2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Characterizing Physical Adversarial Attacks on Robot Motion Planners

Wenxi Wu, Fabio Pierazzi, Yali Du, Martim Brandão

***Abstract*— As the adoption of robots across society increases, so does the importance of considering cybersecurity issues such as vulnerability to adversarial attacks. In this paper we investigate the vulnerability of an important component of autonomous robots to adversarial attacks—robot motion planning algorithms. We particularly focus on attacks on the physical environment, and propose the first such attacks to motion planners: "planner failure" and "blindspot" attacks. Planner failure attacks make changes to the physical environment so as to make planners fail to find a solution. Blindspot attacks exploit occlusions and sensor field-of-view to make planners return a trajectory which is thought to be collision-free, but is actually in collision with unperceived parts of the environment. Our experimental results show that successful attacks need only to make subtle changes to the real world, in order to obtain a drastic increase in failure rates and collision rates—leading the planner to fail 95% of the time and collide 90% of the time in problems generated with an existing planner benchmark tool. We also analyze the transferability of attacks to different planners, and discuss underlying assumptions and future research directions. Overall, the paper shows that physical adversarial attacks on motion planning algorithms pose a serious threat to robotics, which should be taken into account in future research and development.**

I. Introduction

As the adoption of robotics across sectors and in safety-critical applications gradually increases, cybersecurity aspects of robotics become more and more an important concern. Using adversarial attacks, some actors may purposefully cause accidents involving robots, such as to cause economic, reputational, or other harms. For example, a company may provoke an accident involving a competitor company's robot, or it may subtly cause small damage that raises the competitor's maintenance costs, thus obtaining an unfair economic advantage. As another example, motivated by recent incidents of vandalism against robots [1], unhappy users with enough resources may attempt to have robots removed without getting caught, by repeatedly making them fail in subtle ways that lower trust or efficiency of the system.

In this paper we particularly focus on adversarial attacks to motion planning algorithms—which compute feasible trajectories for the motion a robot that lead it to satisfy a given task (e.g. reaching or grasping an object). While various adversarial attacks on computer vision [2], [3] and reinforcement learning algorithms [4] have recently been proposed and analyzed, attacks on traditional motion planning algorithms [5], [6] have not yet been investigated even though they are widely used in robotics.

All authors are with King's College London, UK. This work was supported by the UKRI Centre for Doctoral Training in Safe and Trusted Artificial Intelligence [EP/S023356/1], and EPSRC Grant no. EP/X015971/1. Wenxi and Martim were also supported by The Great Britain Sasakawa Foundation.

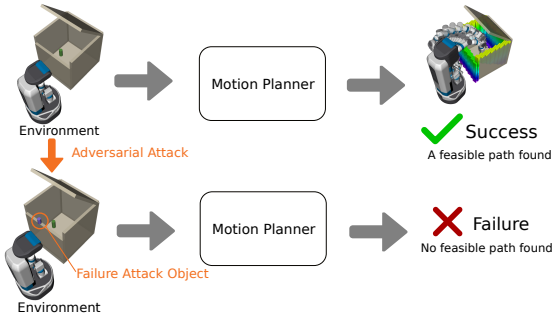


Fig. 1: Illustration of "planner failure" attacks. These make changes to the physical environment so as to make the planner fail to find a solution to the motion planning problem.

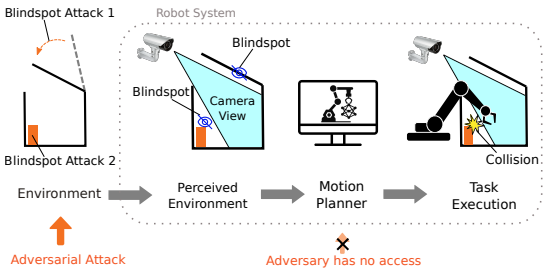


Fig. 2: Illustration of "blindspot" attacks. These make changes to the environment so as to cause a collision between the robot and objects which are not perceived by the camera (due to field of view or occlusion). Attack 1: manipulating an articulated object. Attack 2: placing a new object.

In this paper we address this gap by proposing and characterizing the first adversarial attacks on traditional (sampling-based) robot motion planning algorithms. We focus on the particular case of *physical* adversarial attacks on the environment—i.e. where the attacker has access to the physical environment of the robot, and is thus able to change object positions or place a new object in the scene. This paper will show that appropriate methods can make small, potentially imperceptible, changes to the environment that drastically raise motion planner failure and collision rates.

We propose two types of adversarial attacks, which are illustrated in Fig. 1 and 2. The first is called a "planner failure" attack. The idea is that placing objects in carefully chosen locations could lead a motion planner to fail, i.e. to not find a feasible solution to a problem. This could be because the object location made the problem unsolvable, or because it made it drastically more difficult to solve and thus unsolvable within the planner's computation time budget. The second adversarial attack is called a "blindspot attack". The idea is that an attacker can exploit a robot's (lack of)

field of view, and purposefully cause a collision with an object that is visually occluded. This could be done by either occluding certain parts of the environment or placing obstacles in visually-occluded areas. So, the attack makes the motion planner find a path that *seems* to be collision-free because obstacles are in a blindspot, but is actually in collision.

The contributions of the paper are the following:

1) We propose and characterize the first physical attacks on robot motion planning algorithms: called "planner failure" and "blindspot" attacks;
2) We evaluate the feasibility of the attacks, both qualitatively and quantitatively in motion planning problems from an existing motion planning benchmark;
3) We show that both attacks are able to drastically raise planner failure and collision rates through small changes in the environment, thus making these important attacks to consider in robot security.

II. Related Work

Recent research has shown that it is possible to produce inconspicuous adversarial attacks on computer vision algorithms, for example leading neural networks to misclassify objects [3], [7], [8], [9], to not recognize faces or pedestrians [2], [10], or to overestimate the quality of a grasp [11].

Some distinctions can be made between these attacks. One is between digital and physical attacks. Digital attacks make perturbations to images [9], [8], while physical attacks make perturbations to the physical world [3], [2], [7], [10] (which then gets perceived by a camera). Physical attacks need to be robust against real-world problem-space distortions [12], e.g. camera distances and angles, lighting conditions [7], [10]. Examples of physical attacks include graffiti on stop signs [3], stickers and small objects attached to objects [13], [11], and stickers attached to the lens of a camera [14]. Our paper similarly proposes physical attacks, but it focuses on attacking motion planning instead of computer vision algorithms.

Another distinction of adversarial attacks is on whether they are obtained through continuous optimization methods like gradient descent, or through evolutionary methods. Most deep neural network attacks are based on gradient descent [3], [2], [8], but they are not applicable if the model or feature mapping is neither invertible nor differentiable [12], in which case some have recurred to sampling-based evolutionary methods [13]. In this paper we show results of attacks on traditional sampling-based motion planning algorithms (RRT-Connect [6], KPIECE [15]), which are not differentiable, and so similarly to [13] require sampling methods to optimize the attack.

In robotics, adversarial methods have mainly been proposed as a way to robustify controllers. For example, controllers based on reinforcement learning can be made more robust by having an adversary apply forces on the robot during training [16], [17]. Our paper focuses instead on traditional motion planning algorithms, which are commonly deployed in existing robots.

III. Background

The goal of motion planning algorithms is to compute a feasible path for a robot's degrees of freedom (e.g. its position in the world and its joint angles) that satisfies collision, kinematics, and task constraints—such as avoiding obstacles, staying within robot joint limits, and grasping an object. The algorithm will generate a path that connects a start state to a goal state, such that all the states in the path respect collision, kinematics, or other constraints. Sampling-based motion planning methods [5], [6], [15] can solve high-dimensional and non-convex planning problems by decomposing the path into waypoints which are randomly sampled and connected to a tree that is gradually explored.

In this paper we evaluate our adversarial methods on RRT-Connect [6] and BKPIECE [15] which are two efficient and popular methods used in practice, and integrated into ROS [18]. RRT-Connect expands two trees, one from the start configuration and one from the goal, and tries to connect them by using a greedy heuristic. The algorithm is typically faster than RRT [5]. Bidirectional Kinodynamic Motion Planning by Interior-Exterior Cell Exploration (BKPIECE) [15] is also a bi-directional sampling-based algorithm, which selects nodes for expansion based on their coverage, so as to focus computation on yet-uncovered regions of space. It is typically faster than Expansive Space Trees (EST) [19].

IV. Planner Attack Methods

The goal of both methods proposed in this paper is to make changes to the environment that lead to either planner failure or a collision with the environment with high probability. We now describe the assumptions behind these attacks.

A. Threat Model

1) Objectives: The goal of the adversary is to either make a motion planner fail to find a path ("planner failure" attack) or to make the planner produce a path that is in collision with obstacles ("blindspot" attack). The adversary also has the goal of producing inconspicuous attacks, i.e. changes to the environment are small, or lead to states that are visually similar to regular world states. This is to make failures look natural, or look like an accident, thus lowering the chances that the attack is recognized as such. Our attacks implement inconspicuousness by adding very small objects (e.g. a small-radius sphere) to the environment, or making small changes to the state of articulated objects.

2) Incentives: We assume the adversary's incentives are to cause economic loss to those deploying or maintaining robots, e.g. by lowering efficiency through high failure rates or by raising collisions and thus damage to the robot or environment. The incentive could also be to undermine the reputation of robots, or those deploying robots.

3) Knowledge and Capability: We assume the adversary generates attacks on the motion planner by manipulating the environment where the robot operates. We thus assume the adversary has knowledge of the (expectation of) location of objects in the environment, typical tasks done in that environment (e.g. geometry and location of object that will be grasped at some point in the future), as well as knowledge of the type of planner used and its parameters (e.g. RRT with a 5 second time budget). These are reasonable assumptions, as robots often operate in the same environment over long

periods of time, and conducting similar tasks. Also, the use of certain planners (e.g. RRT) is widespread and so may be easy to guess. Since the adversary has this knowledge, they do not have to make environmental changes in real-time. They can conduct various experiments in a simulated environment so as to decide what change to the environment they should make later on to implement the attack.

B. Common Definitions

Let $\mathcal{E}$ be the space of environments and $E \in \mathcal{E}$ be an environment defined as set of "objects" $E = \{o_1, ...o_N\}$. Each object in an environment is a tuple $o_i = (g_i, \theta_i, r_i)$ of geometry g_i (e.g. a 3D mesh or primitive), a pose θ_i (e.g. position and orientation in $SE(3)$), and a parent object with respect to which the pose is defined r_i (e.g. box with respect to which its lid's pose is defined).

Then let $c : \mathcal{E} \times \mathcal{A} \to \mathcal{E}$ be a function that makes changes to an environment E, using parameters $a \in \mathcal{A}$ and thus obtaining a new environment $E' = c(E, a)$. In this paper we consider two kinds of environment changes, corresponding to two implementations of this environment-change function c:

1) Placing a new object at a given position and orientation in the world, i.e. a is a new object $a = (g, \theta, r)$ and $E' = c_1(E, a) = E \cup a$;
2) Changing the position and orientation of existing objects or parts-of-articulated-objects in the world (e.g. opening/closing a door by a certain degree as illustrated in Fig. 8). Here $a = (i, \theta^a)$ is a tuple of the index i of an existing object in the world and a new pose θ^a, and $E' = c_2(E, a)$ is such that $E' = E$ for all objects except i, where $\theta'_i = \theta^a$.

The adversary has knowledge of a motion planning problem (E, s, t) where s is a random variable representing the start state of the robot in configuration space $\mathcal{S}$, and $t \in \mathcal{T}$ are target constraints in task space. We make t constant but allow s to be a random variable, as it may be easy for an attacker to predict a task (Section IV-A), but difficult for them to predict the initial state of the robot when solving that task.

A "planner" is a function $F_{\text{plan}} : \mathcal{E} \times \mathcal{S} \times \mathcal{T} \to \mathcal{P}$ that returns a feasible path $p \in \mathcal{P}$ that satisfies kinematics, collision, and target constraints. We assume that the planner function F_{plan} first obtains the visible part of the environment $E_v \subseteq E$ (i.e. the part of the environment that is visible by the robot's sensors from state s) and only then runs a motion planning algorithm such as RRT on E_v. We do this in order to obtain a realistic simulation of planning in the real-world, where some objects are not seen by the cameras.[1] If the planner fails to find a feasible path, i.e. a path that satisfies constraints and is not in collision with E_v, then $p = \emptyset$.

Finally, we define a convenience function for the success of a planner $f_{\text{success}} : \mathcal{P} \to \{0, 1\}$, where $f_{\text{success}}(p) = 0$ if $p = \emptyset$ and 1 otherwise. And a convenience function for the existence of collisions between a path and an environment $f_{\text{collision}} : \mathcal{P} \times \mathcal{E} \to \{0, 1\}$, where $f_{\text{collision}}(p, E) = 1$ if p is in collision with E, and 0 otherwise.

[1]This will be important for the blindspot attack, as its goal is to raise the chances of collision with unperceived parts of the environment.

Algorithm 1: Planner failure attack optimization

Data: E, t, and set of start configurations $s_1, ..., s_M$

$a^* \leftarrow []$;
$\sigma^* \leftarrow \frac{1}{M} \sum_{i=1...M} f_{\text{success}}(F_{\text{plan}}(E, s_i, t))$;
for *it = 1, ..., MaxIter* **do**
  $a \leftarrow$ *SamplePhysicallyRealizable*$(\mathcal{A}, E)$;
  $E' \leftarrow c(E, a)$;
  $\sigma \leftarrow \frac{1}{M} \sum_{i=1...M} f_{\text{success}}(F_{\text{plan}}(E', s_i, t))$;
  if $\sigma < \sigma^*$ **then**
    $\sigma^* \leftarrow \sigma$;
    $a^* \leftarrow a$;
  return a^*, σ^* ;

C. Planner Failure Attacks

We define "planner failure" attack as one that changes the robot's environment so as to lower the success rate of the planner. The attack is illustrated in Fig. 1. The goal of the attack is to solve the following optimization problem:

$$\underset{a}{\text{minimize}} \ \mathbb{E}_s[f_{\text{success}}(F_{\text{plan}}(c(E, a), s, t))]. \tag{1}$$

Intuitively, the goal is to find a single environment change parameter a that leads to a new environment $E' = c(E, a)$, on which the expected value of planner success (over many start configurations) is as low as possible.

To solve this optimization problem, in this paper we adopt a simple approach, which is to randomly sample a within physically-realisable values for a fixed number of iterations. We found this strategy to work well in practice, though more advanced evolutionary algorithms [11], [20] could also be used. Pseudo-code of our algorithm is shown in Algorithm 1. As the pseudo-code shows, we compute the expected value $\mathbb{E}_s[f_{\text{success}}(F_{\text{plan}}(c(E, a), s, t))]$ by taking the average of f_{success} over a set of different start configurations $s_1, ..., s_M \in \mathcal{S}$. We obtain these configurations through random sampling for each motion planning problem and keep them fixed throughout the optimization process. The function *SamplePhysicallyRealizable*$(\mathcal{A}, E)$ performs uniform sampling within the space of $\mathcal{A}$ until it finds a value of a that does not lead to a collision between environment objects. In the case where a is a new object to be placed in the environment, i.e. $c = c_1$, then *SamplePhysicallyRealizable* will also guarantee that the new object is placed within a certain threshold distance of E—to make sure that it could be "attached" or "glued" to the environment. Throughout the rest of the paper, therefore, we will refer to *physical attacks* as any attacks on the physical environment, regardless of their realism or implementation difficulty, and we will refer to *physically realizable* attacks as those that also satisfy physical and implementation constraints such as being collision-free and attachable.[2]

In order to make the attacks inconspicuous, in this paper we implement $c_1(E, a)$ as the placement of a *small-radius*

[2]Note that attacks far away from E, even though not attachable, could potentially still be realized, e.g. through a transparent base or string from the ceiling—though for simplicity we will not consider these in the paper.

sphere, i.e. $a = (g, \theta, r)$ where g is a small-radius sphere, $\theta = (x, y, z)$ is a position, and $r = \emptyset$ (i.e. positions are set w.r.t. the world reference frame). This type of attack is similar to the idea of "one-pixel" attacks in computer vision [8], since both make a point-wise change to the input.

D. Blindspot Attacks

We define "blindspot" attacks as those that lead the planner F_{plan} to generate a path that, even though not in collision with the visible environment E_v, is actually in collision with the real environment E. The attack is illustrated in Fig. 2. As shown in the figure, the adversary makes small changes to the environment (e.g. the state of an articulated object such as a door or lid; or the placement of a new obstacle) that lead an object to be placed in an area that is not perceived but likely to be used by the planned path—and thus lead to a collision. The goal of a blindspot attack is to solve the following optimization problem:

$$\underset{a}{\text{maximize}} \ \mathbb{E}_s[f_{\text{collision}}(F_{\text{plan}}(c(E, a), s, t))]. \tag{2}$$

Intuitively, the goal is to find a single environment change parameter a that leads to a new environment $E' = c(E, a)$, on which the expected value of path collision is as high as possible. Similarly to planner failure attacks, we solve this optimization problem using a sampling-based approach equivalent to Algorithm 1, but which replaces f_{success} by $f_{\text{collision}}$ and the inequality in line 7 by $\sigma > \sigma^*$.

V. Results

A. Experimental Setup

We evaluate our attack methods in the manipulation scenes of MotionBenchMaker [21], which is a tool to generate and benchmark datasets of realistic robot motion planning problems. The robot is a mobile manipulator Fetch with a 7-degree-of-freedom arm. We used MotionBenchMaker to generate a set of 20 planning problems in BenchMaker's kitchen scene and box scene. All problems have the same start configuration, but objects in the scene are randomly initialized around the robot. For each problem, the target object that the robot has to grasp is also randomly initialized (on top of a shelf on the kitchen scene, and inside the box on the box scene). The goal of each planning problem is a task-space pose of the end-effector, which is always set to a pre-grasp location with respect to the target object. When the planner is able to obtain a path within a given tolerance of the target pose, 1cm (translation) and 0.01rad (orientation) in each axis, we consider the planner to have succeeded. To evaluate lines 2 and 6 of Algorithm 1 we use $M = 20$ start configurations for each planning problem, and *MaxIter* is set to 50 unless stated otherwise. We consider obstacle placement attacks as physically realizable when they are within a threshold distance of 10cm to the environment (allowing an attacker to physically attach the object to the environment).

We evaluate our attacks on two popular sampling-based motion planners: RRT-Connect [6] and BKPIECE [15]. We selected these planners as they are both efficient and widely adopted [22], [23]. All experiments were run on a laptop PC with an 8-core 3GHz Intel® Core™ i7 Processor.

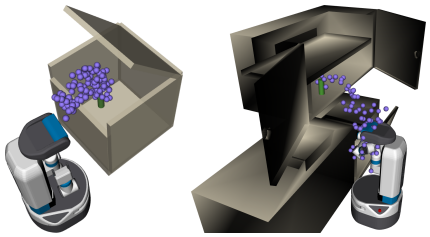
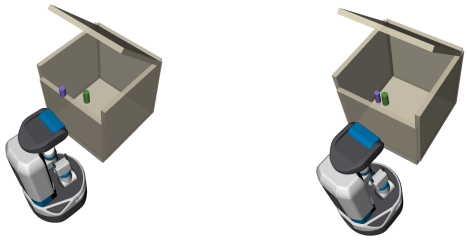


Fig. 3: Visualization of 100 unconstrained "planner failure" attacks (i.e. physical realizability not enforced) on one problem from the box scene and one from the kitchen scene.



(a) Attack object (purple) on the edge of the box.

(b) Attack object (purple) next to target (green)

Fig. 4: Examples of physically realizable "planner failure" attacks.

B. Planner Failure Attacks

Examples. To showcase planner failure attacks we simulated the placement of small fixed-size (4cm-radius) spheres as attacks. Fig. 3 shows 100 unconstrained attacks (i.e. not necessarily physically realizable) to the RRT-Connect planner in one problem from MotionBenchMaker's box and kitchen scenes. It took on average 437.5 seconds to generate 100 attacks. In the box problem, the robot has to reach inside the box and place its end-effector in a pre-grasp pose to grasp the green object. In the kitchen problem the robot has to move the cylinder from the shelf into the dishwasher. All the visualized attacks (purple spheres) lead the motion planner to not find a feasible solution to the problem with non-zero probability. We will analyze these probabilities later.

Most of the attacks shown in Fig. 3 lie in areas of the environment that are occupied by the robot at the goal configuration—however, some of these locations are physically realizable and may be imperceptible to the user due to the inconspicuous object size and location. Only 58% of these attacks are realizable (i.e. within 10cm of the environment, so they could be "attached" to it). Fig. 4 shows two of the realizable attacks of the box problem. One of them involves placing an object on the edge of the box, while the other places an object next to the target.

Effectiveness. We then evaluated the effectiveness of the attacks obtained with Algorithm 1. Fig. 5 shows the success rate of the optimal attacks, averaged over 20 motion planning problems. The figure also compares our method to 1) the success rate of the planner before any attack; 2) a baseline random-attack method (which randomly selects a point in space to place the obstacle at); 3) the success rate of unconstrained attacks that do not need to be physically

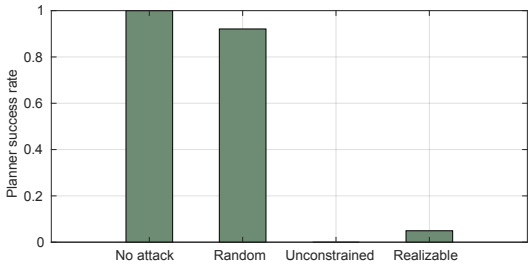


Fig. 5: Average planner success rate under "planner failure" attacks on box-scene problems.

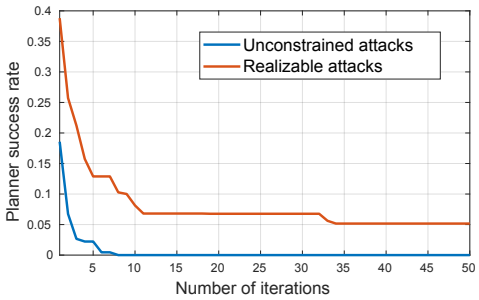


Fig. 6: Planner success rate as a function of the number of iterations of our "planner failure" attack method.

realizable (i.e. where physical realizability is not enforced, and therefore spheres may be flying in space). The figure shows that RRT-Connect always finds a path to the 20 problems (each from 20 different start configurations) when there is no attack. Random attacks slightly reduce the success rate to 92%, but our method leads to a drastically lower average success rate of 5.0%. Interestingly, the figure also shows that, were the attacker not constrained by physics, then unconstrained attacks could lead the planner to have a success rate of 0%—this is because of the higher flexibility available for where to place obstacles (e.g. flying in space).

To investigate the trade-off between attack effectiveness and the amount of computation resources used (i.e. algorithm iterations), we computed the planner success rate achieved when our algorithm is allowed to run for anywhere between 1 and 50 iterations. Fig. 6 shows the results. The figure shows that our algorithm's solution quickly reaches 10% planner success rate, in under 10 iterations. It also shows that physically realizable attacks take longer to converge to the optimal solution, compared to unconstrained attacks, due to the increased complexity of the space. As we used a simple random sampling strategy to generate attacks, both computation time and quality of the attacks could potentially be further improved by using a more advanced evolutionary [11], [20] or heuristic method.

Transferability. Finally, we evaluate the transferability of planner failure attacks. Transferability measures the ability of an attack to remain effective when applied to a different, possibly unknown, planner [24]. We compute the success rate of one motion planner when solving problems that were attacked by assuming a different motion planner. To do this, we used the attacks optimized for RRT-Connect,

TABLE I: Transferability and generalization of "planner failure" attacks: success rate of attacks on different planners*

RRTC-RRTC	RRTC-BKP	BKP-BKP
5.0% ± 20.5%	5.2% ± 18.7%	0% ± 0%

*Note: Attacks are computed for a specific planner, and then tested on a different planner, e.g. RRTC-BKP: computation on RRT-Connect and testing on BKPIECE. Results shown are success rates on the test planner.

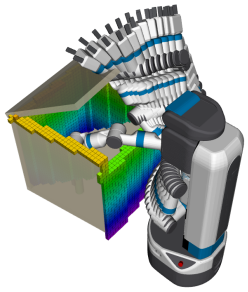


Fig. 7: Example of a "blindspot" attack, where the lid of a box is fixed at an angle where the robot is not able to fully perceive it, thus leading to a trajectory that is thought to be collision-free but is actually in collision.

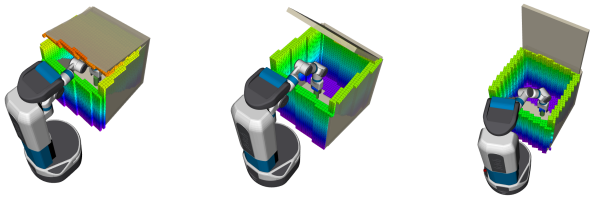
and computed the success rate of BKIPECE at solving those problems. Table I shows the results. For comparison, we also show the success rate of RRT-Connect at solving those problems, and the success rate of BKPIECE at solving problems with attacks optimized for BKPIECE. The table shows that, when we transfered the attacks generated for RRT-Connect to BKPIECE, the planner success rate was still low, almost equal to when tested on RRTC (5.2% on BKP vs 5.0% on RRTC). This shows that the failure attacks were highly transferable to a different planner. Interestingly, the table also shows that attacks optimized for BKPIECE lead this algorithm to have a 0% success rate—meaning that BKPIECE is easier to attack than RRT-Connect, which could be due to lower amount of randomness in the planner's outputs. This also shows that our attack method is generalizable to other planning methods.

The results in this section thus show that it is possible to make very small changes to an environment that lead traditional motion planners to drastically fail to find a solution to a problem—and these attacks are also transferrable to other planners. Such attacks could cause delays during robot use, as they would require a human to intervene, or the robot to start a recovery mechanism.

C. Blindspot Attacks

Example I. Fig. 7 shows an example of a blindpot attack, where the occupancy cubes (colored according to height) represent the regions of space that are perceived by the camera. The trail of the robot arm is a path planned when using the occupancy grid as occupied space. The figure shows that the robot arm collides with the lid of the box because a large section of the lid is not within the camera's field of view.

Effectiveness. To showcase blindspot attacks on this environment, we parameterized the attack a using the angle



(a) 15 Degree (b) 45 Degree (c) 90 Degree

Fig. 8: Real and perceived environment under different box lid angles. Real environment shown in grey, perceived showed with a colored occupancy grid.

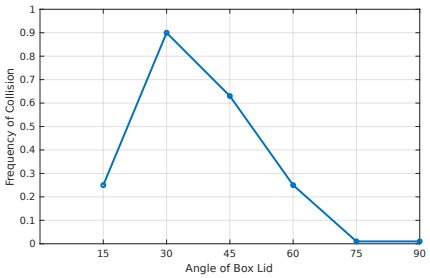


Fig. 9: Collision frequency of paths under different box lid angles, as an example "blindspot" attack. An attacker can change this angle on purpose to make the robot to collide with the environment. Larger angles make the attack more inconspicuous, as 90^o is the default state of the object.

of the box lid. Fig. 9 shows the frequency of plan collisions for several values of a, from 15 to 90 degrees. The figure shows that when the lid is wide open at 90 degrees, the planned trajectory only collides with the environment 1% of the time. This is because even though the lid is not perceived by the camera, as shown in Fig. 8, it is too far away from the region of space that is used by the obtained plans for there to be collisions. However, when a is set to 45 degrees the collision rate drastically increases to 63%, and then to 90% at 30 degrees. This is because the access space into the box is getting smaller but the camera cannot still perceive the lid fully. Only when the lid angle is small—15 degrees—does the robot start to perceive a large section of the lid (see Fig. 8) and can thus compute mostly collision-respecting plans again, with a collision rate of 25%. Fig. 9 thus represents the trade-off between attack effectiveness (i.e. collision frequency) and inconspicuousness (i.e. box lid angle values closer to 90 degrees are more inconspicuous, as this is assumed to be the usual state of the object).

Example II. We then obtained results using the same attack parameterization as that in the planner failure attacks of Section V-B—i.e. a as a new 4cm-radius sphere to add to the scene. Fig. 10 shows the attack obtained by running our method on the box scene. As the figure shows, the attack places the object inside the box, close to a lateral wall of the box but in a location that is on the camera's blindspot. The sphere is not perceived as it is occluded by the frontal wall of the box. RRT-Connect thus succeeds to find a feasible (in the occupancy grid) trajectory, but this trajectory is in collision with the placed obstacle when executed.

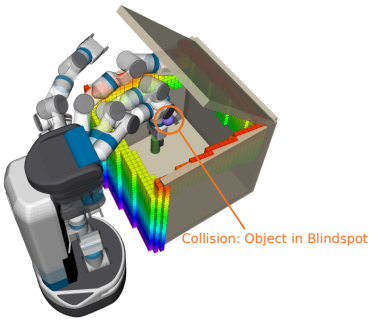


Fig. 10: Example of "blindspot" attack computed by our method, which places an object in a part of the environment that is not perceived by the robot's camera.

These results thus show that it is possible to make very small changes to an environment—slightly lowering the lid of a box from which objects are typically retrieved, or placing a small obstacle in area that is occluded by other objects—that lead traditional motion planners to compute paths that collide with the environment, thus potentially causing damage to the robot or the environment itself.

VI. Conclusions and Discussion

In this paper we proposed and evaluated two kinds of physical attacks on robot motion planners. "Planner failure" attacks occupy part of the perceived workspace, thus making the planner fail to find a solution to the motion planning problem. "Blindspot" attacks occupy space that is not perceived by the robot's sensors, causing planned trajectories to collide with the environment.

Our results show that successful attacks need only to make subtle changes to the real world, in order to obtain a drastic increase in failure rates and collision rates—leading the planner to fail 95% of the time and collide 90% of the time in problems generated with MotionBenchMaker [21]. Compared with other physical adversarial attacks in the literature, ours have relatively high effectiveness (e.g. 55% collision rates on a Reinforcement Learning attack in [4]. The results also showed that there is a trade-off between computational resources spent and the effectiveness of the attacks, but that attacks may transfer well to different planners if optimal solutions are used.

By manual inspection of the generated attacks, we noticed that many failure attacks that can significantly decrease the success rate of the planner are actually very close to the target object. This is expectable, as placing an object just next to the target object will make grasping it impossible, but at the same time leads the attack to be highly noticeable to humans, which means they are more likely to be identified as attacks. In the future, better metrics of inconspicuousness should be investigated, for example through user studies.

Other directions of future research include the investigation of possible defenses, such as the detection of small obstructing objects, recognition of adversarial attacks, automatic recovery plans that move objects around to make a problem feasible [25], or mapping algorithms for blindspot-minimization.

REFERENCES

[1] J. A. Oravec, "Robo-rage against the machine: Abuse, sabotage, and bullying of robots and autonomous vehicles," in *Good Robot, Bad Robot: Dark and Creepy Sides of Robotics, Autonomous Vehicles, and AI*. Springer, 2022, pp. 205–244.

[2] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter, "Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 1528–1540. [Online]. Available: https://doi.org/10.1145/2976749.2978392

[3] I. Evtimov, K. Eykholt, E. Fernandes, T. Kohno, B. Li, A. Prakash, A. Rahmati, and D. Song, "Robust physical-world attacks on machine learning models," *CoRR*, vol. abs/1707.08945, 2017. [Online]. Available: http://arxiv.org/abs/1707.08945

[4] X. Pan, C. Xiao, W. He, S. Yang, J. Peng, M. Sun, M. Liu, B. Li, and D. Song, "Characterizing attacks on deep reinforcement learning," in *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, 2022, pp. 1010–1018.

[5] S. LaValle and J. Kuffner, "Randomized kinodynamic planning," in *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, vol. 1, 1999, pp. 473–479 vol.1.

[6] J. Kuffner and S. 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, 2000, pp. 995–1001 vol.2.

[7] S.-T. Chen, C. Cornelius, J. Martin, and D. H. Chau, "Shapeshifter: Robust physical adversarial attack on faster r-cnn object detector," in *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2018, Dublin, Ireland, September 10–14, 2018, Proceedings, Part I 18*. Springer, 2019, pp. 52–68.

[8] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 5, pp. 828–841, 2019.

[9] C. Xie, J. Wang, Z. Zhang, Y. Zhou, L. Xie, and A. Yuille, "Adversarial examples for semantic segmentation and object detection," 2017.

[10] S. Thys, W. Van Ranst, and T. Goedemé, "Fooling automated surveillance cameras: adversarial patches to attack person detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, 2019, pp. 0–0.

[11] N. W. Alharthi and M. Brandao, "Physical and digital adversarial attacks on grasp quality networks," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, May 2024.

[12] F. Pierazzi, F. Pendlebury, J. Cortellazzi, and L. Cavallaro, "Intriguing properties of adversarial ml attacks in the problem space," in *2020 IEEE symposium on security and privacy (SP)*. IEEE, 2020, pp. 1332–1349.

[13] X. Wei, Y. Guo, and J. Yu, "Adversarial sticker: A stealthy attack method in the physical world," 2022.

[14] J. Li, F. R. Schmidt, and J. Z. Kolter, "Adversarial camera stickers: A physical camera-based attack on deep learning systems," *CoRR*, vol. abs/1904.00759, 2019. [Online]. Available: http://arxiv.org/abs/1904.00759

[15] I. A. Sucan and L. E. Kavraki, "A sampling-based tree planner for systems with complex dynamics," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 116–131, 2012.

[16] L. Pinto, J. Davidson, and A. Gupta, "Supervision via competition: Robot adversaries for learning tasks," 2016.

[17] L. Pinto, J. Davidson, R. Sukthankar, and A. Gupta, "Robust adversarial reinforcement learning," in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. PMLR, 06–11 Aug 2017, pp. 2817–2826. [Online]. Available: https://proceedings.mlr.press/v70/pinto17a.html

[18] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng, *et al.*, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[19] D. Hsu, J.-C. Latombe, and R. Motwani, "Path planning in expansive configuration spaces," in *Proceedings of international conference on robotics and automation*, vol. 3. IEEE, 1997, pp. 2719–2726.

[20] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, Apr 2002.

[21] C. Chamzas, C. Quintero-Peña, Z. K. Kingston, A. Orthey, D. Rakita, M. Gleicher, M. Toussaint, and L. E. Kavraki, "Motionbenchmaker: A tool to generate and benchmark motion planning datasets," *CoRR*, vol. abs/2112.06402, 2021. [Online]. Available: https://arxiv.org/abs/2112.06402

[22] M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review," *IEEE Access*, vol. 2, pp. 56–77, 2014.

[23] C. Zhou, B. Huang, and P. Fränti, "A review of motion planning algorithms for intelligent robots," *Journal of Intelligent Manufacturing*, vol. 33, no. 2, pp. 387–424, 2022.

[24] A. Demontis, M. Melis, M. Pintor, M. Jagielski, B. Biggio, A. Oprea, C. Nita-Rotaru, and F. Roli, "Why do adversarial attacks transfer? explaining transferability of evasion and poisoning attacks," in *28th USENIX security symposium (USENIX security 19)*, 2019, pp. 321–338.

[25] Q. Liu and M. Brandao, "Generating environment-based explanations of motion planner failure: Evolutionary and joint-optimization algorithms," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, May 2024.