S*: On Safe and Time Efficient Robot Motion Planning

Riddhiman Laha*,1, Wenxi Wu*,2, Ruiai Sun*,1, Nico Mansfeld*,1,
Luis F.C. Figueredo*,1, and Sami Haddadin[1]

***Abstract*— As robots and humans increasingly share the same workspace, the development of safe motion plans becomes paramount. For real-world applications, nonetheless, it is critical that safety solutions are achieved without compromising performance. The computation of safe, time-efficient trajectories, however, usually requires rather complex often decoupled planning and optimization methods which degrades the nominal performance. In this work, instead, we cast the problem as a graph search-based scheme that enables us to solve the problem efficiently. The graph search is guided by an informed cost balance criterion. In this context we present the S* algorithm which minimizes the total planning time by equilibrising shortest time-efficient paths and paths with higher safe velocities. The approach is compatible with standards and validated both in rigorous simulation trials on a 6 DoF UR5 robot as well as real world experiments on a Franka Emika 7 DoF research robot.**

I. Introduction

In close human-robot interaction (HRI) scenarios, physical contact is part of the process and potentially hazardous collisions may occur [1], [2]. Therefore, it is of primary importance to ensure human safety in order to bring robots outside the industrial fences. Robots deployed for HRI require proper decision-making, motion planning, and control capabilities that account for and ensure safety in the event of unforeseen contacts [3], [4].

Traditional pre-collision schemes often rely on anticipatory scaling of motion velocities – which is naturally detrimental to performance and fluidness of interaction. Especially in industrial settings, the trade-off between safety and performance often makes HRI applications uneconomical. Time-efficient motion planning solutions [5], [6] are sought, which are typically related to maximization of resources, and therefore performance, by means of exploitation of joint and task-space capabilities, i.e., velocities, accelerations, and jerks. The results are, however, hardly applicable for shared workspaces and an additional safety-layer is required that refines, scales the time, and/or increases interpolation of the original path as to ensure safety constraints are met. This additional layer often degrades the nominal trajectory performance substantially.

In order to plan trajectories that are safe and fast, several optimization-based schemes were proposed in literature [7]–[10]. Such methods can have drawbacks in terms of complexity, computation time, etc. In this work, instead, we propose a *search-based* approach, that can be solved efficiently by making use of a well-informed graph. This graph is obtained by combining two of our previously introduced safety tools, namely a) the Safe Motion Unit [11], a local trajectory scaling scheme that embeds biomechanics injury data for safe velocity control, and b) the Safety Map, a tool that evaluates the safety characteristics for the entire (discretized) robot workspace or task-dependent subsets [12]. We leverage the properties of both methods in order to bridge the gap between robot design and control. More specifically, we connect the nodes of the safety configuration space (Safety Map) and efficiently navigate through this field with a Lazy approach; see illustrative example in Fig. 1. The proposed real-time capable task-space *Lazy S** planner generates safe and efficient robot trajectories that comply with safety standards (e.g., ISO/TS 15066) and the robot's motion constraints. In this paper, we frame the basic search problem, introduce the novel *Lazy S** planner, validate, and compare the approach with a time-optimal version of A* along with its lazy variant, as well as new modified safety-scaled approaches for A* and RRT* we introduce for comparison.

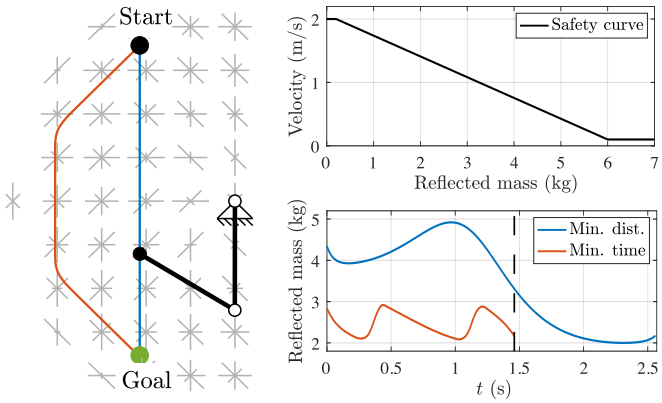

Fig. 1: Illustrative example for planning safe, time-efficient trajectories. The workspace grid of a 2R planar robot is shown on the left. For every position, the maximum safe velocity (v_{safe}) in the direction of the adjacent positions is determined via the safety curve (upper right plot) with magnitude depicted within the gray lines – the longer, the higher the v_{safe} in that direction. The blue line represents the minimum-distance trajectory via the A* planner; the red the minimum-time safe trajectory from S*. Both trajectories are subject to the safety-curve constraint, i.e., safety is ensured at every time instant. The bottom plot shows that the S* is faster than the conventional approach as higher safe velocities can be exploited.

Illustrative Example

To better illustrate the proposed framework and the overall concept concerning safe and time-efficient motion planning, let us consider a planar 2R robot. As shown in Fig. 1, we want to compute a safe, time-efficient trajectory from

*Equal contribution.

[1]The authors are with Technical University of Munich (TUM), Munich Institute of Robotics and Machine Intelligence (MIRMI), Munich, Germany. Email:{riddhiman.laha, ruiai.sun, nico.mansfeld, luis.figueredo, haddadin}@tum.de.[2]Author is with King's College London, London, United Kingdom. Email: wenxi.wu@kcl.ac.uk.

the start to the goal position for the robot[1]. The distance between the discretized robot's workspace positions is $10\,\mathrm{cm}$. For every grid position, a corresponding joint configuration is determined via the robot's inverse kinematics. Then, for each position, the maximum safe velocity in the direction of the adjacent grid positions is determined via the safety curve depicted in Fig. 1 (upper right). These velocities are represented by gray lines in the workspace. The longer a line, the higher the maximum safe velocity in the considered direction.

Provided with the distances and speed limitations between the grid positions, and assuming that the velocity is constant when traveling from one position to another, we can now determine the time required to traverse through the grid. This information can be used to find a time-efficient path from start to goal using our proposed graph search-based algorithm. In Fig. 1, the red line represents the minimum-time solution obtained via B-spline interpolation. It is compared to the blue minimum-distance trajectory. Both trajectories are subject to the safety-curve constraint, i.e., safety is ensured at every time instant. The bottom right figure shows that the minimum-time trajectory is faster than the minimum-distance trajectory because it explicitly seeks regions with lower reflected mass that allow higher safe velocities.

II. Related Work

Safety in motion planning [13] is a vivid field of research. There exist many pre- and post-collision safety schemes in literature [7], [8], [10], [14], [15]. The biomechanics-based Safe Motion Unit (SMU) has often been used as a safety-layer between planning and control. The idea is to constrain the previously planned trajectory velocities in a sequential manner (if necessary), which may degrade the performance compared to the (unsafe) initial trajectory. In this work we aim to extend the approach for a search based navigation capturing the safety requirements. For the case of sampling based planners [16], the overall performance is sensitive to the sampling strategy used. Although some recent efforts have been put to impose additional constraints like moving obstacles [17] and danger index for safe interaction [18], integrating safety aspects intrinsically in traditional motion planning algorithms using dedicated safety indicators is still an ongoing challenge [19].

In this regard, the authors in [20], [21] propose an offline improved version of RRT-Connect that checks for collision avoidance and safety criteria based on a predefined cost function containing visibility, inertia and human-robot centre of mass distance criterions is used to plan geometric paths. However, the path that is obtained at the end is not necessarily optimal and study does not map to injury data as we do – hence with a poorer assessment of collision and safety. A planner operating with an RRT paradigm where the main idea is to grow the trees towards safer regions is presented in [22]. However, the heuristic that they use for guiding the expansion is based on danger fields [23], [24] which is a generalisation of the potential field approach [25], and is able to capture only kinematic state and velocities of the particular point of interest. We search for points in the task space in the same vein as [26] which differs from most of the associated literature– the benefit being more direct exploration of empty regions in the lower dimensional task space instead of the full configuration space. This drastically reduces the set of potential solutions that need to be traversed.

As far as time-optimal and time-efficient motion planning is concerned, there has been no systematic investigation apart from heuristic-based sampling bias in the realm of sampling based algorithms. From the few papers that are present, [27], [28] deal with running an RRT algorithm multiple times, improving in each run with the goal of converging to an optimal solution. In [29] a bidirectional RRT is presented which is probabilistically complete and uses backward search as a guiding heuristic for the forward expansion.

A different class of algorithms that received widespread attention in this context is graph search algorithms, similar to ours, that spans over a finite discretization [30]–[32]. However, few approaches take safety constraints into account. In this work, we take a fundamentally different approach by integrating safety aspects arising from the dynamics of the robot. This strategy reinvigorates the original decision-making problem towards a new formulation that embeds a cost resolution between safety and time-efficiency.

III. Problem Definition

In this paper, we are interested in finding a minimum-time trajectory in free task space (TS) from an initial pose $\underline{\boldsymbol{x}}_0$, for a n-DoF robot, to a goal pose, $\underline{\boldsymbol{x}}_f(\underline{\boldsymbol{x}}_0, \underline{\boldsymbol{x}}_f, \in \text{TS} \subseteq SE(3))$. This trajectory should satisfy safety constraints in the case of a collision and the robot's motion constraints.

The safety constraint for dynamic collisions is given in terms of a *safety curve*, a functional relation between the robot reflected mass $m_u(\boldsymbol{q})$ in the direction of motion [33] and the operational speed; see Fig. 1. Associated to a certain contact geometry (blunt, edgy, or sharp) and a human body part, a safety curve determines the maximum safe speed v_{safe} for the current effective mass. For a certain application, one may select the safety curves provided in the ISO/TS 15066, which stem from contact modeling, or a data-driven relation between robot mass, velocity, contact geometry, and injury as proposed in [11], for example. In the sequel, safety curves are referred to as $\mathcal{S}_{\text{CG}}$. For the sake of simplicity, we assume that potentially dangerous collisions with the robot occur at the end-effector (EE) and that collisions with other locations along the robot structure are safe, given the light mass properties and blunt surface geometry of tactile and collaborative robots[2]. In this work, we assume the human to be (quasi-) static, however, our method can be extended to dynamic environments, which requires additional sensors.

In our work, a planning query takes the input set $(\boldsymbol{q}_0, \underline{\boldsymbol{x}}_f, \boldsymbol{Q}_{\text{lim}}, \mathcal{S}_{\text{CG}})$, where $\boldsymbol{Q}_{\text{lim}} = \{\underline{\boldsymbol{q}}, \overline{\boldsymbol{q}}, \underline{\dot{\boldsymbol{q}}}, \overline{\dot{\boldsymbol{q}}}, \underline{\ddot{\boldsymbol{q}}}, \overline{\ddot{\boldsymbol{q}}}, \}$ is the set that contains the joint position, velocity, and acceleration limits. These limits are used to maximize resources which combined with the safety curve $\mathcal{S}_{\text{CG}}$ ensures a safe, time-efficient trajectory. To solve a planning query, herein, we

[1]The robot is gravity-free and the length of each robot link is $0.5\,\mathrm{m}$, at the distal end of each link a $2\,\mathrm{kg}$ point mass is located.

[2]The extension to multiple contact locations is subject to future work.

explore a similar approach to [26], [34] for planning in task-space. Notwithstanding, instead, we propose in this work a modified task-space A* planner [35]. To this aim, we first uniformly voxalize the $SE(3)$ workspace to build a task-space graph that accelerates the search.

The search problem is performed in the direct graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of vertices and $\mathcal{E}$ is the set of edges connecting pairs of vertices (nodes). A cornerstone of our proposed planner are the definitions below:

- Each node $\boldsymbol{v} = (\underline{\boldsymbol{x}}, \boldsymbol{Q}) \in \mathcal{V}$ which consists of a task-space point $\underline{\boldsymbol{x}} \in \text{TS} \subseteq SE(3)$ and a point in configuration space $\boldsymbol{Q} = \{\boldsymbol{q}, \dot{\boldsymbol{q}} \in \mathbb{R}^n \mid \text{FKM}(\boldsymbol{q}) = \underline{\boldsymbol{x}}\}$, where FKM is the robot's forward kinematics;
- The edge between two nodes, $\forall i, j,\ (\boldsymbol{v}_j, \boldsymbol{v}_i) \in \mathcal{E}$ takes a safety transition time t_{safe} as a cost function $\boldsymbol{G}$, in addition to the distance heuristic $\boldsymbol{H}$, as in standard A*, such that $\boldsymbol{F} = \boldsymbol{G} + \boldsymbol{H}$, $\boldsymbol{F} : \mathcal{E} \to \mathbb{R}^+$. The cost t_{safe} is defined by the maxima time between robot configuration space capabilities ($\boldsymbol{Q}_{\text{lim}}$) to transfer from two nodes and the task-space safety-bounds stemming from v_{safe}. It is important to note here that we normalize the time and distance heuristic appropriately such that there is no conflict of matching units (detailed in Sec. IV-A).

A. Formulation

Given the planning query described above and the direct graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, our goal is to find the minimum-cost path on $\mathcal{G}$, starting from an initial pose $\boldsymbol{v}_{\text{start}}$ and ending at a goal pose $\boldsymbol{v}_{\text{goal}} = \underline{\boldsymbol{x}}_f$. The resulting joint space trajectory must also satisfy the robot's position, velocity, and acceleration constraints – in addition to being continuous and smooth, at least C^2. Since edge evaluation is usually expensive in high dimensions, we delay it to the point when it is absolutely necessary as in the *lazy paradigm* [36]. The sum of the weights of the edges gives the *true* ($\hat{\boldsymbol{F}}(\mathcal{P})$) and *lazy* cost ($\boldsymbol{F}(\mathcal{P})$) of a valid path $\mathcal{P}$,

$$\boldsymbol{F}(\mathcal{P}) = \sum_{\mathcal{E} \in \mathcal{P}} \boldsymbol{F}(\mathcal{E}), \quad \text{and} \quad \hat{\boldsymbol{F}}(\mathcal{P}) = \sum_{\mathcal{E} \in \mathcal{P}} \hat{\boldsymbol{F}}(\mathcal{E}), \tag{1}$$

The edge evaluation consists of the cost balance resolution between time efficient motion plans and safe motion directions guided by the heuristic function $\boldsymbol{G}$. The safety constraint is given by the safety curves described previously. Since the paths are not fully evaluated in the algorithm (unlike traditional weighted A*), each path $\mathcal{P}$ can be disintegrated into $\mathcal{P}_{\text{front}}$ and $\mathcal{P}_{\text{back}}$. The collision free edges in $\mathcal{P}_{\text{front}}$ will be fully evaluated and edges in $\mathcal{P}_{\text{back}}$ will face lazy evaluation. Therefore, the total estimated cost $\tilde{\boldsymbol{F}}(\mathcal{P})$ is designated by,

$$\tilde{\boldsymbol{F}}(\mathcal{P}) = \boldsymbol{F}(\mathcal{P}_{\text{front}}) + \hat{\boldsymbol{F}}(\mathcal{P}_{\text{back}}). \tag{2}$$

In this equation, the true cost $\hat{\boldsymbol{F}}$ steers the search in the sense that edge evaluation, taking into account the safety constraints and time efficiency, is postponed until the actual intention (cost) of the planner. In other words, the framework allows instantaneous computation of a maximum safety-aware velocity that is computed based on the Cartesian velocity of the robot, its reflected mass range, and contact geometries. This information can be acquired in real-time and helps in the decision between safety (low reflected mass) and efficiency

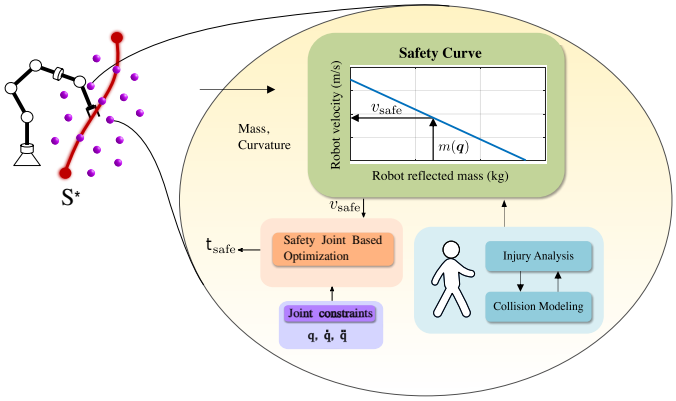

Fig. 2: Concept for generating biomechanically safe robot velocities based on [11]. The configuration-dependent robot reflected mass $m(\boldsymbol{q})$ is related to a safe velocity v_{safe} via a safety curve, which can be derived from impact experiments or contact models. The safety-aware instantaneous velocity bound gives us a desired joint-velocity and convergence time to a subsequent node in the search graph. This is compared to the joint-space constraints (position, velocities, accelerations and jerks) to provide the worst-case convergence time that satisfies both robot and safety constraints.

(high EE velocity). These trajectories might be deviated from the desired nominal path that does not respect the safety constraints. Hence, a non-greedy (likely graph-based) approach is required to obtain the near-optimal trajectory that resolves the cost balance between shortest path and path with higher allowable velocities. Mathematically, the problem we seek to solve can be stated as follows:

Problem Statement 1: *Given a path planning problem* $(\mathcal{V}_{\text{free}}, \boldsymbol{v}_{\text{start}}, \boldsymbol{v}_{\text{goal}})$ *and a cost function* $\tilde{\boldsymbol{F}} : \mathcal{E} \to \mathbb{R}^+$*, find a feasible path* $\mathcal{P}$ *such that* $\boldsymbol{F}^*(\mathcal{P}) = \min\{\tilde{\boldsymbol{F}}(\mathcal{P}) : \mathcal{P}$ *is feasible* $\}$*. If such a path does not exist, report failure.* More specifically, in this work, we seek a time-optimal trajectory based on minimum-cost-graph-search solutions that satisfies both safety constraints in terms of Cartesian velocities – and therefore, time costs – and joint-space robot capabilities – in terms of position, velocities, accelerations, and jerks available.

IV. Time Efficient Safe Path Planning (Lazy S*)

This section introduces our approach for safety-aware time efficient trajectory generation. First, we present an overview of the proposed planner based on the Lazy weighted A* algorithm. Second, we introduce the safety aspects emerging from the *Safe Motion Unit* and how the safety-aware time information is fed to the planner. Finally, we show the safety joint based optimization that we deploy during node expansion and edge assessment in order to achieve safe planning.

A. Algorithmic details and description

Our algorithm is illustrated in Alg. 1. It starts with initializing the *OPEN* and *CLOSED* sets as well as the cost function on the voxelized map (lines 2, 3). The goal is to grow and maintain a shortest path tree over the graph $\mathcal{G}$ containing nodes residing in $\mathcal{V}_{\text{free}}$. Firstly, we set the nodes $\boldsymbol{v}_{\text{start}}$ (the root of the tree) and $\boldsymbol{v}_{\text{goal}}$. The algorithm maintains two lists/queues to aid the search process. The *OPEN* list stores the nodes that have been discovered using minimum $\boldsymbol{F}$ value (line 5), but haven't been expanded. The

already evaluated nodes are a part of the *CLOSED* list and at the end form the path $\mathcal{P}$. We start by setting the first discovered node as the current node $\boldsymbol{v}_{\text{c}}$ (line 5). Due to space limitations, Alg. 1 only illustrates the S^* concept which is later implemented in a standard lazy fashion to reduce the computational burden during edge assessment, while iteratively removing the cheapest states from the *OPEN* list until $\boldsymbol{v}_{\text{goal}}$ is reached.

Once the exploration starts (line 10), we proceed by checking if the state belongs to the *CLOSED* list. The algorithm terminates if a collision-free shortest path to $\boldsymbol{v}_{\text{goal}}$ is found during evaluation (lines $8-9$). For each neighboring node $\boldsymbol{v}_{\text{i}}$, we examine feasibility using *IKFAST*, which returns all possible solutions for the given pose in task space due to redundancy. For redundancy resolution, we use a minimal weighted norm heuristic to select the closest IK solution $\boldsymbol{q}_j$ from the current robot configuration $\boldsymbol{q}_c$ to avoid larger inefficient joint motions given by $\boldsymbol{q}_i = \arg\min||\boldsymbol{q}_c - \boldsymbol{q}_i||_M$. This choice of redundancy resolution is of interest as it determines completeness features of the algorithm. Here, $\boldsymbol{M}$ is the diagonal matrix of weights – most often given higher importance to lower joints – $\boldsymbol{q}_c$ is the current joint configuration, and $\boldsymbol{q}_i$ is the set of feasible joint-configurations for a given node expansion and edge evaluation. The selected IK values during exploration are stored for future use; nodes that returned no solutions are excluded from the search tree.

1) Edge Evaluation: Time Optimal Joint Optimization (TOJO): From the node assessment and IK mapping, we get the direction of motion also in the joint-space. This allows us to explore the traversal time $t_{\lim(q)}$ between the system in accordance to robot joint-space capabilities, i.e., velocities, acceleration and jerks. This alone would provide us with a time-optimal solution if integrated directly in the cost function (5) during edge evaluation. Nonetheless, given our interest in finding the safety-aware time-optimal solution to the planning problem, we integrate the safety-aware instantaneous maximum velocity solution from [12].

Next, with the selected target joint configuration from the IK search (obtained during node expansion as described above), we perform a real-time optimization in the configuration space. Formally, we want to solve the following problem: Given an initial joint-configuration $\boldsymbol{q}_c$ and a target configuration $\boldsymbol{q}_d$, we search for the time optimal trajectory $\boldsymbol{q}^*(t)$, given by

$$\begin{aligned} t_{\lim} &= \min_{\underline{\boldsymbol{x}}(t)} 1 \cdot T, \\ \text{s.t.} \quad \boldsymbol{q}(0) &= \boldsymbol{q}_0 \quad \underline{\boldsymbol{q}} \le \boldsymbol{q} \le \overline{\boldsymbol{q}} \\ \boldsymbol{q}(t_{\lim}) &= \boldsymbol{q}_d \quad \underline{\dot{\boldsymbol{q}}} \le \dot{\boldsymbol{q}} \le \overline{\dot{\boldsymbol{q}}} \\ & \qquad\quad \underline{\ddot{\boldsymbol{q}}} \le \ddot{\boldsymbol{q}} \le \overline{\ddot{\boldsymbol{q}}} \end{aligned} \tag{3}$$

We adopt the approach in [37] to solve the above optimization problem. Herein, each DoF is considered independently, and a time synchronization is performed for every DoF that reaches its target at the given trajectory duration.

2) Edge Evaluation: Safe velocity constraint: As mentioned in the previous sections, the safety curves provide us with a relationship between the reflected mass perceived at the end-effector $m_u(\boldsymbol{q})$ and the maximum instantaneous

Algorithm 1 S* Algorithm

1: **procedure** PLANMINTIMESAFETRAJ($\boldsymbol{q}_0, \boldsymbol{v}_{\text{goal}}, \boldsymbol{Q}_{\lim}, \mathcal{S}_{\text{CG}}$)
2: CLOSED $\leftarrow \emptyset$, OPEN $\leftarrow \boldsymbol{v}_{\text{start}}$;
3: $\boldsymbol{G}, \boldsymbol{H} \leftarrow$ Initialize cost and heuristic function ;
4: **while** OPEN $\neq \emptyset$ **do**
5: $\boldsymbol{v}_{\text{c}} \leftarrow$ min $\boldsymbol{v} \in$ OPEN $\mid \boldsymbol{F}(\boldsymbol{v}) \le \boldsymbol{F}(\boldsymbol{v}_i), \forall \boldsymbol{v}_i \in$ OPEN;
6: Remove $\boldsymbol{v}_c$ from OPEN;
7: CLOSED $\leftarrow$ Add $\boldsymbol{v}_{\text{c}}$ to closed list;
8: **if** $\boldsymbol{v}_{\text{c}} == \boldsymbol{v}_{\text{goal}}$ **then**
9: **return** CLOSED;
10: break;
11: **for** each neighbour $\boldsymbol{v}_i$ of $\boldsymbol{v}_{\text{c}}$ **do**
12: **if** $\boldsymbol{v}_i \notin$ CLOSED **then**
13: $\boldsymbol{Q}_\eta \leftarrow$ Find η IK solutions for $\underline{\boldsymbol{x}} \leftarrow \boldsymbol{v}_i$;
14: Update $\boldsymbol{v}_i \leftarrow$ Solve redundancy resol. $\boldsymbol{Q}_\eta$;
15: $t_{\lim} \leftarrow$ opt-time (3) w/ ($\boldsymbol{Q}_{\lim}$,$\boldsymbol{q}_d \leftarrow \boldsymbol{v}_i$);
16: $m_u \leftarrow$ Get reflected mass based on $\underline{\boldsymbol{x}} \leftarrow \boldsymbol{v}_i$);
17: $v_{\text{safe}} \leftarrow$ Obtain safe velocity (m_u, $\mathcal{S}_{\text{CG}}$);
18: $t_{\text{SMU}} \leftarrow$ Get safe traversal time (v_{safe});
19: $\boldsymbol{G}_{\text{new}} \leftarrow$ Update max $\{t_{\text{SMU}}, t_{\lim}\}$;
20: $\boldsymbol{H}_{\text{new}} \leftarrow$ Set new $\boldsymbol{H}$ value;
21: $\boldsymbol{F}_{\text{new}} \leftarrow \boldsymbol{G}_{\text{new}} + \boldsymbol{H}_{\text{new}}$;
22: OPEN $\leftarrow$ Add to open list;
23: continue;
24: **return** failure;

directional (let us denote this direction by $\boldsymbol{u}$) safe velocity v_{safe} (see Fig. 1 for illustration). The reflected mass perceived at the end-effector in the unit direction $\boldsymbol{u}$ is given by

$$m_u(\boldsymbol{q}) = \left(\boldsymbol{u}^{\mathsf{T}} \boldsymbol{\Lambda}_\nu(\boldsymbol{q})^{-1} \boldsymbol{u}\right)^{-1}, \tag{4}$$

where $\boldsymbol{\Lambda}_\nu(\boldsymbol{q})^{-1}$ is the upper 3×3 part of the Cartesian mass matrix inverse $\boldsymbol{\Lambda}(\boldsymbol{q})^{-1}$ [38].

For the edge evaluation, we strive to compute the fastest transport time satisfying safety-constraints, that is, $t_{\text{SMU}(q)}$. During tree expansion (line 10 in Alg. 1), the node propagation determines the EE direction of motion. We use the joint velocities $\boldsymbol{q}$ obtained from TOJO and the geometric jacobian $\boldsymbol{J}(\boldsymbol{q})$ to compute the task-space directional velocity $\dot{\boldsymbol{x}}_u$, along the the normalized direction $\boldsymbol{u}$. Provided with the reflected mass at the a point of interest (POI) at the EE or payload, we can evaluate the maximum biomechanically safe speed v_{safe} with $\mathcal{S}_{\text{CG}}$; see Fig. 2. Finally, from v_{safe} we directly compute the fastest transport time $t_{\text{SMU}(q)}$ – which is used for resolution of the cost balance during edge evaluation.

3) Resolution of cost balance as safety heuristic: In this section, we provide an intuition on how we resolve the cost balance between shortest paths and and paths between higher allowable safety-velocities. Given a graph $\mathcal{G}$, we define the optimal cost $\boldsymbol{F}^*(\mathcal{P})$ for path $\mathcal{P}$ connecting $\boldsymbol{v}_{\text{start}}$ and $\boldsymbol{v}_{\text{goal}}$. During node expansion we want to minimize this cost while satisfying the physical constraints of the robot. Therefore, the edge asssessment cost w_{cost} can be computed as

$$w_{\text{cost}} = t_{\text{safe}} = \max\{t_{\lim}, t_{\text{SMU}}\}. \tag{5}$$

In other words, our algorithm selects safer paths even when faster options are available as depicted in the experimental section. In the worst case, we always end up with a slower safe path. For heuristic consistency, we define weights and normalize both $\boldsymbol{G}$ and $\boldsymbol{H}$ before adding them up (line 21). These weights could be tuned further for specific behaviors.

B. Remark on completeness

One common concern when new planners are introduced – especially for sampling-based or grid-based planners in task-space [26], [34], [39], [40] – is the concept of probabilistic completeness, i.e., if a feasible path exists, the probability of finding the solution approaches on as the computation time goes to infinity [40]. Considering the space of all possible configurations, task-space planners are often not complete [26], [39]. Notwithstanding, the planner will explore all possible grid-based task-space solutions in infinte computational time, which are always validated in configuration space. Hence, in our case, it is only applicable to the discrete-task-space search, i.e., along the task-space grid. This limitation is only due to the fact that during the edge-evaluation, the planner stores only the greedy time-optimal choice among null-space solutions of a given grid node – lines 12-15 of Alg. 1. Probabilistic completeness in the configuration space would require exploring all possible solutions and costs for the redundancy resolution (line 12), which in turn would make the planner less time efficient. As previously argued, this efficiency is more valid in practical real-world applications than configuration-space completeness [40].

C. Implementation details

We design and describe our algorithm as an extension in the task space of the lazy version of weighted A*, first introduced in [41], which is a variation of the classic A* search algorithm that employs a one-step lookahead to reduce the number of edge evaluations. More specifically, we want to heuristically guide the search in a similar fashion to A*, which orders the search list based on a sum of the cost and an estimation of the same which in our case is the safety heuristic. Firstly, we seek to determine a robot's feasibility transitions which could be potentially observed as an analysis within a reachability map [42], [43]. Therefore, we discretize the workspace and use IKFast [44] to check for feasibility of a neighboring node during expansion. If a feasible solution is found, we evaluate the edges and search for the path considering that node. Secondly, to accelerate graph search for effective implementation, and alleviate the influence of high dimensionality in time-perfomance we evaluate the discretized workspace in a 4D approach[3]. The orientation, in this case, is computed through an interpolated and relative distance from the original orientation to the goal orientation,

$$\boldsymbol{r}_c = \exp(\log(\boldsymbol{r}_0^* \boldsymbol{r}_1)c)\,, \tag{6}$$

where $\boldsymbol{r}_0$, $\boldsymbol{r}_1$, and $\boldsymbol{r}_c$ depict the quaternion orientation of the starting, ending and current poses. The scalar $c \in [0, 1]$ is defined along a range and represents how far along the path the robot is at the moment, and could be selected in different ranges of the path.

V. Quantitative Assessment And Experiments

This section investigates quantitative aspects of performance and constraint satisfaction of the proposed safety-aware time efficient planner under different conditions. All simulated scenarios were deployed using the UR5 robot from Universal Robots, a non-redundant 6-DoF manipulator, with planning algorithms implemented in C++ using ROS as a middleware. Lower level controllers and kinematics of the robot were implemented using the DQ-Robotics library [45]. We used a modified *IKFAST* python plugin [46] for the safety-aware feasibility check, and node-expansion with integration to Ruckig [37] for time-optimal edge evaluation. For real-world experiments, we used the 7-DoF Franka Emika robot. As far as the contact geometry of the robot EE is concerned, the shape of the robot's gripper brackets is similar to the spherical impactor with $12.5\,\mathrm{mm}$ radius that was tested for abdominal injury in [11]. In our experiments, we use the safety curve of this impactor, where we reduce the safe speed by a conservative factor of three for prevention of execution of potentially "unsafe" motions given the inertial parameters.

A. Planning Efficiency: A comparison study

To measure the performance of the proposed planning scheme, also to compare with existing alternative planners that can be adapted to account for both safety and performance, we devised a set of 100 scenarios (in an obstacle free environment) in Coppelia Robotics simulated environment [47] – running the physical engine Bullet 2.78. Although computational costs can be critical in search-based planning for online applications, our non-optimized code is used in an offline fashion for the experiments.

As pointed out in Sec. II, existing decoupled strategies usually limit the speed of the nominal planned trajectory to comply with the safety criteria. This planning scheme is defined as the baseline for comparison. Notice that this exact planning design does not exist in literature and comparison with existing frameworks mentioned in the related work is part of the future work. Hence, for comparison purposes in addition to the proposed S* and Lazy-S* planners, we also propose four new sub-optimal motion planners:

1) A*+ SJBO,
2) Lazy-A*+ SJBO,
3) RRT*+ SJBO,
4) Lazy-A* + TOJO

The first three planners are based on traditional graph and/or road-map approaches embedded in the configuration space. They frame the problem as a graph-search for the shortest path. In the formulation of the three aforementioned planners, we obtain the shortest-path result, as expected from the planners, and evaluate the resulting joint-trajectory by means of the Safety-Joint-Based-Optimizaton (SJBO). The SJBO scheme, based on Ruckig algorithm [37] together with the safety-margins, redefines the time between resulting nodes to satisfy the task-space safety-velocity constraints and joint velocity, acceleration, and jerk constraints. Notice that this can be viewed as a cascade solution from higher level motion planning to joint-level safety-aware optimization. On the other end, the 4th approach, Lazy-A* + TOJO, aims to find the path that takes the shortest time to the goal without considering the safety effects.

Table I summarizes the average time-to-goal execution time (i.e., the time required to complete the motion task given the

[3]For a complete 6D analysis we plan to include orientation bounds that we consider for each node $\boldsymbol{v}$ in the task space in a future investigation. Also, robot dexterity including redundancy and self-motion will be considered.

TABLE I: Planning efficiency and comparison between the proposed planners in 100 randomized scenarios under different environment conditions. Quantitative assessment includes the average time from starting to goal pose, whilst satisfying safety constraints, the total path length in both Cartesian and joint space normalized against the nominal-non-safe trajectory from A*, success rate, the average maximum velocity (and the velocity-ratio) normalized from the maximum velocity stemming from safety criteria.

	A*+ SJBO	Lazy-A*+ SJBO	RRT*+ SJBO	Lazy-A*+TOJO	S*	Lazy-S*
Time to Goal (s)	6.7865 ± 2.1735	6.3302 ± 1.7094	5.3516 ± 2.7771	2.3181 ± 1.5343	3.6030 ± 2.3363	3.4175 ± 2.3219
Safe	Yes	Yes	Yes	No	Yes	Yes
Joint-space Length (rad)	22.8664	22.3804	5.3645	9.4249	11.6896	10.9286
Success rate (%)	91	94	100	95	100	93
Max. EE Velocity (m/s)	0.2654	0.2625	0.2222	0.4272	0.5331	0.5409
Velocity-Ratio	0.1591	0.1609	0.2387	0.8591	0.2454	0.2756

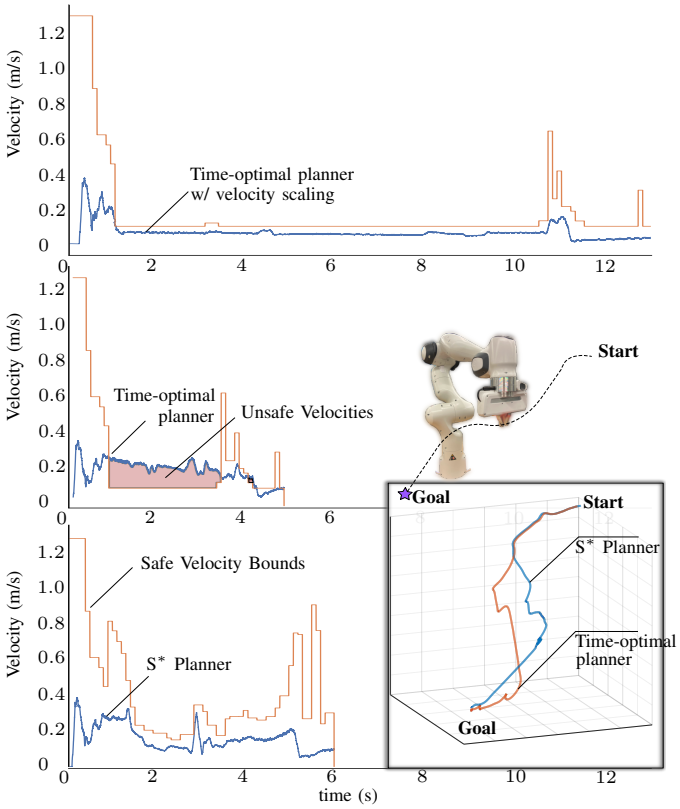

Fig. 3: Trajectories obtained from the three approaches during real robot experiments. The weights on the heuristics are - A*(2), Time-Optimized A*(0.4), and S*(0.05).

safety-constraints), path length in Cartesian space, success rate (if the robot was able to reach the goal), and average velocity rate (normalized by the instantaneous safety velocity constraints).

As expected, S* gives the best performance in terms of time taken to reach the goal while still being safe. Lazy S* finds paths faster than S* due to reduced edge evaluations. The lower success rate, as compared to S*, might be attributed to timeouts in some cases. Due to the planning in the configuration space, RRT*+ SJBO results in the smallest change in joint-configuration at each step. The Lazy-A* + TOJO has the highest velocity, thus the shortest time to the goal. Notwithstanding, this leads to unsafe motions which raise hazard issues for HRI scenarios. Indeed, the time-efficiency of the Lazy-A* + TOJO was obtained in exchange of having 70.5% of the trajectory outside the safety bounds. Our proposed solution, instead was able to achieve sufficient time-efficiency performance (47% slower than the unsafe planner) without compromising safety. In contrast to other planners that satisfy safety constraints in a decoupled manner, the Lazy-S* was able to achieve higher performance. Indeed, similar trajectories planned with RRT*+SJBO, Lazy-A*+SJBO, and A*+SJBO took overall 56.6%, 85.2%, and 98.6% more time to be completed, respectively.

B. Real robot experiments

In this experiment, the Franka Emika robot performs a pick and place task in the vicinity of a human. The goal is to analyze the performance of the robot with three different planners: (a) Time-optimal planner (b) Time-optimal planner with velocity scaling (c) Lazy S* planner. Figure 3 shows the data collected from the experiment. The nominal velocity scaling strategy, although remaining restricted inside the safe zone takes double the time to finish the same path length as compared to other approaches. The *Lazy S** planner is able to maintain the Cartesian EE velocity in accordance with the safe velocity bound v_{safe}. However, with an implementation of time-optimized path without our safety heuristic following the same initial conditions and parameters, the EE violates the safe velocity threshold, thereby making the setup unsuitable for close human-robot collaboration. Furthermore, we also see in Fig. 3 (bottom right corner) that our heuristic influences the planner to explore different paths to the goal. This validates our hypothesis presented in Fig. 1 where we claim that S* produces time-efficient yet safe trajectories.

VI. Conclusion

This paper presents a novel framework for integrating safety constraints into motion plans that are optimized w.r.t. time. We introduce the S* algorithm (along with its lazy variant), which terminates in a finite number of iterations and is cost effective. Simulation studies and real robot experiments highlight the fact that our method respects the safety requirements during task execution. In future works, we plan to extend the method to be able to run online with dynamic obstacles and perform further experimental validation in realistic HRI applications.

Acknowledgement

This work was funded by Lighthouse Initiative Geriatronics by StMWi Bayern (Project X, grant 5140951), KI.FABRIK Bayern (grant DIK0249), and SafeRoBAY (grant DIK0203/01), European Union's Horizon 2020 research and innovation programme as part of the project Darko under grant 101017274. Please note that S. Haddadin has a potential conflict of interest as shareholder of Franka Emika GmbH.

REFERENCES

[1] S. Haddadin, A. Albu-Schäffer, and G. Hirzinger, "Requirements for safe robots: Measurements, analysis & new insights," *International Journal of Robotics Research*, vol. 28, no. 11-12, pp. 1507–1527, 2009.

[2] R. Behrens, *Biomechanische Grenzwerte für die sichere Mensch-Roboter Kollaboration*. Springer, 2019.

[3] M. Beetz, G. Bartels, A. Albu-Schäffer, F. Bálint-Benczédi, R. Belder, D. Beßler, S. Haddadin, A. Maldonado, N. Mansfeld, T. Wiedemeyer, R. Weitschat, and J.-H. Worch, "Robotic agents capable of natural and safe physical interaction with human co-workers," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 6528–6535.

[4] A. Kanazawa, J. Kinugawa, and K. Kosuge, "Adaptive motion planning for a collaborative robot based on prediction uncertainty to enhance human safety and work efficiency," *IEEE Transactions on Robotics*, vol. 35, no. 4, pp. 817–832, 2019.

[5] J. Luo and K. Hauser, "An empirical study of optimal motion planning," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 1761–1768.

[6] J. D. Gammell and M. P. Strub, "Asymptotically optimal sampling-based motion planning methods," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 4, pp. 295–318, 2021.

[7] A. M. Zanchettin, N. M. Ceriani, P. Rocco, H. Ding, and B. Matthias, "Safety in human-robot collaborative manufacturing environments: Metrics and control," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 2, pp. 882–893, 2015.

[8] R. Weitschat and H. Aschemann, "Safe and efficient human–robot collaboration part II: Optimal generalized human-in-the-loop real-time motion generation," *IEEE Robotics and Automation Letters (RA-L)*, vol. 3, no. 4, pp. 3781–3788, 2018.

[9] R. A. Rojas, M. A. R. Garcia, L. Gualtieri, and E. Rauch, "Combining safety and speed in collaborative assembly systems–an approach to time optimal trajectories for collaborative robots," *Procedia CIRP*, vol. 97, pp. 308–312, 2021.

[10] A. Palleschi, M. Hamad, S. Abdolshah, M. Garabini, S. Haddadin, and L. Pallottino, "Fast and safe trajectory planning: Solving the cobot performance/safety trade-off in human-robot shared environments," *IEEE Robotics and Automation Letters (RA-L)*, vol. 6, no. 3, pp. 5445–5452, 2021.

[11] S. Haddadin, A. Khoury, T. Rokahr, S. Parusel, R. Burgkart, A. Bicchi, and A. Albu-Schäffer, "On making robots understand safety: Embedding injury knowledge into control," *International Journal of Robotics Research*, vol. 31, no. 13, pp. 1578–1602, 2012.

[12] N. Mansfeld, M. Hamad, M. Becker, A. G. Marin, and S. Haddadin, "Safety map: A unified representation for biomechanics impact data and robot instantaneous dynamic properties," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1880–1887, 2018.

[13] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, S. Thrun, and L. Kavraki, *Principles of Robot Motion: Theory, Algroithms, and Implementation*. Cambridge: MIT Press, 2005.

[14] A. Pereira and M. Althoff, "Safety control of robots under computed torque control using reachable sets," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 331–338.

[15] N. Lucci, B. Lacevic, A. M. Zanchettin, and P. Rocco, "Combining speed and separation monitoring with power and force limiting for safe collaborative robotics applications," *IEEE Robotics and Automation Letters (RA-L)*, vol. 5, no. 4, pp. 6121–6128, 2020.

[16] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[17] H. Liu, X. Deng, and H. Zha, "A planning method for safe interaction between human arms and robot manipulators," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2005, pp. 2724–2730.

[18] D. Kulić and E. A. Croft, "Safe planning for human-robot interaction," *Journal of Robotic Systems*, vol. 22, no. 7, pp. 383–396, 2005.

[19] B. Lacevic and P. Rocco, "Towards a complete safe path planning for robotic manipulators," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2010, pp. 5366–5371.

[20] V. Rajendran, P. Carreno-Medrano, W. Fisher, and D. Kulić, "Human-aware rrt-connect: Motion planning for safe human-robot collaboration," in *2021 30th IEEE International Conference on Robot & Human Interactive Communication (RO-MAN)*. IEEE, 2021, pp. 15–22.

[21] K. Ikuta, H. Ishii, and M. Nokata, "Safety evaluation method of design and control for human-care robots," *International Journal of Robotics Research*, vol. 22, no. 5, pp. 281–297, 2003.

[22] B. Lacevic, P. Rocco, and M. Strandberg, "Safe motion planning for articulated robots using rrts," in *2011 XXIII International Symposium on Information, Communication and Automation Technologies*. IEEE, 2011, pp. 1–7.

[23] B. Lacevic and P. Rocco, "Kinetostatic danger field-a novel safety assessment for human-robot interaction," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2010, pp. 2169–2174.

[24] B. Lacevic, P. Rocco, and A. M. Zanchettin, "Safety assessment and control of robotic manipulators using danger field," *IEEE Transactions on Robotics*, vol. 29, no. 5, pp. 1257–1270, 2013.

[25] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *International Journal of Robotics Research*, vol. 5, no. 1, pp. 90–98, 1986.

[26] A. Shkolnik and R. Tedrake, "Path planning in 1000+ dimensions using a task-space voronoi bias," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 2061–2067.

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

[28] N. A. Wedge and M. S. Branicky, "On heavy-tailed runtimes and restarts in rapidly-exploring random trees," in *Twenty-third AAAI conference on artificial intelligence*, 2008, pp. 127–133.

[29] S. Nayak and M. W. Otte, "Bidirectional sampling-based motion planning without two-point boundary value solution," *IEEE Transactions on Robotics*, 2022.

[30] M. Likhachev, G. J. Gordon, and S. Thrun, "Ara: formal analysis," 2003.

[31] J. P. Bailey, A. Nash, C. A. Tovey, and S. Koenig, "Path-length analysis for grid-based path planning," *Artificial Intelligence*, vol. 301, p. 103560, 2021.

[32] C. Zammit and E.-J. Van Kampen, "Comparison between a* and rrt algorithms for uav path planning," in *2018 AIAA guidance, navigation, and control conference*, 2018, p. 1846.

[33] O. Khatib, "Inertial properties in robotic manipulation: an object-level framework," *Int. Journal of Robotics Research*, vol. 14, no. 1, pp. 19–36, 1995.

[34] N. Larkin, A. Short, Z. Pan, and S. Van Duin, "Task space motion planning decomposition," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 1688–1694.

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

[36] A. Nash, S. Koenig, and C. Tovey, "Lazy theta*: Any-angle path planning and path length analysis in 3d," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 24, no. 1, 2010, pp. 147–154.

[37] L. Berscheid and T. Kröger, "Jerk-limited real-time trajectory generation with arbitrary target states," *arXiv preprint arXiv:2105.04830*, 2021.

[38] O. Khatib, "Object manipulation in a multi-effector robot system," in *ISRR*, R. Bolles and B. Roth, Eds., 1988, pp. 137–144.

[39] M. Behnisch, R. Haschke, and M. Gienger, "Task space motion planning using reactive control," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 5934–5940.

[40] G. Mesesan, M. A. Roa, E. Icer, and M. Althoff, "Hierarchical path planner using workspace decomposition and parallel task-space rrts," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 1–9.

[41] B. Cohen, M. Phillips, and M. Likhachev, "Planning single-arm manipulations with n-arm robots," in *International Symposium on Combinatorial Search*, vol. 6, no. 1, 2015.

[42] F. Zacharias, *Knowledge representations for planning manipulation tasks*. Springer Science & Business Media, 2012.

[43] N. Vahrenkamp, T. Asfour, and R. Dillmann, "Robot placement based on reachability inversion," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2013, pp. 1970–1975.

[44] R. Diankov, "Automated construction of robotic manipulation programs," Ph.D. dissertation, Carnegie Mellon University, 2010.

[45] B. V. Adorno and M. Marques Marinho, "Dq robotics: A library for robot modeling and control," *IEEE Robotics Automation Magazine*, pp. 0–0, 2020.

[46] R. Diankov and J. Kuffner, "Openrave: A planning architecture for autonomous robotics," *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-08-34*, vol. 79, 2008.

[47] E. Rohmer, S. P. N. Signgh, and M. Freese, "V-rep: a versatile and scalable robot simulation framework," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2013.