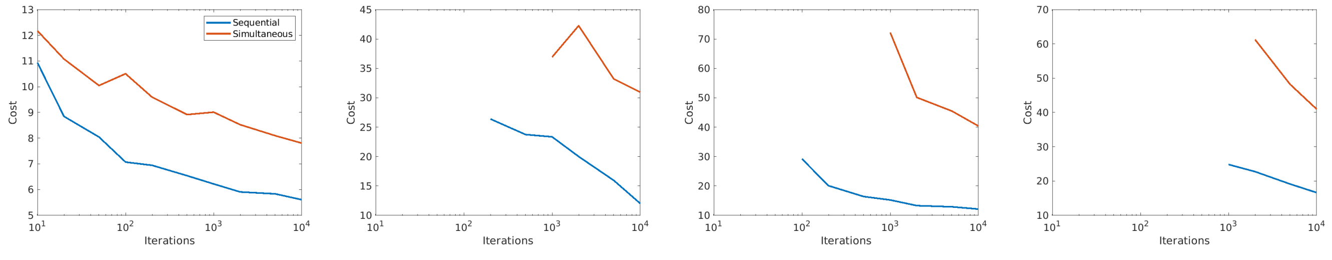
1) Generated Trajectories: The trajectories generated by the two methods differ in several regards, as shown in Figure 3. The most striking upon visual inspection is that the sequentially generated trajectories are much simpler and direct. The simultaneously generated trajectories on the contrary are convoluted.

Figure 3a shows the sequentially generated trajectory for the ***Three UAVs through Window*** scenario. The trajectory of Tello 2 is very direct connecting the start state to the end state. Tello 3 has a more complex route so that it can avoid the paths of Tello 1 and 2. Figure 3b shows the simultaneously planned variant of the same scenario. All Tellos follow complex trajectories. Since the simultaneous case plans in a state space that is a composite of all the individual Tellos' state spaces, the convergence to an optimal solution is much slower. The sequential method is able to benefit from running the KRRT* algorithm twice in a state space with half the dimensions. These facts correlate to the wider trajectories of the simultaneous case due to the much larger state space. Table II shows the durations of the various trajectories. The sequential trajectories are faster in all scenarios.

2) Trajectory Cost vs Iteration: Figure 4 shows the costs for each planner at 10, 20, 50, 100, 200, 500, 1000, 2000, 5000, and 10000 iterations of the MK-RRT*. These values were found by averaging the costs of 10 runs up to that many iterations. For the iteration counts where there is no data, the run was unable to find a trajectory.

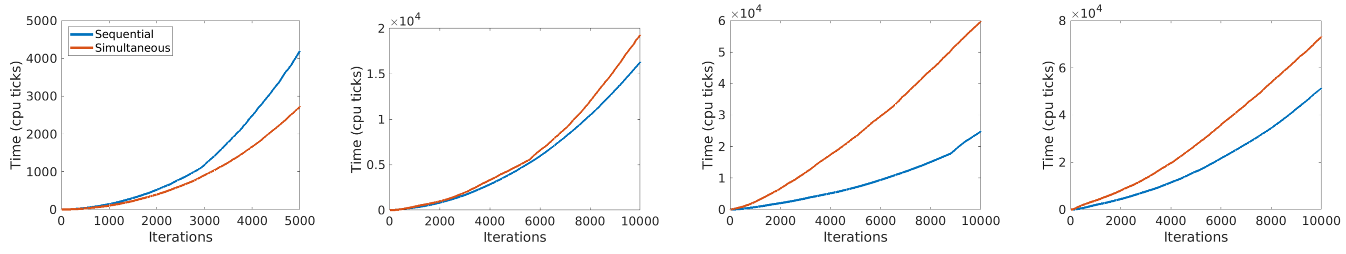
The cost of the best path was recorded at each iteration. For the simultaneous case, this cost is very easy to compare, as each iteration has only a single *Tree*. To yield the same cost function for the sequential case, as there are multiple trees, the total cost at the n$^{\text{th}}$ iteration is calculated as:

$$J = \sum_{i=1}^{N}\left(\int_0^{\tau_i}(u_i[t]^T R u_i[t])dt\right) + \max\left(\int_0^{\tau_i}\alpha dt\right)$$



(a) ***Obstacle-Free Swap*** (b) ***Through Window Swap*** (c) ***Three UAVs through Window*** (d) ***Obstacle Course Swap***

Fig. 4: Plots of the trajectory cost of each MK-RRT* method at each iteration. The plot begins at the first iteration in which a valid trajectory was found.



(a) ***Obstacle-Free Swap*** (b) ***Through Window Swap*** (c) ***Three UAVs through Window*** (d) ***Obstacle Course Swap***

Fig. 5: Plots of the cumulative run-time of each method at each iteration.

The KRRT* problem is run sequentially for all robots up to the predefined number of nodes. The total cost at the nth iteration is the control cost for each drone plus the maximum trajectory time.

3) Computation Time vs Iteration: The computation time of each iteration was recorded to show the relative speeds of the iterations for each method. Figure 5 shows the cumulative time at each iteration. For the sequential timings, the timing at each individual's nth iteration were summed to find the total at that iteration. In the environments with obstacles, the simultaneous MK-RRT* was slower, especially in the case of three Tellos. In the ***Obstacle-Free Swap*** scenario, the simultaneous method was faster than the sequential method.

B. Experimental Results

For all four scenarios considered, three runs were performed using the Tello UAVs. The main motivation for the experimental testing is to validate that the generated trajectories can be effectively tracked by a controller on a real platform without having to deviate much from the planned control effort. Finally, experimental testing helps identify issues that might not arise in simulations.

Figures 6 and 7 show the planned and actual trajectories for a test run on the ***Through Window Swap*** and ***Obstacle Course Swap*** scenarios respectively. To make the trajectories more visible, the obstacles have been removed and only the openings are shown. The increased complexity of the simultaneous planner trajectories is visible in both cases.

A linear MPC is used to track the planned trajectories generated by the proposed frameworks. The MPC has a horizon of $1sec$ and generates controls at a rate of $20Hz$. The cost function of the controller uses the same penalty on the control inputs as the planners. To force the controller to track the planned trajectory as faithfully as possible, the penalty on the state error was much higher than the penalty on the control input.

Three test runs were performed for each scenario. Table III shows the Root Mean Square Error for the different scenarios. From the results, it is evident that the MPC was able to track the planned trajectory. Moreover, Figure 8 shows a comparison between the planned and actual costs for each trajectory. Again, small discrepancies are observed between the planned and actual costs.

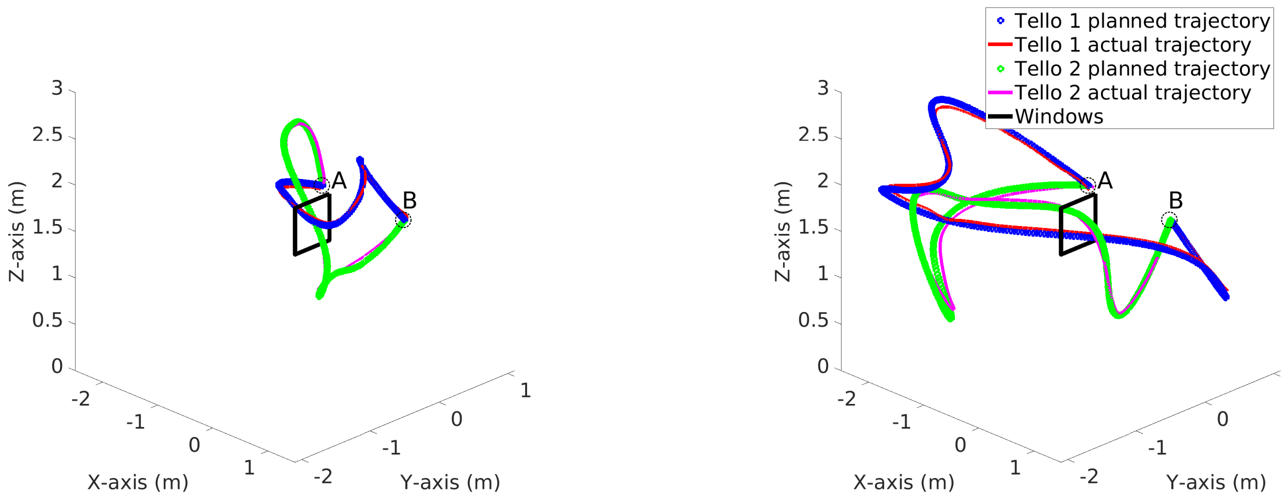
TABLE III: RMSE for each trajectory

Scenario	Sequential	Simultaneous
Obstacle-Free Swap	3.5cm	2.9cm
Through Window Swap	5.1cm	2.6cm
Three UAVs through Window	4.4cm	6.1cm
Obstacle Course Swap	3.6cm	4.8cm

VI. Discussion

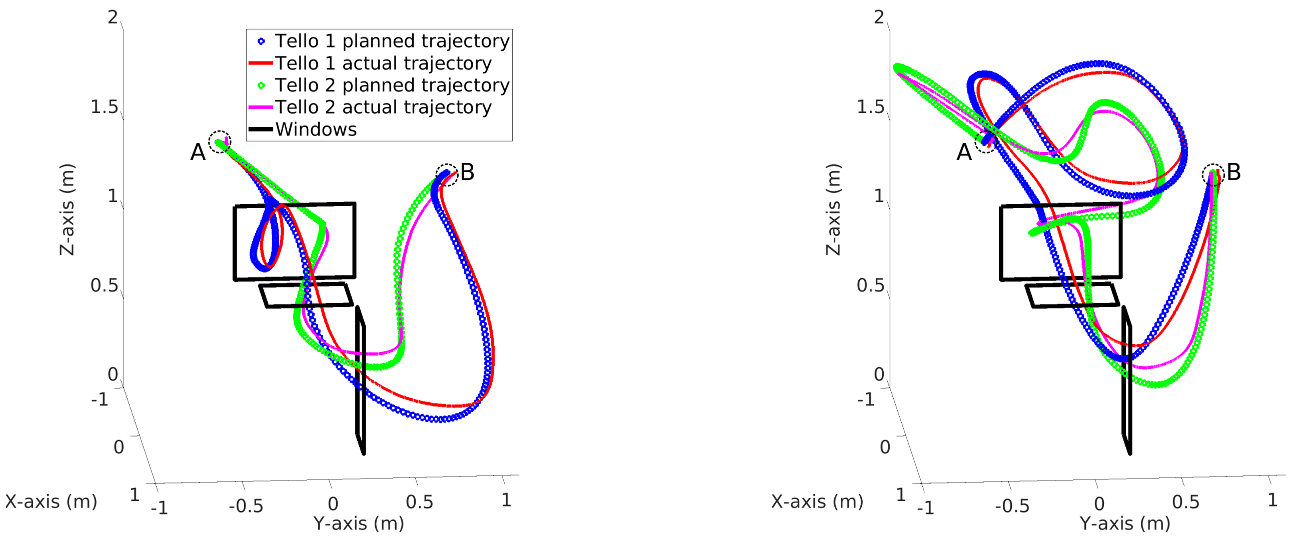
A. Trajectory cost and computation time

It is evident that the sequential method is generally faster than the simultaneous method, though there is an exception in the ***Obstacle-Free Swap*** scenario. It is also evident that the sequential method generally results in a lower cost than the simultaneous method. This is expected as distributed methods were originally proposed in [10] to provide a faster solution to multi-robot path planning problems. Since sampling in lower dimensions converges more quickly to an optimal solution in each robots problem description, each planner often finds significantly more direct routes.



(a) Sequential (b) Simultaneous

Fig. 6: ***Through Window Swap*** experiment results. The two Tello UAVs need to swap positions going through a window. The start point of the first Tello is point A and its goal point is B and vice versa for the second Tello. The blue curve is the planned trajectory for the first Tello with the red being the actual trajectory as it was captured by the MoCap system. For the second Tello, the planned trajectory is in green with the actual one in magenta.



(a) Sequential (b) Simultaneous

Fig. 7: ***Obstacle Course Swap*** experiment results. The two Tello UAVs need to swap positions going through an obstacle course. The start point of the first Tello is point A and its goal point is B and vice versa for the second Tello. The blue curve is the planned trajectory for the first Tello with the red being the actual trajectory as it was captured by the MoCap system. For the second Tello, the planned trajectory is in green with the actual one in magenta.

B. Asymptotic Optimality

Authors in [5] determined that Kinodynamic RRT* is asymptotically optimal and the probability of finding the optimal trajectory approaches 1 as the iterations approach infinity. The tightly-coupled approach shares this quality, as no change was made to the core algorithm, the only changes made were to the CollisionFree method and the plant. The loosely-coupled variant however does not share this quality. Figure 9b shows the cost of swapping two Tellos locked to a specific height. The Tellos begin at [0.25, 0.5]m and [0.75, 0.5]m and are restricted to a 1x1 square, as shown in figure 9a. The original model is used, with the vertical controller removed. Initially, the sequential case is able to find the faster solution by optimally connecting Tello 1's trajectory and planning Tello 2's trajectory with respect to it. However, given sufficient time, the simultaneous method finds better routes than the sequential method. This is an example of a problem in which the sequential case will not converge to the optimal solution, but the simultaneous planner will. Although, for more complex problems such as the ones used

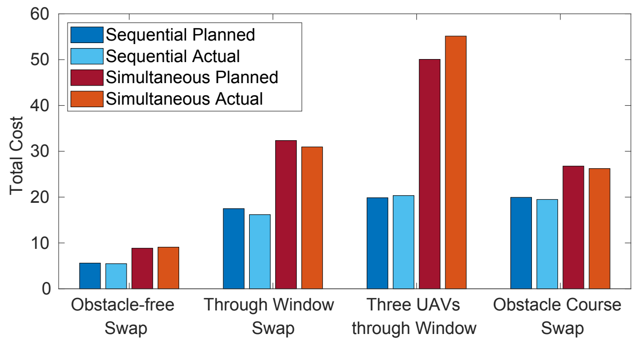


Fig. 8: Predicted and experimental costs of trajectories for each scenario.

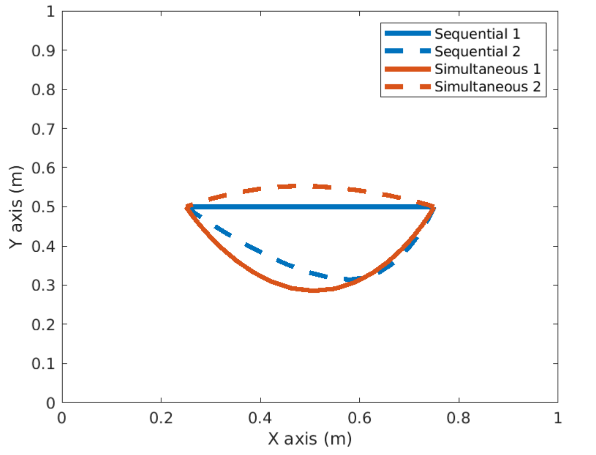
in the experiments presented here, the additional time that the simultaneous planner needs to find a better solution is too high, making it an infeasible solution. For example, for the ***Obstacle Course Swap*** experiment, using the data shown in Figure 4, we extrapolated that the cost of the simultaneous method would become equal or lower than the sequential method around the one millionth iteration. This extrapolation is probabilistic and is not guaranteed to be lower at this point.

Additionally, there are potential deadlock conditions in the sequential case. The possibility of deadlocks alone can remove the optimality guarantee, since the problem has a solution, but the sequential method may never be able to find such a solution given infinite iterations. Deadlocks may occur when the trajectory of higher priority robot blocks the trajectory of lower priority robot at some critical time until and including the final point in the trajectory.

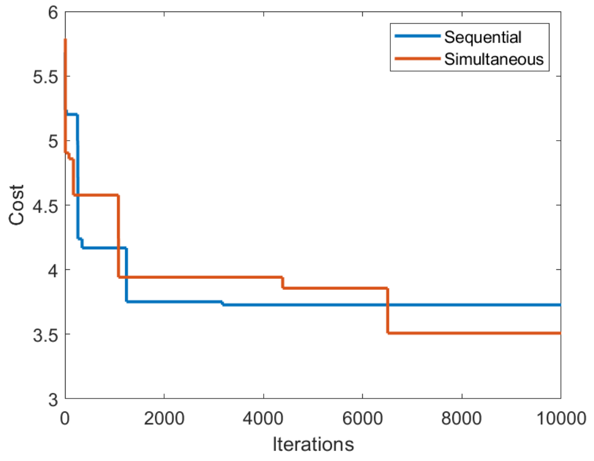
Figure 10a shows one such scenario where a deadlock may occur. If the states of the team were planned simultaneously, Tello 1 would have the chance to move down, allowing Tello 2 past. Using sequential planning however, a deadlock will occur. Figure 10b shows the sequential planner in the deadlock described above. Tello 1 starting at [0.6, 0.8]m is able to find a valid trajectory to [-0.8, 0.8]m (shown as a green line). Tello 2 starting at [-0.6, 0.8]m and ending at [0.8, 0.8]m is able to trajectory plan in the region that Tello 1 has not yet arrived. The farthest that Tello 2 is able to travel is the approximate middle of Tello 1's trajectory, as they will both arrive around the same time. Figure 10c shows the simultaneous solver with a solution to this deadlock situation. If there is a valid solution, the simultaneous solver will find it, given sufficient iterations.

VII. Conclusion and Future Work

In this paper, we introduced the Multi-robot Kinodynamic RRT* (MK-RRT*) algorithm for trajectory planning and demonstrated a comparison between a tightly- and loosely-coupled MK-RRT*. The trajectories generated through these methods were verified in real flights using multiple Ryze Tello EDU UAVs. Each trajectory was shown to plan valid trajectories for the linear, time-invariant models. The loosely-coupled, sequential approach generally produced faster results with lower costs, but are subject to possible deadlocks.



(a) Trajectories of sequential and simultaneous final solutions. Solid lines are Tello 1 travelling left to right, dashed are Tello 2 from right to left. Sequential is blue, simultaneous is orange.



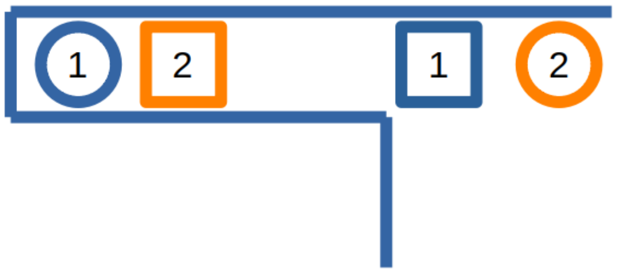
(b) Cost over time of each method.

Fig. 9: Convergence of simultaneous method toward optimal solution.

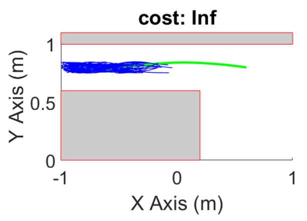
The tightly-coupled, simultaneous approach generally gave slower results with higher costs, but without the potential deadlocks of the sequential method. Furthermore, the sequential method is not asymptotically optimal, as only the first trajectory planned is guaranteed to have this quality. The simultaneous method, however, is asymptotically optimal, as it is a redefinition of the original problem, where the major change is the coupling of the controllers.

For the purposes of this work, the proposed algorithms were implemented in Matlab to allow for quick prototyping and testing. Consequently, the current implementation is sub-optimal and unable to run in real-time. As we aim to run the proposed frameworks in real-time to allow for online replanning, the algorithms will be reimplemented and optimized in a language such as C++.

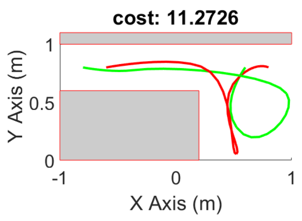
In the current implementation, the Tello UAVs are modeled as spheres to allow adequate space around them. From our experimental testing, we noticed that when the generated



(a) Deadlock scenario



(b) Sequential



(c) Simultaneous

Fig. 10: (a) a potential deadlock condition in sequential planning; circles represent goals and squares represent initial positions, (b) the solution for Tello 1 in green and the *Tree* for Tello 2, (c) the simultaneous solver was able to find a solution.

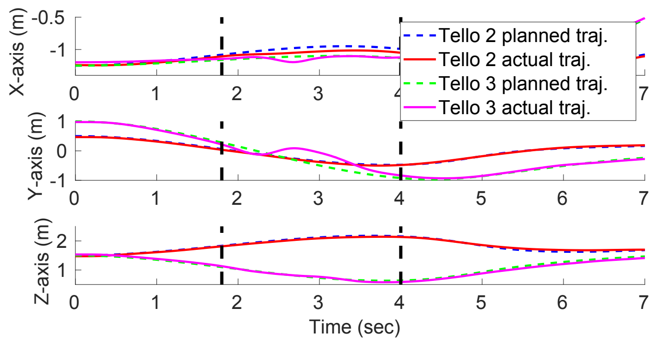


Fig. 11: Disturbance due to downwash.

trajectories placed UAVs above one another, the downwash of the top UAV would affect the flight of the UAV bellow as shown in Figure 11. This could be avoided either by modeling the UAVs as tall cylinders when checking for UAV to UAV collisions or by adding a cost to the vertical distance between UAVs.

In addition, our cost function did not take the distance between robots into account. To create safer paths, the distance between robots and obstacles at each instant could be used in the calculation of the cost to generate safer trajectories.

As with RRT*, this method may be slow to initially find a solution and converge in challenging environments such as through narrow openings. Variants of RRT* such as [15] and [16] provide smarter methods for sampling and connections to improve convergence rates. These informed methods may improve the performance of MK-RRT*.

References

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

[2] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, “Probabilistic roadmaps for path planning in high-dimensional configuration spaces,” *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996. [Online]. Available: https://doi.org/10.1109/70.508439

[3] J. H. Reif, “Complexity of the mover's problem and generalizations,” in *20th Annual Symposium on Foundations of Computer Science (sfcs 1979)*. IEEE, Oct. 1979. [Online]. Available: https://doi.org/10.1109/sfcs.1979.10

[4] 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, June 2011. [Online]. Available: https://doi.org/10.1177/0278364911406761

[5] 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, May 2013. [Online]. Available: https://doi.org/10.1109/icra.2013.6631299

[6] M. Ragaglia, M. Prandini, and L. Bascetta, “Poli-RRT*: Optimal RRT-based planning for constrained and feedback linearisable vehicle dynamics,” in *2015 European Control Conference (ECC)*. IEEE, July 2015. [Online]. Available: https://doi.org/10.1109/ecc.2015.7330917

[7] R. Luna and K. E. Bekris, “Efficient and complete centralized multi-robot path planning,” in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, Sept. 2011. [Online]. Available: https://doi.org/10.1109/iros.2011.6095085

[8] J. Peng and S. Akella, “Coordinating multiple robots with kinodynamic constraints along specified paths,” *The International Journal of Robotics Research*, vol. 24, no. 4, pp. 295–310, Apr. 2005. [Online]. Available: https://doi.org/10.1177/0278364905051974

[9] S. LaValle and S. Hutchinson, “Optimal motion planning for multiple robots having independent goals,” *IEEE Transactions on Robotics and Automation*, vol. 14, no. 6, pp. 912–925, 1998. [Online]. Available: https://doi.org/10.1109/70.736775

[10] J. Barraquand and J.-C. Latombe, “Robot motion planning: A distributed representation approach,” *The International Journal of Robotics Research*, vol. 10, no. 6, pp. 628–649, Dec. 1991. [Online]. Available: https://doi.org/10.1177/027836499101000604

[11] S. Buckley, “Fast motion planning for multiple moving robots,” in *Proceedings, 1989 International Conference on Robotics and Automation*. IEEE Comput. Soc. Press, 1989. [Online]. Available: https://doi.org/10.1109/robot.1989.100008

[12] M. Erdmann and T. Lozano-Perez, “On multiple moving objects,” in *Proceedings. 1986 IEEE International Conference on Robotics and Automation*. Institute of Electrical and Electronics Engineers, 1986. [Online]. Available: https://doi.org/10.1109/robot.1986.1087401

[13] M. Čáp, P. Novák, J. Vokřínek, and M. Pechoucek, “Multi-agent rrt*: Sampling-based cooperative pathfinding (extended abstract),” 02 2013.

[14] M. Ragaglia, M. Prandini, and L. Bascetta, “Multi-agent poli-rrt* optimal constrained rrt-based planning for multiple vehicles with feedback linearisable dynamics (workshop version),” 06 2016.

[15] 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 *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, Sept. 2014. [Online]. Available: https://doi.org/10.1109/iros.2014.6942976

[16] A. H. Qureshi and Y. Ayaz, “Intelligent bidirectional rapidly-exploring random trees for optimal motion planning in complex cluttered environments,” *Robotics and Autonomous Systems*, vol. 68, pp. 1–11, June 2015. [Online]. Available: https://doi.org/10.1016/j.robot.2015.02.007