2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Informable Multi-Objective and Multi-Directional RRT* System for Robot Path Planning

Jiunn-Kai Huang, Yingwen Tan, Dongmyeong Lee, Vishnu R. Desaraju, and Jessy W. Grizzle

***Abstract*— Multi-objective or multi-destination path planning is crucial for mobile robotics applications such as mobility as a service, robotics inspection, and electric vehicle charging for long trips. This work proposes an anytime iterative system to concurrently solve the multi-objective path planning problem and determine the visiting order of destinations. The system is comprised of an anytime informable multi-objective and multi-directional RRT* algorithm to form a simple connected graph, and a solver that consists of an enhanced cheapest insertion algorithm and a genetic algorithm to solve approximately the relaxed traveling salesman problem in polynomial time. Moreover, a list of waypoints is often provided for robotics inspection and vehicle routing so that the robot can preferentially visit certain equipment or areas of interest. We show that the proposed system can inherently incorporate such knowledge to navigate challenging topology. The proposed anytime system is evaluated on large and complex graphs built for real-world driving applications. C++ implementations are available at: https://github.com/UMich-BipedLab/IMOMD-RRTStar.**

I. Introduction and Contributions

Multi-objective or multi-destination path planning is a key enabler of applications such as data collection [1]–[3], traditional Traveling Salesman Problem (TSP) [4], [5], and electric vehicle charging for long trips. More recently, autonomous "Mobility as a Service" (e.g., autonomous shuttles between user-selected points) has become another important application of multi-objective planning such as car-pooling [6]–[9]. Therefore, being able to efficiently find paths connecting multiple destinations and determining the visiting order of the destinations (essentially, a relaxed TSP) are critical for modern navigation systems deployed on autonomous vehicles and robots. This paper seeks to solve these two problems by developing a system composed of a sampling-based anytime path planning algorithm and an approximate relaxed-TSP solver.

To represent multiple destinations, graphs composed of nodes and edges stand out for their sparse representations. In particular, graphs are a popular representation of topological landscape features, such as terrain contour, lane markers, or intersections [10]. Topological features do not change often; thus, they are maintainable and suitable for long-term support compared to high-definition (HD) maps. Graph-based maps such as OpenStreetMap [10] have been developed over the past two decades to describe topological features and are readily available worldwide. Therefore, we concentrate on developing the proposed informable multi-objective and multi-directional rapidly-exploring random trees (RRT*) system for path planning on large and complex graphs.

A multi-objective path planning system is charged with two tasks: **1)** find weighted paths (i.e., paths and traversal costs), if they exist, that connect the destinations. This operation results in an undirected and weighted graph where nodes and edges correspond to destinations and paths connecting destinations, respectively. **2)** determine the visiting order of destinations that minimizes total travel cost. The second task, called relaxed TSP, differs from standard TSP in that we are allowed to (or sometimes have to) visit a node multiple times; see Sec. II-C for a detailed discussion. Several approaches [11]–[14] have been developed to solve each of these tasks separately, assuming either the visiting order of the destinations is given, or a cyclic/complete graph is constructed and the weights of edges are provided. However, in real-time applications, the connectivity of the destinations and the weights of the paths between destinations (task 1) as well as the visiting order of the destinations (task 2) are often unknown. So it is crucial that we to solve these two tasks concurrently in an anytime manner, i.e., the system must provide suboptimal but monotonically improving solutions at any time throughout the path cost minimization process.

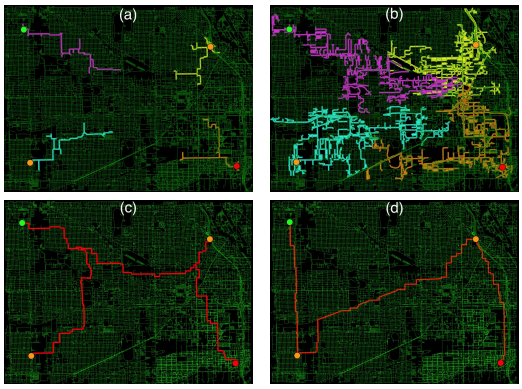

Fig. 1: Illustration of the proposed informable multi-objective and multi-directional RRT* (IMOMD-RRT*) system evaluated on OpenStreetMap of Chicago, containing $866,089$ nodes and $1,038,414$ edges. The green, red, and orange dots are the source, target, and objectives, respectively. (a) shows the initial stage of the tree expansion of each destination. (b) shows the trees from each destination form a connected graph. (c) shows the first path and visiting order from the IMOMD-RRT*. Given more computation time, (d) shows that IMOMD-RRT* returns a better path and order.

In this paper, we seek to develop an anytime iterative system to provide paths between multiple objectives and to determine the visiting order of destinations; moreover, the system should be informable, i.e., it can accommodate prior knowledge of intermediate nodes, if available. The proposed system consists of two components: **1)** an anytime informable multi-objective and multi-directional RRT* (IMOMD-RRT*) algorithm to form a connected weighted-undirected graph, and **2)** a relaxed TSP solver, ECI-Gen, that consists of an enhanced version of the cheapest in-

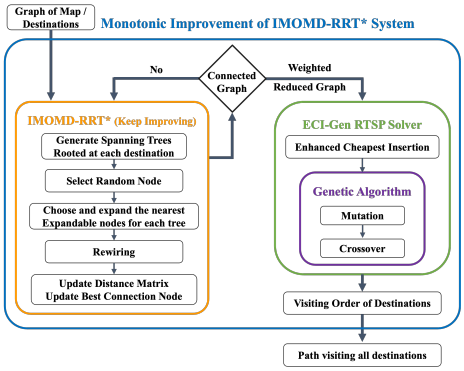

Fig. 2: Illustration of the proposed IMOMD-RRT* system. It consists of the anytime informable multi-objective and multi-directional RRT* (IMOMD-RRT*) algorithm to construct a connected weighted-undirected graph, and ECI-Gen, a polynomial-time relaxed TSP solver. The solver consists of an enhanced version of the cheapest insertion algorithm [15] and a genetic algorithm [16]–[18]. The full system (the blue box) will continue to run to improve the solution over time.

sertion algorithm [15] and a genetic algorithm [16]–[19]. The proposed system is evaluated on large and complex graphs built for real-world driving applications, such as the OpenStreetMap of Chicago containing $866,089$ nodes and $1,038,414$ edges shown in Fig. 1.

The contributions of the overall system (Fig. 2) include:

1) An anytime informable multi-objective and multi-directional RRT* system that runs on large and complex graphs. Anytime implies that the system can, in the allotted time, construct a path on a large-scale undirected weighted graph that meets the existence constraint (the path must traverse each objective at least once), and the order constraint (with fixed start and end points), if any such path exists, or terminates and indicates no such path exists. Therefore, the resulting weighted reduced graph containing the objectives, source, and target is connected, whenever a solution exists.

2) The problem of determining the visiting order of destinations in a connected graph is a relaxed TSP with fixed start and end nodes, though intermediate nodes can (or sometimes must) be revisited. We introduce the ECI-Gen solver that combines an enhancement of the cheapest insertion algorithm [15] and a genetic algorithm [16]–[19] to solve approximately the R-TSP in polynomial time.

3) We show that prior knowledge (such as reference waypoints) for robotics inspection of a pipeline or factory can be readily integrated into the IMOMD-RRT*. In addition, providing prior knowledge to the planner can help with navigating challenging topology.

4) We show on large graphs that the proposed IMOMD-RRT* outperforms bi-directional A* [20], ANA* [21], and kA* [22] in terms of speed and memory usage. We also demonstrate by providing reference waypoints, the IMOMD-RRT* escapes from bug traps (e.g., single entry neighborhoods) in complex graphs.

5) We open-source the C++ implementation of the system at https://github.com/UMich-BipedLab/IMOMD-RRTStar.

The remainder of this paper is organized as follows. Section II summarizes the related work. Section III explains the proposed anytime IMOMD-RRT* to construct a simple connected graph. The ECI-Gen solver to determine the ordering of destinations in the connected graph is discussed in Sec. IV. Experimental evaluation of the proposed system on large and complex graphs is presented in Sec. V. Finally, Sec. VI concludes the paper with avenues for future work.

II. Related Work

In this section, we review several types of path planning algorithms and techniques to improve their efficiency. We also place our work in context with existing literature on car-pooling/ride-sharing and the traveling salesman problem.

A. Common Path Planners

Path planners aim to find the shortest path from a single source to a single target. Graph-based and sampling-based algorithms are the two prominent categories of planners.

Graph-based algorithms [20], [21], [23]–[27] such as Dijkstra [24] and A* [23] discretize a continuous space to an undirected graph composed of nodes and weighted edges. They are popular for their efficiency on low-dimensional configuration spaces and small graphs. There are many techniques [20], [21], [25]–[27] to improve their computation efficiency on large graphs. Inflating the heuristic value makes the A* algorithm likely to expand the nodes that are close to the goal at the expense of optimality. Anytime Repairing A* (ARA*) [26] uses weighted A* and decreases the weight parameter at each iteration, leading to a better solution. Anytime Non-parametric A* (ANA*) [21] is as efficient as ARA* and spends less time between solution improvements. R* [25] is a randomized version of A* to improve performance. The efficient many-to-many path planning algorithm [28] re-uses the former explored trees when the current source or target is unchanged from the former source or target. Heuristic search for one-to-many shortest path kA* [22] considers all goals as a single compound goal by combining heuristic estimates of all goals into one. It then consumes less time and memory than k executions of A* by expanding the common route for the k destinations only once.

Algorithms that improve exploration efficiency, such as Jumping Point Search [27], only work for grid maps. However, graph-based algorithms inherently suffer from bug traps, whereas sampling-based methods can overcome bug traps more easily via informed sampling; see Sec. V for a detailed discussion. Sampling-based algorithms such as rapidly-exploring random trees (RRT) [29] and its asymptotically optimal version, RRT* [30]–[32] stand out for their low complexity and high efficiency in exploring high-dimensional, continuous configuration spaces. Sampling-based algorithms on discrete spaces such as RRLT and d-RRT* have also been applied to multi-robot motion planning [33]–[36]. We leverage RRT* to construct a simple connected graph containing multiple destinations from a large and complex map and embed prior knowledge of a reference path.

B. Car-Pooling and Ride-Sharing

Problems such as car-pooling, ride-sharing, food delivery, or public transportation handle different types of constraints such as maximum seats, time window, battery charge, number of served requests along with multiple destinations

Fig. 3: (a) shows a *simple connected graph* where the objectives have to be visited twice to visit all destinations. (b) shows the case where revisiting the source v_s allows a shorter path when triangular inequality does not hold.

[6]–[9], [37]–[45]. These problems are usually solved by Genetic Algorithms [6], Ant Colony Optimization [37], [45], Dynamic Programming [41], [42] or reinforcement learning [8], [44]. These methods assume, however, that weighted paths between destinations in the graph are known [6], [8], [37], [38], [40], [42], [45]. In practice, the connecting paths and their weights are unknown and must be constructed.

C. Traveling Salesman Problem Solvers

Determining the travel order of nodes in an undirected graph with known edge weights is referred to as a Traveling Salesman Problem (TSP). The TSP is an NP-hard problem to find the shortest possible cycle that visits every node exactly once and returns to the start node. Another variant of TSP, the shortest Hamiltonian path problem [4], is to find the shortest path that visits all nodes exactly once between a fixed starting node (v_s) and a fixed terminating node (v_t). The problem can be solved as a standard TSP problem by assigning a large negative cost to the edge between v_s and v_t [4], [5].

We are inspired by the work of [11], [13], where the authors propose to solve the problem through a single-phase algorithm in simulated continuous configuration spaces. They first leverage multiple random trees to solve the multi-goal path planning problem and then solve the TSP via an open-source solver. In particular, they assume that the paths between nodes in the continuous spaces can be constructed in a single pass and that the resulting graph can always form a cycle. The problem can then be solved by a traditional TSP solver. In practice, however, the graph might not form a cycle (e.g., an acyclic graph or forest) as shown in Fig. 3a. Even if the graph is cyclic or there exists a Hamiltonian path, there is no guarantee the path is the shortest. In Fig. 3b, the Hamiltonian path simply is $v_s \rightarrow o_1 \rightarrow v_t$, and the traversed distance is 12. However, another shorter path exists if we are allowed to traverse a node (v_s in this case) more than once: $v_s \rightarrow o_1 \rightarrow v_s \rightarrow v_t$ and the distance is 7. We, therefore, propose a polynomial-time approximate solver for this relaxed TSP problem; see Sec. IV for further discussion.

III. IMOMD-RRT* Algorithm

This section introduces our anytime informable multi-objective and multi-directional Rapidly-exploring Random Tree* (IMOMD-RRT*) algorithm.

A. Multi-objective and Multi-directional RRT* on Graphs

In this paper, we use *map* to refer to the input graph, which might contain millions of nodes, and use *graph* to refer to the graph composed of only the destinations including the source and target node. The proposed IMOMD-RRT* differs from the original RRT* [30] and graph-based RRT* [33]–[36] in six aspects when growing a tree. First, the sampling

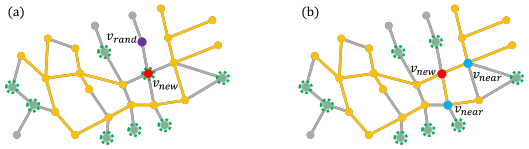

Fig. 4: Illustration of tree expansion and connection nodes of a tree. The unexplored paths in the graph and the spanning tree are represented by the gray and yellow lines, respectively. Green-dashed circles denote expandable nodes. (a) the tree $\mathcal{T}_i$ extends toward v_{rand} by connecting to the closest expandable node, v_{new}. (b) $\mathcal{X}_i$ is the updated set of expandable nodes and the tree is rewired around v_{new}. The blue dots indicate rewired nodes, v_{near}.

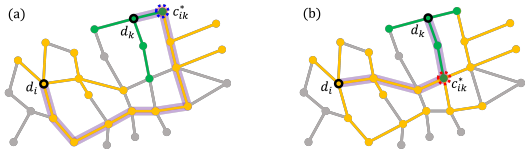

Fig. 5: Illustration of better connection nodes resulting in a better path. Two trees rooted at d_i and d_k are marked in yellow and green, respectively. The highlighted purple line shows the shortest path between d_i and d_k. (a) shows the path with the original connection node (dashed-blue circle). The newly extended node (dashed-red circle) is added to a set of connection nodes $\mathcal{C}_{ik}$, as shown in (b). The element of the distance matrix, A_{ik}, and the connection node c^*_{ik} that generates the shortest path between the destination d_i and d_k are updated as a shorter path is found.

is performed by picking a random v_{rand} in the map, and not from an underlying continuous space. The goal bias is not only applied to the target but also the source and all the objectives. Second, a steering function directly finds the closest expandable node as v_{new} to the random node v_{rand}, without finding the nearest node in the tree (Fig. 4a). Unlike graph-based RRT*, this process speeds up exploration by not selecting nodes that already exist in the tree. Note that instead of directly sampling from the set of expandable nodes, sampling from the map ameliorates the bias of sampling on the explored area. Third, the parent node is chosen from the nodes connected with the new node v_{new}, called the neighbor nodes. Among the neighbor nodes, the node that yields the lowest path cost from the root becomes the new node's parent. Fourth, the jumping point search algorithm [27] is also leveraged to speed up tree exploration. Fifth, IMOMD-RRT* rewires the neighborhood nodes to minimize the accumulated cost from the root of a tree to v_{new} (Fig. 4b). Lastly, if v_{new} belongs to more than one tree, this node is considered a connection node, which connects the path between destinations (Fig. 5).

Our proposed graph-based RRT* is summarized below with notation that follows graph theory [46]. The map (i.e., input graph) $\mathcal{G}$ is an ordered triple $(\mathcal{V}(\mathcal{G}), \mathcal{E}(\mathcal{G}), \Phi_{\mathcal{G}})$, where $\mathcal{V}(\mathcal{G}) = \{v \in \zeta\}$ is a set of nodes in the robot state space ζ, $\mathcal{E}(\mathcal{G})$ is a set of edges (disjoint from $\mathcal{V}(\mathcal{G})$), and an indication function $\Phi_{\mathcal{G}}$ associating each edge of $\mathcal{G}$ with an unordered pair (not necessarily distinct) of nodes of $\mathcal{G}$.

Given a set of destinations $\mathcal{D} = \{d_i | d_i \in \{v_s, v_t\} \cup \mathcal{O}\}_{i=1}^{m+2}$, where $v_s \in \mathcal{V}(\mathcal{G})$ is the source node, $v_t \in \mathcal{V}(\mathcal{G})$ is the target node, and $\mathcal{O} \subseteq \mathcal{V}(\mathcal{G})$ is the set of m objectives, the IMOMD-RRT* solves the multi-objective planning problem by growing a tree $\mathcal{T}_i = (V, E)$, where $V \subseteq \mathcal{V}(\mathcal{G})$ is a set of nodes connected by edges $E \subseteq \mathcal{E}(\mathcal{G})$, at each of the destinations $d_i \in \mathcal{D}$. Thus, it leads to a family of trees

$\mathcal{T} = \{\mathcal{T}_1, \cdots, \mathcal{T}_m, \mathcal{T}_{m+1}, \mathcal{T}_{m+2}\}$.

The proposed IMOMD-RRT* explores $\mathcal{G}$ by random sampling from $\mathcal{V}(\mathcal{G})$ and extending nodes to grow each tree. We explain the key functions of IMOMD-RRT* below.

1) Tree Expansion: Let $\mathcal{N}(v)$ be the neighborhood of node v (i.e., the set of nodes directly connected to v) [46]. Node v is expandable if there exists at least one unvisited node and one node of the tree in $\mathcal{N}(v)$, shown as the dashed-green circles in Fig 4a. Let $\mathcal{X}_i$ be the set of expandable nodes of the tree $\mathcal{T}_i$. A random node v_{rand} is sampled from $\mathcal{V}(\mathcal{G})$. Next, given a distance function $\texttt{Dist} : \mathcal{V}(\mathcal{G}) \times \mathcal{V}(\mathcal{G}) \to [0, \infty)$, find the nearest node v_{new} in the set of expandable nodes $\mathcal{X}_i$,

$$v_{\text{new}} = \underset{v \in \mathcal{X}_i}{\arg\min}\, \texttt{Dist}(v, v_{\text{rand}}), \tag{1}$$

and in the case of multiple minimizing nodes, we take the first one found. We use the Jumping Point Search algorithm [27] to speed up the tree expansion. If the current v_{new} has only one neighbor that is not in the tree, v_{new} is added to the tree and that neighbor is selected as the new v_{new}. This process continues until v_{new} has at least two neighbors not in the tree, or it reaches v_{rand}.

2) Parent Selection: Let the set $\mathcal{N}_i(v_{\text{new}})$ be the neighborhood of the v_{new} in the tree $\mathcal{T}_i$. The (first) node v_{near} in $\mathcal{N}_i(v_{\text{new}})$ that results in the smallest cost-to-come, $\texttt{Cost}(\cdot, \cdot)$, is the parent of the v_{new} and is determined by:

$$v_{\text{parent}} = \underset{v_{\text{near}} \in \mathcal{N}_i(v_{\text{new}})}{\arg\min}\, \{\texttt{Cost}(\mathcal{T}_i, v_{\text{near}}) + \texttt{Dist}(v_{\text{near}}, v_{\text{new}})\}. \tag{2}$$

Next, all the unvisted nodes in $\mathcal{N}_i(v_{\text{new}})$ are added to the set of expandable nodes $\mathcal{X}_i$.

3) Tree Rewiring: After the parent node is chosen, the nearby nodes are rewired if a shorter path to the node through v_{new} is found (Fig. 4b). The rewiring step guarantees asymptotic optimality, as with the classic algorithm.

4) Tree Connection Update: A node is a connection node if it belongs to more than one tree. Let $\mathcal{C}_{ik}$ be the set of connection nodes between $\mathcal{T}_i$ and $\mathcal{T}_k$, and let c^*_{ik} denote the (first) node that connects $\mathcal{T}_i$ and $\mathcal{T}_k$ with the shortest distance

$$c^*_{ik} = \underset{c \in \mathcal{C}_{ik}}{\arg\min}\, \{\texttt{Cost}(\mathcal{T}_i, c) + \texttt{Cost}(\mathcal{T}_k, c)\}. \tag{3}$$

Let $A_{(m+2)\times(m+2)}$ be a distance matrix that represents pairwise distances between the destinations, where m is the number of objectives. The element $A_{i,k}$ indicates the shortest path between destinations d_i and d_k, as shown in Fig. 5b:

$$A_{i,k} = \texttt{Cost}(\mathcal{T}_i, c^*_{ik}) + \texttt{Cost}(\mathcal{T}_k, c^*_{ik}). \tag{4}$$

B. Discussion of Informability

As mentioned in Sec. I, applications such as robotic inspection or vehicle routing might consider prior knowledge of the path, so that the robot can examine certain equipment or areas of interest or avoid certain areas in a factory. The prior knowledge can be naturally provided as a number of "pseudo destinations" or samples in the IMOMD-RRT*. A pseudo destination is an artificial destination to help IMOMD-RRT* to form a connected graph. However, unlike *true* destinations that will always be visited, a pseudo destination might not be visited after rewiring (Fig. 6). Pseudo destinations can also encode prior knowledge for traversing challenging topology, such as bug-traps; see Sec. V.

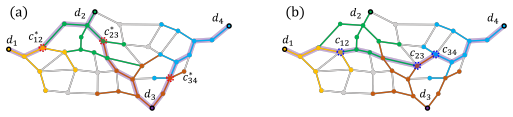

Fig. 6: Illustration of the rewired path of pseudo-destinations. $d_1 - d_4$ are the destinations and marked in different colors. The dashed-red circles are the connection nodes between two trees. The thick purple line is the final path. (a) shows the resulting path as if $d_1 - d_4$ are "true" destinations, and (b) is the resulting path as if $d_2 - d_3$ are "pseudo" destinations. The real destinations must be visited as in (a), whereas pseudo destinations are artificial destinations to help form a connected graph, and might no longer be visited after rewiring

IV. ECI-Gen Algorithm

This section introduces a polynomial-time approximate solver for the relaxed traveling salesman problem (R-TSP).

A. Relaxed Traveling Salesman Problem

The R-TSP differs from standard TSP [4], [5], [47] in two perspectives. First, nodes are allowed to be visited more than once, as mentioned in Sec. II-C. Second, we have a source node where we start and a target node where we end. Therefore, the R-TSP can also be considered a relaxed Hamiltonian path problem [4], [47]. We propose the ECI-Gen solver, which consists of an enhanced version of the cheapest insertion algorithm [15] and a genetic algorithm [16]–[18] to solve the R-TSP. The complexity of the proposed solver is $O(N^3)$, where N is the cardinality of the destination set $\mathcal{D}$.

B. Graph Definitions and Connectivity

A graph is simple or strict if it has no loops and no two edges join the same pair of nodes [46], [48]. In addition, a path is a sequence of nodes in the graph, where consecutive nodes in the sequence are adjacent, and no node appears more than once in the sequence. A graph is connected if and only if there is a path between each pair of destinations. Once all the destinations form a simple-connected graph, there exists at least one path π that passes all destinations $\mathcal{D}$. We can then consider the problem as an R-TSP (see Sec. IV-A), with a source and target node as well as several objectives to be visited. Therefore, we impose the graph connectivity and simplicity as sufficient conditions to solve the R-TSP. The disjoint-set data structure [49], [50] is implemented to verify the connectivity of a graph.

C. Enhanced Cheapest Insertion Algorithm

The cheapest insertion (CI) algorithm [15] provides an efficient means to find a sub-optimal sequence that guarantees less than twice the optimal sequence cost. However, it does not handle the case where revisiting the same node makes a shorter sequence. Therefore, we propose an enhanced version of the cheapest insertion algorithm (ECI), which comprises of a set of actions: **1)** in-sequence insertion, $\lambda_{\text{in-sequence}}$, as in regular cheapest insertion; **2)** in-place insertion, $\lambda_{\text{in-place}}$, to allow existing nodes to be revisited; and **3)** swapping insertion, $\lambda_{\text{swapping}}$, which is inspired by genetic algorithms. The sequence is then refined at the end of the algorithm.

Let $\mathcal{S}_{\text{current}} = \{v_s, s_1, \cdots, s_i, s_{i+1}, \cdots, s_n, v_t\}$ be the current sequence to indicate the visiting order of destinations, where $s_{\{\cdot\}} \in \mathcal{D}$ and $(n+2)$ is the number of destinations in the sequence. The travel cost $\theta(\cdot, \cdot)$ is the path distance

between two destinations provided by IMOMD-RRT*. The $\mathcal{S}_{\text{current}}$ is constructed by Dijkstra's algorithm on the graph.

Let $\mathcal{K}$ denote the set of destinations to be inserted, and let the to-be-inserted destination be d_k and its ancestor be s_i, where $d_k \in \mathcal{K}$ and $s_i \in \mathcal{S}_{\text{current}}$. Given $\mathcal{S}_{\text{current}}$, the location to insert s^* and the action of insertion λ^* are determined by

$$\lambda^*, s^* = \underset{\lambda_j \in \Lambda,\ s_i \in \mathcal{S}_{\text{current}}}{\arg\min} \lambda_j(s_i, d_k), \tag{5}$$

where $\Lambda = \{\lambda_{\text{in-sequence}}, \lambda_{\text{in-place}}, \lambda_{\text{swapping}}\}$ is the set of the insertion actions.

1) In-place Insertion $\lambda_{\textit{in-place}}$: This step detours from s_i to d_k. The resulting sequence is $\mathcal{S}_{\text{modified}} = \{v_s, s_1 \cdots, s_i, d_k, s_i, s_{i+1}, \cdots, s_n, v_t\}$, yielding

$$\lambda_{\text{in-place}}(s_i, d_k) = 2\theta(s_i, d_k). \tag{6}$$

2) In-sequence Insertion $\lambda_{\textit{in-sequence}}$: This step inserts d_k between s_i and s_{i+1} ($\forall s_i \in \{\mathcal{S}_{\text{current}}/(v_t)\}$), and the resulting sequence is $\mathcal{S}_{\text{modified}} = \{v_s, s_1, \cdots, s_i, d_k, s_{i+1}, \cdots, s_n, v_t\}$. The insertion distance is

$$\lambda_{\text{in-sequence}}(s_i, d_k) = \theta(s_i, d_k) + \theta(d_k, s_{i+1}) - \theta(s_i, s_{i+1}). \tag{7}$$

3) Swapping Insertion $\lambda_{\textit{swapping}}$: The swapping insertion changes the order of nodes right next to the newly inserted node. There are three cases in swapping insertion: swapping left, right, or both. For the case of swapping left, the modified sequence is $\mathcal{S}_{\text{modified}} = \{v_s, s_1 \cdots, s_{i-2}, s_i, s_{i-1}, d_k, s_{i+1}, \cdots, s_n, v_t\}$ by inserting d_k between s_i and s_{i+1} ($\forall s_i \in \{\mathcal{S}_{\text{current}}/(v_s, s_1)\}$), then swapping s_i and s_{i-1}. The insertion distance becomes

$$\begin{aligned}\lambda_{\text{swapping (left)}}(s_i, d_k) &= \theta(d_k, s_{i+1}) - \theta(s_i, s_{i+1}) \\ &+ \theta(s_{i-1}, d_k) + \theta(s_{i-2}, s_i) - \theta(s_{i-2}, s_{i-1}).\end{aligned} \tag{8}$$

The right swap is similar except that it swaps s_{i+1} and s_{i+2} instead. Lastly, the case of swapping both does a left swap (s_i and s_{i+1}) and then a right swap (s_{i+1} and s_{i+2}).

4) Sequence Refinement: In-place insertion occurs when the graph is not cyclic or the triangular inequality does not hold on the graph. This could generate redundant revisited nodes in the final result and lead to a longer sequence. We further refine the sequence by skipping revisited destinations when the previous destination and the next destination are connected. The refined sequence of destinations, $\mathcal{S}_{\text{ECI}}$, with cardinality $r \leq n$ is the input to the genetic algorithm.

D. Genetic Algorithm

We apply a genetic algorithm (GA) [16]–[18] to refine the sequence from ECI. The GA selects a parent[1] sequence and generates a new offspring sequence from it by a mutation or crossover process. We take the ordered sequence $\mathcal{S}_{\text{ECI}}$ from ECI as our first and only parent for the mutation process.

1) Mutation: There are three steps for each mutation process; see [51, Fig. 8 (a)]. First, the ordered sequence $\mathcal{S}_{\text{ECI}}$ from the enhanced cheapest insertion is randomly divided into k segments. Second, random inversion (reverses the order of destinations in the segment) is executed for each segment except the first and last segments, which contain the source and the target. Lastly, the segments in the middle are randomly reordered and spliced together. Let the resulting offspring sequence be $\psi = \{v_s, p_1, p_2, \cdots, p_r, v_t\}$, where v_s is the source node, v_t is the target node, $(r+2)$ is the number of destinations in the sequence (the cardinality of S_{ECI}), and $\{p_i\}_{i=1}^{r}$ is the re-ordered destinations, which may contain the start and end. The cost Θ of the offspring is:

$$\Theta = \theta(v_s, p_1) + \Sigma_{i=1}^{r}\theta(p_i, p_{i+1}) + \theta(p_n, v_t), \tag{9}$$

where $\theta(\cdot, \cdot)$ is the path distance between two destinations provided by the IMOMD-RRT*.

We perform the mutation process thousands of times, resulting in thousands of offspring. Only the offspring with a lower cost than the parent are kept for the crossover process.

2) Crossover: Let $\Psi = \{\psi_i\}_{i=1}^{h}$ be the set of mutated sequences, where h is the number of offspring kept after the mutation process. For each generation, the crossover process is performed thousands of times and only the offspring with a lower cost than the previous generation are kept. Each crossover process combines sub-sequences of any two sequences ($\psi_i, \psi_j \in \Psi$) to generate a new offspring; see [51, Fig. 8 (b)]. The probability of a sequence ψ_i being picked is:

$$P_\psi(\psi = \psi_i) = \frac{\rho_i}{\sum_{i=1}^{w}\rho_i}, \quad \rho_i = \frac{1}{\Theta_i}, \tag{10}$$

where w is the number of the remaining offspring from each generation after the $(i-1)^{\text{th}}$ generation and ρ_i is the fitness of the sequence ψ_i. Given the two selected sequences and an empty to-be-filled offspring, a segment of one of the two sequences is randomly selected, and random inversion is performed on it. The resulting segment is randomly placed inside the empty sequence of the offspring. Lastly, the remaining elements of the offspring are filled by the order of the other sequence except the elements that are already in the offspring sequence. After a few generations, the offspring with the lowest cost, ψ^*, is the final sequence of destinations, $\mathcal{S}_{\text{ECI-Gen}}$.

Remark 1: Whenever the IMOMD-RRT* provides a better path (due to its asymptotic optimality), the ECI-Gen solver will be executed to find a better visiting order of the destinations. Therefore, the full system provides paths with monotonically improving path cost in an anytime fashion.

E. Discussion of time complexity

As mentioned in Sec. IV-C, the first sequence $\mathcal{S}_{\text{current}}$ is constructed by Dijkstra's algorithm, which is an $O(N^2)$ process, where N is the cardinality of the destination set $\mathcal{D}$. We then pass $\mathcal{S}_{\text{current}}$ to the ECI algorithm, whose time complexity is $O(N^3)$, to generate a set of parents for the genetic algorithm, which is also an $O(N^3)$ process. Therefore, the overall time complexity of the proposed ECI-Gen solver is $O(N^3)$, and indeed a polynomial solver.

V. Experimental Results

This section presents evaluations of the IMOMD-RRT* system applied to two complex vehicle routing scenarios. The robot state ζ is defined as latitude and longitude. The distance between robot states $\texttt{Dist}(\cdot, \cdot)$ in (1) is defined as the haversine distance [52]. We use bi-directional A* [20], ANA* [21], and kA* [22] as baselines to compare the speed and memory usage (the number of explored nodes) of the IMOMD-RRT* algorithm. Once a connected graph is formed, the proposed

[1]Note that the parent in the GA is a different concept from the parent node in RRT* (Sec. III-A), but we keep the terminology to follow the literature.

TABLE I: Quantitative results of the IMOMD-RRT* system on two large maps (both contain more than one million nodes and edges) built for real robotics and vehicle applications. The proposed system outperforms bi-A*, ANA*, and kA*. Note that all four approaches use the ECI-Gen solver.

		Initial Solution Time [seconds]	Initial Path Cost [kilometers]	Final Memory Usage [# explored nodes]
Seattle	IMOMD-RRT*	**0.44**	**501,342**	**49,768**
	Bi-A*	4.40	808,416	3,240,515
	ANA*	1.71	1,089,873	234,457
	kA*	3.97	1,089,443	1,038,055
San Francisco	IMOMD-RRT*	**1.10**	156,807	**61,785**
	Bi-A*	9.93	315,061	3,640,863
	ANA*	Failed	Failed	Failed
	kA*	7.38	**130,976**	1,263,448

ECI-Gen R-TSP solver is applied to all the methods to determine the visiting order of the destinations. All hyper-parameters are kept the same throughout all experiments.

We evaluate the IMOMD-RRT* system (IMOMD-RRT* and the ECI-Gen solver) on a large and complex map of Seattle, USA. The map contains $1,054,372$ nodes and $1,173,514$ edges, and is downloaded from OpenStreetMap (OSM), which is a public map service built for real applications (e.g., Apple Map® actually uses OpenStreetMap as their foundation.). We then place 25 destinations in the map. We demonstrate that the IMOMD-RRT* system is able to concurrently find paths connecting destinations and determine the order of destinations. We also show that the system escapes from a bug trap by inherently receiving prior knowledge. The algorithm runs on a laptop with an Intel® Core™ i7-1185G7 CPU @ 3.00 GHz.

To show the performance and ability of multi-objective and determining the visiting order, we randomly set 25 destinations in the Seattle map. There are 25! possible combinations of visiting orders and therefore it is intractable to solve the visiting order by brute force. The results are shown in Fig. 7, where IMOMD-RRT* finds the first path faster than all the three baselines with a lower cost and then also spends less time between solution improvements. Due to the nature of RRT*, the quality of the path is improved monotonically. Additionally, the memory usage of IMOMD-RRT* is less than ANA*, much less than bi-A* and kA*. As shown in Table I, the proposed system provides the first solution 10 times faster than bi-A*, nine times faster than kA*, and four time faster than ANA*. The proposed system also consumes 65 times less memory than bi-A*, 20.8 times less memory than kA*, and 4.7 times less memory usage than ANA*.

Prior knowledge through pseudo destinations can also be leveraged to traverse challenging topology, such as bug-traps [53]. This problem is commonly seen in man-made environments such as a neighborhood with a single entry or cities separated by a body of water, as in Fig. 8. As mentioned in Sec. III-B, prior knowledge is provided as a number of pseudo destinations in the IMOMD-RRT* as a prior collision-free path in the graph for robotics inspection or vehicle routing. The prior path is then rewired by the IMOMD-RRT* to improve the path. We demonstrate this feature by providing the prior knowledge to escape the bug trap in San Francisco, as shown in Fig. 8. The map contains $1,277,702$ nodes and $1,437,713$ edges. As shown in Table I, the proposed system escapes from the trap nine times faster than bi-A*, and 6.7 times faster than kA*, whereas ANA* failed to provide a path within the given time frame. The proposed system also consumes 58.9 and 20.4 times less memory than bi-A* and kA*, respectively.

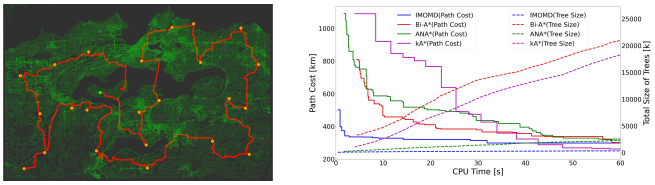

Fig. 7: Quantitative and qualitative results for an OSM of Seattle with 25 destinations. The proposed IMOMD-RRT* outperforms Bi-A*, ANA*, and kA* in term of speed and memory usage (the number of explored nodes).

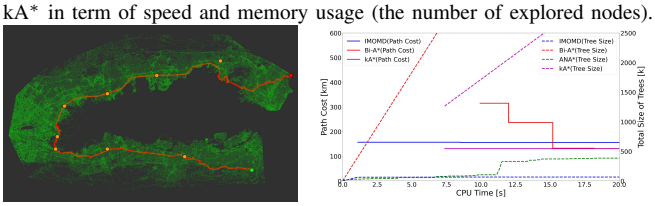

Fig. 8: Providing prior knowledge to the proposed IMOMD-RRT* system to avoid bug traps. The left and the right are the qualitative and quantitative results for a bug trap in San Francisco, respectively. Eight pseudo destinations help traverse the challenging topology (the source and target are separated by a body of water). Note: ANA* failed to provide a solution in the given time.

VI. Conclusion and Future Work

We presented an anytime iterative system on large-complex graphs to solve the multi-objective path planning problem, to decide the visiting order of the objectives, and to incorporate prior knowledge of the potential path. The system contains an anytime informable multi-objective and multi-directional RRT* to connect the destinations to form a connected graph and the ECI-Gen solver to determine the visiting order (via a relaxed Traveling Salesman Problem) in polynomial time.

The system was extensively evaluated on OpenStreetMap (OSM), built for autonomous vehicles and robots in practice. In particular, the system solved a path planning problem and the visiting order with 25 destinations (25! possible combinations of visiting orders) on an OSM of Seattle, containing more than a million nodes and edges, in 0.44 seconds. In addition, we demonstrated the system is able to leverage a reference path (prior knowledge) to navigate challenging topology for robotics inspection or vehicle routing applications. All the evaluations show that our proposed method outperforms Bi-A*, ANA*, and kA* in terms of speed and memory usage.

While we showed that the overall system is significantly faster than other approaches, a thorough evaluation of the proposed ECI-Gen R-TSP solver is ongoing work. In the future, we shall use the developed system within autonomy systems [32], [54]–[65] on a robot to perform point-to-point tomometric navigation in graph-based maps while locally avoiding obstacles and uneven terrain. It would also be interesting to deploy the system with multi-layered graphs and maps [66]–[68] to incorporate different types of information.

Acknowledgment

Toyota Research Institute provided funds to support this work. Funding for J. Grizzle was in part provided by NSF Award No. 2118818. First author thanks Wonhui Kim for useful conversations.

REFERENCES

[1] J. Faigl and G. A. Hollinger, "Unifying multi-goal path planning for autonomous data collection," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 2937–2942.

[2] H. Hu, K. Xiong, G. Qu, Q. Ni, P. Fan, and K. B. Letaief, "Aoi-minimal trajectory planning and data collection in uav-assisted wireless powered iot networks," *IEEE Internet of Things Journal*, vol. 8, no. 2, pp. 1211–1223, 2020.

[3] M. Samir, S. Sharafeddine, C. M. Assi, T. M. Nguyen, and A. Ghrayeb, "Uav trajectory planning for data collection from time-constrained iot devices," *IEEE Transactions on Wireless Communications*, vol. 19, no. 1, pp. 34–46, 2019.

[4] M. Jünger, G. Reinelt, and G. Rinaldi, "The traveling salesman problem," *Handbooks in operations research and management science*, vol. 7, pp. 225–330, 1995.

[5] D. L. Applegate, R. E. Bixby, V. Chvátal, and W. J. Cook, *The traveling salesman problem*. Princeton university press, 2011.

[6] C. Ma, R. He, and W. Zhang, "Path optimization of taxi carpooling," *PLoS One*, vol. 13, no. 8, p. e0203221, 2018.

[7] H. Huang, D. Bucher, J. Kissling, R. Weibel, and M. Raubal, "Multimodal route planning with public transport and carpooling," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 9, pp. 3513–3525, 2018.

[8] A. O. Al-Abbasi, A. Ghosh, and V. Aggarwal, "Deeppool: Distributed model-free algorithm for ride-sharing using deep reinforcement learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 12, pp. 4714–4727, 2019.

[9] S. Hulagu and H. B. Celikoglu, "An electric vehicle routing problem with intermediate nodes for shuttle fleets," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[10] OpenStreetMap contributors, "Planet dump retrieved from https://planet.osm.org ," https://www.openstreetmap.org, 2017.

[11] J. Janos, V. Vonásek, and R. Penicka, "Multi-goal path planning using multiple random trees," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 4201–4208, 2021.

[12] D. Devaurs, T. Siméon, and J. Cortés, "A multi-tree extension of the transition-based rrt: Application to ordering-and-pathfinding problems in continuous cost spaces," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 2991–2996.

[13] V. Vonásek and R. Penicka, "Space-filling forest for multi-goal path planning," in *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 2019, pp. 1587–1590.

[14] B. Englot and F. Hover, "Multi-goal feasible path planning using ant colony optimization," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 2255–2260.

[15] D. J. Rosenkrantz, R. E. Stearns, and P. M. Lewis, II, "An analysis of several heuristics for the traveling salesman problem," *SIAM journal on computing*, vol. 6, no. 3, pp. 563–581, 1977.

[16] J.-Y. Potvin, "Genetic algorithms for the traveling salesman problem," *Annals of Operations Research*, vol. 63, no. 3, pp. 337–370, 1996.

[17] C. Moon, J. Kim, G. Choi, and Y. Seo, "An efficient genetic algorithm for the traveling salesman problem with precedence constraints," *European Journal of Operational Research*, vol. 140, no. 3, pp. 606–617, 2002.

[18] H. Braun, "On solving travelling salesman problems by genetic algorithms," in *International Conference on Parallel Problem Solving from Nature*. Springer, 1990, pp. 129–133.

[19] Z. H. Ahmed, "Genetic algorithm for the traveling salesman problem using sequential constructive crossover operator," *International Journal of Biometrics & Bioinformatics (IJBB)*, vol. 3, no. 6, p. 96, 2010.

[20] A. V. Goldberg and C. Harrelson, "Computing the shortest path: A search meets graph theory." in *SODA*, vol. 5. Citeseer, 2005, pp. 156–165.

[21] J. Van Den Berg, R. Shah, A. Huang, and K. Goldberg, "Anytime nonparametric a," in *Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

[22] R. Stern, M. Goldenberg, A. Saffidine, and A. Felner, "Heuristic search for one-to-many shortest path queries," *Annals of Mathematics and Artificial Intelligence*, vol. 89, no. 12, pp. 1175–1214, 2021.

[23] P. Hart, N. 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. [Online]. Available: https://doi.org/10.1109/tssc.1968.300136

[24] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[25] M. Likhachev and A. Stentz, "R* search," *University of Pennsylvania*, 2008.

[26] M. Likhachev, G. J. Gordon, and S. Thrun, "Ara*: Anytime a* with provable bounds on sub-optimality," *Advances in neural information processing systems*, vol. 16, 2003.

[27] D. Harabor and A. Grastien, "The jps pathfinding system," in *International Symposium on Combinatorial Search*, vol. 3, no. 1, 2012.

[28] J. Roth, "Efficient many-to-many path planning and the traveling salesman problem on road networks," *International Journal of Knowledge-based and Intelligent Engineering Systems*, vol. 20, no. 3, pp. 135–148, 2016.

[29] S. M. LaValle *et al.*, "Rapidly-exploring random trees: A new tool for path planning," *Computer Science Dept., Iowa State University*, 1998.

[30] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, 2011.

[31] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the rrt*," in *Proc. IEEE Int. Conf. Robot. and Automation*, 2011, pp. 1478–1483.

[32] J.-K. Huang and J. W. Grizzle, "Efficient Anytime CLF Reactive Planning System for a Bipedal Robot on Undulating Terrain," *arXiv preprint arXiv:2108.06699*, 2021.

[33] S. Morgan and M. S. Branicky, "Sampling-based planning for discrete spaces," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 2. IEEE, 2004, pp. 1938–1945.

[34] M. S. Branicky, M. M. Curtiss, J. A. Levine, and S. B. Morgan, "Rrts for nonlinear, discrete, and hybrid planning and control," in *42nd IEEE International Conference on Decision and Control (IEEE Cat. No. 03CH37475)*, vol. 1. IEEE, 2003, pp. 657–663.

[35] K. Solovey, O. Salzman, and D. Halperin, "Finding a needle in an exponential haystack: Discrete rrt for exploration of implicit roadmaps in multi-robot motion planning," *The International Journal of Robotics Research*, vol. 35, no. 5, pp. 501–513, 2016.

[36] R. Shome, K. Solovey, A. Dobson, D. Halperin, and K. E. Bekris, "drrt*: Scalable and informed asymptotically-optimal multi-robot motion planning," *Autonomous Robots*, vol. 44, no. 3, pp. 443–467, 2020.

[37] S.-C. Huang, M.-K. Jiau, and Y.-P. Liu, "An ant path-oriented carpooling allocation approach to optimize the carpool service problem with time windows," *IEEE Systems Journal*, vol. 13, no. 1, pp. 994–1005, 2018.

[38] Y. Duan, T. Mosharraf, J. Wu, and H. Zheng, "Optimizing carpool scheduling algorithm through partition merging," in *2018 IEEE International Conference on Communications (ICC)*. IEEE, 2018, pp. 1–6.

[39] M. Tamannaei and I. Irandoost, "Carpooling problem: A new mathematical model, branch-and-bound, and heuristic beam search algorithm," *Journal of Intelligent Transportation Systems*, vol. 23, no. 3, pp. 203–215, 2019.

[40] H. K. Suman and N. B. Bolia, "Improvement in direct bus services through route planning," *Transport Policy*, vol. 81, pp. 263–274, 2019.

[41] Y. Lyu, C.-Y. Chow, V. C. Lee, J. K. Ng, Y. Li, and J. Zeng, "Cb-planner: A bus line planning framework for customized bus systems," *Transportation Research Part C: Emerging Technologies*, vol. 101, pp. 233–253, 2019.

[42] M. D. Simoni, E. Kutanoglu, and C. G. Claudel, "Optimization and analysis of a robot-assisted last mile delivery system," *Transportation Research Part E: Logistics and Transportation Review*, vol. 142, p. 102049, 2020.

[43] S. Naccache, J.-F. Côté, and L. C. Coelho, "The multi-pickup and delivery problem with time windows," *European Journal of Operational Research*, vol. 269, no. 1, pp. 353–362, 2018.

[44] M. Nazari, A. Oroojlooy, L. Snyder, and M. Takác, "Reinforcement learning for solving the vehicle routing problem," *Advances in neural information processing systems*, vol. 31, 2018.

[45] E. H.-C. Lu and Y.-W. Yang, "A hybrid route planning approach for logistics with pickup and delivery," *Expert Systems with Applications*, vol. 118, pp. 482–492, 2019.

[46] J. A. Bondy, U. S. R. Murty, *et al.*, *Graph theory with applications*. Macmillan London, 1976, vol. 290.

[47] A. P. Punnen, "The traveling salesman problem: Applications, formulations and variations," in *The traveling salesman problem and its variations*. Springer, 2007, pp. 1–28.

[48] D. B. West *et al.*, *Introduction to graph theory*. Prentice hall Upper Saddle River, 2001, vol. 2.
[49] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT press, 2022.
[50] Z. Galil and G. F. Italiano, "Data structures and algorithms for disjoint set union problems," *ACM Computing Surveys (CSUR)*, vol. 23, no. 3, pp. 319–344, 1991.
[51] J.-K. Huang, Y. Tan, D. Lee, V. R. Desaraju, and J. W. Grizzle, "Informable multi-objective and multi-directional rrt* system for robot path planning," *arXiv preprint arXiv:2205.14853*, 2022.
[52] G. Van Brummelen, "Heavenly mathematics," in *Heavenly Mathematics*. Princeton University Press, 2012.
[53] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006, available at http://planning.cs.uiuc.edu/.
[54] J. Rehder, J. Nikolic, T. Schneider, T. Hinzmann, and R. Siegwart, "Extending kalibr: Calibrating the extrinsics of multiple imus and of individual axes," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 4304–4311.
[55] P. Furgale, J. Rehder, and R. Siegwart, "Unified temporal and spatial calibration for multi-sensor systems," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 1280–1286.
[56] L. Oth, P. Furgale, L. Kneip, and R. Siegwart, "Rolling shutter camera calibration," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 1360–1367.
[57] J. Huang and J. W. Grizzle, "Improvements to Target-Based 3D LiDAR to Camera Calibration," *IEEE Access*, vol. 8, pp. 134 101–134 110, 2020.
[58] J. K. Huang, S. Wang, M. Ghaffari, and J. W. Grizzle, "LiDARTag: A Real-Time Fiducial Tag System for Point Clouds," *IEEE Robotics and Automation Letters*, pp. 1–1, 2021.
[59] J.-K. Huang, W. Clark, and J. W. Grizzle, "Optimal target shape for lidar pose estimation," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1238–1245, 2021.
[60] J.-K. Huang, C. Feng, M. Achar, M. Ghaffari, and J. W. Grizzle, "Global Unifying Intrinsic Calibration for Spinning and Solid-State LiDARs," *arXiv preprint arXiv:2012.03321*, 2020.
[61] R. Hartley, M. G. Jadidi, J. Grizzle, and R. M. Eustice, "Contact-aided invariant extended Kalman filtering for legged robot state estimation," in *Proc. Robot.: Sci. Syst. Conf.*, Pittsburgh, Pennsylvania, June 2018.
[62] R. Hartley, M. Ghaffari, R. M. Eustice, and J. W. Grizzle, "Contact-aided invariant extended kalman filtering for robot state estimation," *The International Journal of Robotics Research*, vol. 39, no. 4, pp. 402–430, 2020.
[63] Y. Gong and J. Grizzle, "Zero dynamics, pendulum models, and angular momentum in feedback control of bipedal locomotion," 2021.
[64] ——, "Angular momentum about the contact point for control of bipedal locomotion: Validation in a lip-based controller," *arXiv preprint arXiv:2008.10763*, 2020.
[65] Y. Gong, R. Hartley, X. Da, A. Hereid, O. Harib, J.-K. Huang, and J. Grizzle, "Feedback control of a cassie bipedal robot: Walking, standing, and riding a segway," in *2019 American Control Conference (ACC)*. IEEE, 2019, pp. 4559–4566.
[66] P. Fankhauser, M. Bloesch, and M. Hutter, "Probabilistic terrain mapping for mobile robots with uncertain localization," *IEEE Robotics and Automation Letters (RA-L)*, vol. 3, no. 4, pp. 3019–3026, 2018.
[67] P. Fankhauser, M. Bloesch, C. Gehring, M. Hutter, and R. Siegwart, "Robot-centric elevation mapping with uncertainty estimates," in *International Conference on Climbing and Walking Robots (CLAWAR)*, 2014.
[68] L. Gan, R. Zhang, J. W. Grizzle, R. M. Eustice, and M. Ghaffari, "Bayesian spatial kernel smoothing for scalable dense semantic mapping," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 790–797, April 2020.