Intelligent Physical Attack Against Mobile Robots With Obstacle-Avoidance

Yushan Li, Student Member, IEEE, Jianping He, Senior Member, IEEE, Cailian Chen, Member, IEEE, and Xiping Guan, Fellow, IEEE

Abstract—The security issue of mobile robots has attracted considerable attention in recent years. In this article, we propose an intelligent physical attack to trap mobile robots into a preset position by learning the obstacle-avoidance mechanism from external observation. The salient novelty of our work lies in revealing the possibility that physical-based attacks with intelligent and advanced design can present real threats while without prior knowledge of the system dynamics or access to the internal system. This kind of attack cannot be handled by countermeasures in traditional cyberspace security. To practice, the cornerstone of the proposed attack is to actively explore the complex interaction characteristic of the victim robot with the environment and learn the obstacle-avoidance knowledge exhibited in the limited observations of its behaviors. Then, we propose shortest-path and hands-off attack algorithms to find efficient attack paths from the tremendous motion space, achieving the driving-to-trap goal with low costs in terms of path length and activity period, respectively. The convergence of the algorithms is proved and the attack performance bounds are further derived. Extensive simulations and real-life experiments illustrate the effectiveness of the proposed attack, beckoning future investigation for the new physical threats and defense on robotic systems.

Index Terms—Intelligent attack, intentional learning, mobile robots, obstacle-avoidance.

I. INTRODUCTION

With integrated communication, computation, and control to support the operations in the physical world, the mobile robots can be seen as a typical cyber-physical system (CPS), attracting extensive attention, thanks to their excellent flexibility and scalability. From UAVs to UGVs, either single or multiple coordinated mobile robots are becoming more and more pervasive in industrial and military fields, e.g., logistics transportation, environment exploration, surveillance, and reconnaissance. With this ever-increasing popularity, their security becomes critical yet imperatively challenging [1], given numerous possible malicious or unanticipated behaviors.

This article investigates a novel intelligent physical attack against mobile robots without relying on any prior knowledge. The ultimate goal of the attacker is to learn the obstacle-avoidance mechanism of a mobile robot from external observation, and then leverage it to fool the target robot into a preset trap. This problem is inspired by the intrinsic interaction nature of a robot with the physical environment and its security issues. On the one hand, prior works concerning the security of mobile robots are mainly developed from the perspective of cyberspace [2], [3], while physical attacks are less noticed yet also critically important. It is universally needed for mobile robots in the real world to interact with the physical environment [4]. By disguising as an obstacle and physically approaching a robot, the resulting obstacle-avoidance behavior is inevitable, providing support to trap the robot. On the other hand, numerous defense techniques assume possible attacks are powerful beforehand [5], [6], obtaining robust security performance while incurring more costs, e.g., hardware computation burdens. Indeed, if an attack is figured to be almost impossible to launch, then there is no need to design sophisticated countermeasures along with performance degradations, better guiding the systems defense design.

Despite some similar features between the attack problem in this article and classic pursuit-evasion [7], [8], [9] and herding problems [10], [11], [12], we point out that there are still major differences in their application scenarios and basic model formulation. First, the physical attack here is not purely about controlling a robot to capture or track the victim robot but making the victim robot move to a preset deterministic position. Second, unlike the aforementioned works that usually assume complete or partial knowledge about interaction models is available, the attacker here has no prior knowledge about the victim robot. It needs to actively learn the necessary obstacle-avoidance knowledge first, making the attack more generalizable and applicable to a majority of obstacle-avoidance mechanisms, which is a significant and unique component of this article. Besides, compared with directly intercepting (e.g., missile guidance [13]) or decoding-and-manipulating the communication messages (especially when the control communication is protected with strong encryption [14], [15]), the proposed physical attack is more feasible, stealthy, and economically viable, presenting serious threats.
To systematically design and evaluate the intelligent physical attacks on mobile robots, three major challenges must be addressed. First, the attacker is unaware of the prerequisite information that supports the attack (such as the obstacle detection range and goal position), making it an impediment to formulating the attack model by traditional techniques, e.g., parameter identification and model predictive control. Second, the attacker and victim robots are mutually influenced in the motion space, incurring difficulties to find the feasible solution space that meets the attack requirements. Third, even if a feasible solution space is found, it is hard to optimize strategy designs to achieve the final purpose with low attack costs.

This article is an extension of the preliminary work presented in [16], providing a detailed and rigorous treatment of model learning, attack design, and performances analysis, as well as significant novel simulation results. The main contributions of this article are as follows:

1) We investigate the possibility of achieving an intelligent and advanced physical attack against mobile robots, merely utilizing external observations and not relying on any prior information of the system dynamics. Taking the universal obstacle-avoidance mechanism as the attack interface, we provide a systematic attack scheme including knowledge-learning and attack implementation.

2) We propose an intentional excitation-based learning approach to acquire the obstacle-avoidance knowledge. By characterizing the obstacle-avoidance behaviors and disguising the attacker as an obstacle, we demonstrate how to excite various avoidance reactions of the victim robot. Based on observations over this process, we establish the featured data pairs that reflect the underlying mechanism and further regress it by learning-based methods.

3) We design two driving-to-trap attack algorithms by taking the attack path length and activity period as the objectives, respectively. We prove the convergence of the attack algorithms from the perspective of the descent search. The performance bounds of the algorithms are further derived concerning the optimal cost in theory. Extensive simulations and real-life experiments are conducted to illustrate the effectiveness of the proposed attack.

In a larger sense, this article reveals the vulnerability of system security from external observation and excitation. As the obstacle-avoidance behavior is indispensable for a mobile robot to interact with the environment, the proposed attack is hard to defend when the robot has no prior knowledge about the malicious behavior, where the attacker is not always consistently but intermittently active during the process. Therefore, our work lays the foundation to explore more potential and advanced attacks against mobile robots from physical aspects and provides deeper insight to guide the defense design in the future.

The rest of this article is organized as follows. To begin with, Section II presents an overview of the relevant literature. In Section III, the kinematics and obstacle-avoidance modeling for both holonomic and nonholonomic mobile robots are introduced. The learning scheme for the obstacle-avoidance mechanism is proposed in Section IV. Section V presents the attack strategies with performance analysis. Simulations and experiments are shown in Sections VI and VII, respectively. Finally, Section VIII concludes this article. A roadmap of the main results of this article is shown in Fig. 1.

II. RELATED WORK

There are mainly two related topics: CPS security and obstacle-avoidance mechanism. Given the richness of both fields, a brief overview of some representative works is given.

**CPS security**: The security shows the ability of a system to govern malicious behaviors or unanticipated events [17]. In the literature, attacks against mobile robots are mainly from cyberspace, and they can be roughly divided into three categories: denial of service, replay, and deception attacks [2]. In these attacks, communication channels are jammed/disrupted [18], or the control data and measurements are compromised/ altered [19], and thus compromising the systems in critical and adversarial scenarios. Many efforts have been devoted to designing corresponding countermeasures. For instance, Pasqualetti et al. [6] characterized the undetectable and unidentifiable attacks and designed corresponding detection filters. Chang et al. [20] considered the secure estimation problem where the set of attacked nodes can change with time. The motion synchronization methods were developed for robot formation considering the presence of robot malfunctions [21], [22] or undesirable formation variation [23], [24]. A group of robots were deployed to herd an adversarial aerial attacker to defend safety-critical infrastructure in [25]. However, most existing research relied on the assumption that the attacker has some knowledge or can access the robot system. For example, malicious agents have knowledge of the system structure or nodes’ states [5] or the packets transmitted over the network are corrupted [26], [27]. These assumptions may no longer hold given the cryptographic solutions in practice.

In recent years, stealthy physical attacks emerged and presented real threats, where the physical components were targeted to compromise or even destroy the system. For instance, the GPS sensor readings can be disturbed by GPS spoofing attack [28], [29]. Designed acoustic noises can alter gyroscopic sensor data, leading to drone crashes [30]. Even the important values stored in memory (e.g., EEPROM, Flash memory) can be corrupted by heating a memory cell while the device is without any damage [31]. Compared with cyber attacks, these physical attacks have a wider range of applicability, while the traditional detection techniques from the computer security...
community are usually not effective in handling them [1]. Note that these emerging physical attacks are still elementary and not sufficiently intelligent. For instance, the attacks were designed against a specific transducer by utilizing its sensing mechanism and cannot be generalized. In addition, these physical attacks aiming to disturb the system performance were designed with an "open-loop" flavor and without sophisticated control design, which motivates this work to investigate possible physical attacks with intelligence.

Obstacle-avoidance mechanism: Obstacle-avoidance is the major interface for mobile robots to interact with the outside environment. Numerous methods have been developed in the literature, for example, potential field based approach [32], genetic algorithm based approach [33], fuzzy logic based approach [34], neural network based approach [35], etc. (see [36] and [37] for a detailed review). According to their algorithm characteristic, similar to [38], we divide these algorithms into two main types: deterministic (e.g., artificial potential field (APM), learning-based method) and search-based (e.g., dynamic window approach (DWA), genetic approach, evolutionary algorithm). The former is determined-model driven and the solution for the current situation is unique and found quickly. The key idea of the latter one is to search for all feasible solutions and select the best one according to an evaluation function, which is time-consuming. No matter what obstacle-avoidance method is used, the essence of circumventing the obstacle with certain trajectory adjustment toward its goal is identical and directly leaks to external observation. This nature is captured and leveraged in our attack design.

In summary, it remains an open yet critical issue to investigate what kind of intelligent and generalized attack is available from the physical side while without prior knowledge about the system.

III. Preliminary and Problem Formulation

In this section, we first introduce some basics about the kinematics of a mobile robot. The motion dynamics of both holonomic and nonholonomic mobile robots are presented. Following this, we show what an important role the obstacle-avoidance plays. At last, our problem of interest is formulated. Some important notation and definitions are given in Table I.

A. Motion Control for Mobile Robots

It should be mentioned that there are generally two kinds of mobile robots: nonholonomic (car-like wheeled mobile agents, unicycles, etc.) and holonomic. Although the former one appears more common in daily life, their instantaneous movement is restricted laterally [39], making the control more challenging than that of the latter kind.

For a nonholonomic mobile robot, the posture is usually represented by its position \((x, y)\) and orientation \(\theta \in [0, 2\pi)\) with respect to X-axis, denoted as \(p = [x, y, \theta]^T\). The motion is controlled directly by linear velocity \(v\) and angular velocity \(\omega\) or velocities of two driving wheels, which are equivalent to each other. Then, its kinematics is modeled as

\[
\begin{align*}
\dot{x} &= v \cos(\theta) \\
\dot{y} &= v \sin(\theta) \\
\dot{\theta} &= \omega
\end{align*}
\]

(1)

Accordingly, the discrete form of (1) is given by

\[
\begin{align*}
x(t^e_i) &= x(t^e_{i-1}) + v(t^e_{i-1}) \cdot \cos(\theta(t^e_{i-1})) \cdot T \\
y(t^e_i) &= y(t^e_{i-1}) + v(t^e_{i-1}) \cdot \sin(\theta(t^e_{i-1})) \cdot T \\
\theta(t^e_i) &= \theta(t^e_{i-1}) + \omega(t^e_{i-1}) \cdot T
\end{align*}
\]

(2)

where \(T\) is the motion control period, and \(t^e_i (i = 1, 2, \ldots)\) denotes the control time instant satisfying \(t^e_i - t^e_{i-1} = T\). The motion control aims to drive the robot toward its desired posture \(p^* = [x^*, y^*, \theta^*]^T\) to achieve predefined tasks by designing the velocities \(v\) and \(\omega\). Define the instantaneous posture error as \(p_e = p^* - p = [x_e, y_e, \theta_e]^T\), and the controller can be represented as

\[
\begin{bmatrix}
v \\
\omega
\end{bmatrix} =
\begin{bmatrix}
u_e(p_e, \dot{p_e}) \\
\omega_e(p_e, \dot{p_e})
\end{bmatrix}
\]

(3)

where \(u_v\) and \(u_\omega\) are designed to control \(v\) and \(\omega\) such that the tracking error \(\|p_e\|_2\) goes to zero. Note that (3) represents a general first-order motion control for nonholonomic mobile robots. Many works have been developed with stability guarantees [40], [41], [42]. They usually require the measurements of \(\dot{\theta}\) and are of proportional-derivative-trigonometric type.\(^2\)

Considering the nonlinear characteristic of nonholonomic robots, it is common to introduce a "hand" position \(h = [x_h, y_h]^T = [x + d_h \cos \theta, y + d_h \sin \theta]^T\) to simplify the motion control, where \(d_h\) is a distance starting from \(p\) along the orientation axis. Taking \(h\) into (1), one has

\[
\begin{bmatrix}
x_h \\
y_h
\end{bmatrix} =
\begin{bmatrix}
\cos \theta & -d_h \sin \theta \\
\sin \theta & \ d_h \cos \theta
\end{bmatrix}
\begin{bmatrix}
v \\
\omega
\end{bmatrix}.
\]

(4)

\(^2\)The introduction of the trigonometric function is mainly caused by the decomposition of linear velocity. For example, in [42], the motion controller is designed as \(v = v^\theta \cos \theta^* + k_x x^*, \omega = \omega^* + k_\theta \theta^* + v^\theta \sin \theta^*/\theta^*\).
By output feedback linearization, the controller for \((v, \omega)\) can be designed as [43]
\[
\begin{bmatrix}
v \\
\omega
\end{bmatrix} =
\begin{bmatrix}
\cos \theta & \sin \theta \\
-\frac{1}{d_h} \sin \theta & \frac{1}{d_h} \cos \theta
\end{bmatrix}
\begin{bmatrix}
u_x(x_e, \dot{x}_e) \\
u_y(y_e, \dot{y}_e)
\end{bmatrix}
\] (5)
which gives the following simplified point regulation of the hand position:
\[
\begin{bmatrix}
\dot{x}_h \\
\dot{y}_h
\end{bmatrix} =
\begin{bmatrix}
u_x(x_e, \dot{x}_e) \\
u_y(y_e, \dot{y}_e)
\end{bmatrix}.
\] (6)
The controller (6) is decoupled and designed in proportional-derivative form to guarantee the stability of \(h\).

There are a few works focusing on directly controlling the applied torque and force for the wheels based on \(p_e\) [43], [44], which is essentially a second-order controller. In this article, we do not focus on a specific motion controller design but emphasize that the desired posture of the robot can be affected to achieve our attack no matter what motion controller the robot uses.

As for a holonomic mobile robot, the kinematics in two directions is independent and its discrete form is given by
\[
\begin{align*}
x(t^e_{n+1}) &= x(t^e_n) + v_x(t^e_n) \cdot T \\
y(t^e_{n+1}) &= y(t^e_n) + v_y(t^e_n) \cdot T
\end{align*}
\] (7)
where \(v_x\) and \(v_y\) are velocities along X- and Y-axis directions, respectively. Since the motion of a holonomic robot is a direct composition of the motions in two directions, the orientation is usually neglected. Therefore, different from (3), here the motion control is directly formulated as
\[
\begin{bmatrix}
v_x \\
v_y
\end{bmatrix} =
\begin{bmatrix}
u_x(p_x, \dot{p}_x) \\
u_y(p_y, \dot{p}_y)
\end{bmatrix}.
\] (8)
The proposed intelligent attack in this article applies to both kinds of robots. Hereafter, we mainly illustrate our work on non-holonomic robots. It is easy to extend the results to holonomic ones due to the simple motion characteristic, and they will be briefly discussed.

B. Obstacle-Avoidance Behavior Modeling

The obstacle-avoidance mechanism is the major interface for a robot to interact with the physical environment and is universally needed for the robot to work. Let \(p_{ob}\) and \(v_{ob}\) be the state and velocity of the obstacle, respectively. Considering that avoidance behavior is mainly determined by the relative posture and motion between the robot and the obstacle, we formulate it as the following general function:
\[
u_{ob} = f(p_{ob} - p, p^* - p, v_{ob}, v)
\] (9)
where \(p^*\) is the instantaneous desired posture determined by a specific task (e.g., it can represent the target leader posture in formation control). In some algorithm designs, the influence of \((p^* - p)\) is negligible when \(p_{ob}\) is very close to \(p\).
Next, we characterize the obstacle-avoidance behavior under (9). Let \(A_d(p)\) be the obstacle detection area of a robot. The obvious characteristic of \(f\) is its boundedness regardless of the detailed design, which can be described as
\[
\begin{align*}
\|f(\cdot)\|_2 &= 0, & \text{if } p_{ob} \notin A_d(p) \\
0 \leq \|f(\cdot)\|_2 &\leq \bar{u}_{ob}, & \text{if } p_{ob} \in A_d(p)
\end{align*}
\] (10)
where \(\bar{u}_{ob}\) is the output bound. Another salient characteristic is the directionality of the obstacle-avoidance behavior. For ease of expressions, let \(T_\theta\) be a general notation for a simple curve (trajectory) that divides the X-Y plane into two parts, \(T^+\) and \(T^-\) (the subscript \(\theta\) will be replaced when indicating a specific trajectory). Define \(I_{\theta}(T_\theta, p)\) as the indication function
\[
I_{\theta}(T_\theta, p) = \begin{cases} 
0, & \text{if } p \in T_\theta \\
+1, & \text{if } p \in T^+ \\
-1, & \text{if } p \in T^-
\end{cases}
\] (11)
Then, when an obstacle occurs in the detection range at moment \(k\) [i.e., \(p_{ob} \in A_d(p(k))\)] and \(T_m(p)\) is the instantaneous tangent line of the robot’s motion, the obstacle-avoidance behavior satisfies
\[
I_{\theta}(T_m(p(k)), p_{ob}) \cdot I_{\theta}(T_m(p(k)), p(k + 1)) < 0
\] (12)
which indicates that the robot will move toward the opposite side of the obstacle taking \(T_m(p(k))\) as a boundary.
There are some notable points about the obstacle-avoidance modeling in some complicated cases.
1) Geometric shape: In practice, the obstacle is detected by the sensors as a series of dense points, and the avoidance reaction is based on processing these points by certain computation rules. For example, they can be enclosed by a convex geometric shape that has a simple representation (e.g., a circle) and the robot moves around the central point with a known radius [45].
2) Multiple obstacles: When multiple obstacles are involved, the robot will evaluate the influences of the obstacles and obtain their composite impact. In some simple cases, the robot can directly take the nearest obstacle into its avoidance procedures at the moment, which always guarantees a safe avoidance behavior.
The avoidance model (9) covers a majority of algorithms that are based on local detection range and use the obstacle’s instantaneous motion state as the input. Considering when the obstacle is moving, some advanced methods are developed to improve the avoidance efficiency by predicting and evaluating the motion of the obstacle [46], [47], [48], [49]. Nevertheless, these methods usually require some prior knowledge, e.g., the obstacles are assumed to move along deterministic paths [48] or satisfy certain nominal kinematic constraints [49]. Based on the above analysis, we make the following assumption throughout this article.

Assumption 1: The obstacle-avoidance detection range of the victim robot is local, and the obstacle-avoidance algorithm it uses can be modeled by (9) and is an injective mapping. Besides, the observations of the attack robot on the victim robot are corrupted with Gaussian noise \(\mathcal{N}(0, \sigma^2)\).

Remark 1: For the class of moving obstacle avoidance algorithms with prior knowledge acquired, the attack idea and
procedure proposed in this article can still be applied with additional computation costs and higher algorithm complexity, which come from the two aspects. First, the historical positions and velocities of the obstacle itself need to be considered for modeling and learning the mechanism \( f \). Second, the search space during attack implementation should also be increased according to the time horizon used for learning \( f \).

C. Problem of Interest

Denote the malicious attack robot and the victim robot as \( R_v \) and \( R_a \), respectively, and let \( f \) represent the obstacle-avoidance mechanism. Our work applies to a basic and representative application scenario where \( R_v \) is performing a go-to-goal task and able to detect and avoid obstacles occurring in its surroundings by \( f \). Concretely, \( R_v \) aims to fool \( R_a \) into a preset trap \( p_t \) by disguising as an obstacle while it can observe (or sense) \( R_v \) and has limited moving capability in this process. Define \( J \) as the observed dataset reflecting \( f \), and the attack is mathematically transformed to solve the following problems:

1) Learning \( f \): Given \( J \), find the mapping relation \( \phi \) that embodies \( f \) by solving

\[
\min_{\phi(J)=f} \| f - \hat{f} \|_2. \tag{13}
\]

2) Attack strategy design: Given \( \hat{f} \), find a group of attack inputs \( u_{a,0:H} \) that drives \( R_v \) into \( p_t \) by solving

\[
P_0 : \min_{H, u_{a,0:H}} C_0(u_{a,0:H}) \quad \text{s.t.} \quad \| p_v(H) - p_t \|_2 \leq \delta \tag{14}
\]

where \( C_0 \) is the attack cost (e.g., attack path length or activity period) and \( \delta \geq 0 \) is a constant that indicates attack termination.

To solve problem (13), the key point is to capture the essential principle about how the mobile robot will adjust its trajectory after an obstacle is detected nearby and construct a corresponding dataset from observations. For problem (14), it is extremely hard to get an optimal analytic solution due to the implicit form of \( f \). Nevertheless, borrowing ideas from the sampling-based approaches [50], [51], we are able to design efficient algorithms to obtain suboptimal solutions in terms of different attack costs. The whole framework of this article is shown in Fig. 2.

IV. LEARNING SCHEME FOR OBSTACLE-AVOIDANCE MECHANISM

When \( R_a \) encounters an obstacle within its detection area, it will evaluate the obstacle’s influence and take corresponding actions, deviating from its desired trajectory. Inspired by this, a learning scheme for the obstacle-avoidance mechanism is proposed. This scheme consists of the following three parts:

1) Presampling: Ideally, the instantaneous motion information of \( R_v \) (such as orientation, linear, angular, and acceleration velocities) can be obtained based on three consecutive position samplings. This constitutes the cornerstone of the following steps.

2) Intentional excitation: With the ability of observing \( R_v \)’s motion, \( R_a \) intentionally excites \( R_v \) by approaching it as a detected obstacle. Then, \( R_v \) is steered to generate various obstacle-avoidance behaviors, enriching the observations to learn the underlying mechanism.

3) Modeling learning: Based on the collected data of the excitation trials, \( R_a \) computes the featured process variables that exhibit the obstacle-avoidance behavior and regresses the underlying mechanism (9) by learning-based methods (e.g., SVR, RRT).

A. Presampling: Data Acquisition

Recall that the posture \( p \) of a mobile robot is updated every control period \( T \), and its trajectory during the period \( T \) can be approximated as a straight line. Equipped with advanced sensors, \( R_a \) is able to measure its relative displacement with a moving object. Let the sampling period of \( R_v \) be \( T = NT \), and denote by \( t_k \) the kth sampling time satisfying \( t_k - t_{k-1} = \bar{T} \). For simplicity, we use the subscript \( k \) to denote the sampling time, e.g., \( x_k \) represents the \( x(t_k) \).

After three consecutive sampling moments, the instantaneous variables of \( R_v \)’s motion are estimated by

\[
\begin{align*}
    v_{x,k+1} &= \frac{x_{k+1} - x_k}{\bar{T}}, & v_{y,k+1} &= \frac{y_{k+1} - y_k}{\bar{T}} \\
    v_{k+1} &= \sqrt{v_{x,k+1}^2 + v_{y,k+1}^2}, & \theta_{k+1} &= \arctan\frac{v_{y,k+1}}{v_{x,k+1}} \\
    a_{k+1} &= \frac{v_{y,k+1} - v_y}{\bar{T}}, & \omega_{k+1} &= \frac{\theta_{k+1} - \theta_k}{\bar{T}}
\end{align*} \tag{15}
\]

where \( v, w, \) and \( a \) represent the linear, angular, and accelerated velocities, respectively. Note that \( R_v \) should have the latest three groups of observations to calculate (15). For holonomic mobile robots, this process is much easier with higher precision, and \( v_x, v_y \) can be directly used. To make a unified statement, let \( v_1 = v, v_2 = \omega \) if the robot is nonholonomic or \( v_1 = v_x, v_2 = v_y \) if holonomic.

Remark 2: Generally, the motion control period \( T \) is very small in practice (e.g., 0.05 s or 0.1 s), and the sampling period \( \bar{T} \) is determined by the information sensing and processing ability that \( R_a \) could afford. Note that the influence of observation noises can be largely weakened if one set a larger sampling period and then calculate an average value in a smaller period (e.g., \( \bar{T} \)). This will not jeopardize the following analysis for the learning and attack procedures.
Note that (15) is a noise-free formulation and the real observations of $R_a$ are corrupted with Gaussian noise $\mathcal{N}(0, \sigma^2)$. According to the 3σ-principle [52], for Gaussian distribution with zero expectation value, the variable locates in $[-3\sigma, +3\sigma]$ with probability 0.997, and this interval is basically regarded as the actual range. Therefore, we take 3σ as the maximum error bound of the corresponding measurement in this article. And the analysis of the measurement noise will be given later.

For $R_a$, it also needs to discriminate what type $R_a$ is first. This can be done by checking the smoothness of the reaction trajectory; for the velocity direction of the holonomic robot is easy to change suddenly, while that of the nonholonomic robot is strictly constrained. Based on (15), $R_a$ can master $R_v$’s instantaneous motion information of any time. This constitutes the foundation of the following steps.

B. Intentional Excitation: Trajectory Trial

The obstacle detection area of $R_v$ can be generalized to a circular region mostly. If we do not consider obstacles behind the robot when it moves forward, the detection area is modeled as a sector directly. Then, at this stage, the primary objective of $R_v$ is to infer the radius $D$ and angle range $\alpha$ of the sector, which are together denoted as $(D, \alpha)$.

When $R_v$ moves close to $R_a$, $R_v$ makes a record of $R_a$’s relative position, heading and bearing with it. After a period $T$, $R_v$ does the measurement again. With the two readings, $R_v$ is able to calculate $R_a$’s position variation $\Delta s$ and heading variation $\Delta \theta$ after $T$, as shown in Fig. 3(a). During time slot $[t_k^a, t_{k+1}^a]$, we have

$$\begin{align*}
\Delta s(k) &= \sqrt{(\Delta x_{k+1}^a)^2 + (\Delta y_{k+1}^a)^2} \\
\Delta \theta(k) &= \arctan \frac{\Delta y_{k+1}^a}{\Delta x_{k+1}^a} - \alpha
\end{align*}$$

(16)

where $\Delta x_{k+1}^a = x(t_{k+1}^a + T) - x(t_k^a)$ and $\Delta y_{k+1}^a = y(t_{k+1}^a + T) - y(t_k^a)$. Let $p_g$ represent the goal position of $R_v$. In normal situations, $R_v$ goes straight forward to $p_g$, i.e., $\Delta \theta = 0$. Therefore, $R_v$ is detected as an obstacle within $(D, \alpha)$ by $R_a$, if $\Delta \theta \neq 0$. Meanwhile, after the orientation of $R_v$ becomes stable, $R_v$ also records the trajectories of $R_a$ moving straightforward to $p_g$, as shown in Fig. 3(b) and (c).

It is remarkable that when the mission area is obstacle-free, moving straightforward to the goal position is the most efficient way. There are also cases where the robot is specified to go to the goal in a smooth curve, and we can utilize classic polynomial fitting technique to approximate the trajectory [53]. Let $\hat{p}_g$ be the polynomial order, $\tilde{x} = [x^n v, x^{n-1} v, \ldots, x^0]^T$, and $P \in \mathbb{R}^{n+1}$ be the coefficient vector associated with $\tilde{x}$. A point in the trajectory is modeled as

$$y = P^T \tilde{x} + \eta$$

(17)

where $\eta$ is independent identically distributed Gaussian noise satisfying $\mathcal{N}(0, \sigma^2)$. Then, the problem is transformed to find the best $P$ from groups of linear equations (17), where the solving procedures are the same as that of a straight line ($n_t = 1$ in this case). Therefore, we mainly focus on the latter case for simplicity.

Repeatedly, suppose there are $n_z$ groups of trajectories recorded, and the observed points associated with the $l$th trajectory satisfy

$$T_{i}^{(l)} : y_{ij} = P_l \tilde{x}_{ij} + \eta_{ij}, j = 1, 2, \ldots, n_t$$

(18)

where $n_t$ is the number of recorded points in this trajectory. Let $d_T(T_{i}, p)$ be the minimal distance from a position $p$ to $T_{i}$, and we obtain the following result.

Theorem 1 (Estimate the goal position): Given $n_t$ observed points in $T_{i}^{(l)}$, if $n_t \to \infty$, the estimated $T_{i}^{(l)}$ will pass through $p_g$, satisfying

$$\lim_{n_t \to \infty} d_T(T_{i}^{(l)}, p_g) = 0.$$  

(19)

Given trajectories $\{T_{i}^{(l)} \mid l = 1, \ldots, n_z\}$ and let $\hat{p}_g(n_z)$ be the corresponding least square estimator of the goal position $p_g$. If $n_z \to \infty$, $\hat{p}_g$ is accurately estimated, i.e.,

$$\lim_{n_z \to \infty} \| \hat{p}_g(n_z) - p_g \|_2 = 0.$$  

(20)

Proof: The proof is provided in Appendix A.

Remark 3: Theorem 1 illustrates how the accuracy is influenced by the recorded trajectories, where $n_z$ determines how close $p_g$ is to a recorded trajectory, while $n_z$ determines how close $p_g$ is to $\hat{p}_g$. Specifically, according to Chebyshev inequality (as shown in the proof of Theorem 1), the estimation errors of a single trajectory $T_{i}^{(l)}$ and $\hat{p}_g$ go to zero at the rate of $O \left( \frac{1}{n_t} \right)^3$ and $O \left( \frac{1}{n_z} \right)$, respectively.

If there are other obstacles occurring in $R_v$’s detection area, an efficient way for $R_v$ is adding up the impacts of the obstacles and using the composite impact as the instantaneous recorded input. Meanwhile, the attack process of $R_a$ will simultaneously take

3For two real-valued functions $f_1$ and $f_2$, $f_1(x) = O(f_2(x))$ as $x \to x_0$ means $\lim_{x \to x_0} |f_1(x)/f_2(x)| < \infty$.  

Fig. 3. Illustration of the learning process for $R_a$. (a) $R_v$’s reaction after detecting $R_a$ as an obstacle. (b) Learning $R_v$’s detection radius $D$. (c) Learning $R_v$’s detection area $O$.
Algorithm 1: Excitation-Based Obstacle-Avoidance Learning.

**Input**: Posture regulating variables $\Delta d, \Delta \alpha$, constant $k_{\text{max}}$.

**Output**: Detection area $(D, \alpha), p_g$ and $f$;

1. **Initialize**: $R_a$ moves to remote posture $p_a$ such that $R_a$ is directly ahead of $R_v$ and $\phi_r = 0$;
2. **while** $\Delta \theta = 0$ **do**
   3. $R_a$ moves towards $R_v$ such that $d(p_a, p_v) = d(p_a, p_v) - \Delta d$, and calculate $\Delta \theta$ by (16);
   4. **end while**
3. Record $D$, and the following trajectory as $\mathcal{T}_r^{(1)}$. Then reset $R_a$ such that $d(p_a, p_v) = D$ and $\phi_r = +90^\circ$
4. **while** $\Delta \theta = 0$ **do**
   7. $R_a$ moves toward $R_v$ such that $\phi_r = \phi_r - \Delta \alpha$, and calculate $\Delta \theta$ by (16);
   8. **end while**
5. **for** $k \leftarrow 1$ to $k_{\text{max}}$ **do**
   9. $R_a$ randomly moves to $p_a \in (p_v, D, \alpha)$, and compute $Q_{\text{in}} = [\theta', v_1, v_2, a, d_{\text{env}}, \phi_r]$ at $t_k^-$;
   10. Wait for a time slot $\mathcal{T}$, compute $Q_{\text{out}} = [\Delta s, \Delta \theta]$ at $t_k^{+1}$, and $\mathcal{J} = \mathcal{J} \cup \{Q_{\text{in}}, Q_{\text{out}}\}$;
   11. **end for**
6. Use $\mathcal{J}$ and learning-based method (e.g., SVR) to regress $f$;
7. **Return** $(D, \alpha), p_g$ and $f$.

Accordingly, the obstacle-avoidance knowledge dataset $\mathcal{J}$ is constructed by

$$\mathcal{J} = \bigcup_{k \in \mathcal{J}} \{Q_{\text{in}}(k), Q_{\text{out}}(k)\}$$

where $\mathcal{F} = \{k \in \mathbb{N}^+: d_{\text{env}}(k) \leq D, \phi_r(k) \in \alpha\}$.

Based on above analysis, the whole steps of intentional learning are designed as Algorithm 1, by which $R_a$ learns the obstacle-detection area $(p_v, D, \alpha)$ and goal position $p_g$ [Lines 2–10] and regresses the obstacle-avoidance mechanism $f$ of $R_v$ [Lines 11–16], respectively. The complexity of Algorithm 1 is simply determined by the number of the required observations, where $R_a$ just moves to obtain the required independent observations and does not involve other specific computation procedures. Taking $p_g$ as an example, two independent groups of observations $(n_z = 2)$ on two nonparallel trajectories $(n_z = 2)$ toward $p_g$ are sufficient to obtain an accurate $\hat{p}_g$ in the ideal noise-free situation. As Remark 3 indicates that the estimation error will decay to zero at a rate of $O(\frac{1}{k_{\text{max}}})$ in noisy settings, setting $n_z = 3$ and $n_{\text{env}} = 30$ would be sufficient for our purpose, which is also verified in simulations. If the position and motion parameter settings change, one only needs to sample more observations to ensure a reliable $\hat{p}_g$.

SVR method has good performance on nonlinear regression and strong generalization ability when the amount of data is not vast. It is insensitive to the models of the objects and has a certain tolerance for data noises due to the $\epsilon$-error tube design [54]. Thus, it is adopted in this article to learn the obstacle-avoidance mechanism using the collected data, which can be formulated as solving the following problem:

$$\min_{\omega, \beta, \xi^-_k, \xi^+_k} \frac{1}{2}||w||^2 + C \sum_{k=1}^{\mathcal{J}} \left(\xi^-_k + \xi^+_k\right)$$

s.t. $f(Q_{\text{in}}(k)) - Q_{\text{out}}(k) \leq \epsilon + \xi^-_k$

$$Q_{\text{out}}(k) - f(Q_{\text{in}}(k)) \leq \epsilon + \xi^+_k$$

$$f(Q_{\text{in}}(k)) = w^T \psi(Q_{\text{in}}(k)) + \beta$$

$$\xi^-_k \geq 0, \xi^+_k \geq 0, k = 1, 2, \ldots, |\mathcal{J}|$$

where $C > 0$ is the regularization parameter for the $\epsilon$-sensitive loss function, $\xi^-_k$ and $\xi^+_k$ are two slack variables, $\psi(Q_{\text{in}}(k))$ represents the mapping feature vector of $Q_{\text{in}}(k)$ and can be dealt by the commonly used kernel tricks. The obtained $\omega, \beta$ and the selected kernels together constitute the final learned obstacle-avoidance mechanism $\hat{f}$. More details about the solving process of the SVR can be found in [54].

It is worth noting that SVR method utilizes a $\epsilon$-tube as a regularization technique to improve the model robustness, which acquires certain tolerance for the observation noises. Also, since the observation noise of $Q_{\text{out}}$ accumulates during the observation period $T$, the learning method will achieve the ideal performance if $\mathcal{T} \rightarrow T$. In addition, although the learned $\hat{f}$ is a mapping from $Q_{\text{in}}$ to $Q_{\text{out}}$, the model is not necessarily a surjective or injective mapping. In other words, $\hat{f}$ cannot be inversely used to obtain the input given outputs in most cases.

$R_v$ and other obstacles into account. Considering the possibility that $R_a$’s sensor view may be blocked by other obstacles, $R_a$ can be equipped with advanced onboard sensors at an appropriate height, e.g., deploying an omnidirectional camera on the top of $R_a$. Also, $R_a$ can actively move to other positions such that $R_v$ is observed, as in the intentional learning stage.

C. Model Learning: SVR-Based Method

When $(D, \alpha)$ of $R_a$ is obtained, $R_a$ is controlled to excite $R_v$ by blocking in $(D, \alpha)$. In this step, $R_a$ also records its relative distance $d_{\text{env}}$ and bearing $\phi_r$ with $R_v$, and $R_a$’s heading deviation $\theta'$ with $p_g$. Once $R_a$ is detected by $R_v$, $R_a$ stores two groups of data during the next period $\mathcal{T}$. The data groups are defined as

$$\begin{align*}
Q_{\text{in}}(k) &= [\theta'(k), v_1(k), v_2(k), a(k), d_{\text{env}}(k), \phi_r(k)]^T \\
Q_{\text{out}}(k) &= [\Delta s(k), \Delta \theta(k)]^T
\end{align*}$$

(21)

$R_v$ and other obstacles into account. Considering the possibility that $R_a$’s sensor view may be blocked by other obstacles, $R_a$ can be equipped with advanced onboard sensors at an appropriate height, e.g., deploying an omnidirectional camera on the top of $R_a$. Also, $R_a$ can actively move to other positions such that $R_v$ is observed, as in the intentional learning stage. 

C. Model Learning: SVR-Based Method

When $(D, \alpha)$ of $R_a$ is obtained, $R_a$ is controlled to excite $R_v$ by blocking in $(D, \alpha)$. In this step, $R_a$ also records its relative distance $d_{\text{env}}$ and bearing $\phi_r$ with $R_v$, and $R_a$’s heading deviation $\theta'$ with $p_g$. Once $R_a$ is detected by $R_v$, $R_a$ stores two groups of data during the next period $\mathcal{T}$. The data groups are defined as
The discrepancy of regression for different obstacle-avoidance algorithms embodies in the internal structure and parameters of the regression model. With the obstacle-avoidance mechanism learned by $R_a$, it fills the gap between the strong assumption (where the attacker has known the information of the target system from the very beginning) and the real implementation (where the attacker needs to acquire the necessary information first).

V. ATTACK STRATEGY DESIGN AND ANALYSIS

In this section, we propose the shortest-path attack and hands-off attack strategies to achieve the final purpose. First, the feasibility of the attack strategies is analyzed. Then, we present the detailed design of the strategies. Finally, the attack performance bounds and conditions are derived to illustrate the effectiveness of our proposed methods.

A. Attack Idea and Feasibility

In this part, we demonstrate the ideas of the proposed attack strategies and analyze their feasibility.

Since path distance is a commonly-used optimization objective in robot path planning [55], we first propose the shortest-path attack strategy for $R_o$, which aims to drive $R_e$ into the preset trap with minimal path cost. Note that $R_a$ is in a consistently active state during the shortest-path attack process, which is not desirable in some cases. For example, long-time obstacle occurrence nearby may cause an anomaly alarm of $R_a$ and even make $R_e$ alter the control strategy. To alleviate these effects and make $R_a$ stealthier, we further propose hands-off attack strategy. “Hands-off” means the control effort over the plant can be maintained exactly at zero over a time interval and aims to obtain the sparsest control or the minimal active interference with the plant [56], [57]. The feasibility of our proposed strategies lies in two aspects. First, the proposed attack is launched from the physical world by making $R_a$ disguise as an obstacle in front of $R_e$, and the presence of $R_a$ is taken into consideration by $f$, making $R_e$ unable to ignore the influence. Second, the learning scheme proposed in the last section enables $R_a$ to acquire the obstacle-avoidance knowledge of $R_e$, providing reliable support for a more efficient and intelligent strategy design. Specifically, the major issue is to address the coupled dynamics of $R_a$ and $R_e$ and the nonanalytic property brought by the learned $f$.

Based on the learned knowledge about $f$, the obstacle detection area of $R_e$ is formulated as

$$\mathcal{A}_d(p_e) = \{p : \|p - p_v\|_2 < D, \alpha_1 \leq \langle \overrightarrow{pp_v}, \mathcal{T}_m(p_e) \rangle \leq \alpha_2\}$$

(24)

where $\langle \overrightarrow{pp_v}, \mathcal{T}_m(p_e) \rangle$ is the angle between the vector $\overrightarrow{pp_v}$ and the motion orientation $\mathcal{T}_m(p_e)$.

Regardless of the robot type, the motion updates of $R_e$ and $R_a$ are represented as

$$p_e(k + 1) = h_e(p_e(k), u_e(k))$$

(25)

$$p_a(k + 1) = h_a(p_a(k), u_a(k))$$

(26)

where $u_e(k)$ is the velocity control input of $R_e$ and $u_a(k)$ is attack input of $R_a$. With learned $f$, the velocity input and position of $R_e$ are, respectively, predicted by

$$\dot{u}_e(k) = f(p_a(k), p_v(k), u_e(k - 1))$$

(27)

$$\dot{p}_e(k + 1) = h_v(p_v(k), \dot{u}_e(k)).$$

(28)

B. Shortest-Path and Hands-Off Attack Algorithms

First, we introduce the shortest-path attack. Since the obstacle-avoidance process under $f$ is a dynamic tradeoff between avoiding the obstacle and going to goal, there exists a certain moment before which the obstacle-avoidance part plays a dominant role and after which the go-to-goal does. The key point of the shortest-path attack strategy is to impose a consistent impact on the former one as large as possible, such that $R_e$ is generally dominated by $R_a$ into $p_t$.

Denote the initial attack position as $p_{a0}$, and the attack path is the trajectory of $R_a$ that starts from $p_{a0}$ and ends at $p_t$. Then, to obtain the minimal attack path length, the original attack design problem $P_0$ is rewritten as the following problem:

$$P_1 : \min_{H, u_{a0}, H} C_s(u_{a0}, H) = \sum_{k=0}^{H} \|\dot{p}_v(k + 1) - p_v(k)\|_2$$

(29a)

s.t. $\|p_v(k)\|_2 \leq \mu$ (29b)

$\|p_v(H) - p_t\|_2 \leq \delta$ (29c)

$\eta \leq \|p_a(k) - p_v(k)\|_2$ (29d)

$p_a(k) \in \mathcal{A}_d(p_v(k))$ (29e)

(25), (26), (27), and (28)

where $\mu$, $\delta$, and $\eta$ are all positive constants, and the constraints hold for all $k \in \{0, \ldots, H\}$. In (29), the first constraint (29b) demonstrates the boundness of the attack inputs, and the second constraint (29c) guarantees that $R_a$ is close enough to $p_t$ when the attack stops. The $\eta$ in (29d) is designed to keep a safe distance between $R_a$ and $R_e$ to avoid possible collision at next move, and (29e) makes sure that $R_a$ is always in the detection area during the attack. The last four constraints capture the state dynamics of $R_a$ and $R_e$.

Ideally, the shortest attack path $T_a$ is directly obtained by connecting $p_{a0}$ and $p_t$. However, $T_a$ is almost impossible to be the real optimal trajectory because the mechanical structure and control dynamics constraints of $R_e$ will not allow the sudden turn point and the trajectory rigidity in $T_a$. Even so, we are able to take $T_a$ as evaluation criterion: the closer the obtained attack trajectory is with $T_a$, the better the attack strategy is. Then comes an interesting problem: how to choose the best $p_{a0}$ to begin attacking, which we call entry point.

Denote by $T_p$ the trajectory from the initial position $p_v(0)$ to $p_t$ without being attacked. Then, the definition of the entry point is given as follows.

Definition 1: (Entry point) Given a configuration of $p_v(0), p_t$, and $p_t$, rebuild a coordinate frame $X'-Y'$ where the $Y'$ positive
Algorithm 2: Shortest-Path Attack Strategy.

Input: $f$, the iteration limit $k_{\text{max}}$ and tolerant bound $\delta$.
Output: Terminal horizon $H$, and attack input vector $u_{a,0:H}$.

1: Initialize $p_v(0), p_g, p_t$, and set the attack signal $a_s = 0$;
2: Compute entry point $p_e$ by (30);
3: Randomly select a $p_a(0)$ such that $\|p_a(0) - p_e\| \leq r_d$;
4: $R_a$ starts to run towards $p_g, A_a' = \emptyset$;
5: for $k \leftarrow 1$ to $k_{\text{max}}$ do
6: if $a_s = 0$ and $\|p_t - p_a\| \leq r_d$ then
7: $R_a$ moves into the preattack position, $a_s = k$;
8: else
9: $R_a$ stays still;
10: end if
11: if $a_s \geq 1$ then
12: Uniformly sample a subset $A_{a}^{\text{rand}} \subseteq A_{a}(p_v(0))$;
13: for $p_a(k + 1) \in A_{a}^{\text{rand}}$ do
14: Compute $\hat{p}_v(k + 1)$ by (28);
15: $\hat{d}_v(k + 1) = \|p_v(k + 1) - p_t\|_2$;
16: if $\hat{d}_v(k + 1) \leq \delta$ and $p_a(k + 1) \in A_{a}(p_v(k))$ then
17: Update $A_{a}' = A_{a}' \cup \{p_a(k + 1)\}$;
18: end if
19: end for
20: $k' = k - a_s$,
21: $u_a(k') = \arg_{u_a} \min \{\|p_v(k) - p_t\|_2 : p_a(k + 1) \in A_{a}'(p_v(k)) \}$;
22: if $\|p_v(k) - p_t\|_2 < \delta$ then
23: break;
24: end if
25: end for
26: $H = k'$, and construct attack input vector $u_{a,0:H}$.

Note that the complexity of Algorithm 2 is determined by multiple factors, including the moving offset of $R_a$, the distance between $p_e$ and $p_t$, and the granularity of $A_{a}^{\text{rand}}$, making it hard to be explicitly characterized. Specially, if $R_a$ can make $R_a$ directly move toward the trap with constant offset $L$ and by referring to the complexity analysis of gradient-descent algorithm, the complexity of Algorithm 2 can be characterized by $O(1/\varepsilon_L^3)$ (see [58], Ch. 8.3) for more proof details). This complexity can be regarded as the ideal case where the obstacle-avoidance mechanism is perfectly learned. Despite the intractable complexity analysis for general cases, we can characterize the more concerned performance bounds, which will be provided in Section V-D.

Based on the formulation in the shortest-path attack, we further propose the hands-off attack, which leverages inertia nature that the obstacle-avoidance of $R_a$ is not a one-time effort (i.e., it takes certain time to circumvent the obstacle and adjust to its coordinates).
Algorithm 3: Hands-Off Attack Strategy.

Input: $k_{\max}$, $\delta$, $\gamma$ (as in Algorithm 2), and preset constant $\gamma$;
Output: $H$ and $u_{a,0:H}$;
1: do [Lines 1–4] of Algorithm 2;
2: for $k \leftarrow 1$ to $k_{\max}$ do
3: $d_{vt}(k) = \|p(v) - p(t)\|_2$, $k' = k - a_s$;
4: do [Lines 6–10] of Algorithm 2;
5: if $a_s \geq 1$ and $d_{vt}(k) > \gamma d_{te}$ then
6: if $d_{vt}(k) \leq d_{vt}(k - 1)$ and (32b), (32d) hold then
7: $R_a$ stays inactive, i.e., $u_a(k') = 0$;
8: else
9: do [Lines 12–20] of Algorithm 2 and obtain $u_a(k - a_s)$;
10: end if
11: else if $a_s \geq 1$ and $d_{vt}(k) \leq \gamma d_{te}$ then
12: if $d_{T}(T_m(p(v), p_i)) < \delta$ and (32b), (32d) hold then
13: $R_a$ stays inactive, i.e., $u_a(k') = 0$;
14: else
15: do [Lines 12–20] of Algorithm 2 and obtain $u_a(k')$;
16: end if
17: end if
18: do [Lines 22–24] of Algorithm 2;
19: end for
20: $H = k'$, and construct attack input vector $u_{a,0:H}$;

2) When $d_{vt}(k)$ is large after the attack begins, $R_a$ uses the change of $d_{vt}(k)$ to determine whether to stay still or move [Lines 6–10]. If it needs to make an attack, the attack inputs are sampled from the motion space of both $R_a$ and $R_e$ and to explore the best one from the feasible attack set.

3) When $d_{vt}(k)$ becomes small, $R_a$ alters to a new attack decision criterion, by which $R_a$ can stay still if $d_{T}(T_m(p(v), p_i))$ does not grow larger and the constraints are satisfied [Lines 12–16].

4) Finally, the same stopping criterion as that in Algorithm 2 is adopted [Line 18]. Since Algorithm 3 also enjoys the descentwise nature, it shares the same complexity scale as Algorithm 2.

C. Attack Conditions and Algorithm Convergence

Next, we give the conditions to successfully realize the attack by the proposed algorithms and analyze the convergence. To intuitively describe the trap setting, we mainly use geometric expressions combined with graph illustration. Given $p_{0}$ and $p_{g}$, divide the 2-D plane into the $X'$–$Y'$ coordinate frame defined in Definition 1. First, draw the two circumcircles with radius $r_{\min}$, $T_{c1}$ and $T_{c2}$, tangent to $T_g$ at $p_{0}$ respectively. Then, draw the two tangent lines $T_{l1}$ for $T_{c1}$ and $T_{l2}$ for $T_{c2}$, going through $p_{g}$. In the $X'$–$Y'$ coordinate frame, with deterministic $p_{0}$, $p_{g}$, and $r_{\min}$, the expressions of $T_{c1}$, $T_{c2}$, $T_{l1}$, and $T_{l2}$ are explicitly determined and are represented as:

$$T_{c1} = \{ p : \| p - p_{c1} \|_2^2 = \gamma_{\min}^2 = 0 \}$$

$$T_{c2} = \{ p : \| p - p_{c2} \|_2^2 = \gamma_{\min}^2 = 0 \}$$

$$T_{l1} = \{ p = (x, y)^T : y - a_1 x - b_1 = 0 \}$$

$$T_{l2} = \{ p = (x, y)^T : y - a_2 x - b_2 = 0 \}$$

where $p_{c1}$ and $p_{c2}$ are the centers of $T_{c1}$ and $T_{c2},$ and $\{a_1, b_1\}$ and $\{a_2, b_2\}$ are the slope–intercept terms of $T_{l1}$ and $T_{l2},$ satisfying $a_1, a_2 > 0, b_1 > 0,$ and $b_2 < 0.$ Accordingly, we have

$$I_T(T_{c1}, p) = \text{sign} (\| p - p_{c1} \|_2^2 - \gamma_{\min}^2)$$

$$I_T(T_{l1}, p) = \text{sign} (y - a_1 x - b_1)$$

where $i = 1, 2$ and sign is the indication function. Next, we introduce an auxiliary area variable, given by

$$A_{z1} = \{ p : I_T(T_{l1}, p) \leq 0, I_T(T_{l2}, p) \geq 0, x \leq x^g, y \leq y^g \}$$

$$A_{z2} = \{ p : I_T(T_{c1}, p) \geq 0, I_T(T_{c2}, p) \geq 0, I_T(T_{l1}, p) \leq 0, I_T(T_{l2}, p) \geq 0, \| p - p_g \|_2 \leq \| p_{0} - p_g \|_2 \}.$$
Theorem 2 presents the criterion to set \( p_t \). In fact, as Fig. 5 shows, it also demonstrates how the initial configuration of \( R_a \), \( R_y \), and the trap will affect the final attack result. As long as the initial configurations meet the above conditions, the success of Algorithm 2 by utilizing the obstacle-avoidance characteristic of \( R_a \) is guaranteed, which will also be verified with various cases in Section VI and VII. There are two points that need to be noted. First, although the given condition is only sufficient to achieve the driving-to-trap attack, it covers quite a large area for most trap settings. To further relax the condition, the key is to explore more advanced strategy design based on more agile moving ability of \( R_a \). Second, as indicated in Appendix B, the obstacle-avoidance characteristic corresponds to a descent property and Algorithm 2 can be regarded as a descent search algorithm to solve

\[
\min_{p_o} F(p_o) = \|p_t - p_o\|^2
\]  

(42)

which is convex about \( p_o \) and has only one minima. Note that Algorithm 2 shares a similar form to the conventional gradient-descent method and, thus, effectively solves the problem. The main difference in Algorithm 2 is that the search direction is not a fixed form at each iteration [e.g., not necessarily the negative gradient of \( F(p_o) \)]. By evaluating the most appropriate move from the sampled feasible position set, the algorithm effectively avoids the oscillation issues.

As for the hands-off attack, Algorithm 3 also works as a descent search such as Algorithm 2, except that the attack robot is not always active. However, we note that during an inactive period \([k_1, k_2]\) for the attacker, the positions of the victim satisfy \( \|p_o(k_2) - p_t\| < \|p_o(k_1) - p_t\| \), illustrating that the victim still takes a descent path to the trap. Hence, the victim’s movement during this period is equivalent to a special single step, and the convergence of Algorithm 3 is still guaranteed as Algorithm 2.

### D. Attack Performance Analysis

In this part, we analyze the attack performance of Algorithms 2 and 3 from the perspective of the path length and activity period, respectively. First, denote by \( C^*_a \) the optimal cost of \( P_1 \), and we have the following result.

**Theorem 3 (Existence of \( \varepsilon \)-approximated solutions):** The optimal cost can be approximated with arbitrary precision, i.e., \( \forall \varepsilon > 0 \), there exists a \( u_{a;0:H} \) such that \( C_s(u_{a;0:H}) - C^*_a \| \leq \varepsilon \).

**Proof:** The proof is provided in Appendix C.

Theorem 3 demonstrates the existence of \( \varepsilon \)-approximated solutions for the optimal cost of \( P_1 \). To further investigate how good the solution by Algorithm 2 is, we introduce a two-tuple \((\omega, r)\) to illustrate the visual effect that how a nonholonomic robot avoids an obstacle.\(^4\) As shown in Fig. 6(a), supposing the trap is at the left side of \( R_y \), we denote four extreme positions \( \{p_1, p_2, p_3, p_4\} \), where \( R_y \) could make an effective attack, along with corresponding output denoted as \( \{(\omega_i, r_i), i = 1, \ldots, 4\} \). In terms of the reaction radius \( r, p_t \) is the most threatening position for \( R_y \) while \( p_t \) the least threatening, as \( r_1 < r_2 < r_4 < r_3 \). Let \( r_{\max} = \max\{r_i\}, r_{\min} = \min\{r_i\}, d_{te} = \|p_t - p_e\|_2 \), and denote \( l_{\text{path}} \) as the trajectory length of \( R_y \) under attack. Then, we present the following theorem that gives the performance bounds of Algorithm 2.

**Theorem 4 (Performance bounds of Algorithm 2):** Let \( C_s \) be the cost obtained by Algorithm 2. When \( p_e \in A_1 \) and \( d_{te} > 2 \cdot r_{\min} \), we have

\[
(\pi/2 + \xi - \cos \xi)r_{\min} + d_{te}(\cos \xi - 1) \leq C_s - C^*_a \leq \left( \frac{7}{6} - \pi - \sqrt{3} \right) r_{\max}
\]  

(43)

where \( \xi = \arcsin \left( \frac{r_{\min}}{d_{te} - r_{\min}} \right) \).

**Proof:** The proof is provided in Appendix D.

With the feasibility of the attack guaranteed, Theorem 4 indicates in most situations how close the solution obtained by Algorithm 2 is to ideal trajectory, and how the worst solution could be, which is extremely hard to actually meet.

Concerning the hands-off attack, the existence of the optimal solution of \( P_2 \) is similar to \( P_1 \) and guaranteed. The proof resembles that of Theorem 3 and is omitted here. Then, the attack performance of Algorithm 3 is presented in the following theorem.

**Theorem 5 (Performance bounds of Algorithm 3):** Let \( H_1 \) and \( H_2 \) denote the attack horizon obtained by Algorithm 2 and 3, respectively, then we have

\[
H_2 \leq 2H_1
\]  

(44)

and the active attack period by Algorithm 3 satisfies

\[
C_h(u_{a;0:H})/H_2 \leq 0.5.
\]  

(45)

**Proof:** The proof is provided in Appendix E.

Theorem 5 gives the attack performance bounds of Algorithm 3, in terms of the attack path length (compared with Algorithm 2) and the attack activity ratio. In Section VI, we will illustrate that the worst performance is hard to actually meet. Note that the core insight of Algorithm 3 lies in how to leverage the inertia property of \( R_a \)’s obstacle-avoidance behavior to make \( R_y \) more inactive, not in how to design the best attack move for \( R_y \) when it needs be active. Therefore, it can also be compared with other

---

\(^4\)For a nonholonomic robot, the output \((v, \omega)\) under \( f \) also determines the curvature radius \( r \) as \( v = \omega r \), and \((\omega, r)\) is more intuitive to describe the instantaneous obstacle-avoidance behavior.

\(^5\)There is no need to consider \( p_e \in A_1 \) such that \( dr(T_0, p_t) \leq (\sqrt{5} + 1)r_{\min} \) in Algorithm 2, as the attack can be easily achieved from \( p_e \).

---

**Fig. 5.** Illustration of the condition of successful attack.
attack algorithms by replacing the [Line 9, 15] of Algorithm 3 with the attack move computed by other algorithms, and the similar conclusions to Theorem 5 still hold.

E. Attack Against Holonomic Robots

For holonomic robots, the attack performance (results of Theorems 4 and 5) for $R_a$ is the same under Algorithm 2 and 3. The major difference is that the motions of holonomic robots in two orthogonal directions are independent and involve no coupling constraints, and there is no need to compute their orientations, making learning the obstacle-avoidance mechanism easier for $R_a$. Therefore, attacks against holonomic robots with omnidirectional moving ability are not subject to the limitations of Theorem 3, i.e., the preset trap is arbitrary. However, due to the same reason, a stronger moving ability for $R_a$ is required to move to its optimal attack position.

VI. SIMULATIONS

In this section, we model both nonholonomic and holonomic robots as simulation objects. Specifically, we adopt the popular DWA [59], [60] and APM [32] as the obstacle-avoidance mechanism of the two kinds of robots, respectively. Our evaluation focuses on two aspects: effectiveness and efficiency. For better illustrating and depicting the attack process as detailed as possible in a square figure, we set the initial positions of $R_a$, $R_v$, the trap, and the goal roughly on the four corners (other settings that meet the attack conditions exhibit similar results). First, the critical learning steps of the proposed attack framework are shown. Then, we illustrate the effectiveness of the proposed control designs by presenting different attack scenarios and evaluating the attack costs. Finally, detailed comparisons with other works are provided.

A. Learning Phase

Fig. 7(a) shows that $R_a$ collects a series of $Q_{\text{in}}$ and $Q_{\text{out}}$ by sequential “intentional learning.” It is feasible to regress the obstacle-avoidance algorithm using the data when no prior information of that is available. Fig. 7(b) shows a simple and rough attack: once $R_a$ appears in $(D, \alpha)$, it makes continuous impacts on $R_v$ in one direction (here is to the right). As expected, $R_v$ should keep avoiding $R_a$ all the time. However, this attack cannot proceed consistently. We conclude that the leading cause lies in two parts: 1) the control inputs are not well designed by considering where the next (sub-)optimal attack position is; 2) each prediction will produce certain inaccuracy, which is cumulated to influence the attack effect in this case. Next, we present the results of the proposed attack algorithms. By comparison, the effectiveness of the algorithms is exhibited, remedying the deficiency of the former one.

B. Attack Phase

Basic setup: The goal position of $R_v$ is set as (12,12). For a fair comparison, the same two cases for all attack strategies are designed where the trap is set as (5,12) and $(-10,15)$, respectively. The traps locates in area $A_1$ (defined in Fig. 4), satisfying the conditions in Theorem 3. Fig. 8 shows that the shortest-path attack strategy against a nonholonomic robot is applied. Fig. 9 shows that the hands-off attack strategy against a nonholonomic robot is applied. Examples of the holonomic robot are presented in Fig. 10.

Results analysis: In terms of path cost, from Fig. 8, we can observe that the trajectory of $R_v$ is near to $T_1$ by the shortest-path attack, and the trajectory of $R_v$ by hands-off attack twists and turns a little more, as shown in Fig. 9. Therefore, the former attack strategy is better in the sense of attack path length. For example, it is 5.52 m in Fig. 8(a), while 6.56 m in Fig. 9(a),
Fig. 8. Shortest-path attack against nonholonomic robot. The obstacle-avoidance algorithm of $R_v$ is DWA. (a) Preset trap locates in $(5,12)$. Attack iteration time is 89 steps, and the path length after being attacked is 5.52 m. (b) Preset trap locates in $(-10,15)$. Attack iteration time is 296 steps, and the path length after being attacked is 18.35 m.

Fig. 9. Illustration of the hands-off attack against nonholonomic robot. The obstacle-avoidance algorithm of $R_v$ is DWA. In hands-off control input, the state 1 means $R_a$ is active while state 0 means $R_a$ stays still. The marked parts with green rectangles mean the attack move is intensive. (a) and (b) Preset trap locates in $(5,12)$. Attack horizon is 169 steps, $R_a$ is active only for 20 steps, and the path length after being attacked is 6.56 m. (c) and (d) Preset trap locates in $(-10,15)$. Attack horizon is 509 steps, $R_a$ is active only for 78 steps, and the path length after being attacked is 20.43 m.

consistent with the conclusion in Theorem 5. However, if we focus on the active attack time, it is found that the small sacrifice of path length largely saves the attack times for $R_a$ by hands-off attack. As shown in Fig. 9(b) and (d), $R_a$ only needs to attack in part times of the whole process to implement the attack successfully. The marked parts with green rectangle show this part of trajectory twists and turns, meaning the attack moves in this period are intensive. The reason lies in that $R_a$ needs to make sure that $R_v$ moves toward the trap, not adjusting to the goal direction in time [Lines 6–21, Algorithm 2]. Taking the case of trap $(-10,15)$ as an example, by hands-off attack, the attack horizon is 509 steps and $R_a$ is active only for 78 steps, and the path length after being attacked is 20.43 m.

For a holonomic robot $R_v$, note that its trajectory is not as smooth as that of a nonholonomic robot, i.e., the moving direction of $R_v$ may suddenly change. Even though this property makes the movement of $R_v$ more unconstrained, the response trajectory of $R_v$ is not necessarily shorter than that of a nonholonomic robot, as we can see from Figs. 8(b) and 10(b).

C. Comparisons With Other Works

The proposed attack shares some similar features with conventional pursuit-evasion problems (e.g., [7], [8], [9]) and herding problems (e.g., [10], [11], [12]), as they all utilize the interaction characteristic of the robots to achieve the specific tasks. However, it would be inappropriate to directly compare the method performance with these works due to the major difference in their basic model formulation and application scenarios. For intuitive illustrations, the comparisons with some representative works are provided in Table II, which can be interpreted from the following three aspects.

1) Task scenario: The proposed attack aims to drive the target robot into a preset trap position by utilizing the obstacle-avoidance mechanism. While in pursuit-evasion problems, the aim of the pursuers is to capture the evaders, which is usually turned to closely track the evaders or form an encirclement over the evader.

2) Model knowledge: Our work focuses on learning the knowledge about the interaction mechanism of mobile robots and then utilizing the learned model to design a driving-to-trap attack, while most existing works rarely discuss the former issue about model knowledge, e.g., many herding problem works assume the interaction model of the robot is priorly known.

3) Model types: The proposed attack is not tailored for a specific dynamic model of the victim robot. The dynamics of the robots in the pursuit-evasion is generally modeled as a multiagent LTI systems, and in herding problems, it is usually characterized by specific function forms (e.g.,
TABLE II
COMPARISONS WITH REPRESENTATIVE WORKS OF HERDING PROBLEMS

| Works       | Dynamic model of target robot\(^1\) | Self-goal consideration\(^2\) | Model is not priorly known\(^3\) | Make target robot go preset position | Required number of active robots |
|-------------|-------------------------------------|------------------------------|----------------------------------|-------------------------------------|----------------------------------|
| [61]        | potential function based            |                              | √                               | multiple                            |                                  |
| [11]        | potential function based            |                              | √                               | multiple                            |                                  |
| [12]        | nonlinear Lipschitz function        |                              | √                               | single                              |                                  |
| [62]        | flocking model                      |                              | not fixed                       | single                              |                                  |
| [63]        | flocking model                      |                              | √                               | multiple                            |                                  |
| Our work    | majority of obstacle-avoidance algorithms |                     | √                               | √                                   | single                           |

---

\(^1\) For comparisons in a uniform framework, the herd robot in herding problem and the victim robot in our work are both referred as the target robot, the herder robot in herding problem and the attack robot in our work are both referred as the active robot.

\(^2\) This factor indicates that whether the dynamic model of target robot contains an intrinsic goal position, i.e., the dynamics of the target robot is not only determined by the active robot, but also by its intrinsic goal position.

\(^3\) This means whether the active robot knows the dynamic model of the target robot to design control strategies.

potential function), and only depends on the positions of the herding and target robots.

VII. EXPERIMENTS

A. Platform Description and Experimental Setup

To demonstrate the practical performance of the proposed attack, multiple experiments were conducted in our self-designed mobile robot platform [64]. The AprilTag visual system is adopted for the real-time localization of the robots. The control procedures based on the localization results are implemented by MATLAB in a VMWare ESXI virtual machine, which is equipped with an Intel(R) Xeon(R) Gold 5220R CPU, 2.20 GHz processor and 16 GB RAM.

We validate the result on a 5 m × 3 m rectangular space in practice. Fig. 11 shows the used robots, which are with 17.5 cm × 17.5 cm × 20 cm volumes and designed in omnidirectional form. The robots are subject to the motion model (8), and their real-time positions are detected by the cameras deployed to the lab ceiling. The positions of the robot, goal, trap, and other obstacles are represented by the coordinates of their geometric central points. Note that the unit of all motion states (e.g., position and speed) is described by the picture pixel hereafter, where 1 pixel in the calibrated picture is about 1.08 mm in the practical testbed.

For simplicity and intuitive comparison, the same initial position settings of the victim robot \(R_v\), the attacker \(R_a\), the goal \(p_g\), and the preset trap \(p_t\) are used in different types of experiments. The initial positions of the victim robot and the attacker are \(p_v(0) = [1800, 600]\) and \(p_a(0) = [3800, 600]\), respectively. The goal and trap positions are \(p_g = [3800, 2400]\) and \(p_t = [1900, 2200]\), respectively. According to Definition 1, the entry point for the attacker is \(p_e = [2670, 1373]\). The maximum velocity of the attacker and the victim robot are 400 pixel/s and 200 pixel/s. The obstacle-detection radius is set as \(D = 600\) pixel. The robot is assumed to arrive at the goal or trap position if \(\|p_v - p_g\|_2 < 200\) or \(\|p_v - p_t\|_2 < 200\).

B. Experimental Results

The first presented experiment is an illustration of the intentional learning stage, as shown in Fig. 12. In this experiment, the attacker is assigned three observation points \([2600, 1000]\), \([2900, 1500]\), and \([2900, 1500]\). Fig. 12(a)–(c) depicts the obstacle-avoidance process of the victim robot when the attacker approaches it as an obstacle. The avoidance trajectory of the victim robot is observed by the attacker and is used to construct the input–output pairs in the holonomic form of (21). Fig. 12(d) shows the complete trajectory evolution during this process. By conducting such intentional learning procedures in different positions, we obtain 200 groups of the input–output pairs to infer the goal position by the least squares method and learn the obstacle-avoidance mechanism by the SVR method. The inferred obstacle-detection radius and the goal position are \(\hat{D} = 678\) pixel \((\|\hat{D} - D\| = 78)\) and \(\hat{p}_g = [3742, 2355]\) \((\|\hat{p}_g - p_g\|_2 = 73)\), while the learned model achieves mean directional accuracy of 0.91 and root-mean-square error of 160.1 pixels, which are sufficient to support the following attack implementation.

Next, we focus on the validation of the proposed shortest-path and hands-off attack strategies based on the learned model.
better illustrate the performance and avoid repeating similar tests multiple times, we directly provide the results considering there are also other obstacles in the environment. Note that we use periodic displacements in a detection period of the cameras to describe the speed of the robots.\footnote{Different from the simulation examples, the control period of the practical robots is not strictly a constant but varies in a small range around the nominal value due to different computation burden and system noises. Therefore, we use the periodic displacements of the robots in two consecutive snapshots to plot the speed curve.}

Fig. 13 presents the results of the shortest-path attack, where the coordinates of the obstacles are [2400,1900] and [2600,1700], respectively. The attack starts at step 490 with a total attack horizon of 1164 steps. During the attack process, the attacker is only active for 507 steps in total, thus the active ratio of this example is 0.4356, which corresponds to the conclusion in Theorem 5. All the experiments demonstrate the effectiveness of the proposed attack. More experiment tests can be found in the attached video (see Supplementary Material).

VIII. CONCLUSION

In this article, the security problem of mobile robots is investigated. Different from most existing works, we propose a learning-based intelligent attack framework, where the attacker is able to learn certain dynamics of the target based on external observations, and then utilize it to accomplish its attack purpose in a more feasible and stealthier way. Specifically, without
any prior information, the attacker is able to infer the victim robot’s goal position and obstacle-detection area first. Then, the obstacle-avoidance mechanism is regressed by exploring the intrinsic characteristic of the mechanism and dynamically collecting reliable data in trials. Furthermore, two kinds of efficient control strategies for short path and high concealment attack objectives are designed, respectively. The feasibility analysis of the attacks and their performance bounds are provided, and the conditions for a successful attack implementation are proved. Extensive simulations and real-life experiments confirm the effectiveness of the proposed intelligent attack framework.

There are still many open issues that require further investigations. For example, we mainly focus on attack design in this article, without more detailed consideration of defense for the victim robot. How to constitute an integrated framework containing both attack and defense needs further investigation. Meanwhile, the scenario considered in this article is relatively simple, and applying the idea to more complicated scenarios is quite meaningful, such as multirobot coordination tasks. More generally, questions like what other dynamics information can be learned from external observation and what sophisticated attacks can be designed are also worth further exploration.

**APPENDIX**

**A. Proof of Theorem 1**

**Proof:** First, we prove that the trajectory can be approximated accurately in the asymptotic sense. Recalling \( y_{ij} = P_1^T \tilde{x}_{ij} + \eta_{ij} \) and the trajectory \( T_r^{(l)} \) is a straight line, thus the coefficient vector \( P_1 \) is two-dimensional. For simple expressions, the variables are organized as \( X_I = [\tilde{x}_{i1}, \tilde{x}_{i2}, \ldots, \tilde{x}_{inh}]^T \), \( Y_I = [y_{i1}, y_{i2}, \ldots, y_{inh}]^T \) and \( \Gamma_I = [\eta_{i1}, \eta_{i2}, \ldots, \eta_{inh}]^T \). Then, the problem is transformed to solve

\[
\min_{P_1} \|Y_I - \tilde{X}_I P_1\|_2^2
\]  

(46)

where its least square solution and error vector are given by \( \tilde{P}_I = (\tilde{X}_I^T \tilde{X}_I)^{-1} \tilde{X}_I^T Y_I \) \( \hat{E}_I = \tilde{P}_I - P_1 = (\tilde{X}_I^T \tilde{X}_I)^{-1} \tilde{X}_I^T \Gamma_I \). \( \tilde{P}_I \) denotes the trajectory is limited and each element in \( X_I \) is bounded, and one has \( \|\tilde{X}_I^T \tilde{X}_I\|_2 = O(n_t) \) and \( \|\tilde{X}_I^T \Gamma_I\|_2 = O(1/n_t) \). Then, denote the element with largest absolute value in \( X_I \) as \( \tilde{x}_{\text{max}} \) and one has

\[
E \left\{ \frac{[X_I^T \tilde{\Gamma}_I]_{ij}}{n_t} \right\} = \frac{1}{n_t} \sum_{n_t=1}^{n_t} \frac{1}{n_t} \mathbb{E} \left\{ \frac{\eta_{n_t}}{n_t} \right\} = 0
\]

(47)

\[
\mathbb{E} \left\{ \frac{[X_I^T \tilde{\Gamma}_I]_{ij}}{n_t} \right\} = \sum_{n_t=1}^{n_t} \left( \frac{[X_I^T \tilde{\Gamma}_I]_{ij}}{n_t} \right) \sigma^2 \leq x_{\max}^2 \sigma^2
\]

(48)

By the famous Chebyshev inequality, given arbitrary \( \epsilon > 0 \), one has

\[
\text{Pr} \left\{ \left| \frac{[X_I^T \tilde{\Gamma}_I]_{ij}}{n_t} \right| < \epsilon \right\} \geq 1 - \frac{\mathbb{E} \left\{ [X_I^T \tilde{\Gamma}_I]_{ij} \right\}}{\epsilon^2} \geq 1 - \frac{x_{\max}^2 \sigma^2}{n_t \epsilon^2}
\]

(49)

\[\hat{E}_I = \hat{P}_I - P_1 = (\tilde{X}_I^T \tilde{X}_I)^{-1} \tilde{X}_I^T \Gamma_I \]

(50)

Let \( \epsilon = (n_t)^{\frac{1}{2}} \) and substitute it into (50), one infers that

\[
\lim_{n_t \to \infty} \text{Pr} \{ \|\hat{E}_I\|_2 = 0 \} = 1.
\]

(51)

Since \( p_g \) locates on the noise-free \( T_r^{(l)} \), it follows from (51) that \( \lim_{n_t \to \infty} d_T(T_r^{(l)}), p_g) = 0 \), which proves the first statement.

Next, consider obtaining the goal \( p_g \) from the estimated trajectories \( \{T_r^{(l)}, l = 1, 2, \ldots, n_g\} \). Let \( \tilde{P}_I = [\tilde{a}_I, \tilde{b}_I]^T \), \( \tilde{A} = [\tilde{A}_0, -\tilde{I}_{n_g}] \) (\( \tilde{I}_{n_g} \) represents all-one vector), \( \tilde{A}_0 = [\tilde{a}_I, \ldots, \tilde{b}_I]^T \) and \( \tilde{B} = [\tilde{b}_1, \ldots, \tilde{b}_n]^T \). Then, \( p_g \) can be obtained by solving

\[
\min_{p_g} \|\hat{A} p_g - \tilde{B}\|_2^2.
\]

(52)

Like (46), the least square solution of (52) is given by \( \hat{p}_g = (\tilde{A}^T \tilde{A})^{-1} \tilde{A}^T \tilde{B} \). \( \hat{p}_g \) is a straight line, thus the trajectory is limited and each element in \( X_I \) are not at the same side of \( p_{\text{max}} \), it follows from (51) that \( \lim_{n_t \to \infty} \|\hat{E}_I\|_2 = 0 \), which proves the first statement.

**B. Proof of Theorem 2**

**Proof:** First, we prove that \( p_t \notin A_{21} \setminus A_{22} \) is a necessary condition by limit performance analysis and logical deduction. The nature of the proposed attack to generate feasible \( p_a \) whose impact could offset the impact of \( p_g \) while lead \( R_c \) to \( p_a \). In the proposed algorithms, this is done by keeping \( p_t \) and \( \{p_a, p_g\} \) stay in different sides of \( T_m(p_v) \) after the attack begins, i.e.,

\[
\begin{align*}
I_T(T_m(p_v), p_a) \cdot I_T(T_m(p_v), p_g) &> 0 \\
I_T(T_m(p_v), p_a) \cdot I_T(T_m(p_v), p_g) \cdot I_T(T_m(p_v), p_t) &< 0
\end{align*}
\]

(53)

Recall that the error vector \( \hat{A} p_g - \tilde{B} \) is also Gaussian according to (48). Therefore, the estimation error \( \|\hat{p}_g - p_g\|_2 \) will goes to zero when \( Z \to \infty \). The analysis is the same as that of \( \hat{P}_I \) and the details are omitted here.

In the extreme case where \( R_c \) circumvents the obstacle with minimal curvature radius \( r_{\min} \), if \( p_t \notin A_{22} \), then there exists a certain moment \( k \) when \( T_m(p_v) \) is parallel with \( T_0 \) (or \( T_2 \)). After \( k \), \( p_a \) and \( p_g \) are not at the same side of \( T_m(p_v) \), i.e., (54) does not hold, indicating that the impact of \( p_g \) to offset the impact of \( p_g \) reversely change and the movement of \( R_c \) is determined by \( p_g \). Then, it follows that \( R_c \) cannot impose desired impact on \( R_c \), and the proposed algorithms cannot work to achieve the attack. On the contrary, for \( p_t \notin A_{21} \), an attack trajectory, which satisfies (54) while does not meet the extreme obstacle-avoidance behavior indicated by \( r_{\min} \), can always be found by Algorithm 2 and Algorithm 3.

Next, we prove the convergence of Algorithm 2 from the optimization perspective when. Note that Algorithm 2 can be seen as a kind of one-dimensional search method, and iteratively solves the following problem:

\[
\mathbf{P}_3 : \min_{p_v} F(p_v) = \|p_t - p_v\|_2^2
\]

(55)

Apparently, the global optimal solution of \( \mathbf{P}_3 \) is \( p_v = p_t \) and this solution is unique. Therefore, we need to prove that the sequence \( \{p_v(k)\} \) produced by Algorithm 2 will converge to \( p_t \) as \( k \) goes to infinity.
Note that the iterative searching process is of the form
\[ p_v(k + 1) = p_v(k) + \lambda_k S_k \] (56)
where vector \( S_k \) is called the search direction, and \( \lambda_k \geq 0 \) is called the step size. Similar to classic gradient-descent method, Algorithm 2 is essentially updating the search direction and step size. The feasible choice of the search direction is determined by the movement of \( R_\alpha \). For simple expression, we denote the search procedures of Algorithm 2 (i.e., Lines 7–15) by a mapping \( \phi: \mathbb{R}^2 \rightarrow \mathbb{R} \). Specifically, \( \phi \) is equivalent to the following composite mapping:
\[
\phi = \arg \min_{p_a} \{ \phi_0(S_k) : S_k = \hat{p}_v(p_a) - p_v(k), p_a \in A_\nu^f \}
\] (57)
where the inner mapping \( \phi_0 \) represents the step of updating \( \lambda_k \) and is given by
\[
\phi_0 = \min \{ h(p_v(k) + \lambda S_k) : \lambda \geq 0, \lambda \|S_k\|_2 \leq \mu \}. \] (58)

With the feasible attack set \( A_\nu^f \) determined, \( S_k = \hat{p}_v(p_a) - p_v(k) \) is a descent search direction of \( F(p_v) \), i.e.,
\[
S_k^T \nabla F(p_v(k)) < 0.
\] (59)
Then, one infers that \( d_v(k + 1) \leq d_v(k) \) and
\[
F(p_v(k + 1)) \leq F(p_v(k)).
\] (60)
Combining the differentiability and convexity of \( F(p_v) \), it follows that \( \phi \) is a closed mapping when \( p_v \neq p_t \), and \( F(p_v) \) is a descent function about \( p_t \) and \( \phi \). By the convergence theorem of search algorithms (refer to [58], Ch. 7), the two conditions ensure the sequence \( \{v(k)\} \) produced by \( \phi \) will converge to \( p_v \).

### C. Proof of Theorem 3

**Proof:** In essence, the motion control of a mobile robot is in discrete form, thus we consider formulating the problem using breadth traversal analysis. All the solutions of the problem are represented by a tree, where the initial attack position \( p_a(\alpha + 1) \) is the root node and a node in floor \( k' \) denotes the attack inputs at attack iteration \( k' \).

First, we prove the existence of a feasible move in each iteration. A feasible move means \( R_\alpha \) will move closer to \( p_t \), which can be achieved by making \( R_\alpha \)'s position satisfy
\[
\begin{align*}
p_a & \in A_\nu(p_v) \\
I_T(T_m(p_v), p_a) \cdot I_T(T_m(p_v), p_t) & < 0.
\end{align*}
\] (61)

Apparently, there are multiple choices of \( p_a \) in each iteration such that (61) holds, and the next step is to determine the best \( p_a \) to implement the attack.

Based on (29b), multiple attack inputs from \( \{ u_a : \|u_a(k')\|_2 \leq \mu \} \) at attack step \( k' \) are sampled. Suppose the number of sampling groups is \( n \), for every two adjacent node \( u_{a_k}(k') \) and \( u_{a_k-1}(k') \), the deviation \( \|u_{a_k}(k') - u_{a_k-1}(k')\|_2 \) is the same. Then, we define the subnode set of \( u_a(k') \) as \( S_n(u_a(k')) \), whose cardinal number is \( n \). Due to the constraint (29d), \( R_\alpha \) cannot stay unmoved all the time. Besides, at iteration \( k' \), \( u_a(k' + 1) \) is computed such that the distance \( d(k + 1) = \|p_v(k + 1) - p_t\|_2 \leq d(k) = \|p_v(k) - p_t\|_2 \). As a consequence of the two factors, it is deduced that the length \( H \) of \( u_a(0:H) \) is finite. Denote the maximum attack depth as \( H \) and construct a solution as
\[
\bar{u}_{a,0:H} = \{ \bar{u}_a(1), \bar{u}_a(2), \ldots, \bar{u}_a(H) \}
\] (62)
where \( \bar{u}_a(k' + 1) \in S_n(u_a(k')) \), \( k' = 1, 2, \ldots, H - 1 \). By the constraints (29d) and (29e), many subnodes of a node \( u_a(k') \) are excluded in Algorithm 2. Therefore, the total number of all feasible solutions \( n_f \ll n^H \).

Given a sampling size \( n \), let \( \bar{u}_{a,0:n}^* \) be the best solution among \( n_f \) feasible solutions, and it is intuitive to have
\[
C_\nu(\bar{u}_{a,0:n}^*) \leq C_\nu(\bar{u}_{a,0:H})
\] (63)
which is monotonically decreasing toward \( C_\nu^* \) as \( n \) growing. Therefore, when \( n \) is large enough, there exists a \( \bar{u}_{a,0:n}^{*,n+1} \) (not necessarily unique) such that \( |C_\nu(\bar{u}_{a,0:H}^n) - C_\nu| \leq \varepsilon \). ■

### D. Proof of Theorem 4

**Proof:** We prove this result by two steps. First, supposing the reaction radius \( r \) of \( R_\alpha \) keeps unchanged under an attack pattern of \( R_\alpha \), we establish the bounds of \( l_{\text{path}} \) in the situation. Then, we scale the value of \( r \) to obtain the final bounds.

**Step 1: Establish the bounds with unchanged \( r \).**

First, consider that \( R_\alpha \) attacks \( R_\nu \) by being at the same relative position with it. This process continues until the orientation of \( R_\alpha \) is heading toward \( p_t \), the path length of this process is denoted as \( l_{\text{path1}} \). Then \( R_\alpha \) moves along with \( R_\nu \) such that \( R_\nu \) goes straightforward to \( p_t \). The path length is of this process is \( l_{\text{path2}} \). The pattern is represented by Pattern 1, as shown in Fig. 6(b), and the angle between the second line and \( T_a \) is
\[
\xi = \arcsin \left( \frac{r}{d_{\text{ct}} - r} \right)
\] (64)
By the calculation formula for arc length, we obtain
\[
l_{\text{pathA1}} = (\pi/2 + \xi) r, \quad l_{\text{pathA2}} = (d_{\text{ct}} - r) \cos \xi.
\] Then we have
\[
l_{\text{pathA}} = (\pi/2 + \xi - \cos \xi) r + d_{\text{ct}} \cos \xi.
\] (65)
Next, consider another attack pattern such that \( R_\nu \) moves two circular arcs, and the end is tangent to the line \( T_a \). The pattern is represented by Pattern 2, as shown in Fig. 6(c). By the geometrical representation, the length of the arc part \( R_\nu \) passed is computed as \( \frac{\xi}{3} \pi r \), and the length of the straight segment part is \( d_{\text{ct}} - (1 + \sqrt{3}) r \). Then, we obtain
\[
l_{\text{pathB}} = \left( \frac{7}{6} \pi - 1 - \sqrt{3} \right) r + d_{\text{ct}}.
\] (66)
Note that \( d_{\text{ct}} > (\sqrt{3} + 1) r \) and \( 0 < \xi < \arcsin \sqrt{3} \pi \), then
\[
l_{\text{pathB}} - l_{\text{pathA}} = \left( \frac{2\pi}{3} - 1 - \sqrt{3} - \xi + \cos \xi \right) r + d_{\text{ct}}(1 - \cos \xi) \geq \left( \frac{2\pi}{3} - \xi - \sqrt{3} \cos \xi \right) r > 0.16 r > 0.
\] (67)
Equation (67) implicates $l_{\text{path}B} > l_{\text{path}A}$ always holds. In fact, if the reaction radius of $R_s$ is unchanged, the length of $R_s$’s path is bounded by $l_{\text{path}A}$ and $l_{\text{path}B}$, i.e.,

$$
(\pi/2 + \xi - \cos \xi)r + d_{te}\cos \xi \leq l_{\text{path}} \leq \left(\frac{7}{\sqrt{3}}\pi - 1 - \sqrt{3}\right)r + d_{te}.
$$

(68)

Step 2: Scale $r$.

Given $r \in [r_{\text{min}}, r_{\text{max}}]$, the path length is bounded by (68).

Since the path length is monotonically increasing with $r$ increasing, the minimum path length is obtained when $r = r_{\text{min}}$ and the trajectory is Pattern 1. The maximum path length is obtained when $r = r_{\text{max}}$ and the trajectory is Pattern 2. By the design of Algorithm 2, even though the reaction radius of $R_s$ is time-varying, it is also bounded in $[r_{\text{min}}, r_{\text{max}}]$. Then, it is induced that the real path length is between the minimum and the maximum path length. Due to $C_2^* = d_{te}$, we have $(\pi/2 + \xi - \cos \xi)r_{\text{min}} + d_{te} \leq C_2^* \leq \left(\frac{7}{\sqrt{3}}\pi - 1 - \sqrt{3}\right)r_{\text{max}}$. 

E. Proof of Theorem 5

Proof: The performance bounds of Algorithm 3 can be obtained by resorting to analyzing a relaxed version of Algorithm 2, where the $R_s$ does not have to be in the obstacle detection area all the time. Once the attack begins, $R_s$ does the same procedure of [Lines 12–20, Algorithm 2] to move, and the next iteration $R_s$ stays inactive. By alternately running the two operations (equivalent to one attack move in every two iterations), the attack can still be achieved with a longer attack horizon. Note that at every inactive state, $R_s$’s impact on $R_e$ does not persist but only work a little weaker than continuing its attack, i.e., $R_e$ will still move toward $p_t$. Therefore, the attack horizon $H \leq 2H_1$ in Algorithm 3, with the active attack period $\|u_{a,0:H}\|_0 = 0.5H$.

Next, by further relaxing the attack mode such that $R_e$ always stays inactive if $R_s$’s impact still works while its movement limit is not violated, then we have $\|u_{a,0:H}\|_0 < 0.5H$.

ACKNOWLEDGMENT

The authors would like to thank X. Ding and Z. Wang for their help in the attack experiments.

REFERENCES

[1] H. Choi et al., “Detecting attacks against robotic vehicles: A control invariant approach,” in Proc. ACM SIGSAC Conf. Comput. Commun. Secur., 2018, pp. 801–816.

[2] D. Ding, Q.-L. Han, Y. Xiang, X. Ge, and X.-M. Zhang, “A survey on security control and attack detection for industrial cyber-physical systems,” Neurocomputing, vol. 275, pp. 1674–1683, 2018, doi: https://doi.org/10.1016/j.neucom.2017.10.009.

[3] H. S. Sánchez, D. Rotondo, T. Escobet, Y. Puig, and J. Quevedo, “Bibliographical review on cyber attacks from a control oriented perspective,” Annu. Rev. Control, vol. 48, pp. 103–128, 2019, doi: https://doi.org/10.1016/j.arcontrol.2019.08.002.

[4] A. Pandey, S. Pandey, and D. Parhi, “Mobile robot navigation and obstacle avoidance techniques: A review,” Int. J. Robot. Automat. Ind., vol. 2, no. 3, pp. 96–105, 2017.

[5] F. Pasqualetti, A. Bicchi, and F. Bullo, “Consensus computation in unreliable networks: A system theoretic approach.” IEEE Trans. Autom. Control, vol. 57, no. 1, pp. 90–104, Jan. 2012.

[6] F. Pasqualetti, F. Dörfler, and F. Bullo, “Attack detection and identification in cyber-physical systems,” IEEE Trans. Autom. Control, vol. 58, no. 11, pp. 2715–2729, Nov. 2013.

[7] R. Vidal, O. Shakernia, H. J. Kim, D. H. Shim, and S. Sastry, “Probabilistic pursuit-evasion games: Theory, implementation, and experimental evaluation,” IEEE Trans. Robot. Automat., vol. 18, no. 5, pp. 662–669, Oct. 2002.

[8] A. Kolling and S. Carpin, “Pursuit-evasion on trees by robot teams,” IEEE Trans. Robot., vol. 26, no. 1, pp. 32–47, Feb. 2010.

[9] V. G. Lopez, F. L. Lewis, Y. Wan, E. N. Sanchez, and L. Fan, “Solutions for multiagent pursuit-evasion games on communication graphs: Finite-time capture and asymptotic behaviors,” IEEE Trans. Autom. Control, vol. 65, no. 5, pp. 1911–1923, May 2020.

[10] W. Lee and D. Kim, “Autonomous shepherding behaviors of multiple target steering robots,” Sensors, vol. 17, no. 12, 2017, Art. no. 2729.

[11] A. Varava, K. Hang, D. Kragic, and F. T. Pokorny, “Herding by caging: A topological approach towards guiding moving agents via mobile robots,” in Proc. Robot. Sci., 2017.

[12] R. A. Licitra, Z. I. Bell, and W. E. Dixon, “Single-agent indirect herding of multiple targets with uncertain dynamics,” IEEE Trans. Robot., vol. 35, no. 4, pp. 847–860, Aug. 2019.

[13] A. Ratnouo and T. Shima, “Line-of-sight interceptor guidance for defending an aircraft,” J. Guid., Control, Dyn., vol. 34, no. 2, pp. 522–532, 2011.

[14] M. S. Darap, A. Redder, and D. E. Quevedo, “Encrypted cooperative control based on structured feedback,” IEEE Control Syst. Lett., vol. 3, no. 1, pp. 37–42, Jan. 2019.

[15] A. B. Alexandru and G. J. Pappas, “Encrypted LQG using labeled homomorphic encryption,” in Proc. 10th ACM/IEEE Int. Conf. Cyber-Phys. Syst., 2019, pp. 129–140.

[16] Y. Li, J. He, C. Chen, and X. Guan, “Learning-based intelligent attack against formation control with obstacle-avoidance,” in Proc. Amer. Control Conf., 2019, pp. 2690–2695.

[17] Q. Zhu and T. Basar, “Game-theoretic methods for robustness, security, and resilience of cyberphysical control systems: Games-in-games principle for optimal cross-layer resilient control systems,” IEEE Control Syst. Mag., vol. 35, no. 1, pp. 46–65, Feb. 2015.

[18] Z. Feng and G. Hu, “Secure cooperative event-triggered control of linear multiagent systems under DoS attacks,” IEEE Trans. Control Syst. Technol., vol. 28, no. 3, pp. 741–752, May 2020.

[19] R. Su, “Supervisor synthesis to thwart cyber attack with bounded sensor reading alterations,” Automatica, vol. 94, pp. 35–44, 2018.

[20] Y. H. Chang, Q. Hu, and C. J. Tomlin, “Secure estimation based Kalman filter for cyber-physical systems against sensor attacks,” Automatica, vol. 95, pp. 399–412, 2018, doi: 10.1016/j.automatica.2018.06.010.

[21] Z. Liu, W. Chen, H. Wang, Y.-H. Liu, Y. Shen, and X. Fu, “A self-repairing algorithm with optimal repair path for maintaining motion synchronization of mobile robot network,” IEEE Trans. Syst., Man, Cybern. Syst., vol. 50, no. 3, pp. 815–828, Mar. 2020.

[22] Z. Liu, H. Wang, L. Xu, Y.-H. Liu, J. Lu, and W. Chen, “A failure-tolerant approach to synchronous formation control of mobile robots under communication delays,” in Proc. IEEE Int. Conf. Robot. Automat., 2018, pp. 1661–1666.

[23] X. Dong, B. Yu, Z. Shi, and Y. Zhong, “Time-varying formation control for unmanned aerial vehicles: Theories and applications,” IEEE Trans. Control Syst. Technol., vol. 23, no. 1, pp. 340–348, Jan. 2015.

[24] X. Dong and G. Hu, “Time-varying formation tracking for linear multiagent systems with multiple leaders,” IEEE Trans. Autom. Control, vol. 62, no. 7, pp. 3658–3666, Jul. 2017.

[25] Y. S. Chiappe and D. Panagou, “Herding an adversarial attacker to a safe area for defending safety-critical infrastructure,” in Proc. Amer. Control Conf., 2019, pp. 1035–1041.

[26] D. Silvestre, P. Rosa, J. P. Hespanha, and C. Silvestre, “Stochastic and deterministic fault detection for randomized gossip algorithms,” Automatica, vol. 78, pp. 46–60, 2017, doi: 10.1016/j.automatica.2016.12.011.

[27] Y. Dong, N. Gupta, and N. Chopra, “False data injection attacks in bilateral teleoperation systems,” IEEE Trans. Control Syst. Technol., vol. 28, no. 3, pp. 1168–1176, May 2020.

[28] N. O. Tippenhauer, C. Pöpper, K. B. Rasmussen, and S. Capkun, “On the requirements for successful GPS spoofing attacks,” in Proc. 18th ACM Conf. Comput. Commun. Secur., 2011, pp. 75–86.

[29] G. Biench, Y.-C. Liu, and F. Pasqualetti, “Secure navigation of robots in adversarial environments,” IEEE Contr. Syst. Lett., vol. 4, no. 1, pp. 1–6, Jan. 2020.
He was also a Guest Editor of the IEEE/CAA Journal of Automatica Sinica, etc. He was the winner of Outstanding Thesis Award, Chinese Association of Automation, 2015. He was a recipient of the Best Paper Award from IEEE VTC`20-FALL. He was the winner of Outstanding Thesis Award, Chinese Association of Automation, 2015. He was a recipient of the Best Paper Award from IEEE VTC`20-FALL.

Yushan Li (Student Member, IEEE) received the B.E. degree in automatic control from the School of Artificial Intelligence and Automation, Huazhong University of Science and Technology, Wuhan, China, in 2018. He is currently working toward the Ph.D. degree in control science and engineering with the Department of Automation, Shanghai Jiaotong University, Shanghai, China. He is a member of Intelligent of Wireless Networking and Cooperative Control Group. His research interests include robotics, security of cyber-physical system, and distributed computation and optimization in multiagent networks.

Jiapeng He (Senior Member, IEEE) received the Ph.D. degree in control science and engineering from Zhejiang University, Hangzhou, China, in 2013. He is currently an Associate Professor with the Department of Automation, Shanghai Jiao Tong University, Shanghai, China. He was a Research Fellow with the Department of Electrical and Computer Engineering, University of Victoria, Canada, from December 2013 to March 2017. His research interests mainly include the distributed learning, control and optimization, security and privacy in network systems. Dr. He serves as an Associate Editor for IEEE TRANSACTIONS ON CONTROL OF NETWORK SYSTEMS, IEEE OPEN JOURNAL OF Vehicular Technology, and KSII TRANSACTIONS ON Internet and Information Systems. He was also a Guest Editor of IEEE TRANSACTIONS ON AUTOMATIC CONTROL, International Journal of Robust and Nonlinear Control, etc. He was the winner of Outstanding Thesis of the Association of Automation, 2015. He was a recipient of the Best Paper Award from IEEE WCSP'17, the Best Conference Paper Award from IEEE PESGM'17, was a finalist for the Best Student Paper Award from IEEE ICCA’17, and the Finalist Best Conference Paper Award from IEEE VTC`20-FALL.
Cailian Chen (Member, IEEE) received the B.Eng. and M.Eng. degrees in automatic control from Yan- shan University, Qinhuangdao, China, in 2000 and 2002, respectively, and the Ph.D. degree in control and systems from the City University of Hong Kong, Hong Kong, in 2006. She has been with the Department of Automation, Shanghai Jiao Tong University, Shanghai, China, since 2008. She is now a Distinguished Professor. She has authored three research monographs and more than 100 refereed international journal papers. She is the inventor of more than 20 patents. Her research interests include industrial wireless networks, computational intelligence and situation awareness, and Internet of vehicles.

Dr. Chen was a recipient of the prestigious IEEE TRANSACTIONS ON FUZZY SYSTEMS Outstanding Paper Award in 2008, and the Best Paper Award of WCSP’17 and YAC’18. She won the Second Prize of National Natural Science Award from the State Council of China in 2018, First Prize of Natural Science Award from The Ministry of Education of China in 2006 and 2016, respectively, and First Prize of Technological Invention of Shanghai Municipal, China in 2017. She was honored “Changjiang Young Scholar” in 2015 and “National Outstanding Young Researcher” by NSF of China in 2020. She has been actively involved in various professional services. She is a Distinguished Lecturer of IEEE VTS. She serves as the Deputy Editor for National Science Open, and an Associate Editor for the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IET Cyber-Physical Systems: Theory and Applications, and Peer-to-peer Networking and Applications (Springer). She also served as the TPC Chair of ISAS’19, the Symposium TPC Co-chair of IEEE Globecom 2016, and the Track Co-Chair of VTC2016-Fall and VTC2020-Fall.

Xinping Guan (Fellow, IEEE) received the B.Sc. degree in mathematics from Harbin Normal University, Harbin, China, in 1986, and the Ph.D. degree in control science and engineering from the Harbin Institute of Technology, Harbin, China, in 1999. He is currently a Chair Professor with Shanghai Jiao Tong University, Shanghai, China, where he is the Dean of School of Electronic, Information and Electrical Engineering, and the Director of the Key Laboratory of Systems Control and Information Processing, Ministry of Education of China. Before that, he was the Professor and Dean of Electrical Engineering, Yanshan University, Qinhuangdao, China. He has authored and/or coauthored 5 research monographs, more than 270 papers in IEEE transactions and other peer-reviewed journals, and numerous conference papers. As a Principal Investigator, he has finished/been working on many national key projects. He is the leader of the prestigious Innovative Research Team of the National Natural Science Foundation of China. He is an Executive Committee Member of Chinese Automation Association Council and the Chinese Artificial Intelligence Association Council. His current research interests include industrial cyber-physical systems, wireless networking and applications in smart factory, and underwater networks.

Dr. Guan was a recipient of the First Prize of Natural Science Award from the Ministry of Education of China in both 2006 and 2016, and the Second Prize of the National Natural Science Award of China in both 2008 and 2018. He was a recipient of the IEEE TRANSACTIONS ON FUZZY SYSTEMS Outstanding Paper Award in 2008. He is a “National Outstanding Youth” honored by NSF of China, “Changjiang Scholar” by the Ministry of Education of China, and “State-level Scholar” of “New Century Bai Qianwan Talent Program” of China.