Open access • Journal Article • DOI:10.1109/LCSYS.2019.2922166

An Admissible Heuristic to Improve Convergence in Kinodynamic Planners Using Motion Primitives — Source link

Basak Sakcak, Luca Bascetta, Gianni Ferretti, Maria Prandini

Institutions: Polytechnic University of Milan

Published on: 01 Jan 2020 - IEEE Control Systems Letters

Topics: Admissible heuristic and Heuristic

An Admissible Heuristic to Improve Convergence in Kinodynamic Planners Using Motion Primitives

Basak Sakcak, Luca Bascetta, Gianni Ferretti, and Maria Prandini

***Abstract*—This paper introduces a new heuristic function that can be incorporated in any kinodynamic planner using motion primitives, to the purpose of increasing its convergence rate. The heuristic function is proven to be admissible and, hence, the optimality properties of the planning algorithm are preserved. Notably, it can be applied to planning problems with generic agent motion models and cost criteria, since it depends only on the database of motion primitives. The proposed heuristic has been integrated into a randomized sampling-based and a deterministic kinodynamic planner, and its effectiveness has been shown in numerical examples with different agent motion models and cost criteria.**

***Index Terms*—Kinodynamic motion planning, motion primitives, admissible heuristic function, mobile robots**

I. Introduction

KINODYNAMIC motion planning, first introduced in [1], refers to planning with differential constraints, i.e., determining a collision-free, possibly optimal, trajectory that drives an agent from an initial position to a specified target region, while satisfying constraints on the derivative of the agent configuration that represent its motion model. In this way, the planned trajectory is guaranteed to be feasible but the planning problem becomes far more complex. Note that computing an optimal trajectory subject to kinodynamic constraints can be difficult, especially when considering arbitrary motion models.

Search-based planners (e.g., [2]–[4]) address this issue by using a graph-representation of the state space where discrete states are connected with pre-computed optimal trajectories – the *motion primitives* – and then applying an optimal graph-search. Optimal graph-search is a computationally intense operation and both time and memory requirements grow drastically in search-based approaches when increasing the dimensionality or the number of adopted motion primitives.

Sampling-based planners determine a solution by randomly sampling the continuous state space of the agent and creating a graph, where nodes are sampled states and edges are feasible trajectories connecting nodes generated by a steering function. A class of these algorithms can guarantee *asymptotic optimality* such that the probability of finding an optimal solution converges to 1 as the tree cardinality grows to infinity. Kinodynamic-RRT* [5] and variants, can only be applied to a limited class of systems, since asymptotic optimality requires that two nodes are connected exactly and optimally by a steering function. In order to consider arbitrary dynamics, approaches such as sampling the control space [6] and iteratively running a feasible kinodynamic planning algorithm [7] have been introduced. More recently, motion primitives have been adopted also for sampling-based planners, inspired by the search-based approach [8].

The inclusion of an appropriate heuristic within graph-search is typically used to improve computational performance. This is indeed crucial for search-based planners to be able to consider a larger number of motion primitives. To this aim, suitable heuristic based algorithms, such as A* [9] and variants (e.g. ARA* [10]), have to be used. These algorithms can return a resolution-optimal solution (i.e., an optimal solution among the ones that can be obtained with the given set of motion primitives) while expanding a reduced number of nodes if the adopted heuristic is *admissible*, i.e., if it always underestimates the actual cost to reach the goal. Though it is not so crucial as in search-based planning, also sampling-based planners can significantly improve their convergence while preserving their optimality properties, by incorporating an admissible heuristic. Some recent works such as Informed-RRT* [11], BIT* [12], C-FOREST [13] use the length of the shortest path discovered to analytically compute a reduced sampling set and increase the convergence rate. Exploiting the same idea, Anytime RRT [14] improves the solution at each run of the algorithm but it lacks a theoretical guarantee on convergence to the optimal solution. Furthermore, RRT^X [15] proposes a method for repairing the promising parts of the tree. Similarly, assuming that an admissible heuristic exists, RRT$^\#$ [16] ensures that the tree contains the best possible branch, in terms of a sequence of edges, with the current set of nodes.

All the mentioned heuristic based algorithms assume that an admissible heuristic exists. Indeed, most of them solve the *shortest path planning problem* where the path length can be computed using its projection on a Euclidean position space and the Euclidean distance is naturally an admissible heuristic. In [17], the authors propose a heuristic look-up table which carries the shortest path information from each cell of a discretized search space to another. Their approach better approximates the cost to goal for systems with differential constraints, however the size of such a database tends to become intractable as the size of the search space increases. Note that, in principle, these results can be extended to different cost criteria. However, this requires to define an admissible heuristic which is often as challenging as solving the original problem. Therefore, computing a generic heuristic function still remains an open issue, which is addressed here.

This paper proposes a *heuristic for a generic mobile robot motion planner that is based on motion primitives*. This heuristic can be incorporated into any motion planning algorithm that relies on a graph built by concatenating a set of motion primitives, and it is shown to be admissible. It depends only on

The authors are with Politecnico di Milano, Dipartimento di Elettronica, Informazione e Bioingegneria, Piazza L. Da Vinci 32, 20133, Milano, Italy. email: {basak.sakcak, luca.bascetta, gianni.ferretti, maria.prandini}@polimi.it

the database of motion primitives and their associated costs, and can be computed using the same strategy, independently of the system dynamics, representing the agent motion model, and the planning cost criterion. The heuristic is incorporated in a sampling-based and a search-based planner using motion primitives, `RRT*_MotionPrimitives` (`MP-RRT*`) [8] and `A*` respectively, and resolution-optimality (asymptotic in the case of `MP-RRT*`) is proven. The effectiveness of the proposed heuristic in improving the convergence rate of these algorithms is demonstrated in simulation using two motion planning examples with different motion models and cost criteria.

II. Motion Planning Problem Statement

Consider an agent with motion model given by the following dynamical system, with state $\mathbf{s} \in \mathbb{R}^d$ and input $\mathbf{u} \in \mathbb{R}^m$

$$\dot{\mathbf{s}}(t) = f\left(\mathbf{s}(t), \mathbf{u}(t)\right), \qquad \mathbf{s}(0) = \mathbf{s}_{start}, \tag{1}$$

where f is continuously differentiable in both of its arguments and $\mathbf{s}_{start}$ stands for the initial state. As system (1) represents a motion model, its state $\mathbf{s}$ includes the agent position $\boldsymbol{\pi} \in \mathbb{R}^{d_p}$, $d_p \leq 3$, with respect to a given absolute reference frame: $\mathbf{s} = \left[\boldsymbol{\pi}^T, \dots\right]^T$. For example, if the agent moves on a planar surface, then $d_p = 2$ and $\boldsymbol{\pi}$ corresponds to the x and y coordinates of its position on the plane.

State and actuation constraints are also enforced, i.e., $\mathbf{s} \in S$ and $\mathbf{u} \in U$, where $S \subset \mathbb{R}^d$ and $U \subset \mathbb{R}^m$ are both compact sets. An open subset $S_{goal} \subset S$ represents the goal region the agent has to reach. Obstacles are represented by an open subset $S_{obs} \subset S$, and the free space is defined as $S_{free} := S \setminus S_{obs}$. We assume $\mathbf{s}_{start} \in S_{free}$ and $S_{goal} \subset S_{free}$.

Given $\mathbf{s}_0$ and $\mathbf{s}_f$ in S, an agent *trajectory* connecting $\mathbf{s}_0$ to $\mathbf{s}_f$ is a tuple $\mathbf{z} = (\mathbf{s}(\cdot), \mathbf{u}(\cdot), \tau)$, where τ is the duration, and $\mathbf{s}(\cdot) : [0, \tau] \to S$ satisfies the boundary conditions $\mathbf{s}(0) = \mathbf{s}_0$ and $\mathbf{s}(\tau) = \mathbf{s}_f$, and the differential equation (1) for $t \in [0, \tau]$ when the control input $\mathbf{u}(\cdot) : [0, \tau] \to U$ is applied. A trajectory $\mathbf{z} = (\mathbf{s}(\cdot), \mathbf{u}(\cdot), \tau)$ is said to be *collision free*, if $\mathbf{s}(t) \in S_{free}$, $t \in [0, \tau]$. A collision free trajectory connecting $\mathbf{s}_0$ to $\mathbf{s}_f$ is *optimal* if it minimizes the cost criterion

$$J(\mathbf{z}) = \int_0^\tau g\left(\mathbf{s}(t), \mathbf{u}(t)\right) \mathrm{d}t, \tag{2}$$

where $g : \mathbb{R}^d \times \mathbb{R}^m \to \mathbb{R}_{\geq 0}$ is an instantaneous cost function. We assume that optimal trajectories joining two different states have a non zero cost. Given the previous definitions, an *optimal solution of the kinodynamic motion planning problem* is an optimal collision free trajectory $\mathbf{z}^\star = (\mathbf{s}^\star(\cdot), \mathbf{u}^\star(\cdot), \tau^\star)$ connecting $\mathbf{s}_0 = \mathbf{s}_{start}$ to $\mathbf{s}_f \in S_{goal}$.

Planning using motion primitives, as in this paper, and the proposed heuristic function apply only to those systems that satisfy the *translation invariance property* with respect to the position component $\boldsymbol{\pi}$ of the state $\mathbf{s} = \left[\boldsymbol{\pi}^T, \dots\right]^T$. In order to state this property formally in Definition 1, we need to define the *translation vector* $\mathbf{r}_\delta = \left[\delta^T, \mathbf{0}_{1\times(d-d_p)}\right]^T$ associated with the displacement $\delta \in \mathbb{R}^{d_p}$: if we add $\mathbf{r}_\delta$ to the agent's state $\mathbf{s} = \left[\boldsymbol{\pi}^T, \dots\right]^T$, then its position $\boldsymbol{\pi}$ is translated of δ, while the remaining components of $\mathbf{s}$ are preserved.

Definition 1 (translation invariance property)**.** *A system with motion model* (1) *and cost criterion* (2) *satisfies the translation invariance property if when obstacles are neglected and an optimal trajectory* $\mathbf{z}^\star = (\mathbf{s}^\star(\cdot), \mathbf{u}^\star(\cdot), \tau^\star)$ *connecting* $\mathbf{s}_0$ *to* $\mathbf{s}_f$ *is considered, then, for every displacement* $\delta \in \mathbb{R}^{d_p}$*, an optimal trajectory connecting* $\mathbf{s}_0 + \mathbf{r}_\delta$ *to* $\mathbf{s}_f + \mathbf{r}_\delta$ *is given by* $(\mathbf{s}^\star(\cdot) + \mathbf{r}_\delta, \mathbf{u}^\star(\cdot), \tau^\star)$ *and has the same cost of* $\mathbf{z}^\star$*.*

It is worth mentioning that this definition holds for a large class of agent models common for mobile robots (terrestrial, underwater or aerial vehicles) and it has been extensively used in the literature (see e.g., [2]–[4], [8]).

III. Heuristic Function for Motion Primitive Based Planning

In this section, we introduce a heuristic function that is determined from a database of motion primitives and their respective costs, irrespectively of the underlying motion model and cost metric. Such a heuristic can be incorporated in any planning algorithm that relies on such a database to guide the search and the growth in a graph by favoring the nodes that can contribute to finding the optimal solution.

In the following, we briefly explain the concept of motion primitives and planning using motion primitives. Consequently, we introduce the proposed heuristic function.

A. Motion Primitives

In order to define the database of motion primitives, we need to fix a finite set of state values that are used as initial and final state pairs of the motion primitives. To this aim, we grid (apply a uniform discretization at each dimension of the state space) a box in the continuous state space set S, assuming, without loss of generality, that the resulting (finite) set of grid points includes states with position $\bar{\boldsymbol{\pi}}_0 = \mathbf{0}$. The motion primitives are then computed by neglecting obstacles and solving a constrained boundary value optimization problem (the problem defined in Section II) for each pair of initial and final states $(\bar{\mathbf{s}}_0^i, \bar{\mathbf{s}}_f^i)$, such that $\bar{\mathbf{s}}_0^i$ and $\bar{\mathbf{s}}_f^i$ belong to the selected set of grid points and $\bar{\mathbf{s}}_0^i = [\bar{\boldsymbol{\pi}}_0, \dots]$, $\forall i$.

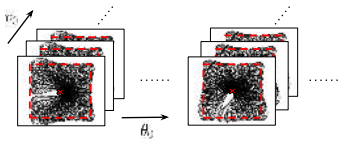


Fig. 1. A database of trajectories that share the same initial position $\bar{\boldsymbol{\pi}}_0 = \mathbf{0} \in \mathbb{R}^2$ (marked with red cross). In particular, only the trajectories that reach the frontier of the bounding box $BB(\bar{\boldsymbol{\pi}}_0)$ (represented using its projection on $\mathbb{R}^2$ with a red dashed line) are shown.

In the sequel, we shall denote as $BB(\boldsymbol{\pi})$, the *bounding box at* $\boldsymbol{\pi}$ given by the set of state values obtained by translating the grid points used for the database construction by $\mathbf{r}_{\boldsymbol{\pi}} = \left[\boldsymbol{\pi}^T, \mathbf{0}_{1\times(d-d_p)}\right]^T$. Hence, $BB(\bar{\boldsymbol{\pi}}_\mathbf{0})$ is the bounding box of the database. Figure 1 shows an illustrative example of a bounding box $BB(\bar{\boldsymbol{\pi}}_0)$ and the associated database of motion primitives for a 4 dimensional state space composed of position, $\boldsymbol{\pi}$, orientation, θ, and velocity v.

B. Motion Primitive Based Planning

The algorithms that use a database of motion primitives rely on a uniform discretization of the state space set S with the same grid size used for constructing $BB(\bar{\boldsymbol{\pi}}_0)$. Let S^{Δ} and $S^{\Delta}_{free} := S^{\Delta} \cap S_{free}$ define the set of grid points that represent the discretized state space and the free discrete state space, respectively. We assume that $\mathbf{s}_{start} \in S^{\Delta}_{free}$ and the set of goal states $S^{\Delta}_{goal} := V^{\Delta}_{free} \cap S_{goal}$ is nonempty. Thanks to the *translation invariance property*, we can define a graph $G^{\Delta}_{free} = (V^{\Delta}_{free}, E^{\Delta}_{free})$ whose nodes, V^{Δ}_{free}, correspond to the set of grid points in S^{Δ}_{free} that can be reached from the initial node, $\mathbf{s}_{start}$, by concatenating a sequence of edges in E^{Δ}_{free}, representing (translated) motion primitives that lie in S_{free}.

An *optimal branch* of G^{Δ}_{free} is composed of the ordered sequence of nodes, $SQ^{\star}_{\mathbf{s}_{start}\to\mathbf{s}_{goal}} := \{\mathbf{s}_{start}, \mathbf{s}^{\star}_1, \mathbf{s}^{\star}_2, \dots, \mathbf{s}_{goal}\}$, which represents a *resolution-optimal* Δ*-trajectory*, a minimum cost trajectory that starts at the initial state $\mathbf{s}_{start}$ and ends in the goal region ($\mathbf{s}_{goal} \in S^{\Delta}_{goal}$), and is obtained by concatenating motion primitives in the database. The *resolution-optimal* Δ*-cost* is the cost $c^{\star\Delta}$ of this optimal trajectory. A motion primitive based algorithm is called *resolution-optimal* if it provides a solution with cost equal to $c^{\star\Delta}$.

C. Database dependent heuristic

Consider a node $\mathbf{s}$ from which the agent can reach the goal region via some branch represented in G^{Δ}_{free}. Let $SQ^{\star}_{\mathbf{s}\to\mathbf{s}_{goal}} := \{\mathbf{s}, \dots, \mathbf{s}_{goal}\}$, where $\mathbf{s}_{goal} \in S^{\Delta}_{goal}$, be an optimal branch represented in G^{Δ}_{free} that connects the node $\mathbf{s}$ to S^{Δ}_{goal} and $c^{\star}_{\mathbf{s}\to\mathbf{s}_{goal}}$ the cost associated to that sequence.

If the optimal cost to reach the goal from node $\mathbf{s}$, i.e., $c^{\star}_{\mathbf{s}\to\mathbf{s}_{goal}}$, is available, optimal graph search algorithms such as A* can guarantee that the number of nodes that are expanded to obtain the optimal solution is minimal [9], thus substantially improving the search efficiency. Similarly, the same cost can also be incorporated in an incremental sampling-based algorithm to expand the tree only from promising nodes, or to prune the unpromising ones. However, computing $c^{\star}_{\mathbf{s}\to\mathbf{s}_{goal}}$ is an issue due to the combinatorial nature of the problem of finding for each node $\mathbf{s} \in V^{\Delta}_{free}$ the minimum cost trajectory represented in G^{Δ}_{free}. Therefore, many algorithms use an approximation of this cost, i.e., a heuristic, in order to improve the search efficiency. However, determining a heuristic function is problem dependent and can be an issue if the minimization objective and/or the agent motion model are not simple. By exploiting the fact that we are considering planners that use a database of motion primitives, we can propose a unified approach to compute a heuristic function, $h(\mathbf{s})$, that approximates the cost, $c^{\star}_{\mathbf{s}\to\mathbf{s}_{goal}}$, $\forall \mathbf{s} \in S^{\Delta}$, and can be computed using the same strategy independently of the problem specificities.

The heuristic, $h(\mathbf{s})$, is defined by exploiting the fact that all trajectories represented in G^{Δ}_{free} are built by concatenating motion primitives extracted from the database. Let us rewrite $SQ^{\star}_{\mathbf{s}\to\mathbf{s}_{goal}}$ as an ordered sequence of nodes starting from $\mathbf{s}_0 = [\boldsymbol{\pi}^T_0, \dots]^T = \mathbf{s}$ such that $SQ^{\star}_{\mathbf{s}\to\mathbf{s}_{goal}} := \{\mathbf{s}_0, \dots, \mathbf{s}_{k-1}, \mathbf{s}_k \dots, \mathbf{s}_{K_s}\}$, where $\mathbf{s}_{K_s} = \mathbf{s}_{goal}$. Any node $\mathbf{s}_k$ should lie within the bounding box of $\mathbf{s}_{k-1}$, i.e., $\mathbf{s}_k \in BB(\boldsymbol{\pi}_{k-1})$. Then the optimal branch belongs to a space comprised of union of K_s bounding boxes at $\boldsymbol{\pi}_k$, i.e., $\{BB(\boldsymbol{\pi}_k)\}$, such that $\forall k = 0, \dots, K_s - 1, \mathbf{s}_k \in SQ^{\star}_{\mathbf{s}\to\mathbf{s}_{goal}}$. Thus, we can approximate the cost of $SQ^{\star}_{\mathbf{s}\to\mathbf{s}_{goal}}$ by computing a lower bound on the number of bounding boxes and on the cost of making a connection within each box. More precisely, the heuristic function $h(\mathbf{s})$ is computed as follows.

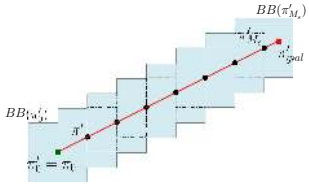


Fig. 2. Procedure to compute the minimum number of bounding boxes: tile the straight line between $\overline{\boldsymbol{\pi}_0\, \boldsymbol{\pi}'_{goal}}$ using the projection of the bounding box on the position space (here shown as $\mathbb{R}^2$).

We can neglect the feasibility of the solution, and compute a lower bound, M_s+1, on the number of boxes that connect $\mathbf{s}$ to $\mathbf{s}_{goal}$, such that $M_s+1 \leq K_s$, as follows. We first tile a straight line $\overline{\mathbf{s}\, \mathbf{s}'_{goal}}$ that connects $\mathbf{s}$ to $\mathbf{s}'_{goal}$ (the node in S^{Δ}_{goal} closest to $\mathbf{s}$ in terms of Euclidean distance with respect to $\boldsymbol{\pi}$), and then we build a sequence of bounding boxes $\{BB(\boldsymbol{\pi}'_m)\}^{M_s}_{m=0}$ such that $\boldsymbol{\pi}'_0 = \boldsymbol{\pi}_0$, $\boldsymbol{\pi}'_m$ corresponds to the intersection of $\overline{\mathbf{s}\, \mathbf{s}'_{goal}}$ and $BB(\boldsymbol{\pi}'_{m-1})$, $m = 1, \dots, M_s-1$, and $BB(\boldsymbol{\pi}'_{M_s})$ intersects with the goal region. As better illustrated in Figure 2, this corresponds to tiling the straight line between $\boldsymbol{\pi}_0$ and $\boldsymbol{\pi}'_{goal}$, i.e., $\overline{\boldsymbol{\pi}_0\, \boldsymbol{\pi}'_{goal}}$, with the projection of $BB(.)$ on the position space $\mathbb{R}^{d_p}$. Note that, the straight line is used to compute a lower bound on the number of boxes and it typically does not correspond to either a feasible or an optimal solution.

Let c_{min} be the minimum among the costs of all the trajectories that reach the frontier of $BB(\bar{\boldsymbol{\pi}}_0)$ (Figure 1). Hence, c_{min} corresponds to the cost of the minimum cost trajectory among the trajectories that start from any initial state and have the final state on the box frontier. Thanks to the translation invariance property, the cost of making a connection within each box $BB(\boldsymbol{\pi}'_m)$ can be lower bounded using the bounding box of the database, $BB(\bar{\boldsymbol{\pi}}_0)$. In particular, c_{min} can be used as a lower bound on the cost of reaching the frontier of $BB(\boldsymbol{\pi}'_m)$, $m = 0, \dots, M_s - 1$. Since it cannot be guaranteed that $\mathbf{s}'_{goal}$ lies on the frontier of $BB(\boldsymbol{\pi}'_{M_s})$, the cost of the last segment of $\overline{\mathbf{s}\, \mathbf{s}'_{goal}}$ is lower bounded with 0. If we then set

$$h(\mathbf{s}) = M_s\, c_{min}, \tag{3}$$

we have that $h(\mathbf{s}) = M_s\, c_{min} \leq c^{\star}_{\mathbf{s}\to\mathbf{s}_{goal}}$, which proves that the following proposition holds.

Proposition 1. *The heuristic function* $h(\mathbf{s})$ *defined in* (3) *is admissible.*

This is a key property since optimality of the solution returned by a planning algorithm that incorporates a heuristic function is guaranteed only if the heuristic is admissible.

IV. Heuristic Guided Motion Planning

In order to show its general applicability, this section describes two classes of planning methods that use motion primitives and that can benefit from the proposed heuristic function. The first approach applies a deterministic graph-search (A*) on the state lattice formed by the regular arrangement of motion primitives, i.e., $G^\Delta_{free} = (V^\Delta_{free}, E^\Delta_{free})$, the second one (MP-RRT*) randomly samples the discrete state space while building a tree of motion primitives.

Before briefly describing the two methods, we introduce some common notation. Both algorithms build a tree, $T = (V, E)$, whose nodes, $\mathbf{s} \in V$, are the states of the dynamic system (1) and edges, $e \in E$, are optimal trajectories according to (2), such that $V \subset V^\Delta_{free}$ and $E \subset E^\Delta_{free}$. Every node $\mathbf{s} \in V$ is connected to $\mathbf{s}_{start}$ via a single sequence of intermediate nodes $\mathbf{s}_j \in V$, $j = 1, \ldots, n-1$, $n \leq N_P$, where N_P is the tree cardinality, and associated edges $e_j = e_{\mathbf{s}_j, \mathbf{s}_{j+1}} \in E$, $j = 0, 1, \ldots, n-1$, with $\mathbf{s}_n = \mathbf{s}$ and $\mathbf{s}_0 = \mathbf{s}_{start}$. We can then associate to $\mathbf{s}_n$ a cost $C(\rightarrow \mathbf{s}_n) = \sum_{j=0}^{n-1} C(e_j)$, where $C(e_j)$ denotes the cost associated with edge $e_j \in E$. Obviously, $C(\rightarrow \mathbf{s}_{start}) = 0$.

A. Graph-Search Using A*

A* [9] is a widely used graph-search algorithm to address the problem of finding the minimum-cost branch of a graph connecting the source node, $\mathbf{s}_{start}$, to a goal node, $\mathbf{s}_{goal}$. It builds on Dijkstra's algorithm [18] with the inclusion of a heuristic function in order to guide the search.

Dijkstra's algorithm maintains two lists of nodes: OPEN and CLOSED. OPEN initially contains only $\mathbf{s}_{start}$ and the CLOSED list is empty. At each iteration, the algorithm selects the node $\mathbf{s}$ in OPEN with the minimum cost $C(\rightarrow \mathbf{s})$ and evaluates its successors, $\mathbf{s}$ is then moved to the CLOSED list. For each successor, the algorithm selects from three actions, which are: (1) if the successor, $\mathbf{s}_{succ}$, belongs to CLOSED it is ignored, (2) if the successor, belongs to OPEN and the cost of reaching it through $\mathbf{s}$ is smaller than its current cost, $C(\rightarrow \mathbf{s}_{succ})$, then the parent of $\mathbf{s}_{succ}$ is updated as $\mathbf{s}$, (3) if it is the first time that $\mathbf{s}_{succ}$ is considered, then it is added to OPEN. The algorithm stops when $\mathbf{s}_{goal}$ is selected from OPEN (returning the branch $SQ^\star_{\mathbf{s}_{start} \rightarrow \mathbf{s}_{goal}}$) or when the list OPEN is empty, i.e., the graph does not contain a branch that reaches $\mathbf{s}_{goal}$.

A* algorithm uses a heuristic function to order the node selection from OPEN. To this end, at each iteration, it selects the node $\mathbf{s}$ with the minimum cost $p(\mathbf{s}) = C(\rightarrow \mathbf{s}) + h(\mathbf{s})$ and evaluates its successors. Thus, Dijkstra's algorithm can be seen as a specific case of A* where $h(\mathbf{s})$ is set to zero. From Proposition 1 and from the theorems on the admissibility (Theorem 1 in [9]) and optimality[1] (Theorem 8 in [19]) of A* the following holds.

Proposition 2. *A* algorithm using the heuristic function defined in* (3) *returns the resolution-optimal solution, $SQ^\star_{\mathbf{s}_{start} \rightarrow \mathbf{s}_{goal}}$, after expanding a minimal[2] number of nodes.*

B. Sampling-Based MP-RRT* Algorithm

In this section, we integrate the proposed heuristic within the sampling-based MP-RRT* algorithm [8], a variant of RRT*, which relies on a discretized state space and a database of motion primitives. We start by recalling the basic procedures in the standard MP-RRT* algorithm.

The MP-RRT* algorithm builds a tree by randomly sampling the set S^Δ_{free}. Each randomly selected state, $\mathbf{s}$, is added to the tree by selecting the best parent, $\mathbf{s}_{best} \in V$, which minimizes the cost $C(\rightarrow \mathbf{s})$, such that a collision-free motion primitive connecting the two states exists. At each iteration, the algorithm also rewires the tree within the local neighborhood of $\mathbf{s}$, i.e., for each neighbor node of $\mathbf{s}$, $\mathbf{s}_{near}$, the edge connecting it to its parent is eliminated and a new edge connecting it to $\mathbf{s}$ is added if that reduces the cost $C(\rightarrow \mathbf{s}_{near})$.

We include the proposed heuristic into MP-RRT* by means of a branch and bound technique. Let $V^i \subset V^\Delta_{free}$ denote the nodes of the tree at i-th iteration of the algorithm, correspondingly c_i is the minimum cost among the costs of all the branches of the tree that reach the goal region. If for a node $\mathbf{s} \in V^i$

$$C(\rightarrow \mathbf{s}) + c^\star_{\mathbf{s} \rightarrow \mathbf{s}_{goal}} > c_i \tag{4}$$

then, any branch that contains $\mathbf{s}$ will not contribute to a solution with a cost lower than c_i. Therefore, such a node can be defined as not expandable such that the algorithm will not consider $\mathbf{s}$ as a potential parent while adding a new node to the tree. Due to the complexity of computing the cost $c^\star_{\mathbf{s} \rightarrow \mathbf{s}_{goal}}$, we use the heuristic function $h(\mathbf{s})$ given in (3) and define the set of expandable nodes V^i_{expand} at each iteration i as follows

$$V^i_{expand} = \{\mathbf{s} \in V^i \mid C(\rightarrow \mathbf{s}) + h(\mathbf{s}) \leq c_i\} \tag{5}$$

Since $h(\mathbf{s})$ is admissible, then, the proposed definition of V^i_{expand} in (5) is not eliminating any node $\mathbf{s} \in V$ that could contribute to finding a solution with a cost lower than c_i.

In [8] we showed that, as the number of iterations goes to infinity, MP-RRT* returns the *resolution-optimal solution*, $SQ^\star_{\mathbf{s}_{start} \rightarrow \mathbf{s}_{goal}}$, associated with the *resolution-optimal Δ-cost* $c^{\star\Delta}$, with probability 1. In the following, we also show that this guarantee is not hampered by using the proposed heuristic guided approach.

Proposition 3. *As the number of iterations goes to infinity, the cost of the trajectory returned by the heuristic guided* MP-RRT**, incorporating the heuristic function defined in* (3)*, converges to the resolution-optimal Δ-cost $c^{\star\Delta}$ with a probability equal to 1.*

Proof. The proposed heuristic guided MP-RRT* returns the resolution-optimal solution once it discovers an optimal sequence $SQ^\star_{\mathbf{s}_{start} \rightarrow \mathbf{s}_{goal}} := \{\mathbf{s}_{start}, \mathbf{s}^\star_1, \mathbf{s}^\star_2, \ldots, \mathbf{s}_{goal}\}$.

[1] Note that, this requires $h(\mathbf{s})$ to be consistent, such that the cost $p(\mathbf{s}_{succ})$ of a successor node should always be higher than its parent $\mathbf{s}$, i.e., $p(\mathbf{s}_{succ}) \geq p(\mathbf{s})$. The proposed heuristic is not consistent in general, however it can be made consistent by applying the *pathmax* equation, i.e., setting $p(\mathbf{s}_{succ}) \leftarrow \max\{p(\mathbf{s}), C(\rightarrow \mathbf{s}) + h(\mathbf{s}_{succ})\}$.

[2] Compared with other informed best-first search algorithms adopting the same heuristic and the tie-braking rule.

Assume that $\{\mathbf{s}_{start}, \mathbf{s}^\star_1, \ldots, \mathbf{s}^\star_{j-1}\}$ is a branch in the tree, and that $\mathbf{s}^\star_j$ is sampled at iteration i. Then, it is added to the tree if $\mathbf{s}^\star_{j-1}$ is expandable, i.e., $\mathbf{s}^\star_{j-1} \in V^i_{expand}$ in (5).

Note that the resolution-optimal Δ-cost can be written as

$$c^{\star\Delta} = C(\rightarrow \mathbf{s}^\star_{j-1}) + c^\star_{\mathbf{s}^\star_{j-1}\rightarrow\mathbf{s}_{goal}},$$

where $c^{\star\Delta}$ is the minimum cost that can be obtained given a particular discretization, and satisfies $c^{\star\Delta} \leq c_i$. Then,

$$C(\rightarrow \mathbf{s}^\star_{j-1}) + c^\star_{\mathbf{s}^\star_{j-1}\rightarrow\mathbf{s}_{goal}} \leq c_i. \tag{6}$$

From Proposition 1 it follows that $h(\mathbf{s}^\star_{j-1})$ satisfies $h(\mathbf{s}^\star_{j-1}) \leq c^\star_{\mathbf{s}^\star_{j-1}\rightarrow\mathbf{s}_{goal}}$. Rewriting (6) in view of this relation, we obtain $C(\rightarrow \mathbf{s}^\star_{j-1}) + h(\mathbf{s}^\star_{j-1}) \leq c_i$, which states that $\mathbf{s}^\star_{j-1}$ is in V^i_{expand}, therefore once $\mathbf{s}^\star_j$ is sampled it is added to the tree. Then, as the number of iterations goes to infinity the optimal sequence will be discovered with probability 1 [8]. □

V. Numerical Examples

We present two numerical examples to show the effectiveness of the proposed heuristic included in MP-RRT* and A* algorithms, using different cost metrics and motion models.

A. Minimum-Time Minimum-Energy

A unicycle like robot with a 4D state space (x, y, θ, v) moving on a planar surface is considered, with motion model

$$\begin{aligned} \dot{x}(t) &= v(t)\cos\theta(t) \qquad & \dot{\theta}(t) &= \omega(t) \\ \dot{y}(t) &= v(t)\sin\theta(t) \qquad & \dot{v}(t) &= a(t) \end{aligned} \tag{7}$$

where (x, y) is the position of the robot and θ its orientation, v and ω are the linear and angular velocity with respect to a global reference frame, and a is the linear acceleration. The control input is represented by $\mathbf{u} = [\omega, a]^T$.

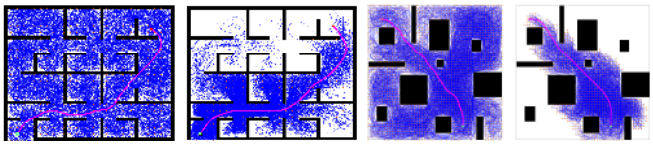
The 293,481 motion primitives in the database are computed for each pair of initial and final states, $\mathbf{s}_0 = [x_0, y_0, \theta_0, v_0]$ and $\mathbf{s}_f = [x_f, y_f, \theta_f, v_f]$, solving the TPBVP for the differential constraints given in (7) and the cost function $J(\mathbf{u}, \tau) = \int_0^\tau \left[1 + \mathbf{u}(t)^T R \mathbf{u}(t)\right] \mathrm{d}t$ that minimizes the total time of the trajectory τ, penalizing the total actuation effort with a weight $R = 0.5 I_2$. The control variables a and ω are bounded as $a \in [-3, 3]$ m/s^2, $\omega \in [-5, 5]$ rad/s.

The initial position for all trajectories in the database is always $(x_0, y_0) = (0, 0)$, the final positions, instead, are selected from a square grid where $(x_f, y_f) \in [-2, 2] \times [-2, 2] \setminus (0, 0)$, and each square cell has a size of half a meter. The initial and final orientations can take 24 equally spaced values in the range $[0, 2\pi)$ rad. The initial and final velocities are selected among 5 equally spaced values in the range $[0, 4]$ m/s.

B. Passenger Comfort

We consider a car-like vehicle with motion model

$$\begin{aligned} \dot{x}(t) &= v(t)\cos\theta(t) \qquad & \dot{\theta}(t) &= \frac{v(t)}{l}\tan\phi(t) \\ \dot{y}(t) &= v(t)\sin\theta(t) \qquad & \dot{v}(t) &= a(t) \end{aligned} \tag{8}$$



(a) (b) (c) (d)

Fig. 3. Trees corresponding to the best solution (projection on $\mathbb{R}^2$) obtained for the minimum-time minimum-energy (a-b) and passenger comfort (c-d) problems using MP-RRT* (a,c), and heuristic guided MP-RRT* (b,d).

where (x, y) is the car position, θ its orientation, v and a are the linear velocity and acceleration, ϕ is the steering angle, and l is the vehicle length here assumed to be equal to $1\,m$. The ISO 2631-1 standard relates passenger comfort with the overall r.m.s acceleration $a_w = 1.4\sqrt{a^2 + v^2\,\dot{\theta}^2}$. In particular, $a_w > 0.8\,m/s^2$ is perceived to be uncomfortable by passengers. In order to maximize passenger comfort, 29,816 motion primitives are generated according to the following objective function $J(a_w, \tau) = \int_0^\tau \left(1 + a_w^2(t)\right) \mathrm{d}t$ which minimizes the total time of the trajectory while penalizing the r.m.s acceleration acting on the human body. Furthermore, the linear velocity is bounded as $v \in [0, 4]\,m/s$, and the overall acceleration a_w is bounded as $a_w \in [0, 0.8]\,m/s^2$, in order to ensure that none of the primitives exceeds the comfort zone. The primitives built in this case share the same characteristics as the ones described in Section V-A, except for the final positions which are selected on a grid characterized by $(x_f, y_f) \in [-3, 3] \times [-3, 3] \setminus (0, 0)$, where each cell has a size of one meter.

C. Results

Simulations are performed on an IntelCore i7@2.40 GHz personal computer with 8Gb RAM and all the algorithms are implemented in MATLAB without any code optimization.

The proposed heuristic has been first tested using the sampling-based approach MP-RRT* described in Section IV-B. For that purpose we considered the standard MP-RRT* algorithm and its heuristic guided version. For each example, both algorithms are run for the same amount of time ($600\,s$) for 10 independent simulations each. In all simulations the goal is reached in less than $10\,s$. Figure 3 shows the tree obtained in a representative run of the two algorithms, together with the best solution, for the two different problems. It is clear that the inclusion of the proposed heuristic channels the computational capability to the areas of the search space that are more likely to contain the optimal solution.

Figure 4 reports the averages of the costs associated to the solution generated by each algorithm with respect to CPU time. These plots clearly show that the inclusion of a heuristic function avoids the expansion of the tree using unpromising nodes and returns a better solution faster.

In order to further show the general applicability of the proposed heuristic function we also used it in A* to search the implicit graph built by concatenating the motion primitives, i.e, G^Δ_{free}. In order to overcome the computational difficulties introduced by branching and the size of the search space, we selected a reduced number of primitives among the

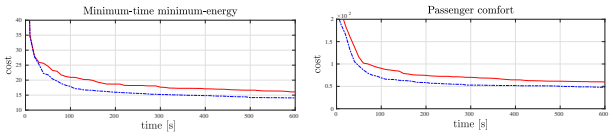


Fig. 4. The solution cost with respect to the computational time for MP-RRT* (red solid line) and heuristic guided MP-RRT* (blue dashed line).

database of trajectories. In particular, for the problem defined in Section V-A the number of primitives in the reduced set is 904 with an average branching factor of 36. Similarly, for the passenger comfort problem (Section V-B) we used 276 primitives with an average branching factor of 12. To support the theoretical results, we run Dijkstra's algorithm and A* to solve the same problem of finding the optimal branch of G^{Δ}_{free} that connects $\mathbf{s}_{start}$ to $\mathbf{s}_{goal}$. We generated 100 random pairs of initial and goal states for both problems and as expected, in all instances A* incorporated with the proposed heuristic returned the same solution as Dijkstra's algorithm (which corresponds to the resolution-optimal solution) while expanding less number of nodes.

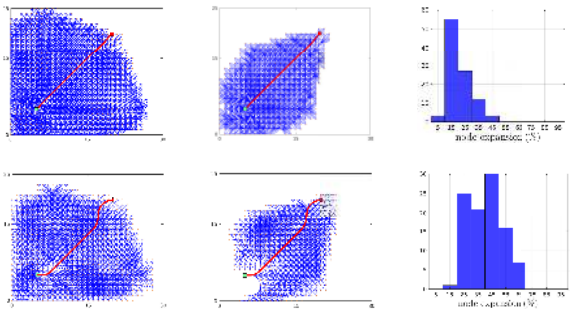


Fig. 5. Searching the implicit graph of motion primitives corresponding to the minimum-time minimum-energy (top) and passenger comfort problems (bottom). Nodes expanded by two different optimal graph-search algorithms: Dijkstra's algorithm (left) and A* using the proposed heuristic (middle), red solid line represents the optimal solution, while green and red squares are the initial and final positions respectively. Histogram of the nodes expanded by A*as percentage of the nodes expanded by Dijkstra's algorithm (right).

An example of an instance of both problems is shown in Figure 5, together with the histogram of the nodes expanded using A* with the proposed heuristic expressed as percentage of the nodes expanded by Dijkstra's algorithm. The computing time is rescaled of the same percentage. In both cases the same solution is obtained but on average in less than half of the time for A*, which shows that the proposed heuristic function is capable of efficiently estimating the cost-to-goal.

VI. Conclusion

In this paper a heuristic function that can be incorporated in any planner using motion primitives is introduced to increase the convergence rate. This heuristic is admissible, thus preserving the optimality properties of the planning algorithm, and it depends only on the database of motion primitives, being applicable to general planning problems with generic agent motion models and cost criteria.

The proposed heuristic has been integrated into a sampling-based and a search-based planner, and the effectiveness of the resulting optimal planner has been shown in simulation using different motion models and cost criteria.

Acknowledgment

This work is supported by TEINVEIN: TEcnologie INnovative per i VEicoli Intelligenti, CUP (Codice Unico Progetto-Unique Project Code): E96D17000110009-Call "Accordi per la Ricerca e l'Innovazione", cofunded by POR FESR 2014-2020 (Programma Operativo Regionale, Fondo Europeo di Sviluppo Regionale-Regional Operational Programme, European Regional Development Fund).

References

[1] B. Donald, P. Xavier, J. Canny, and J. Reif, "Kinodynamic motion planning," *Journal of the ACM*, vol. 40, no. 5, pp. 1048–1066, 1993.

[2] M. Likhachev and D. Ferguson, "Planning long dynamically feasible maneuvers for autonomous vehicles," *The International Journal of Robotics Research*, vol. 28, no. 8, pp. 933–945, 2009.

[3] M. Pivtoraiko, R. A. Knepper, and A. Kelly, "Differentially constrained mobile robot motion planning in state lattices," *Journal of Field Robotics*, vol. 26, no. 3, pp. 308–333, 2009.

[4] M. Pivtoraiko and A. Kelly, "Kinodynamic motion planning with state lattice motion primitives," in *International Conference on Intelligent Robots and Systems*, 2011, pp. 2172–2179.

[5] S. Karaman and E. Frazzoli, "Optimal kinodynamic motion planning using incremental sampling-based methods," in *Conference on Decision and Control*, 2010, pp. 7681–7687.

[6] Y. Li, Z. Littlefield, and K. E. Bekris, "Asymptotically optimal sampling-based kinodynamic planning," *The International Journal of Robotics Research*, vol. 35, no. 5, pp. 528–564, 2016.

[7] K. Hauser and Y. Zhou, "Asymptotically optimal planning by feasible kinodynamic planning in a state-cost space," vol. 32, no. 6, pp. 1431–1443, 2016.

[8] B. Sakcak, L. Bascetta, G. Ferretti, and M. Prandini, "Sampling-based optimal kinodynamic planning with motion primitives," *Autonomous Robots*, 2019, available: https://doi.org/10.1007/s10514-019-09830-x.

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

[10] M. Likhachev, G. J. Gordon, and S. Thrun, "ARA*: Anytime A* with provable bounds on sub-optimality," in *Advances in neural information processing systems*, 2004, pp. 767–774.

[11] 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 *International Conference on Intelligent Robots and Systems*, 2014, pp. 2997–3004.

[12] ——, "Batch informed trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs," in *Int. Conf. on Rob. and Auto. (ICRA)*, 2015, pp. 3067–3074.

[13] M. Otte and N. Correll, "C-forest: Parallel shortest path planning with superlinear speedup," vol. 29, no. 3, pp. 798–806, June 2013.

[14] D. Ferguson and A. Stentz, "Anytime rrts," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, pp. 5369–5375.

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

[16] O. Arslan and P. Tsiotras, "Use of relaxation methods in sampling-based algorithms for optimal motion planning," in *International Conference on Robotics and Automation*, 2013, pp. 2421–2428.

[17] R. A. Knepper and A. Kelly, "High performance state lattice planning using heuristic look-up tables," in *International Conference on Intelligent Robots and Systems*, 2006, pp. 3375–3380.

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

[19] R. Dechter and J. Pearl, "Generalized best-first search strategies and the optimality of A*," *Journal of the ACM*, vol. 32, no. 3, pp. 505–536, 1985.