Journal of Intelligent & Robotic Systems (2020) 99:45–63
https://doi.org/10.1007/s10846-019-01110-1

# Bi-objective Motion Planning Approach for Safe Motions: Application to a Collaborative Robot

**Sonny Tarbouriech**[1,2] · **Wael Suleiman**[1,2]

Received: 23 April 2019 / Accepted: 26 September 2019 / Published online: 14 November 2019
© Springer Nature B.V. 2019

**Abstract**

This paper presents a new bi-objective safety-oriented path planning strategy for robotic manipulators. Integrated into a sampling-based algorithm, our approach can successfully enhance the task safety by guiding the expansion of the path towards the safest configurations. Our safety notion consists of avoiding dangerous situations, e.g. being very close to the obstacles, human awareness, e.g. being as much as possible in the human vision field, as well as ensuring human safety by being as far as possible from human with hierarchical priority between human body parts. Experimental validations are conducted in simulation and on the real Baxter research robot. They revealed the efficiency of the proposed method, mainly in the case of a collaborative robot sharing the workspace with humans.

**Keywords** Motion planning · Safe motion · Collaborative robot · Human-robot cooperation

## 1 Introduction

In recent years, human-robot collaborative tasks has become a very active research field. In manufacturing, it is particularly interesting to carry out collaborative tasks that involve humans and industrial robots. However, the latter have been largely seen as dangerous machines that are kept inside security fences.

Therefore, before sharing the work space and interacting with a potentially harmful equipment, safety issue has always to be guaranteed. In this regard, all phases of the manipulator design have been considered. At the hardware level, modifications of the intrinsic properties of the robot can make it safer. For instance, by using light-weight materials [1], compliant passive systems [2] or safe actuator designs [3].

Another strategy is aimed at reacting in a fast and reliable way to an unexpected collision. To do this, collision detection methods have been developed [4, 5]. In the case of unforeseen contact, a process of collision reaction [6] is started to escape from the hazardous situation. An implementation of a realtime system that ensures human safety while being at very low distances of separation with a standard industrial robot is proposed in [7], the system relies on accurate measurements of human positioning in the workspace using a motion-capture system.

An optimization-based method that considers both the design and the control of a safe human-care robot is proposed in [8], that method has been only validated in simulation scenarios.

Ensuring safety during a human-robot cooperative task can also be done by avoiding any potential collision. The main pre-collision strategy consists in developing a realtime control system that senses the environment and adapts the robot's behavior according to an updated danger assessment [9–11].

In unstructured environments involving humans, it is clear that a realtime process is an indispensable tool to preserve safety. However, the overall efficiency will rely on the way the initial trajectory is defined, for instance, by providing a path that guarantees the human's safety using a motion planning algorithm, and then modify that path in realtime to deal with the dynamic environment.

**Electronic supplementary material** The online version of this article (https://doi.org/10.1007/s10846-019-01110-1) contains supplementary material, which is available to authorized users.

✉ Wael Suleiman
Wael.Suleiman@USherbrooke.ca

[1] Electrical and Computer Engineering Department, Faculty of Engineering, University of Sherbrooke, Sherbrooke, Canada

[2] Institut Interdisciplinaire d'Innovation Technologique (3IT), Universite de Sherbrooke, Sherbrooke, Canada

Most of the modern motion planning algorithms for robotic systems with high degrees of freedom are, generally, based on sampling methods since an exact consideration of the obstacle space is practically impossible [12].

In [13], a safety criterion was included in several motion planning algorithms. It is a simple version of the *Kinetostatic danger field* [14] which takes into account the overall position of the manipulator with respect to the obstacles. The safety measure is then embedded into a heuristic function that guides the exploration of the free configuration space.

An interesting concept of "Legible motion", which is a motion that communicates its intent to a human observer, has been proposed in [15]. However, as pointed out by the authors, this concept is suffering from several limits, for instance optimizing legibility leads to learning and optimizing non-convex functions in high-dimensional spaces.

A manipulation planner especially designed for human-robot interaction was presented in [16]. It generates safe paths that also improve the feeling of comfort perceived by humans during the interaction task. Human-based indications, such as his visibility and posture, lead to the development of specific cost functions that help the planner to select an appropriate path.

## 2 Contribution

The aim of this paper is to introduce a new safety-oriented motion planning approach for robotic manipulators operating in unstructured and human-populated environments (Fig. 1).

Our contribution is proposing a bi-objective optimization method combined with an exact collision checker algorithm [17]. Our method aims to help any sampling-based motion planning algorithm with asymptotic optimality feature to find the safest path, we also introduce a new variant of bi-directional RRT* that integrates our objective functions. The objectives are mainly evaluated in the Configuration-space (C-space) for reasons of speed and efficiency. Our approach relies on reducing the probability of a hazardous situation to occur by maximizing the clearance of the path.

In a manufacturing context, our algorithm will find a compromise between safety and performance of the task. However, it is obvious that our algorithm provides the safest path for a static environment as it is the result of a motion planning algorithm, therefore this path should be modified online to deal with a dynamic environment in realtime. The latter task will be the focus of our future work, however, it is worth mentioning a related work in the literature [18].

## 3 Motion Planning Algorithms

Despite their conceptual simplicity, sampling-based algorithms have been proven as an effective way to solve complex problems in high-dimensional configuration spaces, where the use of deterministic methods is computationally infeasible [19]. The key idea of such an approach is to avoid the exact representation of the C-space by only considering the information provided by randomly sampled states. Then, the problem resolution lies in finding a succession of collision-free connections between the start configuration to a state that respects the goal constraints.

Probably the most widespread sampling-based planners that employ randomization are the rapidly-exploring random tree (RRT) [20] and the probabilistic roadmap algorithm (PRM) [21]. Their inherent simplicity and effectiveness in finding a feasible path in complex spaces have made them a reference in many applications, including diverse robotic areas. These algorithms have the property of being probabilistically complete [22][23], that means their probability to find a solution, if one exists, approaches one as the number of samples tends toward infinity. However, the

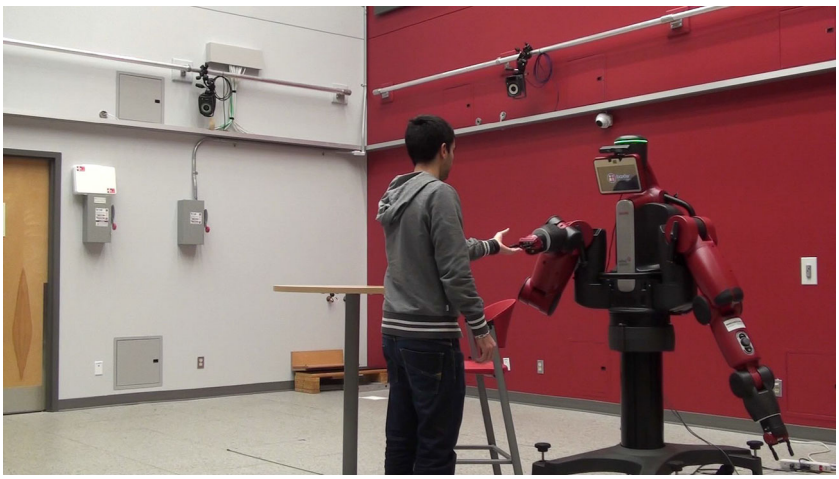


**Fig. 1** Interactive task detection and safe execution

quality of the provided solution is not taken into account during the path elaboration.

Recent researches have led to the development of numerous planners, most of them are extensions of RRT or PRM. In particular, the notion of cost has been introduced to assess the quality of a path. RRT* and PRM* [24] add to their respective basic version the property of being asymptotically optimal, that means that the cost of the returned solution converges almost surely to the optimal one.

Yet, these planners might lead to a solution that is far from optimal in the case of applications that are subject to time constraints, this is mainly because of the slow convergence rate. In particular, this would be a major drawback in our case because of the high dimension of the C-space and the relatively important computational complexity of the cost functions, those two elements may strongly impact the convergence process.

The changing needs in many fields that implies motion planning has resulted in the development of algorithms that focus primarily on finding the best compromise between the quality of the returned solution and the time needed to obtain it, those algorithms are potentially the most suitable for our application. The next paragraphs give an overview of some of those motion planning algorithms.

PRM and its extensions belong to the family of multiple-query algorithms. The main principle is to construct a topological graph that connects sampled states of the C-space; this defines a roadmap. This category of planners is particularly suitable for solving problems where multiple initial state and/or goal states are queried.

Even if PRM* guarantees asymptotic optimality, this condition is fulfilled at the expense of an explosive growth of the constructed graph. Dobson et al. [25] proposed the SPArse Roadmap Spanner (SPARS) algorithm, where the main idea is to relax the optimal property of PRM* to a near-optimal one by using graph spanners [26]. By doing so, a subgraph that contains only useful states and edges is constructed alongside the densest optimal graph. This has shown that better path quality solution can be provided for queries that are time-constrained. In an advanced version, SPARS2 [27], the near-optimally feature is preserved without the need for the dense graph to be developed. This allows for a considerable reduction in memory requirements and a production of high-quality path faster than the original version. In [28], a new stopping criterion for PRM-like methods is proposed, it yields a near-optimal solution within a finite time interval.

However, the above improvements in roadmap-based algorithms only apply when the path length is the unique criterion to optimize. In fact, some inherent features make them inappropriate within the context of this study. The main drawback comes from the graph representation that uses non-oriented edges to define paths between states. As a result of this drawback, these algorithms have to proceed with a two-phase approach: the construction of the graph in the first instance, and then the query phase that seeks for the best solution. That means that the global cost evolution cannot be extracted during the graph construction. As shown in Section 6, this data can significantly improve the performance of the optimization process if it is integrated into heuristic techniques. Moreover, for the same reason, those planners do not allow the specification of a termination condition when a solution that satisfies some criteria is found.

With the Fast Marching Trees algorithm (FMT*) [29], the authors suggest an efficient method for solving complex motion planning problems in high-dimensional configuration spaces. For a given problem, a single batch of samples is generated initially. Paths are then constructed using a marching method and stored in a tree structure. The sequential structure used here allow for a directly ordered search because the knowledge of the C-space does not evolve while the tree is being built. In addition, the planner uses a "lazy" behavior in its dynamic programming recursion. Thus, faster convergence rate to the optimal solution, compared to RRT* and PRM*, has been put forward by numerical experiments. Nevertheless, FMT* does not have the anytime property and, hence, suffers from two main inconveniences: (1) the whole solving process has to be restarted if the current resolution (i.e the C-space coverage) is too low, (2) no intermediate solution can be reported and so no time constraint on the planning task can be imposed.

An interesting quasi-anytime version of FMT* has been proposed in [30]. The presented MPLB planner successively restarts the search with a refined resolution and reuses samples and connections that were previously found. It also uses a heuristic that orders the promising nodes and rejects the one that cannot improve the current best solution. Yet, once again, the planner returns only one solution per batch and the running time is almost twice as long as FMT* for the same final resolution.

Similarly to [28], tree-based asymptotically near-optimal algorithms have been developed. LBT-RTT [31] is able to find solution paths with cost converges to an approximation factor of $1 + \epsilon$ of the optimal one. This algorithm offers a compromise between the speed of RRT and the path quality of RRT*. In this planner, many calls to the collision checker can be avoided, on the other hand, a lot of cost estimations on the edges have to be carried out. As, in our case, the collision checking is included in the cost evaluation (via the clearance objective), this approach is not suitable.

A widely used strategy in motion planning consists of setting up a bi-directional search to solve single-query problems. In this sense, a bi-directional variant of RRT , RRT-Connect [23], is able to find a solution much more

quickly than its original version, particularly in complex scenarios that involve high-dimensional space and cluttered environment. Upgrading to asymptotically-optimal planner has been proposed in [32] and [33]. Both papers present adapted versions of bi-directional RRT* with different heuristics to speed up the search.

The bi-directional RRT* seems to be the most suitable approach in the context of this study, this is because it meets the requirements in terms of computational time and quality of the provided solution and can be easily adapted to support our objective combination strategy. However, heuristics presented in the two previous works are primarily appropriate for the standard objective that aims at minimizing the Euclidean distance and their impact on the performances of the planner are not totally clear.

## 4 Bi-Objective Optimization Path Planning

In this section, we present two objective functions that will be evaluated for each potential path segment connecting two random states. Obviously, the computation time of the algorithm depends on the complexity of the objective functions, therefore we focused our efforts on the development of both relevant and computationally efficient objective functions. Both objective functions are positive functions.

### 4.1 Objectives Combination Strategy

The conventional way of solving a bi-objective optimization problem is to perform a weighted aggregation of all the objective functions in order to obtain a single-objective. Thus, two *positive* objectives, $f_1$ and $f_2$, can be linearly combined with the following function $lc$:

$$\mathrm{lc}(f_1, f_2) = \alpha_1 f_1 + \alpha_2 f_2 \tag{1}$$

where $\alpha_1$ and $\alpha_2$ are positive constants. This combination represents the cost of a path. Subsequently, the optimization process will select the path that minimizes or maximizes the sum of objective functions depending on the desired performance outcome.

In the case of a mixed problem, maximizing $f_1$ and minimizing $f_2$ for instance, a simple solution is to change the sign of one of the objectives. For instance, in our case $f_2^* = -f_2$ becomes the new objective to maximize. A new question, however, arises: is-it possible to unconditionally control the influence of each objective on the final result? In fact, it depends on several parameters: the value of the weighting factor, obviously; the definition of the cost function; and the conditions in which the task is executed. The last point is mainly cumbersome, this is because our objective is to develop a method that can be applied regardless of the environment in which the robot is operating. An example to illustrate the last point is for the clearance objective function, assuming that it is simply defined as the distance to the nearest obstacle, this objective would have less impact in a cluttered space where the robot has no choice but to pass close to obstacles than in a free space. On the other hand, in a clear environment, the clearance objective will tend to prevail over others aside from the weighting factors.

To overcome the above limitations, we developed a cost combination strategy that manages the effect of each objective in an efficient way regardless the operating environment.

An example to illustrate our cost combination strategy is given in Fig. 2, which shows two path segments $P^*$ and $\hat{P}$ in a sampling-based tree planner. $P^*$ and $\hat{P}$ are the unique paths from the root of the tree $q_{start}$ to $q^*$ and $\hat{q}$ respectively. Without loss of generality, let us suppose that we are interested in maximizing $f_1$ and minimizing $f_2$, our strategy to compare $P^*$ and $\hat{P}$ can be summarized as follows:

- We choose a set of weighting factors $\boldsymbol{\alpha} = \{\alpha_1, \alpha_2\}$, where $\alpha_i \geq 0$ and $\alpha_1 + \alpha_2 = 1$.
- Let us define $\mathrm{Cost}(q^*) = v^* = [f_1^* \quad f_2^*]^T$ is the cost associated to $P^*$, and $\mathrm{Cost}(\hat{q}) = \hat{v} = [\hat{f}_1 \quad \hat{f}_2]^T$ is the cost associated to $\hat{P}$.
- We define the following mapping function:

$$\begin{aligned} &\mathfrak{g} : \mathbb{R}^2_{\geq 0} \rightarrow \mathbb{R}_{\geq 0} \\ &\mathfrak{g}(v_1, v_2) = \alpha_1 \frac{v_1[1]}{v_2[1]} + \alpha_2 \frac{v_2[2]}{v_1[2]} \end{aligned} \tag{2}$$

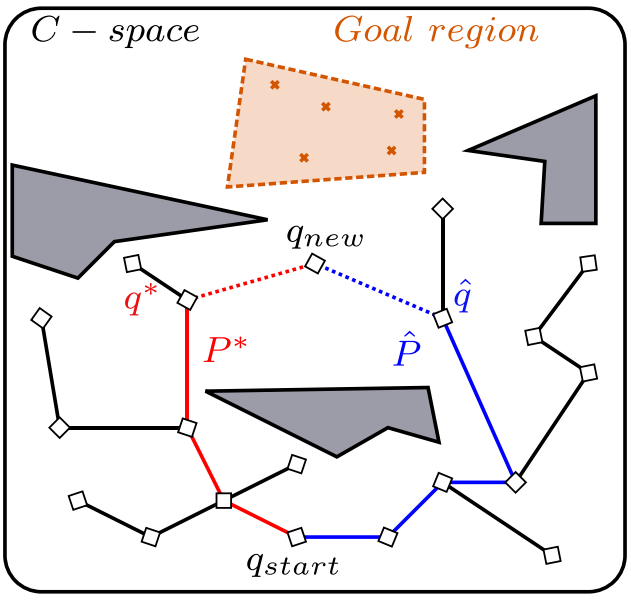



**Fig. 2** Cost function comparison between paths $P_1$ and $P_2$ to figure out to which one the state $q_{new}$ will be connected in a sampling-based tree planner

where $v[i]$ denotes the $i^{th}$ element of vector $v$.

- If $\mathfrak{g}(v^*, \hat{v}) > \mathfrak{g}(\hat{v}, v^*)$, the cost $v^*$ is considered better than $\hat{v}$ (noted $v^* \succ_{\mathfrak{g}} \hat{v}$) and the path $P^*$ is preferred over $\hat{P}$, otherwise the contrary applies.

The proposed strategy can be seen as a dynamic scaling strategy of the objective functions in such a way that the optimization is correctly taking place in non-dimensional and unit-less space.

### 4.2 Clearance Objective

To ensure safety of the robot as well as humans interacting with it, the first objective function to consider is the distance between the robot and the obstacles or humans. We can estimate the clearance along a path segment joining two configurations $q_a$ and $q_b$ as the distance provided by the continuous collision checking method [17], please refer to Appendix A for more details.

When the robot's environment is shared with humans, the clearance objective has to consider human safety at the highest level. To this end, each obstacle evaluated in Algorithm 4 (in Appendix A) is combined with a weighting factor $(wf)$ that reflects the will to get away from it. We classified obstacles in four categories and assigned a specific value to each, by increasing order of safety priority:

- For an object, $wf = 1$
- For a human arm or leg part, $wf = 0.8$
- For a human torso, $wf = 0.6$
- For a human head, $wf = 0.5$

Note that another set of weighting factors can be used as long as it satisfies the above order of safety priority.

Figure 3 illustrates the purpose of the above weighting factor choice. Recall that our objective is to maximize the distance to obstacles, therefore an obstacle is perceived closer than it actually is when its distance is multiplied by a smaller weighting factor.

To take this parameter into account, the procedure presented in Algorithm 4 remains exactly the same, but the sorting rules of the priority queue that gathers the segments have to be adapted. To know which of two segments $s_1$ and $s_2$ needs to be evaluated first, we call Algorithm 1. In doing so, weighting factors are considered only when the two segments are ensured to be collision free. If this is not the case, we keep evaluating the most likely colliding segment first to avoid computational overhead.

The cost resulting from the combination of two path segments, which have costs of $c_1$ and $c_2$, is computed by considering the worst case : Note that, the cost returned by Algorithm 4 is no longer necessarily the minimum obstacle distance but it is the one evaluated as the most dangerous.

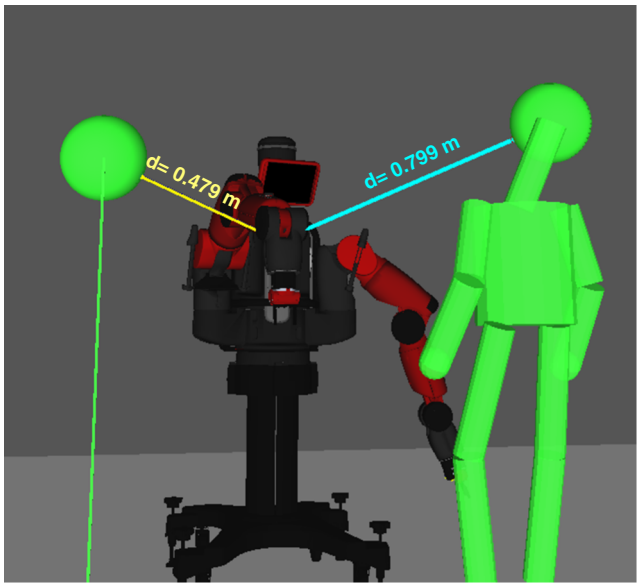



**Fig. 3** Effect of the weighting factor: even if the balloon is closer to the robot than the human head, the latter is considered more dangerous when the distances are multiplied by the corresponding factors

It always has to be associated with its weighting factor, particularly for the cost combination.

### 4.3 Human Awareness Objective

Another important issue regarding the human safety is the human-robot awareness [34]. When one or many people enter in the vicinity of the robot (inside or just outside the reachable space), it is necessary to ensure that the manipulator movements are seen by everyone, otherwise that may increase the risk of creating a dangerous situation. In order to achieve this, we developed an objective function that focuses on keeping the robot arm in the humans' field

**Algorithm 1** Priority queue sorting rule.

```
procedure PRIORITY_SEGMENT(s1, s2)
    if s1.δ > 0 AND s2.δ > 0 then
        if (s1.δ * s1.wf) < (s2.δ * s2.wf) then
            return s1
        else
            return s2
        end if
    else
        if s1.δ < s2.δ then
            return s1
        else
            return s2
        end if
    end if
end procedure
```

of view. The objective is to minimize the angular distance between the human gaze direction and the axis defined by the robot's end-effector position and the human head, 3D occupancy grid of the human awareness is given in Fig. 4. For the cost of this objective between two configurations $q_a$ and $q_b$, we linearly interpolate along the path and return the worst value among the tested states.

### 4.4 Cost function definition

Our cost function has two definitions depending on the root of the segment:

- By default, the cost function for a state $q^*$ (Fig. 2), is defined as follows:

$$\text{Cost}(q^*) = \begin{bmatrix} P^*.c \times P^*.wf \\ P^*.\bar{c} \end{bmatrix} \in \mathbb{R}^2_{\geq 0} \tag{3}$$

  Where $P^*$ is the unique path from the root of the tree to $q^*$. $P^*.c$ and $P^*.wf$ are respectively the clearance cost and weighting factor defined in Section 4.2. $P^*.\bar{c}$ is the human awareness cost defined in Section 4.3.
- When the root is different from the tree root, we define the cost function as follows:

$$\text{Cost}_{q^*}(q_{new}) = \begin{bmatrix} \mathfrak{L}(q^*, q_{new}).c \times \mathfrak{L}(q^*, q_{new}).wf \\ \mathfrak{L}(q^*, q_{new}).\bar{c} \end{bmatrix} \tag{4}$$

  where the function $\mathfrak{L}(q_a, q_b)$ denotes the straight-line path from $q_a$ to $q_b$:

$$\mathfrak{L}(q_a, q_b) = (1 - \lambda)q_a + \lambda\, q_b : \quad \lambda \in [0, 1] \tag{5}$$

  Note that the cost in Eq. 4 consists of the worst costs of clearance and awareness objectives among the states between $q^*$ and $q_{new}$, which are obtained by linear interpolation.

We also define a new operator $\oplus$ to combine two cost functions:

$$\text{Cost}(q^*) \oplus \text{Cost}(\hat{q}) = \min\left(\text{Cost}(q^*), \text{Cost}(\hat{q})\right) \tag{6}$$

where $\min(v_1, v_2)$ is the componentwise minimum of the vectors $v_1$ and $v_2$.

Given a tree $G = (V, E)$, where $V$ and $E$ are respectively the vertices and edges, in the same way as in [24], we introduce the function Parent : $V \rightarrow V$ as a function that maps a vertex $v \in V$ to the unique vertex $u \in V$ such that $(u, v) \in E$. By convention, if $q_0 \in V$ is the root vertex of $G$, $\text{Parent}(q_0) = q_0$.

It can be easily verified that:

$$\text{Cost}(q) = \text{Cost}(\text{Parent}(q)) \oplus \text{Cost}_{\text{Parent}(q)}(q) \tag{7}$$

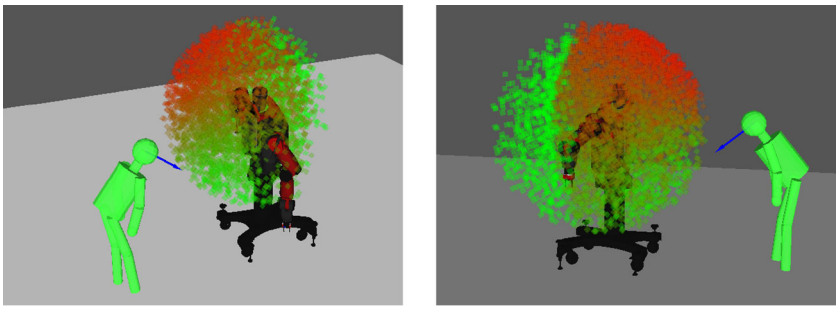
## 5 Variant of Bi-directional RRT*

Now we introduce our proper variant of the bi-directional RRT* algorithm, this algorithm uses effective methods that limit as much as possible the inherent computational burden of our cost estimation and provide a high-quality solution in a reasonable time. Some of the implemented heuristics are inspired by those presented in [32] and [33], but adapted to our case of bi-objective and greatly improve the performance in this context.

Our main contribution here lies in the development of techniques that are particularly suitable for the use of minimax objective functions (that aim at finding the "best" worst case). Furthermore, we integrated the C-Forest parallelization framework [35] to take full advantage of the multi-core technology that almost every recent computer is equipped with.

First, let us introduce the variant of RRT*, Algorithm 2, that we developed for our framework. It is mainly based on the conventional implementation of RRT* proposed in [24] while the difference is highlighted for clarity purpose.

Algorithm 2 shows that, similarly to RRT*, the following procedure is applied each time an attempt is made to add a



(a) (b)

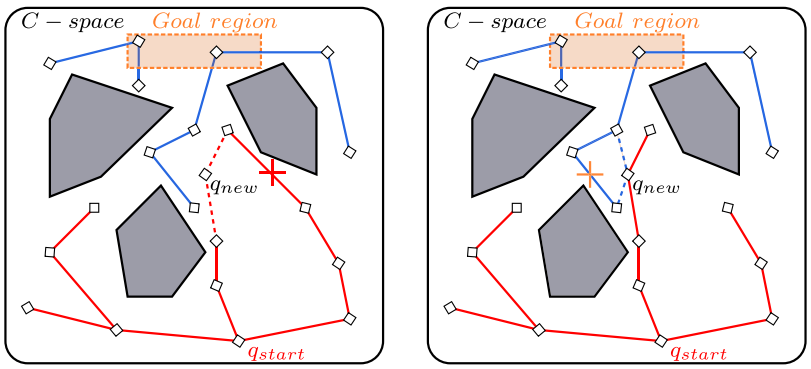
**Fig. 4** 3D occupancy grid of the human awareness: colors vary from green to red according to the human gaze deviation from the end-effector. If the distance between the human and the end-effector is greater than a threshold, the criterion is then considered fulfilled

**Algorithm 2** Variant of RRT*.

$V \leftarrow \{q_{start}\}$; $E \leftarrow \emptyset$;
**for** $i \leftarrow 1 : n$ **do**
  $q_{rand} \leftarrow \text{SampleFree}_i$;
  $q_{nearest} \leftarrow \text{Nearest}(G = (V, E), q_{rand})$;
  $q_{new} \leftarrow \text{Steer}(q_{nearest}, q_{rand})$ ;
  **if** ObtacleFree$(q_{nearest}, q_{new})$ **then**
    $Q_{near} \leftarrow \text{Near}(G = (V, E), q_{new}, \min\left(\gamma_{RRT^*}(\log(card(V))/card(V))^{1/d}, \eta\right)$;
    $V \leftarrow V \cup q_{new}$;
    $q^* \leftarrow q_{nearest}$; $v^* \leftarrow \text{Cost}(q_{nearest}) \oplus \text{Cost}_{q_{nearest}}(q_{new})$;
    **for each** $q_{near} \in Q_{near}$ **do** ▷ Connect along a best-cost path
      **if** CollisionFree$(q_{near}, q_{new}) \wedge \text{Cost}(q_{near}) \oplus \text{Cost}_{q_{near}}(q_{new}) \succ_{\mathfrak{g}} v^*$ **then**
        $q^* \leftarrow q_{near}$; $v^* \leftarrow \text{Cost}(q_{near}) \oplus \text{Cost}_{q_{near}}(q_{new})$
      **end if**
    **end for**
    $E \leftarrow E \cup (q^*, q_{new})$;
    **for each** $q_{near} \in Q_{near}$ **do** ▷ Rewire the tree
      **if** CollisionFree$(q_{new}, q_{near}) \wedge \text{Cost}(q_{new}) \oplus \text{Cost}_{q_{new}}(q_{near}) \succ_{\mathfrak{g}} \text{Cost}(q_{near})$ **then**
        $q_{parent} \leftarrow Parent(q_{near})$;
      **end if**
      $E \leftarrow (E \setminus \{(q_{parent}, q_{near})\}) \cup \{(q_{new}, q_{near})\}$
    **end for**
  **end if**
**end for**
**return** $G = (V, E)$;

state in the tree: a connection is attempted between a random state and its nearest state in the tree. If the resulting path segment is allowed, that means that it satisfies all constraints and remains in the free space, then the state is added to the tree as a leaf. A second phase intends to ascertain the best position of this state in the tree by testing the rewire options with its nearest neighbors, and then keeping the connections that maximize or minimize a given cost function. The rewiring process is illustrated in Fig. 5.

Our bi-directional RRT* planner is based on the above-mentioned variant of RRT*, and consists primarily of a merger between RRT-Connect and RRT*. As in previous bi-directional RRT algorithms, two complementary operations are executed at each iteration. First, one of the trees is extended toward a newly sampled state (Fig. 5a) and, if it succeeded, it is attempted to connect the two trees through this latest configuration (Fig. 5b). In prior works [32, 33], two trees are grown from the start and goal nodes and expanded in order to establish one or more connections between them. We extended the scope of application to consider the goal not only as a unique configuration but also as a region in the C-space, depending on the task definition.




(a) The start tree is extended toward a new sample $q_{new}$ (b) A connection is made between the trees through $q_{new}$

**Fig. 5** Illustration of the bi-directional RRT* functioning when the traveled distance is optimized

In fact, this feature is particularly useful when planning for redundant arm manipulators because it is often queried to reach a specific pose for the end-effector, and, most of the time, this could be accomplished by different joint states. Thus, we are referring here to a set of goal trees that are rooted in different states, all satisfying the goal constraints. Their data about vertex and edges are stored in a shared structure that let the algorithm behave in a similar way to the case of a unique tree.

## 5.1 Heuristics

A description of each specific feature that we developed and is part of the final implementation is given below:

### 5.1.1 Heuristic rejection

At each iteration, a newly sampled state $q_n$ is treated. Before attempting to integrate it as a candidate waypoint, it is relevant to assess its inherent potential to provide an improvement to the research of a path. Therefore, we use a set of admissible heuristics (one for each objective) to ascertain that a best cost could be obtained if the motion passes through this state. For our application, we only need one heuristic function to cover the objective functions.

A heuristic function for minimax objectives:

$$h_{minimax}(q_n) = \text{Cost}_{q_{start}}(q_n) \oplus \text{Cost}_{q_{goal}}(q_n) \tag{8}$$

If many goal configurations exist, we consider the nearest from $q_n$.

The costs of Eq. 8 is combined according to our strategy, and the resulting optimistic cost is then compared to the current best solution. If it is worse, the sample is directly rejected and a new iteration is started.

### 5.1.2 Fast Estimation of Motion Cost

When an attempt is made to insert a new sample, the best position in a given tree is chosen by testing the rewire options with its nearest neighbors, and then keeping the connections that optimize a given cost function. As shown by Karaman and Frazzoli [24], this method guarantees asymptotic-optimality of RRT* and this property is preserved for the bi-directional version [33]. In this process, the number of selected neighbors for evaluation increases with the number of states in the tree. Because the computation of the exact motion cost is relatively cumbersome in our case, the rewiring operation can be separated into two phases. A fast estimation of motion cost is computed for all potential connections that could link the new state and its neighbors. Similar to the previous feature, this evaluation uses an admissible heuristic, which is based solely on the costs of the states that delimit the motion. For our objectives, however, determining the motion cost usually requires interpolation, therefore the following formula is first applied to give an upper bound estimation of the cost between $q_a$ and $q_b$:

$$\text{Fast\_Path\_Cost}(q_a, q_b) = \text{Cost}(q_a) \oplus \text{Cost}(q_b) \tag{9}$$

In the second phase, the most promising neighbor (the one having the better estimated cost from the root of the tree) is selected and the exact motion cost is computed. If this neighbor remains the most promising (the real motion cost is still higher than the upper bound estimation of other neighbors), then the connection is established without considering the other candidates. Otherwise, the operation is reiterated while another rewiring option can possibly be better. The neighbor selection process is detailed in Algorithm 3. This procedure makes a call to the function Sort(), which employs our combination strategy to reorder the state vectors according to their associated costs in order to get the most promising neighbor first. The function WorstCost(), that is also called in this algorithm, returns the set of all individual objective functions taken in the worst case. If an objective is to be maximized then the worst value is 0, otherwise an infinite value is returned when the objective is to be minimized.

**Algorithm 3** Neighbor selection using fast cost estimation.

```
procedure SELECT_BEST_NEIGHBOR(q_new, tree)
    neighbors ← tree.K_Nearest(q_new)                ▷ k nearest
    for i ← 1 : k do
        inc_cost ← Fast_Path_Cost(neighbors[i], q_new)
        costs[i] ← Combine(neighbors[i].cost, inc_cost)
    end for
    Sort(neighbors, costs)
    best_cost ← Worst_Cost()
    best_nbh ← NULL
    for i ← 1 : k do
        inc_cost ← Path_Cost(neighbors[i], q_new)
        costs[i] ← combine(neighbors[i].cost, inc_cost)
        if Collision_Free_Path(neighbors[i], q_new) then
            if costs[i] ≻_g best_cost then
                best_cost ← costs[i]
                best_nbh ← neighbors[i]
                if i ≠ k AND costs[i] ≻_g costs[i + 1] then
                    break
                end if
            end if
        end if
    end for
    return best_nbh
end procedure
```

#### 5.1.3 Local biasing

Biasing in sampling-based motion planning is a common practice that aims at steering the samples towards the most interesting regions of the C-space rather than leaving this process totally random. In this regard, numerous techniques have been proposed and their interests strongly depend on the targeted application. As minimax objectives are used in our framework, the development of a particularly adapted local biasing strategy is necessary. Because the quality of a solution can be highly degraded by a small part of the path, it is opportune to guide the search towards the part that affects the most the final cost, an example of the local biasing strategy is given in Fig. 6.

Note that the local biasing is attempted once a feasible collision-free path is found, hence the objective is to enhance the path quality by locally improving the critical path segments.

#### 5.1.4 Anytime Behavior

The purpose of this functionality is to obtain a path as fast as possible and using the remaining available time to improve its quality. This is made possible by disabling the optimality research until a first solution is found. Thus, up to that moment, the algorithm behaves exactly like RRT-Connect. Concretely, the planner considers a unique nearest neighbor when trying to connect a new node. As the aforementioned heuristic rejection feature needs a solution to be found before being activated, the anytime behavior is expected to contribute to improving the convergence rate.

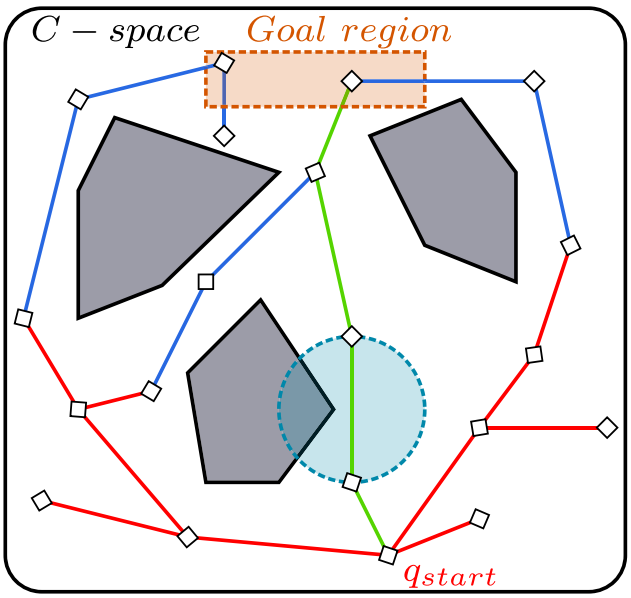



**Fig. 6** Local biasing example when the optimization process aims at maximizing the clearance. Considering the current best solution (green), the sampled area (blue) is located in the vicinity of the segment that has the worst clearance (as it is the only minimax objective here). The selected region is defined by a ball that passes through the two endpoints of the segment

#### 5.1.5 Avoid Minimax Objective Side Effects

As mentioned before, minimax objective are particularly suitable for our application to avoid any vulnerable situation. However, their efficiency is bounded by the costs of imposed configurations (start or goal states). Thus, if one of these states has a poor cost, the optimization process will have no room to improve the quality of the path; this is because our cost combination strategy only takes into account the worst case.

To get around this problem, we implemented a double-layer cost system which operates as follows: the first layer *(Layer 1)* is the standard one that holds the information about the worst value obtained from the root of the tree. The second layer *(Layer 2)* considers if the robot is escaping from an undesired posture; i.e. an objective keeps improving, from the root of the tree, we store the best value when combining two costs. From the moment at which the objective function starts decreasing, the combination considers the worst case again. An illustration of the implementation of the two layers is given in Fig. 7.

When comparing a minimax objective for two path segments, the following cases arise:

- The associated values in *Layer 1* of the segments are distinct: in this case only the values in *Layer 1* are considered as before.
- The associated values in *Layer 1* are equal: *Layer 2* is called, in this case, to decide which path segment has to be prioritized based on the criterion in question.

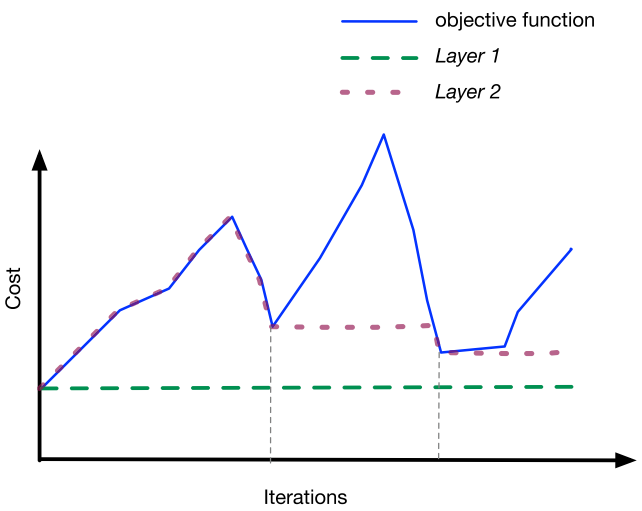
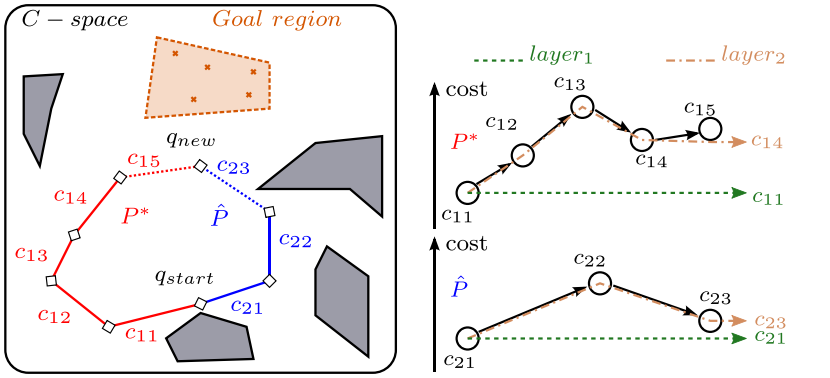



**Fig. 7** Illustration of the implementation of Layer 2 to avoid minimax objective side effects




(a) Representation of the C-space (b) Path cost combination behavior

**Fig. 8** Example: Trying to connect the state $q_{new}$ to the tree through either path $P^*$ or $\hat{P}$ when the clearance is the only objective to be optimized. $c_{1i}$ and $c_{2i}$ are respectively the clearance costs of the segment $i$ of paths $P^*$ and $\hat{P}$. Considering the worst case only, no choice can be made because $P^*$ or $\hat{P}$ have the same cost $c_{11} = c_{21}$. However, with the second layer, path $P^*$ has cost $c_{14}$ and $\hat{P}$ has cost $c_{23}$, so $P^*$ is chosen

Another illustration example of our two-layer strategy is presented in Fig. 8.

#### 5.1.6 C-Forest Framework

We adapted the C-Forest parallelization framework for our bi-directional RRT* implementation. The C-Forest framework consists of data exchange strategy between multiple threads of the planner that are executed in parallel. In particular, the states that constitute the current best solution are shared with the other threads. For more information about this framework, the reader is referred to [35]. Implementation details are given in [36].

#### 5.1.7 Pruning Process

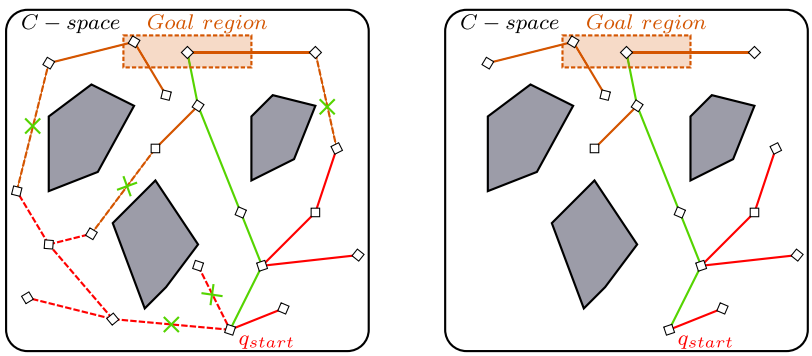
This functionality is integrated in the C-Forest framework but we present it separately because it can also be used as an independent feature. Once a solution is found, there may exist some states in the tree that become useless because they cannot improve the solution. Thus, it would be appropriate to remove them from their respective tree. This allows the algorithm to avoid unnecessary efforts that only slow down the process. In practice, that means that the heuristic rejection presented above is now applied to vertex which are already in the data structure. Note that removing a state also erases the branches that are attached to it. It is therefore likely that a potentially interesting state in the tree could be deleted due to a bad foregoing configuration. The overall performance of the pruning process will be evaluated in Section 6. An illustration of the pruning process is given in Fig. 9.

### 5.2 Analysis

For any motion planning algorithm, two important properties, which are the probabilistic completeness and asymptotic optimality, need to be assessed.

**Lemma 1** *The RRT* variant in Algorithm 2 with the proposed heuristics is probabilistically complete.*

*Proof* Let us first recall the definition of probabilistic completeness. As stated in [24], Algorithm 2 is probabilistically




(a) Undesirable paths and their children are selected for removal (b) New representation of the trees after pruning

**Fig. 9** Illustration of the pruning process applied to both start and goal trees when the clearance is optimized. The cost of the current best solution (green) is used as a reference

complete, if, for any robustly feasible path planning problem $(Q_{free}, q_{start}, Q_{goal})$,

$$\liminf_{n\to\infty} \mathbb{P}\left\{\exists q_{goal} \in V_n \cap Q_{goal} : q_{start} \text{ is connected to } q_{goal} \text{ in } G_n\right\} = 1$$

where $G_n = (V_n, E_n)$ is the graph at the iteration $n$.

The probabilistic completeness of Algorithm 2 is a direct result from the probabilistic completeness of RRT* [24] as long as the proposed heuristics preserve that feature. Among the proposed heuristics, only the heuristic of fast cost estimation in Section 5.1.2 (Algorithm 3) could impact the probabilistic completeness. However, one can figure out from Algorithm 3 that the graph connectivity is forced by an incremental construction, hence preserving the probabilistic completeness [20]. □

**Lemma 2** *The RRT* variant in Algorithm 2 is asymptotically optimal with respect to the mapping function* $\mathfrak{g}$ Eq. 2.

*Proof* Algorithm 2 is asymptotically optimal with respect to the mapping function $\mathfrak{g}$ if, for any path planning problem $(Q_{free}, q_{start}, Q_{goal})$,

$$\mathbb{P}\left(\lim_{n\to\infty}\left\{\sup_{\mathfrak{g}} Y_n = v^*\right\}\right) = 1$$

where:

– for a sequence $\{X_n \in \mathbb{R}^2_{\geq 0}\}_{n\in\mathbb{N}}$ of random vectors:

$$\sup_{\mathfrak{g}} X_n = X^* \text{ such that } \mathfrak{g}(X^*, X_n) \geq \mathfrak{g}(X_n, X^*) : \forall n$$

– $v^* \in \mathbb{R}^2_{\geq 0}$ is the cost of the robustly optimal path, $\sigma^* \in Q_{free}$, satisfies that for any sequence of collision-free paths $\{\sigma_n\}_{n\in\mathbb{N}}$, such that:

$$\lim_{n\to\infty} \sigma_n = \sigma^*$$
$$\lim_{n\to\infty} \mathfrak{g}(v^*, \text{Cost}(\sigma_n)) \geq \mathfrak{g}(\text{Cost}(\sigma_n), v^*)$$

– $Y_n \in \mathbb{R}^2_{\geq 0}$ is the extended random vector corresponding to the cost of best-cost solution, with respect to the mapping function $\mathfrak{g}$, included in the graph returned by the algorithm at the end of iteration $n$.

Using **Theorem 38 (Asymptotic optimality of RRT*)** in [24], Algorithm 2 is Asymptotically optimal if the following three conditions are satisfied:

1. **Condition 1**: Algorithm 2 is probabilistically complete.
2. **Condition 2**: The Cost() function is monotonic with respect to $\mathfrak{g}$.
3. **Condition 3**: $\gamma_{RRT^*} > (2\,(1+1/d))^{1/d}\left(\frac{\mu(Q_{free})}{\xi_d}\right)^{1/d}$. Where $\mu(Q_{free})$ and $\xi_d$ design, respectively, the Lebesgue measure of the obstacle-free space and the volume of the unit ball in the d-dimensional Euclidean space.

**Condition 1** is the result of Lemma 1.

**Condition 2** is satisfied, and it can be easily verified using Eq. 7 in a recursive way.

**Condition 3** is forced to be true in the same way as in the standard implementation of RRT*. As a result, Algorithm 2 is Asymptotically optimal with respect to $\mathfrak{g}$. □

## 6 Experimental Results

The efficiency and performance of the described method have been tested in simulation and on the real robot Baxter from Rethink Robotics Inc. Baxter is equipped with two 7-DOF manipulator arms. In this study, we only focus on the control of one arm but our method can be easily extended to also consider both arms simultaneously. Our program has been implemented using the platform MoveIt! [37] that integrates the motion planning library OMPL [38] and communicates via ROS (Robotic Operating System) API interface [39].

### 6.1 Distance Approximation Function

Even though the dynamic collision checking method [17] tends to limit the amount of distance computations, those operations remain widely called during the planning process. Moreover, computing the exact minimum distance between two complex and concave objects is a computationally expensive operation. Moveit! uses the Flexible Collision Library (FCL) [40] as its primary collision checking library. Besides collisions check, FCL can also compute the distance between two non-overlapping objects.

To accelerate the execution of our algorithm, we developed a computationally efficient distance computation function that gives a reasonable approximation of the distance between a link of the robot and an obstacle. This function is based on simplifying the environment representation by transforming each obstacle into its corresponding oriented bounding box (OBB). The Baxter robot is also approximated by its collision geometry model from the Unified Robot Description Format (URDF) file. Thus, the arms are modeled as cylinders and boxes. Each part of the arm is then subdivided into an optimal number of spheres that encompass the initial volume. Finally, the minimum distance between an obstacle and a link becomes the minimum distance between the OBB and the spheres, hence it can be easily computed.

### 6.2 Environment Representation

In order to create an autonomous system, a RGB-D sensor, Asus Xtion PRO 3D, has been attached on top of the Baxter robot's head display. The movement of the Baxter's head pan joint offers the ability to visually scan a large part of the environment. The OctoMap framework [41] integrated into Moveit! is then used to generate a 3D occupancy grid that gives a collision model of the space. The sonar sensors positioned around the robot's head are used to detect human presence and orientate the camera accordingly. A specific model is provided to represent humans in the environment. In particular, it allows us to distinguish the different parts of the body for the assessment of the hazard level.

In order to reduce the computation complexity, we developed a simplified model for the obstacles to reduce the amount of distance evaluations. To this end, two operations are applied to refine the OctoMap, which is an input for our method. First, all static objects that are outside a certain distance from the limits of the robot's workspace are neglected. Then, we attempt to gather all remaining occupied leaves of the octree nodes into larger boxes. Our model is based on a compromise between a fair representation of the environment and having a minimal number of obstacles. An example of such representation is given in Fig. 10. Note that this representation is only used in the real experiments with the Baxter robot; on the other hand, the simulation experiments, such as those in Fig. 11, use the geometric models of the environment.

### 6.3 Validation in Simulation

The first validation phase of our approach was made through a simulated scenario using the Gazebo simulator [42] as an interface for the Baxter robot. All experiments are performed on Intel(R) Core(TM) i5-3470 CPU @ 3.20 GHz PC with 8 GB RAM. As a first step, we tested each unit independently in order to figure out the influence of the parameters on the overall performance of our algorithm.

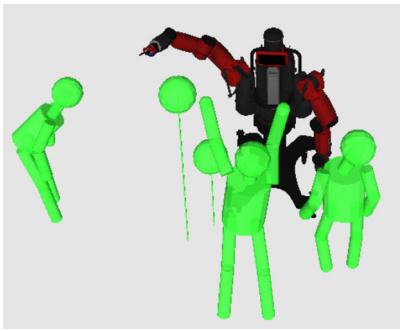


**Fig. 11** Validation scenario with human presence

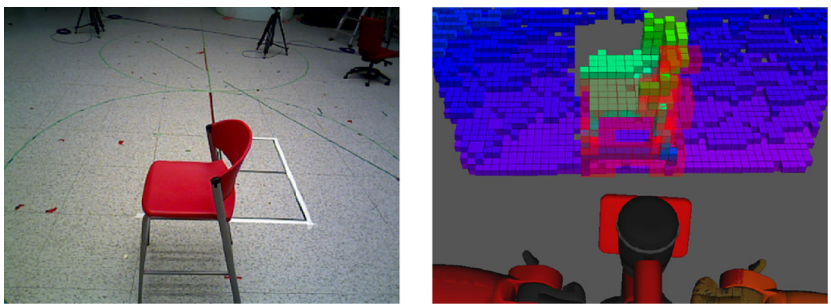
The considered scenario is the following: the right arm's end-effector is queried to reach successively two particular configurations for the joints. These states are located on the sides of two balloons tethered to the ground by thin rods, the robot working space is also populated by people as depicted in Fig. 11.

#### 6.3.1 Comparing with RRT*

We first compare the results of our bi-RRT* planner with those of RRT*. However, note that the developed framework does not allow a fair comparison with existing planners because the implemented bi-objective strategy is part of our contribution. However, it is worth investigating the performance of our algorithm with respect to the algorithm that is inherited from, i.e. RRT*. Recall that RRT*, by default, minimizes the path length.

A cost combination is defined as the following set of values: $v$ = {Clearance; Awareness}. We have chosen two sets of $\boldsymbol{\alpha}$, the first one prioritizes the human-safety $\boldsymbol{\alpha}_1$={0.7; 0.3}, while the second prioritizes the human-awareness $\boldsymbol{\alpha}_2$={0.3; 0.7}.

200 motion planning tasks have been conducted and, if no specification is provided, we always consider the mean value of the collected data to get an idea of the algorithm overall performance. The comparison is given in Fig. 12, it



(a) Camera view (b) Collision model in Moveit!

**Fig. 10** Representation of the scene: the chair and the ground are modeled with occupancy boxes using OctoMap. Only transparent red boxes are considered in the motion planning process

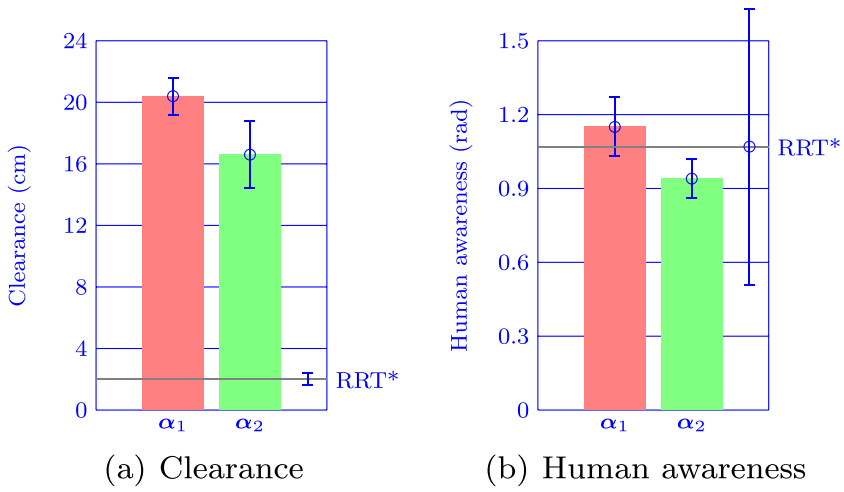



**Fig. 12** Cost results in simulation

shows that, as expected, the returned values of the clearance and human awareness are correlated with $\boldsymbol{\alpha}$. Moreover, as RRT* minimizes the path length, the obtained path is generally close to obstacles as shown in Fig. 12.

### 6.3.2 Analyzing the performance of our algorithm

In this section, we analyze in depth the performance of our planner with $\boldsymbol{\alpha}_1$={0.7; 0.3}, which prioritizes the human-safety. In this case, 200 motion planning tasks have been also conducted, and four threads of the planner are always launched in parallel and they exchange information only if the C-Forest framework is enabled. At a given time, only the best solution among all threads is considered.

Instead of testing the effect of activating only one feature at a time on our bi-RRT* planner, we have found that

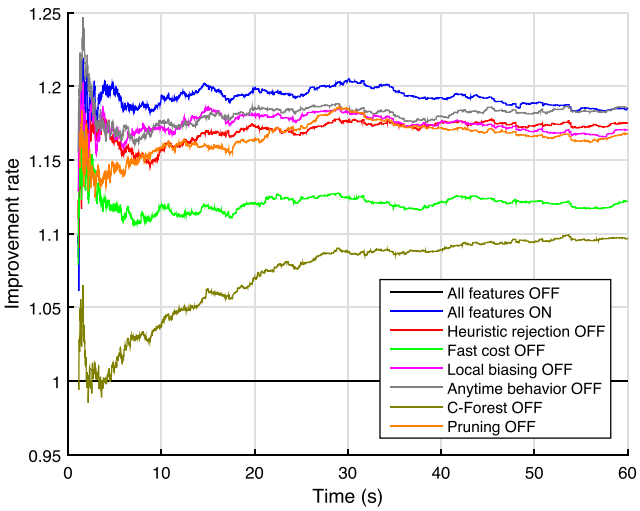



**Fig. 13** Evolution of the improvement rate over time

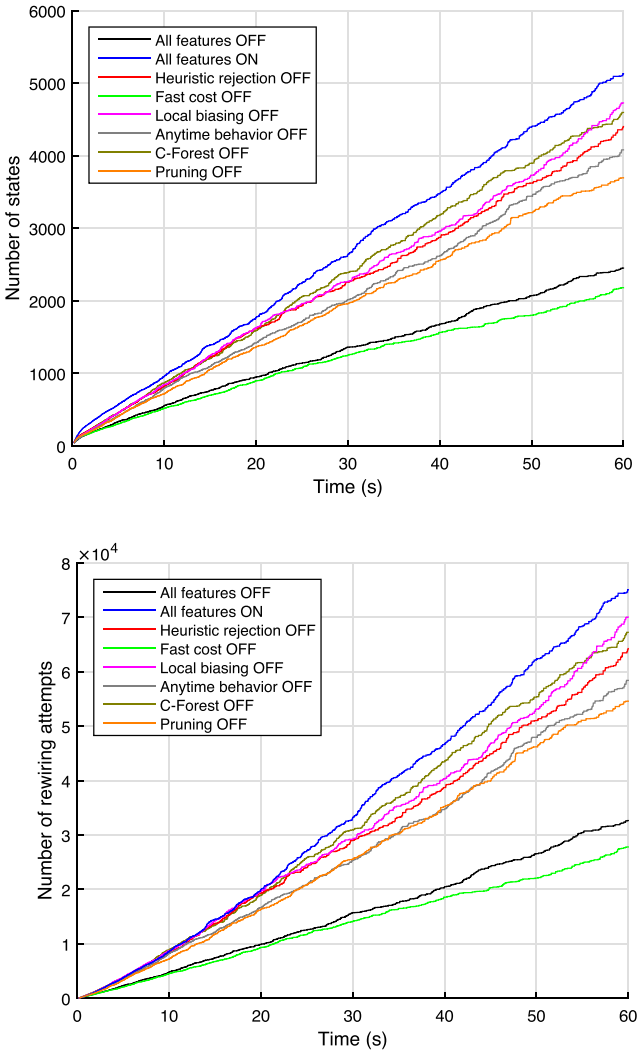
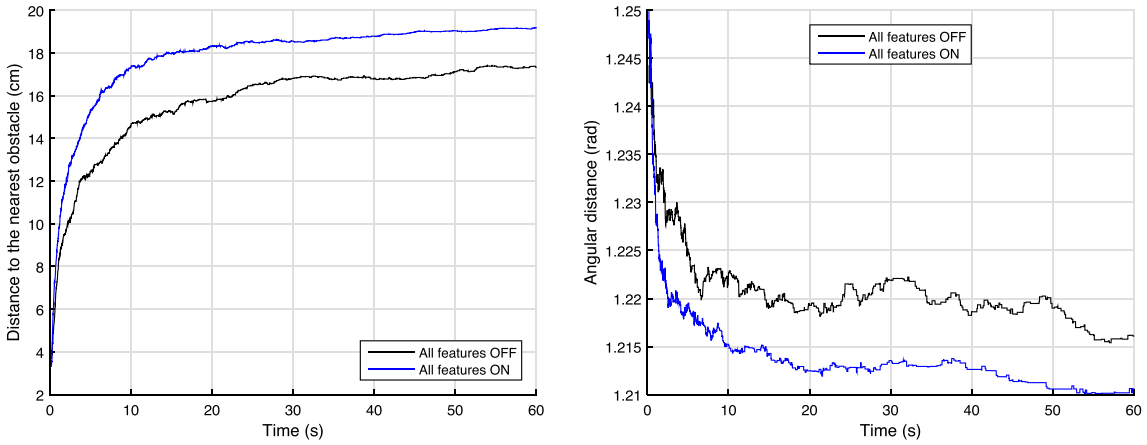



**Fig. 14** Evolution of the number of states generated and rewire options tested over time




(a) Clearance (b) Awareness

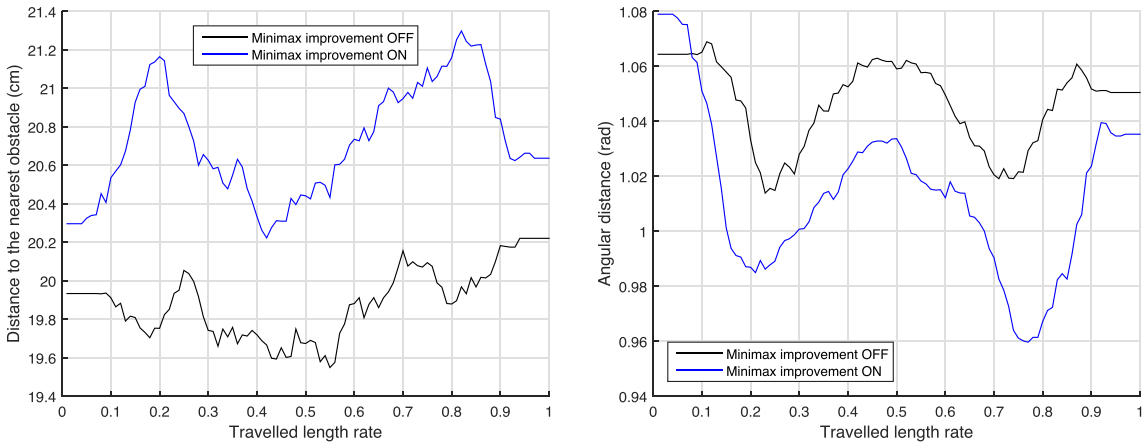
**Fig. 15** Comparative evolution of costs

it is more significant to study the effect of deactivating one feature at a time. Indeed, the numerical experiments show that there is a considerable dependency between the features. Moreover, a relevant benchmarking cannot be made for each objective independently because they are optimized as a set and strongly depend on the value of their associated factor. The comparison is therefore made following the rules of the method presented in Section 4.1. First, a function $\mathcal{F}_1$ is computed when all features are deactivated and serve as a reference. Then, a new function $\mathcal{F}_2$ that results from a path cost that integrates some of the features is computed. We evaluate the improvement produced by the later configuration by generating the rate $\frac{\mathcal{F}_2}{\mathcal{F}_1}$, results are reported in Fig. 13.

The first important observation is that the developed bi-RRT* with all features activated leads to a non-negligible improvement over the basic version. The global quality of the obtained costs is about 20% better. These values does not change a lot over time. It is worth noting that the manner in which we defined our objectives (with a minimax optimization) and the compromise imposed by the numerous constraints to be satisfied do not allow to obtain a huge improvement.

From Fig. 13, it can be noticed that when all features are activated, we generally get the best paths. That means that all features make a contribution of varying degrees to the quality of obtained solutions. The one that has the greatest impact is the C-Forest framework, as its deactivation is the most meaningful.

Moreover, the heuristic rejection feature is clearly less efficient than the fast estimation of path cost feature. This makes sense since the first strategy only considers




(a) Clearance (b) Awareness

**Fig. 16** Profile of typical cost evolutions along the planned path

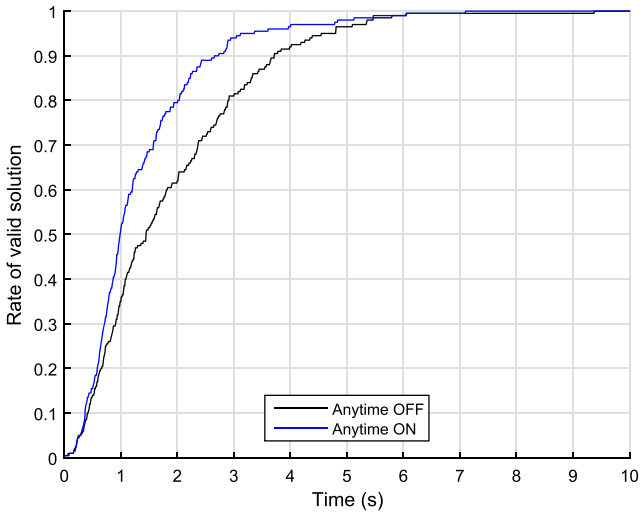



**Fig. 17** Rate of valid solution (number of motion planning tasks for which at least one collision-free path has been found) over time for a single thread

the cost of a unique state to predict the best path that could be generated by including it, this results in an overestimation that generally does not allow the algorithm to reject the state because it has the inherent potential to be part of a better solution, Fig. 14 confirms this observation. It gives the corresponding number of states and rewire options inside the trees that have been carried out during the path planning. This shows that the fast cost estimation has a strong influence on those two parameters and allows the planner to have a better knowledge of the space more quickly. Other features are more or less useful depending on context, however, the combination of all features provides an interesting tool for applications in unconstrained environments.

The comparisons of the cost evolution when turning all the features ON/OFF, for each objective, are presented in Fig. 15. They effectively reveal the significant advantage of the complete framework regarding constraint satisfaction. Note that all the criteria are improved simultaneously and significantly.

The feature that aims at avoiding minimax objective side effects has also been evaluated. Figure 16 illustrate the evolution of each minimax objective along the motion when this feature is activated/deactivated. Since the planner is bi-directional, the process of escaping from a bad initial state is applied for both of the start and goal trees. That lets the planner produce really safer path when looking at the overall displacement, and avoid the problem of getting stuck in an imposed local minimum.

Even when a complex task is being carried out, e.g. the initial or goal pose is close to obstacles, the planner is able to find a solution quickly. The bi-directional implementation doubtless plays a major role to this end. The anytime behavior also contributes to reducing the time needed to find a first solution. Figure 17 indicates that the usefulness of this feature is more significant when a trivial solution does not exist.

### 6.4 Validation on the Real Baxter Research Robot

To validate our planner in the presence of humans, we have carried out an experiment where the Baxter robot is executing a motion planning task while considering human avoidance and awareness, a snapshot of this experiment is given in Fig. 18. In this experiment, the human presence and direction is first detected by the robot sonar sensor, the robot's head pan joint then points the RGB-D sensor in that

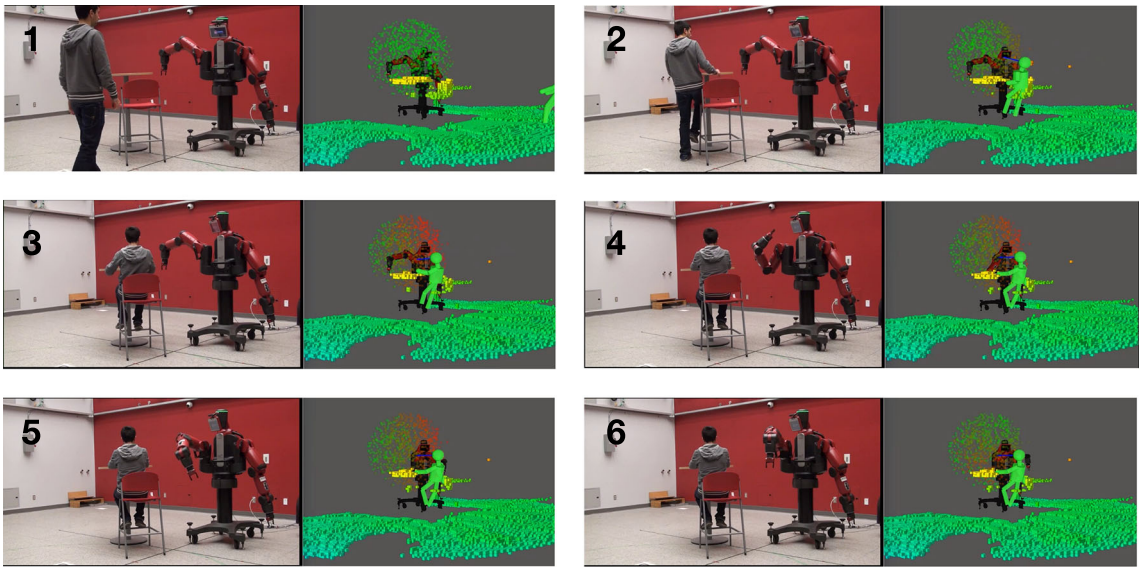



**Fig. 18** Snapshots of the Baxter robot executing a motion planning while considering human avoidance and awareness

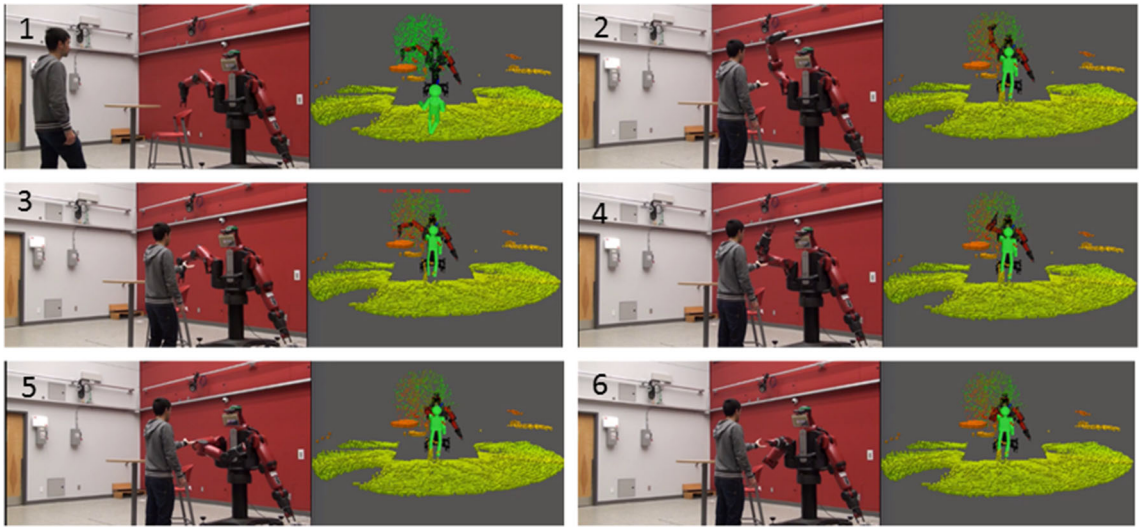



**Fig. 19** Snapshots of the Baxter robot detecting an interactive task intention from a human and executing the corresponding motion planning

direction. After that a skeleton is extracted from the point clouds, and our implementation is called to plan a motion between pre-defined initial and goal poses for the robot's end-effector.

A process that detects an interactive task intention from a human and figures out the corresponding goal to reach for the end-effector has been also developed as shown in Fig. 19. This process mainly observes the human gaze direction and the arm posture, when an interactive task is detected, as shown in Fig. 1, the robot end-effector goal pose is then defined near the human hand and our implementation is called to reach that goal pose from the robot current pose.

## 7 Conclusion

A path planning framework for robotic manipulators that operate in unconstrained environment has been presented. A particular focus is given to ensure the safety of humans that may enter the robot workspace. This is accomplished through the combination of an exact collision checker and two relevant and computationally efficient objectives that guide the planner's exploration.

An adaptation of the bi-directional RRT* planner constitutes the core of our algorithm. It integrates all the safety modules and several features that aim at accelerating the optimization process. However, our approach can be implemented into any sampling-based motion planning algorithm with asymptotic optimality feature.

Future work will focus on reducing the computational time to make the planner even more suitable for time-constrained applications, like human-robot interactions. To this end, we will investigate a more appropriate way to generate the self-collision checking. This process is for now the most time-consuming and an efficient approximation method, like the one developed for the collision with obstacles, could be a good alternative to obtain better performances.

Besides, a more general distance evaluation function will be developed to reduce the approximation error due to bounding boxes. The integration of the second arm of the robot will be investigated in order to consider bi-manipulation tasks. Considering kinodynamic motion planning algorithms [20, 43] to deal with moving objects will also be investigated.

Moreover, it would be interesting to compare our combination strategy for the bi-objective optimization with Pareto-optimal methods.

Realtime aspects will be a major issue in forthcoming work. The initially optimal path should be deformed, due to changes in the environment, to respect the human safety throughout the whole task execution, physical aspects of the interaction will be therefore examined in depth to allow a complete and fully safe human-robot cooperation.

**Acknowledgments** This research was partially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).

## Appendix A: Bisection Continuous Collision Checking Method

In this section, we give an overview of the modified bisection continuous collision checking method [17], which can efficiently handle the case of spherical and two revolute joints by providing tight motion bounds, thus increasing the success rate of checking collision-free paths. Collision checking is an essential step in motion planning as it

ensures the path to be collision-free. The main challenge relies on determining whether the continuous path between two states in C-space is in collision or not. Bisection collision checking method [44] is one of the Continuous Collision Detection (CCD) methods, the main idea behind this method is to establish a sufficient condition of collision-free by computing the geometric path of rigid bodies in the workspace (Fig. 20). A sufficient condition to guarantee that two rigid objects, $A_1$ and $A_2$, do not collide at any configuration $q$ located on the path segment $\pi$, which is joining two configurations $q_a$ and $q_b$, is to verify the following inequality:

$$\lambda_1(q_a, q_b) + \lambda_2(q_a, q_b) < \eta_{12}(q_a) + \eta_{12}(q_b) \tag{10}$$

where $\eta_{12}(q_i)$ is the minimum distance between objects $A_1$ and $A_2$ for a given configuration $q_i$, and $\lambda_i(q_a, q_b)$ refers to the maximum Euclidean displacement of all the points in object $i$ along the path segment $\pi$.

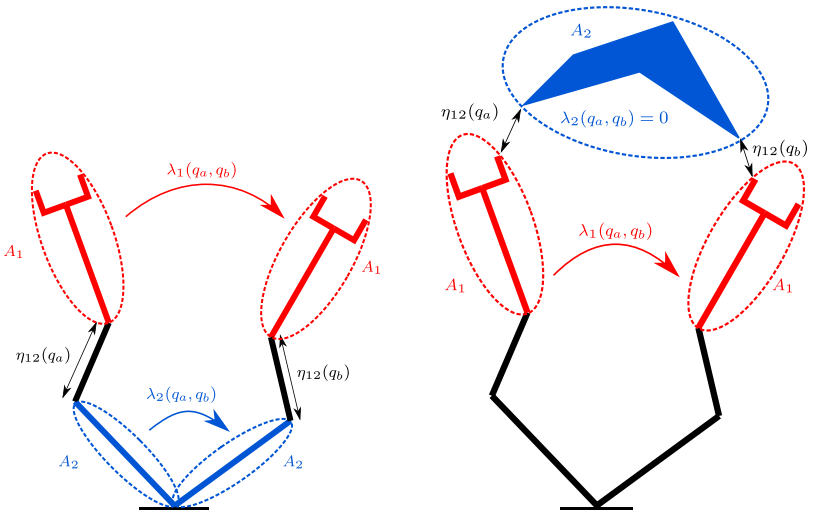
If $A_1$ is a link of the robot and $A_2$ is a fixed obstacle, we define the *estimated clearance* for a path between two configurations $q_a$ and $q_b$ as follows:

$$\begin{aligned}\delta &= \frac{\eta_{12}(q_a) + \eta_{12}(q_b) - \lambda_1(q_a, q_b)}{2} \\ &= dist(q_a, q_b, A_1, A_2)\end{aligned} \tag{11}$$

The procedure to compute the minimum clearance along a path segment and sorting collision-free segment paths according to their clearance is given in Algorithm 4. Note that each element of the structure segment refers to a specific pair of link/obstacle evaluated between two states and is used to store the corresponding distance information. Parameter $\epsilon$ can be defined as the maximum admissible error in the distance estimation. It is a positive user-defined constant that affects the performances of the algorithm: decreasing it improves the returned distance estimation accuracy whereas increasing it reduces the required computational burden to generate the estimation.

**Algorithm 4** Minimum obstacle distance along a path.

```
procedure CLEARANCE(q_a, q_b)
    priority_queue Q                          ▷ segments sorted by increasing δ
    for i ← 1 : nb_links do
        for j ← 1 : nb_obstacles do
            δ ← dist(q_a, q_b, A_i, A_j)                                  ▷ Eq. 11
            if (η_ij(q_a) ≤ 0)||(η_ij(q_b) ≤ 0) then
                return COLLISION
            else
                Q.Add_Segment(q_a, q_b, i, j, δ)
            end if
        end for
    end for
    repeat
        segment s ← Q.Top()
        Q.Pop_Top()
        q_new ← Interpolate(s.q_a, s.q_b)
        δ_1 = dist(s.q_a, q_new, s.link, s.obst)
        δ_2 = dist(q_new, s.q_b, s.link, s.obst)
        if η_{s.link,s.obst}(q_new) ≤ 0 then
            return COLLISION
        else
            Q.Add_Segment(s.q_a, s.q_new, s.link, s.obst, δ_1)
            Q.Add_Segment(s.q_new, s.q_b, s.link, s.obst, δ_2)
            end if
    until Q.Top().λ ≤ ϵ                       ▷ λ is defined in Eq. 10
    if Q.Top().δ ≤ 0 then
        return COLLISION
    else
        return Q.Top().δ
    end if
end procedure
```




(a) Auto-collision (b) Collision with an external object

**Fig. 20** Example: Two types of collision analysis for a 3-DOF robotic arm

The estimated and exact distances to obstacles satisfy the following inequality:

$$\delta_{exa} - \frac{\epsilon}{2} \leq \delta \leq \delta_{exa} \tag{12}$$

where $\delta_{exa}$ and $\delta$ are, respectively, the exact and estimated minimum distances between two objects $A_1$ and $A_2$, where $A_1$ moves from configuration $q_a$ to $q_b$ and $A_2$ is a fixed obstacle.

## References


1. Loughlin, C., Albu-Schäffer, A., Haddadin, S., Ott, C., Stemmer, A., Wimböck, T., Hirzinger, G.: The DLR lightweight robot: design and control concepts for robots in human environments. Ind. Robot: Int. J **34**(5), 376–385 (2007)
2. Wang, H., Chen, W., Lei, Y., Yu, S.: Kinematic analysis and simulation of a 7-DOF cable-driven manipulator. In: IEEE International Conference on Control and Automation(ICCA), pp 642–647 (2007)
3. Ham, R.V., Sugar, T.G., Vanderborght, B., Hollander, K.W., Lefeber, D.: Compliant actuator designs. Robot. Autom. Mag. IEEE **16**(3), 81–94 (2009)
4. Haddadin, S., Robots, T.owards.S.afe.: Approaching Asimov's, 1st edn. Springer Publishing Company, Incorporated (2013)
5. Cho, C.N., Kim, Y.-L., Song, J.-B.: Adaptation-and-collision detection scheme for safe physical human-robot interaction. In: IEEE 9th Asian Control Conference (ASCC), pp 1–6 (2013)
6. De Luca, A., Albu-Schaffer, A., Haddadin, S., Hirzinger, G.: Collision detection and safe reaction with the DLR-III lightweight manipulator arm. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1623–1630 (2006)
7. Lasota, P.A., Rossano, G.F., Shah, J.A.: Toward Safe Close-Proximity Human-Robot Interaction with Standard Industrial Robots. In: 2014 IEEE International Conference on Automation Science and Engineering (CASE), pp. 339–344 (2014)
8. Nokata, M., Ikuta, K., Ishii, H.: Safety-optimizing method of human-care robot design and control. In: IEEE International Conference on Robotics and Automation, vol. 2, pp. 1991–1996 (2002)
9. Kulić, D., Croft, E.: Pre-collision safety strategies for human-robot interaction. Auton. Robot. **22**(2), 149–164 (2007)
10. Zanchettin, A.M., Lacevic, B.: Sensor-Based Trajectory Generation for Safe Human-Robot Cooperation. In: IEEE/RSJ Int. Conf. Intell. Robot. Syst., Workshop Motion Planning Online, Reactive, Real-Time, Algarve, Portugal (2012)
11. Flacco, F., Kroger, T., De Luca, A., Khatib, O.: A depth space approach to human-robot collision avoidance. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 338–345 (2012)
12. LaValle, S.M.: Planning algorithms. University Press, Cambridge (2006)
13. Lacevic, B., Rocco, P.: Safety-oriented path planning for articulated robots. Robotica **31**(06), 861–874 (2013)
14. Lacevic, B.: Kinetostatic danger field-a novel safety assessment for human-robot interaction. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2169–2174 (2010)
15. Dragan, A., Srinivasa, S.: Generating legible motion. In: Proceedings of Robotics: Science and Systems, Berlin, Germany (2013)
16. Sisbot, E.A., Alami, R.: A human-aware manipulation planner. IEEE Trans. Robot. **28**(5), 1045–1057 (2012)
17. Tarbouriech, S., Suleiman, W.: On bisection continuous collision checking method: Spherical joints and minimum distance to obstacles. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 7603–7609 (2018)
18. Quinlan, S.: Real-Time Modification of Collision-Free Paths. Ph.D. dissertation, Stanford University (1994)
19. Choset, H.M.: Principles of robot motion: theory, algorithms, and implementation. MIT Press, Cambridge (2005)
20. LaValle, S.M., Kuffner, J.J.: Randomized kinodynamic planning. Int. J. Robot. Res. **20**(5), 378–400 (2001)
21. Kavraki, L.E., Švestka, P., Latombe, J.-C., Overmars, M.H.: Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Trans. Robot. Autom. **12**(4), 566–580 (1996)
22. Kavraki, L.E., Kolountzakis, M.N., Latombe, J.-C.: Analysis of probabilistic roadmaps for path planning. IEEE Trans. Robot. Autom. **14**(1), 166–171 (1998)
23. Kuffner, J.J., LaValle, S.M.: RRT-connect: An efficient approach to single-query path planning. In: IEEE International Conference on Robotics and Automation (ICRA), vol. 2, pp. 995–1001 (2000)
24. Karaman, S., Frazzoli, E.: Sampling-based algorithms for optimal motion planning. Int. J. Robot. Res. **30**(7), 846–894 (2011)
25. Dobson, A., Krontiris, A., Bekris, K.E.: Sparse roadmap spanners. In: Algorithmic Foundations of Robotics X. Springer, Berlin, pp. 279–296 (2013)
26. Peleg, D., Schäffer, A.A.: Graph spanners. J Graph Theory **13**(1), 99–116 (1989)
27. Dobson, A., Bekris, K.E.: Improving sparse roadmap spanners. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 4106–4111 (2013)
28. Dobson, A.: A study on the finite-time near-optimality properties of sampling-based motion planners. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1236–1241 (2013)
29. Janson, L., Schmerling, E., Clark, A., Pavone, M.: Fast marching tree: a fast marching sampling-based method for optimal motion planning in many dimensions. The International Journal of Robotics Research, pp 883–921 (2015)
30. Salzman, O., Halperin, D.: Asymptotically-optimal motion planning using lower bounds on cost, arXiv:1403.7714 (2014)
31. Salzman, O.: Asymptotically near-optimal rrt for fast, high-quality, motion planning. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 4680–4685 (2014)
32. Akgun, B., Stilman, M.: Sampling heuristics for optimal motion planning in high dimensions. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2640–2645 (2011)
33. Jordan, M., Perez, A.: Optimal bidirectional rapidly-exploring random trees, Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, Tech. Rep MIT-CSAIL-TR-2013-021 (2013)
34. Corke, P.I.: Safety of advanced robots in human environments, Discussion paper for IARP (1999)
35. Otte, M., Correll, N.: C-forest: Parallel shortest path planning with superlinear speedup. IEEE Trans. Robot. **29**(3), 798–806 (2013)
36. "Cforest parallelization framework. http://ompl.kavrakilab.org/CForest.html, (Visited on 04/23/2019)
37. Sucan, I.A., Chitta, S.: Moveit! [Online]. Available: http://moveit.ros.org
38. Şucan, I.A., Moll, M., Kavraki, L.E.: The Open Motion Planning Library. IEEE Robot. Autom. Mag. **19**(4), 72–82 (2012). http://ompl.kavrakilab.org

39. Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y.: ROS: an open-source robot operating system, vol. 3 (2009)
40. Pan, J., Chitta, S., Manocha, D.: FCL: A general purpose library for collision and proximity queries. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 3859–3866 (2012)
41. Hornung, A., Wurm, K.M., Bennewitz, M., Stachniss, C., Burgard, W.: OctoMap: An efficient probabilistic 3D mapping framework based on octrees, Autonomous Robots, software available at http://octomap.github.com (2013)
42. Koenig, N., Howard, A.: Design and use paradigms for gazebo, an open-source multi-robot simulator. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), vol. 3, pp. 2149–2154 (2004)
43. Fiorini, P., Shiller, Z.: Motion planning in dynamic environments using velocity obstacles. Int. J. Robot. Res. **17**(7), 760–772 (1998). [Online]. Available: https://doi.org/10.1177/027836499801700706
44. Schwarzer, F., Saha, M., Latombe, J.-C.: Adaptive dynamic collision checking for single and multiple articulated robots in complex environments. IEEE Trans. Robot. **21**(3), 338–353 (2005)

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

**Sonny Tarbouriech** received the M.Sc. in Electrical and Computer Science Engineering in 2014 from the École Polytechnique de Montpellier and a second M.Sc. in Robotics in 2016 from the University of Sherbrooke, Canada in 2016. Since 2016, he has been working at Tecnalia and has been a Ph.D. student at LIRMM, Université de Montpellier, France. His research interests include motion planning and dual-arm manipulation.

**Wael Suleiman** received the Master's and Ph.D. degrees in automatic control from Paul Sabatier University, Toulouse, France in 2004 and 2008, respectively. He has been Postdoctoral researcher at AIST, Tsukuba, Japan from 2008 to 2010, and at Heidelberg University, Germany from 2010 to 2011. He joined University of Sherbrooke, Quebec, Canada, in 2011, and is currently Associate Professor at Electrical and Computer Engineering Department, Faculty of Engineering. His research interests include collaborative and humanoid robots, motion planning, nonlinear system identification and control and numerical optimization.