Generating Environment-based Explanations of Motion Planner Failure: Evolutionary and Joint-Optimization Algorithms

Qishuai Liu and Martim Brandão

***Abstract*— Motion planning algorithms are important components of autonomous robots, which are difficult to understand and debug when they fail to find a solution to a problem. In this paper we propose a solution to the failure-explanation problem, which are automatically-generated environment-based explanations. These explanations reveal the objects in the environment that are responsible for the failure, and how their location in the world should change so as to make the planning problem feasible.**

Concretely, we propose two methods—one based on evolutionary optimization and another on joint trajectory-and-environment continuous-optimization. We show that the evolutionary method is well-suited to explain sampling-based motion planners, or even optimization-based motion planners in situations where computation speed is not a concern (e.g. post-hoc debugging). However, the optimization-based method is 4000 times faster and thus more attractive for interactive applications, even though at the cost of a slightly lower success rate. We demonstrate the capabilities of the methods through concrete examples and quantitative evaluation.

I. Introduction

Motion planning algorithms are important components of autonomous robots, which compute a trajectory for a robot's degrees of freedom so as to achieve a given task, such as navigation or manipulation. Unfortunately they are known to lack interpretability and explainability [1], [2], as they often fail to find solutions to problems with no clear explanation provided. User studies with practitioners have shown that debugging and understanding failure is indeed difficult even for experts [2], [3], and that automatically generated explanations of failure could be useful in solving this problem [2], [3]. Explanations of planner failure could not only help users and engineers understand why planners fail, when they fail, but also predict future failures and make changes to the environment so as to *avoid* failure [1], [2].

In this paper we propose two methods to generate *environment-based* explanations for motion planning failure, or in other words: to answer the question "why did the planner fail to find a solution to this problem?". Our automatically-generated explanations provide an answer of the type "because of the location of object X. If it had been in location B instead of A, then the planner would have succeeded". This is called a contrastive explanation because it provides a different situation (an environment change) to contrast the failure to. This has been shown to be an important property of explainable algorithms [4]. The explanation is also actionable [5], [3] because it allows a user (or the robot itself) to enact that change on the environment so the robot can solve the original problem.

Both authors are with King's College London, UK.

Our explanation methods work by computing environment changes either within an evolutionary optimization algorithm, or within a single optimization problem which solves motion planning and explanation (i.e. environment-configuration) problems simultaneously.

Our contributions are thus the following:

1) We propose two environment-based explanation-generation algorithms to explain motion planner failure: targeted at both sampling and optimization-based motion planners,
2) We demonstrate the methods through concrete examples, evaluate them, and discuss advantages and disadvantages of each.

II. Related work

This paper is aligned with a recent research interest in explainable AI [6] and explainable task planning [7]. Here we focus on generating explanations for motion planning [1], which robotics practitioners believe could help solve important problems in robot deployment [2]. Recent user studies have shown different properties of explanations which are desirable in motion planning [2] and robotics [8]—including the presence of environmental context [8], contrastiveness [1], and actionability [2]. Our proposed explanations possess all these properties, since they are based on *environment* object changes, they provide an environment *change* that leads to successful planning, and they can be used to actually *move* objects in the world and allow the planner to succeed.

Perhaps the closest work to our own is Hauser's "Minimum Constraint Removal" problem [9], which computes a minimal subset of objects to remove from an environment in order to make a sampling-based motion planner succeed. Compared to [9], our evolutionary method is general enough to be applied to both sampling-based and other kinds of motion planning algorithms (e.g. trajectory optimization). Additionally, the explanations we focus on do not eliminate objects, but instead identify pose changes that lead to success—which could be helpful in triggering recovery behaviour that moves those objects before solving the original problem.

Other related work includes that on *communicating* planner failure [10], which computes plans that are as close as possible to achieving the goal but do not break any other feasibility constraints (e.g. do not break collision constraints). However, such algorithms do not explicitly identify objects or object states that lead to failure.

Other approaches to explanation-generation in motion planning include disconnection proofs [11] and feasibility visualization methods [12]. Approaches targeted at complete robot systems include the use of machine learning methods to compute causal models of robot failure [13]. Finally, and also similar in spirit to our method are "excuse generation" methods in the task planning field—which compute changes to an infeasible (task planning) problem so as to make it feasible [14].

III. Background

The goal of robot motion planning algorithms is to compute a trajectory for a robot's degrees of freedom from a start configuration $q_1 \in \mathcal{C}$ to a target configuration $q_T \in \mathcal{C}$, where T is the number of time steps. The space of robot configurations $\mathcal{C}$ is typically equal to $\mathbb{R}^D$ where D is the number of degrees of freedom (e.g. joint angles), or it can also be the composition of various spaces (e.g. joint angle space and $SE(3)$). The trajectory can be defined as a set of waypoints $\xi = \{q_1, \dots, q_T\}$, and the motion planning problem can be written as an optimization problem:

$$\begin{aligned} \underset{\xi}{\text{minimize}} \quad & f(\xi) & & (1)\\ \text{s.t.} \quad & g_i(\xi) \le 0 & i = 1, \dots, n_{\text{ineq}} \\ & c_i^{\text{re}}(\xi) \le 0 & i = 1, \dots, n_{\text{ineq\_re}} \\ & c_i^{\text{rr}}(\xi) \le 0 & i = 1, \dots, n_{\text{ineq\_rr}} \\ & h_i(\xi) = 0 & i = 1, \dots, n_{\text{eq}} \end{aligned}$$

where f is the objective function to optimize (e.g. trajectory length), c_i^{re} are inequalities representing collision constraints between the robot and the environment, c_i^{rr} are similar self-collision constraints (i.e. between different parts of the robot), g_i are other inequality constraints on the trajectory (e.g. joint angle limits), and h_i are equality constraints (e.g. start configuration, position of a link of the robot at time T). All these functions are scalar functions. In this paper we distinguish between different types of inequalities g_i, c_i^{re} and c_i^{rr} as this will be convenient later on when defining our method. One common objective function is the total length of the trajectory, defined as:

$$f(\xi) = \sum_{i=1}^{T-1} \|q_{i+1} - q_i\|^2 \quad (2)$$

Such motion planning problems can be solved using optimization-based methods [15], [16] or sampling-based methods [17].

IV. Methodology

Motion planning algorithms can fail to find a solution to a problem. As we have already described, our goal in this paper is to develop a method that can automatically compute an *explanation* for a failure. In particular, we focus on environment-based explanations: which are explanations that identify the part of the environment that is responsible for the failure [1]. In other words, our goal is to compute a small change to an environment E (e.g. from E_A to E_B) that leads the motion planning algorithm to succeed to find a plan. In this case, an environment-based explanation for the failure is that "the planner failed because the environment is not E_B". Please see Fig. 2 for an example.

Formally, we define the environment $E = \{O, G\}$ as a set of objects with poses $O = \{o_1, ..., o_n\}$, and arbitrary geometries $G = \{\gamma_1, ..., \gamma_n\}$, where each pose $o_i \in SE(3)$. This definition is general enough to encompass environments defined as set of object primitives, 3D meshes, or occupancy grids.

We assume there is a motion planning problem of the type (1), with environment objects O, that a planner cannot solve. We further distinguish between two kinds of objects: those whose pose we allow to serve as an explanation for failure O_c (i.e. whose pose we assume could be different), and those we don't allow to serve as an explanation for failure O_u (i.e. whose pose we will always leave unchanged). In this paper we assume that all *movable* objects can serve as an explanation, and assume that an oracle algorithm is capable of identifying whether each object is movable or not, and thus usable for the explanation or not.

Our goal is then to compute a new pose vector O_c' that is as close as possible to O_c, but leads the motion planner to find a solution to the problem. This explanation problem can also be stated as an optimization problem. For this we define a new variable $X = \{\xi, O_c'\}$ and rewrite (1) as:

$$\begin{aligned} \underset{X}{\text{minimize}} \quad & f(X) & & (3)\\ \text{s.t.} \quad & g_i(X) \le 0 & i = 1, \dots, n_{\text{ineq}} \\ & c_i^{\text{re}}(X) \le 0 & i = 1, \dots, n_{\text{ineq\_re}} \\ & c_i^{\text{rr}}(X) \le 0 & i = 1, \dots, n_{\text{ineq\_rr}} \\ & h_i(X) = 0 & i = 1, \dots, n_{\text{eq}} \end{aligned}$$

However, problem (3) does not consider collisions between O_c' and O_u, which may happen as a result of making O_c' an optimization variable—and which would lead to the generation of physically unrealistic explanations. Therefore, we add an extra "environment-environment" collision constraint $c_i^{\text{ee}}(X)$ which avoids collisions between O_c'–O_c' and O_c'–O_u:

$$\begin{aligned} \underset{X}{\text{minimize}} \quad & f(X) & & (4)\\ \text{s.t.} \quad & g_i(X) \le 0 & i = 1, \dots, n_{\text{ineq}} \\ & c_i^{\text{re}}(X) \le 0 & i = 1, \dots, n_{\text{ineq\_re}} \\ & c_i^{\text{rr}}(X) \le 0 & i = 1, \dots, n_{\text{ineq\_rr}} \\ & c_i^{\text{ee}}(X) \le 0 & i = 1, \dots, n_{\text{ineq\_ee}} \\ & h_i(X) = 0 & i = 1, \dots, n_{\text{eq}} \end{aligned}$$

The objective function $f(X)$ can now consider not only the quality of the robot trajectory ξ, but also the quality of the explanation (i.e. the environment change). Explanation quality is often modeled as the length of an explanation [1]—which can also be defined as the degree of change to the original environment [18]. We thus model the objective function as the weighted sum of two parts:

$$f(X) = f_1(\xi) + \alpha f_2(O_c') \quad (5)$$

where α is a constant weight parameter, f_1 is the same cost function (2) used to (unsuccessfully) solve the original motion planning problem and f_2 is defined as:

$$f_2(O_c') = \|O_c' - O_c\|_1 \quad (6)$$

This objective thus ensures that the explanation will make small changes to the original problem.

In order to solve (4), we propose two methods: one evolutionary method (particularly suited to sampling-based motion planners) and one continuous-optimization method (particularly suited to optimization-based motion planners).

A. Evolutionary method

The main motivation for the evolutionary method is that some motion planning algorithms are not differentiable, and thus we cannot differentiate the value of trajectory costs or constraints with respect to O'_c. In order to solve the explanation problem in such cases, we thus propose to turn (4) into a bi-level optimization problem:

$$\underset{O'_c}{\text{minimize}} \quad f_1(\xi^*) + \alpha f_2(O'_c) \tag{7}$$

$$\text{s.t.} \quad \xi^* = \underset{\xi}{\operatorname{argmin}}\, f_1(\xi) \tag{8}$$

$$\begin{aligned} \text{s.t.} \quad & g_i(\xi, O'_c) \le 0 && i = 1, \ldots, n_{\text{ineq}} \\ & c_i^{\text{re}}(\xi, O'_c) \le 0 && i = 1, \ldots, n_{\text{ineq\_re}} \\ & c_i^{\text{rr}}(\xi, O'_c) \le 0 && i = 1, \ldots, n_{\text{ineq\_rr}} \\ & c_i^{\text{ee}}(\xi, O'_c) \le 0 && i = 1, \ldots, n_{\text{ineq\_ee}} \\ & h_i(\xi) = 0 && i = 1, \ldots, n_{\text{eq}} \end{aligned}$$

where (8) is solved using the original motion planning algorithm (e.g. RRT*), and (7) is solved by an evolutionary algorithm.

In this paper, we use Particle Swarm Optimization (PSO) to solve (7). PSO basically mimics the behavior of flocks of birds when searching for food. If one of the birds finds food—one of the particles finds a low-cost objective—PSO will use this information to move the swarm towards that location gradually. We choose PSO instead of other methods because it can handle non-differentiable problems, is computationally inexpensive, and requires tuning only a small number of parameters [19]. However, other evolutionary algorithms could similarly be used.

Pseudo-code for the algorithm is shown in Algorithm 1. Basically, each particle in the swarm holds a value of O'_c, and each particle is evaluated by attempting to solve the corresponding motion planning problem (where O_c is changed to O'_c). This motion planning problem is solved in line 8 of Algorithm 1 and corresponds to solving equation (8). If the planner succeeds, then a trajectory ξ is obtained, and the value of the particle is set to $f_1(\xi) + \alpha f_2(O'_c)$ (line 12). Otherwise, the value is set to a large number to avoid having the swarm move in the direction of this particle. The algorithm keeps track of the best particle found O_c^*—which is the one with the lowest objective. We use an open implementation of PSO [20] to implement this algorithm.

B. Continuous joint-optimization method

Optimization-based motion planning algorithms should by definition be able to solve (4) explicitly. In practice, however, most planners' implementations do not provide capability to model environment-parameter variables by default. This is the case of two popular optimization-based motion planners: Trajopt [16] and CHOMP [15].

Algorithm 1: PSO for explanation-generation

```
   Data: Environment objects ($O_u$, $O_c$), problem
         constraints ($g$, $c^{\text{re}}$, $c^{\text{rr}}$, $c^{\text{ee}}$, $h$), population size
         ($S$)
 1 $O_c^* \leftarrow \emptyset$ ;
 2 $P \leftarrow$ RandomlySampleParticles($O_c$, $S$) ;
 3 for $it$ = 1, ..., MaxIter do
 4     for $p$ in $P$ do
 5         UpdateParticleVelocity($p$, $O_c^*$) ;
 6         UpdateParticlePosition($p$) ;
 7         $O'_c \leftarrow$ GetParticlePosition($p$) ;
 8         $\xi \leftarrow$ MotionPlanner($O_u$, $O'_c$, $g$, $c^{\text{re}}$, $c^{\text{rr}}$, $c^{\text{ee}}$, $h$)
            ;
 9         if $\xi = \emptyset$ then
10             UpdateParticleValue($p$, $\infty$) ;
11         else
12             UpdateParticleValue($p$, $f_1(\xi) + \alpha f_2(O'_c)$) ;
13     $O_c^* \leftarrow$ UpdateBestParticleFoundSoFar($O_c^*$, $P$) ;
14 return $O_c^*$
```

For the experiments in this paper we used a practical re-formulation of (4) that allows the problem to be solved explicitly by Trajopt (through Sequential Quadratic Programming) without changes to the planner's implementation. The main idea is to remove objects O_c from the environment, and include them as virtual *links* in the robot model. These virtual links are connected to the robot's base link by virtual joints whose values are new optimization variables.

Formally, let $b \in SE(3)$ be the pose of the base link of the robot in the world. And let $C = |O_c|$ be the number of objects we allow to serve as an explanation. We then define a set of virtual-joint variables $V = \{v_1, \ldots, v_C\}$, where $v_i \in SE(3)$, such that $bv_i = o'_i$ for all $i = 1, \ldots, C$. Intuitively, v_i is the transformation from the base link of the robot to virtual link i, which corresponds to object i in O'_c. The configuration space of this new virtual robot model is now $\xi_v = \{\xi, \mathcal{V}\}$, where $\mathcal{V} = \{V_1, \ldots, V_T\}$ are T waypoints for the virtual joint states. The explanation problem now becomes a regular motion planning problem:

$$\underset{\xi_v = \{\xi, \mathcal{V}\}}{\text{minimize}} \quad f_1(\xi) + \alpha f_2(\xi_v) \tag{9}$$

$$\begin{aligned} \text{s.t.} \quad & g_i(\xi_v) \le 0 && i = 1, \ldots, n_{\text{ineq}} \\ & c_i^{\text{re}}(\xi_v) \le 0 && i = 1, \ldots, n_{\text{ineq\_re}} \\ & c_i^{\text{rr}}(\xi_v) \le 0 && i = 1, \ldots, n_{\text{ineq\_rr}} \\ & h_i(\xi_v) = 0 && i = 1, \ldots, n_{\text{eq}} \\ & V_i = V_{i+1} && i = 1, \ldots, T-1 \qquad (10) \end{aligned}$$

where collisions between O'_c and O_u are now implemented as "robot"-environment collisions c_i^{re}, and collisions between the real robot and O'_c are now implemented as "self"-collisions c_i^{rr}. The second component of the objective function $f_2(\xi_v)$ is still as shown in (6), where O'_c is computed as $O'_c = [bv_1, \ldots, bv_C]$. The constraint (10) ensures the explanation consists of a static environment (i.e. we compute a single

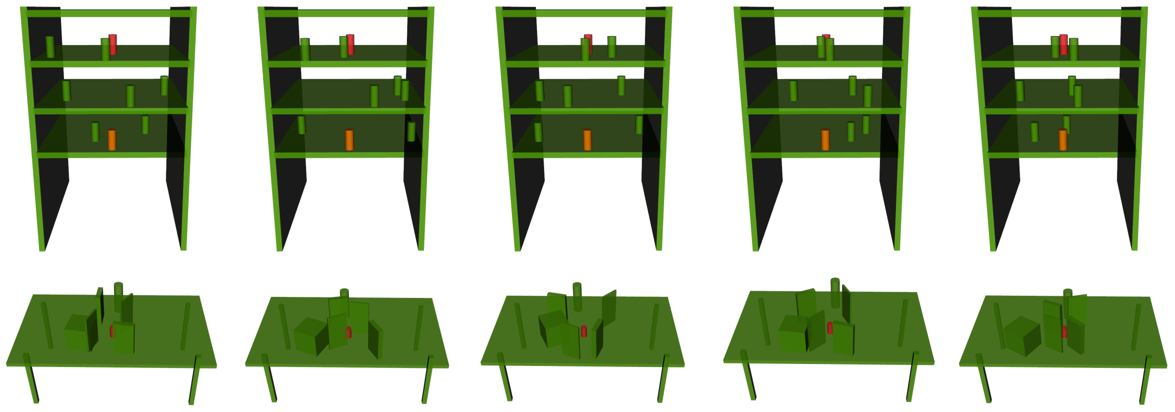

Fig. 1. Infeasible problems for the shelf scene (top) and table scene (bottom). The problem involves moving the robot's end-effector from the orange cylinder (or under the table) to the red cylinder.

environment change that is constant at all times $1, \ldots, T$).

To summarize, then, if we try to solve a motion planning problem with Trajopt and it fails to find a solution, we automatically generate an environment-based explanation in two steps. First, we create a new virtual robot model where objects that we allow to serve as explanation are added as virtual links connected to the base link. In this paper we assume an oracle algorithm is capable of identifying which objects are movable, and allow those to be used as explanation. This virtual robot model is then used to solve the motion planning problem (9). If the planner is successful, then the result tells one possible change in environment object locations that leads the original problem to be feasible.

V. Results

In order to demonstrate and evaluate our explanation-generation methods, we first generated a set of unsolvable motion planning problems, and then used our methods to obtain explanations for motion planner failure. Our experiments use RRT [21] as the sampling-based motion planner, Trajopt [16] as the optimization-based motion planner, and DEAP [20] for the implementation of PSO.

We used MotionBenchMaker [22], a tool to generate motion planning problems by randomly setting poses of objects in pre-specified environments, to generate 40 unsolvable motion planning problems. We generated 20 problems in a grasping scenario on a table, and 20 problems in an object re-placement scenario on a bookshelf (both available through MotionBenchMaker). We show a subset of 5 problems from each scene in Fig. 1. In the bookshelf scenario the robot has to move its end-effector from the orange object to the red object, while in the table scenario the robot starts with the end-effector under the table and has to move it to the red object on top of the table. The problems are unsolvable because there are other objects in the environment that the robot would have to collide with in order to be able to grasp the red object.

We run PSO for *MaxIter*=150 generations, which was sufficient for the method to converge, and we use population size S=100. The explanation weight α was set to 100, such as to obtain small-change explanations.

We start by qualitatively analyzing the results of our methods in each of these scenarios, and then report on efficiency and success-rate results.

A. *Qualitative results*

Fig. 2 shows the result of applying the evolutionary method to explain the failure of RRT in one of the bookshelf problems. The problem is infeasible because one of the cylinders (highlighted in yellow) is too close to the target object, and thus prevents the robot from reaching it without collision. To generate the explanation, we assume all objects except the bookshelf are movable, and thus usable for explanation. The objects are furthermore constrained to lie on their respective shelves—i.e. O_c is the (x, y) position of all green and yellow cylinders. Fig. 2 (middle) shows the computed explanation, which is a minimal displacement of the yellow object's position that leads RRT to find a plan. The figure could be provided to a user as an explanation for failure, together with the text: "The motion planner failed because of the object shown in transparent-yellow. If the object was in the position shown in solid-yellow, then the planner would succeed". This explanation could be used as part of a user interface, that asks a user for help (to move the yellow object) so the robot can complete its task. Alternatively, it could be used to trigger a recovery behavior that plans motion to move the yellow object before going back to the original task. Even though the positions of all (yellow and green) cylinders are part of the optimization variable O'_c, the minimization of $\|O'_c - O_c\|_1$ leads our method to find a displacement that affects only one object—the one that has an effect on planner success rate. Fig. 2 (right) shows a plan obtained by RRT once the object is moved according to the explanation.

We also computed an explanation to this problem using our continuous joint-optimization method. Fig. 3 shows the result, where we can see that the planner arrives at a similar explanation to that found by PSO (i.e. Trajopt failed because of the yellow cylinder, though it succeeds if the cylinder is moved to the right). In this image the explanation is shown in grey, while the original position of the cylinder is shown in yellow.

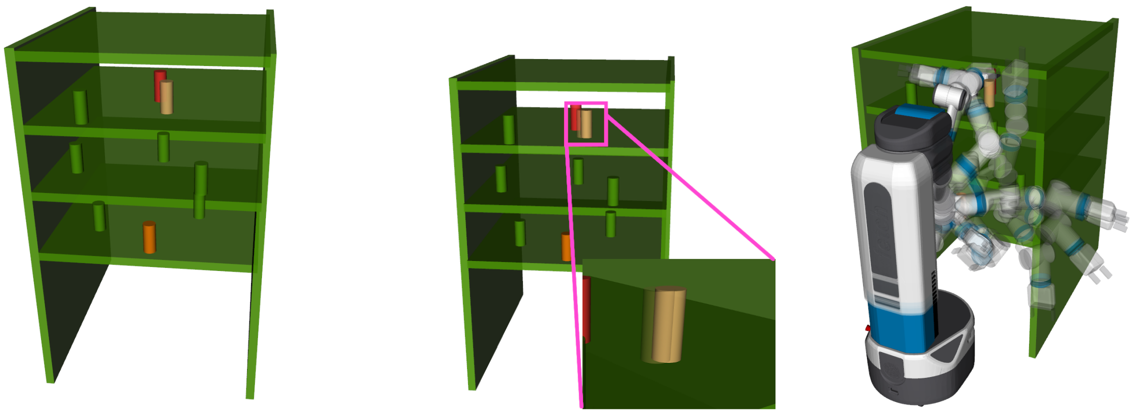

Fig. 2. Example explanation problem on a shelf scene. Left: The original planning problem, where the robot needs to move end-effector from the orange cylinder in the bottom, to the red one in the top. The green and yellow cylinders are obstacles which we assume to be movable. The yellow cylinder is shown in a different color to highlight that it is the one responsible for failure. Center: The explanation for failure computed with PSO, which shows that moving the object from the transparent to the solid location would make the problem feasible. Right: A feasible trajectory found by RRT, after the yellow object is moved according to the explanation.

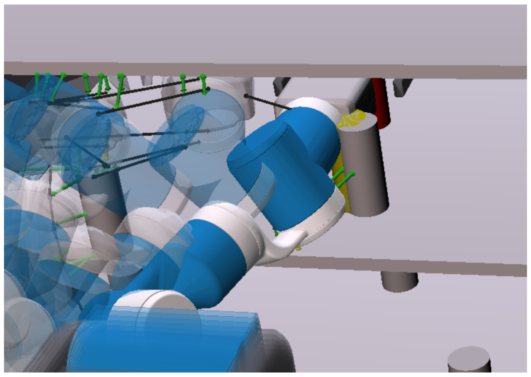

Fig. 3. Explanation-generation example using our joint trajectory-and-environment optimization method on Trajopt, where both robot trajectory and explanation (i.e. object displacement) are optimized. The yellow cylinder represents the obstacle's location on the original problem, grey represents the explanation (required movement such as to make the planner succeed). Black lines and green arrows represent collision constraints.

Another example, on the table scene, is shown in Fig. 4. Here, the yellow box prevents the robot from reaching its target object without collision, thus making RRT fail. In this scenario we include object yaw-orientation as part of the optimization variables, since unlike the bookshelf scenario the objects are not rotation-symmetric. Fig. 4 (middle) shows the explanation computed by PSO: which identifies the object responsible for failure and a minimal pose-change that leads RRT to succeed. Fig. 4 (right) shows the resulting trajectory obtained by RRT after the yellow object is moved according to the explanation.

B. Quantitative evaluation

We then ran a quantitative evaluation of the evolutionary and optimization-based explanation algorithms, on 20 infeasible problems of each scene. We solved the explanation problems using:

1) The evolutionary (PSO) method using RRT as the planner, and a 3-second computation time budget for RRT,
2) The evolutionary (PSO) method using Trajopt as the planner,
3) The joint-optimization method, using Trajopt.

Table I shows the results in all conditions. The metrics used to compare the different methods are: explanation cost, success rate of the explanation algorithm, and computation time. The explanation cost is the value of Eq. (5). The success rate indicates the number of problems for which an explanation was computed (i.e. for which the method could find an environment-change that made the planning problem feasible). Computational time is the average time to compute an explanation per problem. The evolutionary method was around twice as fast to solve the problem when using Trajopt as the planner, when compared to RRT. This was expectable as Trajopt is very fast to fail when it fails, compared to RRT that has to wait for a computation time budget. In both cases the method succeeded to find an explanation in 100% of the problems, and the cost of the explanation (i.e. equation (5)) was similar with both planners. The computation times were quite high, up to 2 hours using RRT and up to 1 hour using Trajopt. This indicates the evolutionary method is appropriate when computational time is not a concern, such as in post-hoc debugging or accident investigations [23], or when there is no other choice because the motion planner in use is a black-box (e.g. closed-source) or non-differentiable.

The continuous joint-optimization method on the other hand, as shown on Table I, was around 4000 times faster than the evolutionary method. When the evolutionary method computes an explanation for Trajopt-failure, it takes between 30 and 60 minutes, while the joint-optimization method (solved with Trajopt itself through problem reformulation as described in Section IV-B) takes less than a second. This demonstrates the advantage of using our continuous optimization formulation. However, there are two disadvantages to this method: 1) it is only applicable to optimization-based

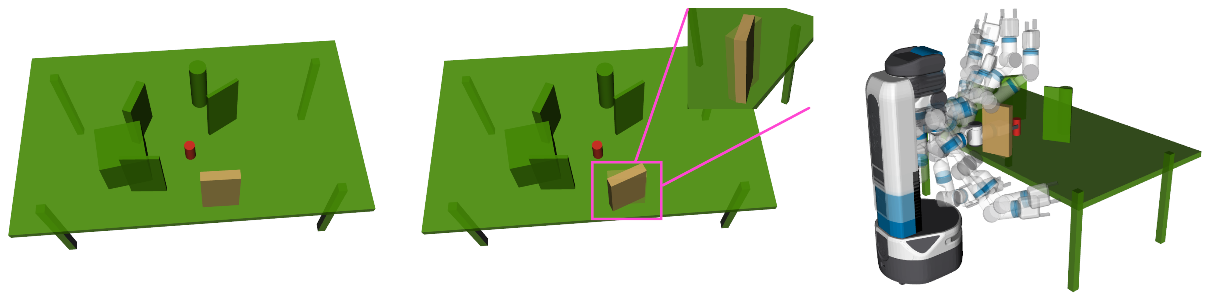

Fig. 4. Example explanation problem on a table scene. Left: The original planning problem, where the robot needs to move end-effector from below the table to the red cylinder on top of the table. The green and yellow objects (except the table) are obstacles which we assume to be movable. The yellow box is shown in a different color to highlight that it is the one responsible for failure. Center: The explanation for failure computed with PSO, which shows that moving the object from the transparent to the solid location would make the problem feasible. Right: A feasible trajectory found by RRT, after the yellow object is moved according to the explanation.

TABLE I

QUANTITATIVE RESULTS ON 20 PROBLEMS OF EACH SCENE

	Evolutionary method (RRT)			Evolutionary method (Trajopt)			Joint-Optimization method (Trajopt)		
	Explanation cost	Success rate	Comp. time	Explanation cost	Success rate	Comp. time	Explanation cost	Success rate	Comp. time
Bookshelf scene	18.34 ± 1.24	100%	124.2 ± 9.54min	18.46 ± 1.35	100%	62.3 ± 5.05min	16.27 ± 0.26	90%	0.879 ± 0.054s
Table scene	9.90 ± 1.15	100%	73.3 ± 7.76min	9.80 ± 1.25	100%	34.6 ± 4.46min	8.87 ± 0.16	95%	0.526 ± 0.023s

planners; and 2) as Table I shows, the method had a success rate of 90% on the shelf scene and 95% on the table scene, which means that the method is more likely not to be able to find a solution to an explanation problem. This is due to the added complexity of the optimization problem, even though some techniques such as "multi-starts" [24] and stochastic optimization [25] could potentially alleviate this issue.

VI. Conclusions

In this paper we proposed two new methods for generating environment-based explanations of motion planner failure.

One method uses evolutionary optimization to find minimal changes to the environment that lead the motion planner to succeed to find a plan. This method can be applied to any motion planning algorithm, since it is only calling an existing motion planner in every iteration with a new environment. However, as our experiments show, this process is inefficient and leads to slow computation times which would be impractical for interactive interfaces—although it could be used in post-hoc debugging or accident investigation scenarios [23].

We also proposed another method targeted specifically at optimization-based motion planning algorithms, which works by framing the explanation problem as a joint trajectory-and-environment optimization problem. We show this can be implemented as a new motion planning problem with virtual robot links and self-collision constraints, and it leads to very fast computation times (approximately 4000 times faster). The results showed that the method can have a slightly lower success rate than the evolutionary one, due to the increased complexity of the optimization problem. In the future, this flaw could potentially be tackled with improvements to the motion planning algorithm (e.g. through randomization [24], [25]).

The explanations produced by our algorithms could be used as part of future user interfaces, where the robot can ask humans to help move an object out of the way so the robot can complete its task. Alternatively, the explanations could be used for user probing (e.g. for answering a user that asks "why did you fail?"), or they could even be used to automatically trigger recovery behaviour that plans and executes motion to move the object before returning to the original task. Possible future research directions include user studies to understand the reception of users to these explanations in different contexts, other models of explanation quality, and computer vision algorithms for detecting movable objects and their physical constraints. Scalable extensions of the methods to point cloud and occupancy grid representations are also important research directions.

References

[1] M. Brandao, G. Canal, S. Krivic, and D. Magazzeni, "Towards providing explanations for robot motion planning," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, Jul 2021, pp. 3927–3933.

[2] M. Brandao, M. Mansouri, A. Mohammed, P. Luff, and A. Coles, "Explainability in multi-agent path/motion planning: User-study-driven taxonomy and requirements," in *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, May 2022, p. 172–180.

[3] M. Brandao, G. Canal, S. Krivic, P. Luff, and A. Coles, "How experts explain motion planner output: a preliminary user-study to inform the design of explainable planners," in *IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, Aug 2021, pp. 299–306.

[4] T. Miller, "Explanation in artificial intelligence: Insights from the social sciences," *Artificial Intelligence*, vol. 267, 2019.

[5] R. Singh, T. Miller, H. Lyons, L. Sonenberg, E. Velloso, F. Vetere, P. Howe, and P. Dourish, "Directive explanations for actionable explainability in machine learning applications," *ACM Transactions on Interactive Intelligent Systems*, 2023.

[6] A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins, *et al.*, "Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai," *Information fusion*, vol. 58, pp. 82–115, 2020.

[7] T. Chakraborti, S. Sreedharan, and S. Kambhampati, "The emerging landscape of explainable ai planning and decision making," *arXiv preprint arXiv:2002.11697*, 2020.

[8] D. Das, S. Banerjee, and S. Chernova, "Explainable ai for robot failures: Generating explanations that improve user assistance in fault recovery," in *Proceedings of the 2021 ACM/IEEE International Conference on Human-Robot Interaction*, 2021, pp. 351–360.

[9] K. Hauser, "The minimum constraint removal problem with three robotics applications," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 5–17, 2014.

[10] M. Kwon, S. H. Huang, and A. D. Dragan, "Expressing robot incapability," in *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, 2018, pp. 87–95.

[11] J. Basch, L. J. Guibas, D. Hsu, and A. T. Nguyen, "Disconnection proofs for motion planning," in *ICRA*. IEEE, 2001, pp. 1765–1772.

[12] J. Kottinger, S. Almagor, and M. Lahijanian, "Maps-x: Explainable multi-robot motion planning via segmentation," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 7994–8000.

[13] M. Diehl and K. Ramirez-Amaro, "Why did i fail? a causal-based method to find explanations for robot failures," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 8925–8932, 2022.

[14] M. Göbelbecker, T. Keller, P. Eyerich, M. Brenner, and B. Nebel, "Coming up with good excuses: What to do when no plan can be found," in *Proceedings of the International Conference on Automated Planning and Scheduling*, ser. ICAPS'10. AAAI Press, 2010, p. 81–88.

[15] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "Chomp: Gradient optimization techniques for efficient motion planning," in *2009 IEEE international conference on robotics and automation*. IEEE, 2009, pp. 489–494.

[16] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.

[17] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[18] M. Brandao, A. Coles, and D. Magazzeni, "Explaining path plan optimality: Fast explanation methods for navigation meshes using full and incremental inverse optimization," in *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, Aug 2021, pp. 56–64.

[19] F. Marini and B. Walczak, "Particle swarm optimization (pso). a tutorial," *Chemometrics and Intelligent Laboratory Systems*, vol. 149, pp. 153–165, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169743915002117

[20] F.-A. Fortin, F.-M. De Rainville, M.-A. G. Gardner, M. Parizeau, and C. Gagné, "Deap: Evolutionary algorithms made easy," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 2171–2175, 2012.

[21] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 2. IEEE, 2000, pp. 995–1001.

[22] C. Chamzas, C. Quintero-Pena, Z. Kingston, A. Orthey, D. Rakita, M. Gleicher, M. Toussaint, and L. E. Kavraki, "Motionbenchmaker: A tool to generate and benchmark motion planning datasets," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 882–889, 2021.

[23] A. F. Winfield, K. Winkle, H. Webb, U. Lyngs, M. Jirotka, and C. Macrae, "Robot accident investigation: a case study in responsible robotics," *Software engineering for robotics*, pp. 165–187, 2021.

[24] M. Brandao, K. Hashimoto, and A. Takanishi, "Sgd for robot motion? the effectiveness of stochastic optimization on a new benchmark for biped locomotion tasks," in *17th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, Nov 2017.

[25] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: Stochastic trajectory optimization for motion planning," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 4569–4574.