robotics

MDPI

Article

Learning-Based Shared Control Using Gaussian Processes for Obstacle Avoidance in Teleoperated Robots

Catalin Stefan Teodorescu *, Keir Groves and Barry Lennox

Depterment of Electrical and Electronic Engineering, The University of Manchester, Manchester M13 9PL, UK
* Correspondence: s.teodorescu@manchester.ac.uk; Tel.: +44-7-308-010180

Abstract: Physically inspired models of the stochastic nature of the human-robot-environment interaction are generally difficult to derive from first principles, thus alternative data-driven approaches are an attractive option. In this article, Gaussian process regression is used to model a safe stop maneuver for a teleoperated robot. In the proposed approach, a limited number of discrete experimental training data points are acquired to fit (or learn) a Gaussian process model, which is then used to predict the evolution of the process over a desired continuous range (or domain). A confidence measure for those predictions is used as a tuning parameter in a shared control algorithm, and it is demonstrated that it can be used to assist a human operator by providing (low-level) obstacle avoidance when they utilize the robot to carry out safety-critical tasks that involve remote navigation using the robot. The algorithm is personalized in the sense that it can be tuned to match the specific driving style of the person that is teleoperating the robot over a specific terrain. Experimental results demonstrate that with the proposed shared controller enabled, the human operator is able to more easily maneuver the robot in environments with (potentially dangerous) static obstacles, thus keeping the robot safe and preserving the original state of the surroundings. The future evolution of this work will be to apply this shared controller to mobile robots that are being deployed to inspect hazardous nuclear environments, ensuring that they operate with increased safety.

Keywords: Gaussian process regression; semi-autonomous vehicle; shared control; obstacle avoidance; nuclear robotics

Citation: Teodorescu, C.S.; Groves, K.; Lennox, B. Learning-Based Shared Control Using Gaussian Processes for Obstacle Avoidance in Teleoperated Robots. *Robotics* **2022**, *11*, 102. https://doi.org/10.3390/robotics11050102

Academic Editors: Shuai Li, Dechao Chen, Mohammed Aquil Mirza, Vasilios N. Katsikis, Dunhui Xiao and Predrag Stanimirović

Received: 27 July 2022
Accepted: 15 September 2022
Published: 21 September 2022

1. Introduction

Gaussian process regression (GPR), also referred to as Kriging [1] in geostatistics, is a generic supervised learning method that is designed to solve regression problems. GPR models can be used to predict the effect variables of a partially observed physical process by building a *continuous* map with artificial data. Such maps have been used to model spatial phenomena, including temperatures [2], magnetic field intensity [3], radiation field [4], spatial localization of minerals in mining geology [1], and epidemic growth [5]. Key strengths of GPR modeling include the ability to use repeated observations, collected at the same location, within the training data [4]; acceptance of both sparse and highly clustered data [4]; data acquisition sampled at variable time and/or space [1]; ability to learn an effective representation of nonlinear dynamics even using small data sets [6]; and predictions consist of two types of information, including the *mean function* (corresponding to the best estimation in average) and an associated confidence (expressed using the standard deviation). Although often neglected when applying GPR, in this article, we make explicit use of the confidence metrics.

Combining GPR with control systems is an emerging topic that has had increasing interest in recent years [7,8]. This is a versatile method that has numerous applications in robotics. From a broader perspective, in our work, we seek to design bespoke control algorithms that are improved by integrating the *stochastic* nature of a real process (or plant) into their design. In this respect, GPR has been demonstrated to be a highly effective approach to modeling, and it is the technique utilized in this work.

1.1. Literature Review

To explain the opportunities for combining a GPR model within a control system for use with robotic systems we employ the classical closed-loop schematic diagram provided in Figure 1. In the following discussion, we separate the literature review into research related to plant, control and reference.

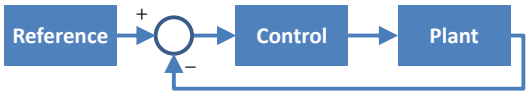

Figure 1. Closed-loop system.

1.1.1. Plant

It is relatively common when developing control systems to utilize an existing model of the plant. For example, the dynamic model, defined in Equation (1), may be available:

$$x_{k+1} = g_{\text{nom}}(x_k, u_k)\,, \tag{1}$$

where g_{nom} is often derived from first principles (e.g., a kinematic model where the velocity is the derivative of the position; a mechanical model derived from Euler–Lagrange equations that make use of the mass, inertia, gravity, etc.; and an electrical model containing resistors, capacitors, inductance, etc.); in Equation (1), x_k is the discrete state variable and u_k is the control action. Recognizing that such models are often imperfect, the model can be further improved by augmenting it using a learned part, $g_{\text{lrn}}(x_k, u_k)$, such that the overall plant is defined as shown in Equation (2):

$$x_{k+1} = g_{\text{nom}}(x_k, u_k) + g_{\text{lrn}}(x_k, u_k)\,. \tag{2}$$

In [8,9], $g_{\text{lrn}}(x_k, u_k)$ can be identified using GPR with the assumption that both x_k and u_k are measurable, and $g_{\text{lrn}}(x_k, u_k)$ is defined such that it can explain the difference between x_{k+1} and $g_{\text{nom}}(x_k, u_k)$. The function, $g_{\text{lrn}}(x_k, u_k)$, is identified, or learned, using data collected when the same task is executed repeatedly. The function, $g_{\text{lrn}}(x_k, u_k)$, can be defined in a variety of ways (e.g., linear *versus* nonlinear), but it is important that, following identification, it can be applied to novel situations [8].

In situations where an existing model of the plant g_{nom} in (2) is not available, the system dynamics relies entirely on the learned part g_{lrn}. In [10], this is defined as $g_{\text{lrn}} := x_k + \Delta_k$, and, using GPR, a one-step ahead prediction model is built using (x_k, u_k) as training inputs and the difference $\Delta_k := x_{k+1} - x_k$ as the training target. An advantage of this approach is that the system dynamics, x_{k+1}, do not fall back to zero in unexplored areas but instead remain at x_k. In a different work [6], such models are called Gaussian process dynamical models and are used to predict latent trajectories (in the context of generating human motion, such as walking).

1.1.2. Reference

To improve the performance of a model-based control system, refs. [11–13] applied GPR to learn a reference periodic signal from noisy data. The resulting model was then used within the control system to predict the most likely future evolution of the reference signal to improve control performance. The technique was applied to a medical robotics application. In related work, refs. [4,14] used GPR models in conjunction with Simultaneous localization and mapping (SLAM) to build *continuous* radiological maps of a priori unknown radioactive areas.

In [15], a GPR model was used to address a quality prediction problem, where the aim was to accurately and robustly predict post-process responses from online measurements.

1.1.3. Control

In relation to control, we can divide the literature into work where GPR models contribute indirectly and directly to the control solution.

To begin, we will consider research where GPR models contribute *indirectly* to the control solution, in the sense that a standard (off-the-shelf) control law is used in conjunction with a model of the plant or reference, that makes use of a GPR model. Such applications are described in [9], which utilized an H_2 robust controller; [11–13], which used GPR models within a linear model predictive controller (MPC); and [8], which integrated a GPR model into a nonlinear MPC. An insightful review on learning-based MPC is presented in [7]. Furthermore, a GPR model-based policy search method (reinforcement learning) is presented in [10], which is shown to be very efficient in terms of required interaction time for learning the control policy.

GPR models can also contribute directly to the control solution. For example, the inverse kinematics problem of robotic arm manipulators was solved using a GPR model in [16] and in several articles mentioned in [8]. Relevant research was also carried out in [17], where a cost function modeled using a GPR $f(x)$ was learned (the state x consisted of the tuning parameters of a PI controller) and then the predicted minimum x_* was used to explore the real system behavior, experimentally.

The work presented in this paper adds to the use of GPR modeling to directly contribute to the control solution.

1.2. Theory on GPR Modeling

Learning in a Bayesian framework involves using probability to express all forms of uncertainty [18]. Specifically, a joint probability distribution is established over all unknown quantities, which is then used to express the relation between these quantities. In case other quantities are accessible and known, a (joint) conditional probability is used instead. GPR modeling relies on the Bayesian framework. Let us assume that values $x_1, \ldots, x_M$ are known, each belonging to $\mathbb{R}^D$ with dimension $D \in \mathbb{Z}_{\geq 1}$, and scalar continuous function values $f(x_1), \ldots, f(x_M)$ are unknown, each belonging to $\mathbb{R}$. The *prior*, based on *past* knowledge, expresses the relation between an arbitrary number of the unknowns ($M \in \mathbb{Z}_{>0}$) treated as continuous random variables, in the absence of any new data as [19,20]:

$$[f(x_1), \ldots, f(x_M)] \sim \mathcal{N}([m(x_1), \ldots, m(x_M)], \Sigma)\,, \tag{3}$$

where $m(\cdot)$ is a known *mean function*, and the covariance matrix Σ consists of elements $\Sigma_{ij} := k(x_i, x_j)$ with known kernel function $k(\cdot, \cdot)$; the notation $\mathcal{N}$ stands for the (multivariate) normal distribution. The prior (3) can be written in a more compact form:

$$f \sim \mathcal{GP}(m, k), \quad \text{or} \quad p(f) = \mathcal{N}(m, \Sigma)\,, \tag{4}$$

emphasizing the fact that a Gaussian process is fully defined by the pair (m, k) [10]. An example is shown in Figure 2.

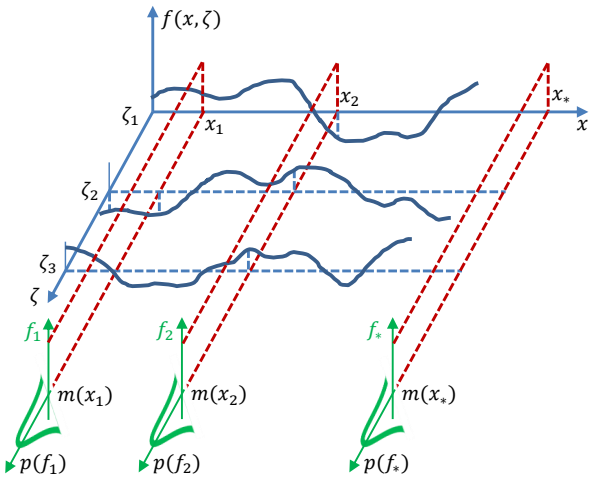

Figure 2. Schematic diagram of a Gaussian Process $f(x,\zeta)$ emphasizing it is a stochastic process [21] that can be described in the continuous, x, or discrete, ζ, domains. Marginal distributions $p(f_1)$, $p(f_2)$ and $p(f_*)$ are defined on function values $f_1 := f(x_1)$, $f_2 := f(x_2)$ and $f_* := f(x_*)$. Adapted from Figure 5.1 in [22].

We now split $x_1, \ldots, x_M$ into *training* $X := [x_1, \ldots, x_N]$ and *testing* $X_* := [x_{N+1}, \ldots, x_M]$, with $1 \leq N < M$. In the particular case of one testing location ($N+1 = M$), the scalar notation $x_* := x_{N+1}$ is used, as in Figure 2. The noisy observations random variables $y_i = f(x_i) + \epsilon_i$, where $\epsilon_i \sim \mathcal{N}(0, \sigma_y^2)$ and $i = 1, \ldots, N$, form the observations vector $y := [y_1, \ldots, y_N]$. Thus, the derived joint density is

$$\begin{pmatrix} y \\ f_* \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} m(X) \\ m(X_*) \end{pmatrix}, \begin{pmatrix} k(X,X) + \sigma_y^2 I & k(X,X_*) \\ k(X,X_*)^{\mathrm{T}} & k(X_*,X_*) \end{pmatrix} \right), \tag{5}$$

where the (noise-free) function values at the test points X_*, $f_* := [f(x_{N+1}), \ldots, f(x_M)]$, are unknown; I is the identity matrix, and notation $^{\mathrm{T}}$ stands for the transpose. The reasoning within Bayesian inference is to use new experimental training data, namely a sample $\mathcal{D} = \{(x_1, \mathring{y}_1), \ldots, (x_N, \mathring{y}_N)\} = (X, \mathring{y})$, with $\mathring{y} = [\mathring{y}_1, \ldots, \mathring{y}_N]$, the recorded observations, to "update" the prior distribution into the posterior predictive density [19]:

$$\begin{aligned} p(f_*|X_*, \mathcal{D}) &= \mathcal{N}(f_*|\mu_{*|X}, \Sigma_{*|X}) \\ \mu_{*|X} &= m(X_*) + k(X,X_*)^{\mathrm{T}}(k(X,X) + \sigma_y^2 I)^{-1}(\mathring{y} - m(X)) \\ \Sigma_{*|X} &= k(X_*,X_*) - k(X,X_*)^{\mathrm{T}}(k(X,X) + \sigma_y^2 I)^{-1} k(X,X_*)\,, \end{aligned} \tag{6}$$

where $\mu_{*|X} := E\{f_*\}$ is the mean function and $\Sigma_{*|X} := E\{(f_* - E\{f_*\})(f_* - E\{f_*\})^{\mathrm{T}}\}$ is the covariance matrix. The notation E stands for the expected value operator. This is depicted in Figure 3.

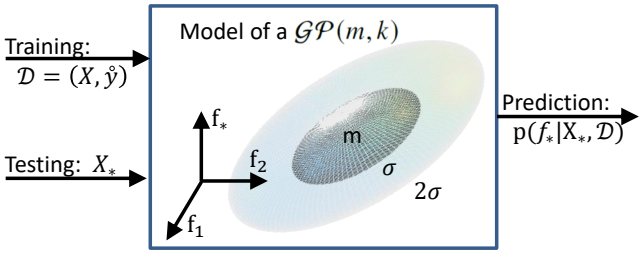

Figure 3. GPR illustrated as an input–output block diagram. The image in the middle introduces isosurfaces (ellipsoids) in function space $f = [f_1\ f_2\ f_*]$ of the joint probability density function $p(f) = \mathcal{N}(m, \Sigma)$ in (4), with $f_1 := f(x_1)$, $f_2 := f(x_2)$ and $f_* := f(x_*)$ from Figure 2. The inner σ-ellipsoid and the outer 2σ-ellipsoid regroup about 68% and 95%, respectively, of the total energy of $p(f)$.

The proof makes use of the relations between the marginals and conditionals of a multivariate normal distribution [19,23]. All these distributions implicitly depend on hyperparameters (σ_y, θ), with θ parameterizing k appearing in (4). These hyperparameters can be experimentally identified by optimizing the log marginal likelihood [19].

The noisy predictive observations

$$y_* = f_* + \epsilon_*\,, \tag{7}$$

where $y_* = [y_{N+1}, \ldots, y_M]$ and $\epsilon_* = [\epsilon_{N+1}, \ldots, \epsilon_M]$, have the mean function

$$\mu_{y_*} \triangleq E\{y_*\} = E\{f_* + \epsilon_*\} = E\{f_*\} = \mu_{*|X}\,, \tag{8}$$

and variance

$$\begin{aligned} \sigma^2_{y_*} &\triangleq E\{(y_* - E\{y_*\})^2\} = E\{(f_* + \epsilon_* - E\{f_* + \epsilon_*\})^2\} \\ &= E\{(f_* - E\{f_*\})^2\} + E\{(\epsilon_* - E\{\epsilon_*\})^2\} \\ &\quad + E\{2(f_* - E\{f_*\})(\epsilon_* - E\{\epsilon_*\})^2\} \\ &= \mathrm{diag}(\Sigma_{*|X} + \sigma_y^2 I)\,, \end{aligned} \tag{9}$$

where the operator 'diag' takes the diagonal of a matrix.

1.3. From Model-Based to Data-Driven Control

In a context where uncertainties play a fundamental role, deciding whether to use a model-based or data-driven approach involves a trade-off between available project resources. Mechanistic models can be difficult and time-consuming to identify and may contain significant errors. Their main advantage is the analytical insight (theory), the ability to relate physical quantities to experimental observations. Quite often, the model-based approach involves solving a complex optimization problem. In [24], a robust concurrent design for a collaborative 2-degrees-of-freedom robot (cobot) is presented, where the uncertainty lies in the human–robot interaction; the robust concurrent design ensures that differences in operators have little impact on the system performance, whereas in our user-centered design, the shared control accounts for the specificity of each operator (Section 2.1); the optimization takes into account, simultaneously, the dynamic model of the mechanical structure as well as the controller equations. A different theoretical framework to deal with the robustness to human operators is presented in [25]: by lowering the inertia of motors that drive an upper-limb exoskeleton prototype, it is shown that control performance is high and also insensitive to uncertainties in the human dynamics (the wearer of the exoskeleton).

Addressing uncertainties using the alternative sensor-based, data-driven approach is often simpler to implement, although it can be time consuming in terms of experimentation

and often lacks the analytical insight (theory). Reinforcement learning (RL) discovers optimal behaviors through trial and error in a simulated (see, for example, [26]) or real environment, by employing a reward function that penalizes undesired behaviors. In [26], RL was used in learning a robotic manipulator's motion planning and control, then grasping was demonstrated using that robotic manipulator in the context of obstacle avoidance.

1.4. Obstacle Avoidance in Nuclear Robotics

The aim of this work is to design control systems that are able to support the deployment of robotic systems in hazardous nuclear environments, such as those described in [4,14,27,28]. In such environments, it is typically necessary to avoid getting too close to radioactive material, which may damage electronic components within the robot or cause the robot to become contaminated. For example, the Spot robot, manufactured by Boston Dynamics, has, by default, a low-level automation system that ensures that the robot keeps an arbitrary distance away from obstacles. This low-level ability means that maneuvering the robot through tele-operation is significantly simplified, particularly in cluttered environments where it is tele-operated beyond the line of sight [4,29].

Figure 4 illustrates four types of tele-operated robots, each belonging to a different category of robot: land based (e.g., the SuperDroid HD2 in Figure 4a), water based (e.g., the BlueROV2 in Figure 4b), air (e.g., the DJI Matrice 300 RTK in Figure 4c) and manipulator (e.g., the KUKA LBR iiwa 14 R820 in Figure 4d). All of these robots have features in common. For example, they can each be tele-operated using a joypad or similar device (e.g., the Sony PlayStation DualShock 4 illustrated in the middle of Figure 4) and they can all be used for safety-critical applications in hazardous (or extreme) environments. In such situations, they might all require the same (low-level) functionality to be implemented, such as the obstacle avoidance control algorithm we present in Section 2.

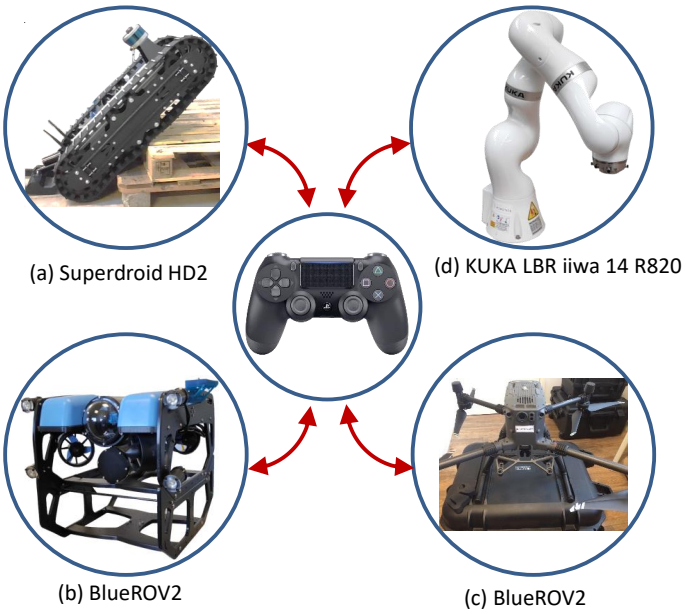

Figure 4. Tele-operated robots that would be suitable candidates for the control algorithms presented in this paper.

General-purpose obstacle avoidance algorithms can be derived from path planning using various techniques, including the artificial potential fields method, which introduces repulsive forces [30,31], sampling-based methods [30] and the dynamic window approach, which has been implemented in ROS [32]. When using these techniques, the final goal location needs to be specified in advance, whereas when using the method proposed in this article, this information is not required.

1.5. Research Contribution

In this article, we propose a novel obstacle avoidance shared control algorithm based on GPR model predictions, in a stochastic context. First, to the best of our knowledge, making use of the confidence interval of GPR models has not been proposed yet in the scientific literature for addressing obstacle avoidance problems. It is our intention to make a significant contribution in this setting. Second, a key message of this article is that improved control performance can be achieved by using the predicted posterior distribution of a Gaussian process. When using GPR models for prediction, it is typical for the *mean function* to be used, but the associated confidence metric is typically ignored. In this article, we propose to use the information contained within the confidence metric, defined within the GPR model. Using this information, we demonstrate that it is possible to design a control system that is particularly suited to safety-critical robotic applications. This research opportunity is novel and has not been exploited yet.

Continuing the work presented in [33], we extend the assist-as-needed (or shared control) algorithm dealing with static obstacle avoidance such that it incorporates *learning* and *prediction* based on a GPR model, rather than using the first-principle, mechanistic models that were utilized in [33]. More specifically, we make the transition from a physically inspired model-based controller [33] to a new data-driven control solution. In this respect, a GPR model ensures a smooth transition, as (some of) the *hyperparameters* can be directly related to the physics of the real process. The relationship that exists for the physical properties of the robot contributed to our decision to use GPR in favor of other more abstract models, such as neural networks [34]. This is particularly important in nuclear applications, where transparency is typically required to ensure regulatory approval for new technologies [35].

Having provided a background to this work, in Section 2, we present the proposed shared control algorithm for safe obstacle avoidance in the context of tele-operated robots. In Section 3, we show that this shared control relies on a GPR model to inform how it should make a safe stop maneuver. Finally, the paper ends with conclusions in Section 5.

2. Shared Control

The concept of *shared control* relies on the cooperation (or collaboration) between two actors: in this work, a human and a machine. The principle of the technique is to combine the best capabilities of each to produce better results than would be obtained when each is used in isolation. It is therefore essential that the algorithms are designed such that they avoid conflict (or contradiction) between the two parties.

2.1. A User-Centered Design

There will be some level of variability in driving style when the operation of a robot is transferred from one user to another. For example, an experienced operator may well control the robot differently to a novice, and even for the same user, executing the same task repeatedly might result in different styles at the beginning and end of a working day, as the operator tires. Hence, we believe that using a "one-size-fits-all" shared control solution is not sensible. One possible way to address this problem would be to group users into clusters and define specific operating modes with purposely-tuned control algorithms for each cluster. However, this does not exclude that some individuals within the same group might feel unhappy with the end result. For this reason, we are interested in a personalized and highly customizable shared control algorithm, able to capture and incorporate the notion of variability at the user level.

2.2. Addressing Uncertainty

In our previous work [33], using a highly complex offline optimization-based procedure, we identified a relatively simple shared control algorithm that is illustrated in Figure 5. In this work, we propose to replace the optimization procedure, ultimately used to compute the blue curve in Figure 5, with a process able to learn this curve from a small

number of experiments. All (x^{obst}, v)-points on the blue curve in Figure 5 correspond to initial states, and from any of these initial states, the controller should be able to advance the robot toward an obstacle in front of it, as shown in Figure 5b, and stop it safely in a fixed but otherwise arbitrary amount of time Δt_{stop} next to the obstacle.

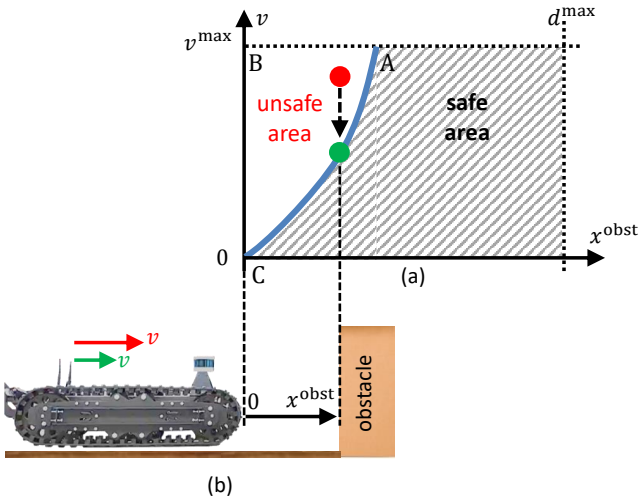

Figure 5. Shared control, adapted from [33]: (**a**) Conceptual drawing in state-space (v, x^{obst}) of the working principle: the algorithm reduces any unsafe linear velocity v within the area ABC (see, for example, the red solid circle) toward a safe value (see the green solid circle located on the blue boundary curve); (**b**) the red and green arrows suggest a reduction in linear velocity as a consequence of applying shared control; x^{obst} is measured from the obstacle toward the robot; and $d^{\max}$ is the maximum observed distance to the obstacle.

In reality, the shape of the blue curve will be dependent on the human operator's style of driving and also on the environment. For example, it may be desirable for a robot advancing on sand to behave differently to a robot moving on a concrete floor. The process is stochastic in that consecutive repetitions of the same task might involve (slightly) different actions from the operator, as well as interactions of the robot with the environment, e.g., slipping and bouncing when braking. To summarize, the sources of uncertainty are the human operator (the driving style), the robotic platform (software and hardware) and the environment conditions. Consequently, we might have to deal with a family of curves associated with a particular driver and a particular environment, and the question is how to make intelligent usage of all this information in the shared control algorithm.

2.3. Advancing Forward and Turning

The shared control algorithm adjusts both the linear and the angular velocities. Figure 5 explains how the linear velocity control is implemented. In this case, unsafe decisions from the human operator, such as requesting a linear velocity $v \in [0, v^{\max}]$ that is too high in situations where the robot is too close to an obstacle $x^{\text{obst}} \geq 0$, would make the shared control algorithm act by reducing v toward the pre-computed blue boundary curve as shown in Figure 5a. Not shown in that figure, the angular velocity is also reduced proportionally, thus completing the algorithm. Whilst the above analysis assumes the robot advances in a forward direction only, this is simply a convention and analogously, the same principles apply when the vehicle moves backward, although this situation is out of the scope for this paper.

2.4. Algorithm

In this section, the case of the robot advancing toward a forward-facing obstacle is analyzed. The shared control Algorithm 1 has five inputs and two outputs. The inputs are as follows: the linear velocity and angular velocity (v, ω) measured by the odometry;

the joypad data corresponding to the desired linear velocity and desired angular velocity (v_d, ω_d) (note, there is a one-to-one mapping between the location of the joystick and the steady-state velocity [33]); and x_{co}^{obst} is the distance to the closest obstacle located straight ahead along the x^{obst}-axis in Figure 5b, measured using a 3D LiDAR (the Velodyne VLP16 in Figure 6c). The outputs of the algorithm are the requested linear velocity and requested angular velocity (v_r, ω_r), sent to the robot's controller.

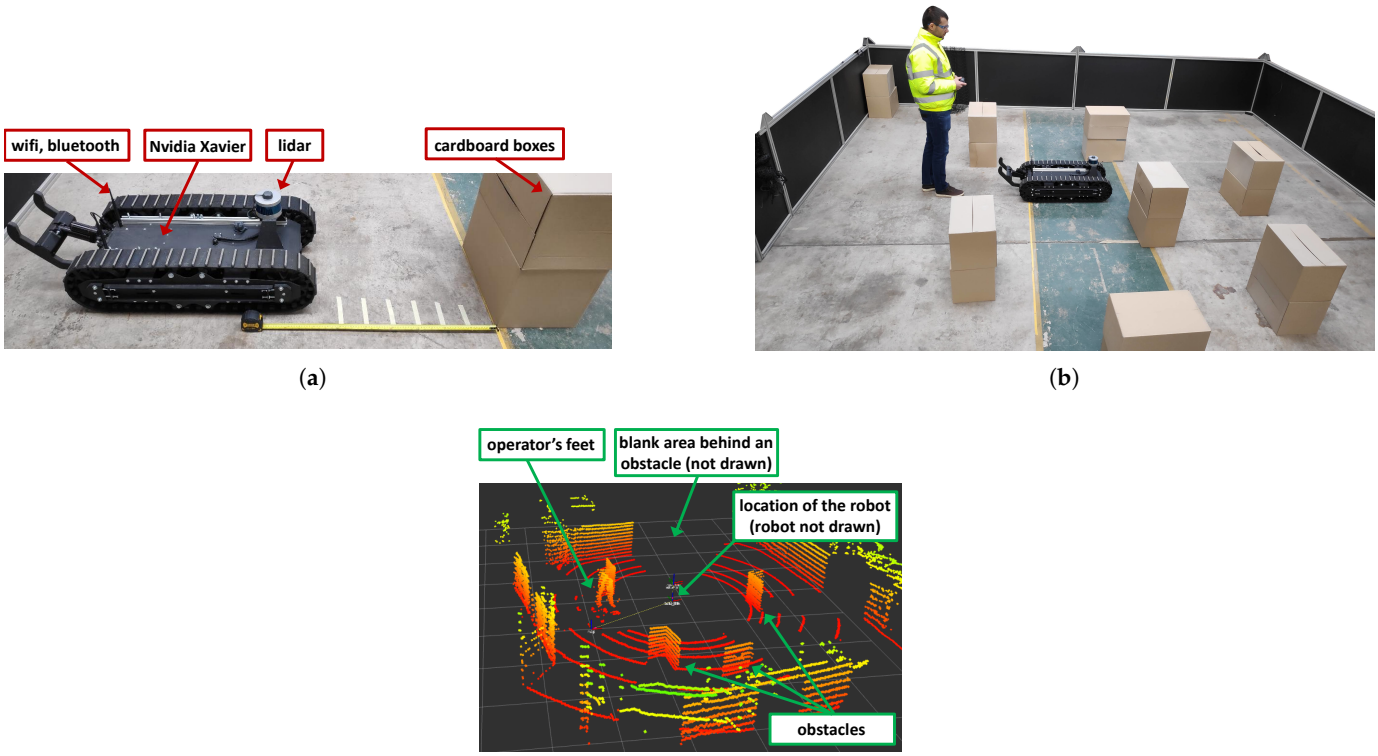

Figure 6. Experiments: (**a**) Collecting training data for the GPR model; (**b**) Testing the shared control algorithm in a circuit with obstacles (made of cardboard boxes). The human operator tele-operates the robot using a joypad; (**c**) Visualization in Rviz of the 3D LiDAR data corresponding to Figure 6b.

Algorithm 1 The proposed obstacle avoidance shared control: the case of a vehicle advancing forward with an obstacle located straight ahead

$v_r = v_d$; $\omega_r = \omega_d$ ▷ initialization: by default shared control is disabled
if $v > 0$ and $\omega = 0$ ▷ vehicle advancing forward without rotation
and $v_d > 0$ ▷ driver expresses intention to advance forward
then
Find x_{co}^{obst} ▷ identify the closest obstacle
$v_r = \min\left(v_d, \text{blue\_curve}(x_{co}^{obst})\right)$ ▷ blue_curve from Figure 5a
$\omega_r = \omega_d \, v_r / v_d$ ▷ reduce ω_d proportionally to the decrease in v_r vis-à-vis v_d
end if

Not presented here, the backward motion can be easily derived analogously to Algorithm 1. Currently, we are working on extending this algorithm to the situation where the robot simultaneously advances forward $(v > 0)$ and rotates $(\omega \neq 0)$. Although not shown here, the algorithm described in this article is applicable to all the categories of robots shown in Figure 4.

2.5. Software Architecture

To test the algorithm, we used the land robot SuperDroid HD2, shown in Figure 4a. The software architecture relies on ROS [32]. The corresponding C++ code used for our experiments is freely available on https://github.com/StefanT83/SC_obst_avoid_GPR_1D.git (accessed on 14 September 2022) and uses three ROS topics as inputs and generates one ROS topic as output. The input ROS topics are as follows:

(i) `/velodyne_point` data, issued by the Velodyne ROS driver; https://github.com/ros-drivers/velodyne/ (accessed on 14 September 2022)
(ii) `/joy_raw` is the joypad data published by the standard joy package; http://wiki.ros.org/joy (accessed on 14 September 2022)
(iii) `/wheel_odom` is the odometry, computed by the default ROS packages the robot came with.

The output ROS topic was a new (virtual) joystick data, `/joy_out`, computed by the shared control algorithm described in Section 2.4. This information was subsequently being used by the robot's `joy_teleop` node and converted into a velocity command `/cmd_vel` used by the `roboteq_driver`.

3. Experiments

The test rig used in this study consists of a SuperDroid HD2 robot illustrated in Figures 4a, 5b, and 6a,b, moving on concrete floor inside the mobile arena shown in Figure 6. It was equipped with a Velodyne 3D LiDAR, ensuring perception of the environment (see Figure 6c), which was used to measure experimentally x^{obst}, the distance to the nearest obstacle. The linear velocity v and the angular velocity of the robot were estimated using odometry, and based on measurements from motor encoders. The maximum linear velocity $v^{\max}$ was approximately 0.4 m/s, and the robot was remotely actuated using a Bluetooth joypad, specifically the Sony PlayStation DualShock 4, illustrated in the middle of Figure 4.

In this paper, we focus on the *method* of how we can build a GPR model, and then use its predictions inside the shared control algorithm. We used *one* human operator, but in future work, we intend to run a more in-depth study involving more participants, enabling the statistical significance of the control solution to be compared against other approaches, such as the popular dynamic window approach, that has been implemented in ROS [32].

3.1. Training Data for the GPR Model

We propose to *learn* the characteristics of the user's driving style in the context of the environment depicted in Figure 6a. In the experiments, the robot is located at a standstill at an arbitrary location, facing toward an obstacle, in this case, made of cardboard boxes, and the following protocol was followed. The only instruction given to the human operator was that they should make use of their driving skills to move the robot close to the obstacle in one attempt, without colliding with it. During this procedure, we recorded the linear velocity $v(t)$ and kept track of the distance to the obstacle $x^{\text{obst}}(t)$, where t represents time. We then recorded the initial instant of time, t_0, and instant of time, t_1, when the robot reached a standstill and stopped safely near the obstacle. Next, we marked the point $(x^{\text{obst}}(t_1), v(t_1))$ on a graphic using a cross, as in Figure 7a. Lastly, we recorded the amount of time, $\Delta t_{\text{stop}} := t_1 - t_0$, elapsed to stop safely. The value of Δt_{stop} can be considered to be a tuning parameter, and in this work, was set to $\Delta t_{\text{stop}} = 4$ s. In other words, we were only interested in those experiments that lasted 4 s and disregarded the rest. This is discussed later in Section 4. Next, we repeated the scenario above four more times and collected data from each test. In Figure 7a, a total of five experimental data points (or noisy observations), $(x^{\text{obst}}(t_1), v(t_1))_i$ are illustrated by crosses. The index of the repetition, or iteration, $i = 1 \ldots 5$. The fact that so few experimental data points are needed to make the algorithm work is a major benefit of this proposed GPR-based approach.

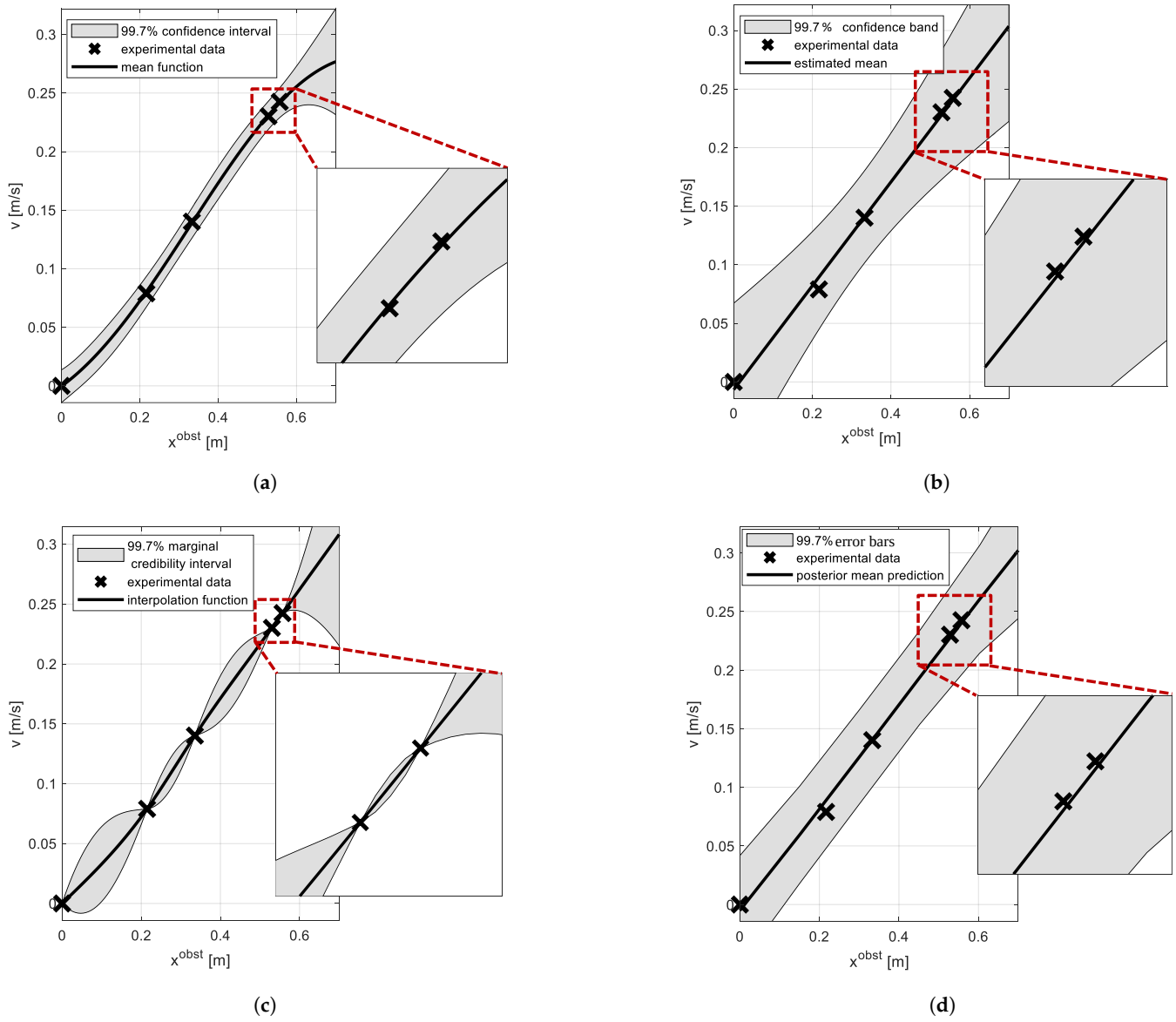

Figure 7. Comparison between predicting the continuous evolution of the safe stop maneuver using (**a**) the proposed method, and (**b**–**d**) other readily available (off-the-shelf) algorithms. The same training experimental data were used in all figures. (**a**) Using Gaussian process regression, specifically the mean function μ_{y_*} in (8), and three times the standard deviation σ_{y_*} in (9) to build the shown 99.7% confidence interval – code largely inspired by `demoRegression.m` in [36]; (**b**) Using simple linear regression – code largely inspired by `regression_line_ci.m` in [37]; (**c**) Using the method of interpolating noise-free data using a Gaussian with prior precision $\lambda = 200$—code largely inspired by `gaussInterpDemo.m` in [34]; (**d**) Using a Bayesian neural network—code largely inspired by `mlpRegEvidenceDemo.m` in [34] and `demev1.m` in [38].

These points were then used to build a GPR model (4) by choosing (i) the *mean function* m to be zero, translating the fact that if we extract a sample consisting of any of the *random variables* that constitute the Gaussian process prior in (3), (4), (5), their mean approaches zero as the size of that sample increases; this choice is typical for situations where little or no prior information about the mapping f is available [3,15]; the mean value of any *random variable* that constitute the Gaussian process posterior in (6) converges to zero as the predictive location X_* gets further away from the collected data locations X [8]; (ii) the

covariance function (or kernel function) k is defined to be equal to the squared-exponential (SE) covariance function in one dimension [16,34]:

$$k(x_p, x_q) := \sigma_f^2 \exp\left(-\frac{1}{2l^2}(x_p - x_q)^2\right) + \sigma_y^2 \delta_{pq}, \tag{10}$$

where δ_{pq} is the Kronecker delta, with p, q indices. The reason is that functions sampled from a GPR with a SE kernel are infinitely differentiable, and therefore very smooth, which is convenient for our application, whereas other options such as the Matern kernel produce "rougher" functions [4,19]. The three *hyperparameters* in (10) are as follows: (i) *Noise variance* σ_y^2, which can be estimated using the *maximum likelihood estimate* $\hat{\sigma}_y^2$ in a series of additional experiments, as follows. Fix a convenient value along the horizontal axis in Figure 7a. Then, using a series of repetitions, collect a *sample* of v values and compute the sample variance $\hat{\sigma}_y^2$. Next, use this estimate in place of σ_y^2 in (10). (ii) *Signal variance* σ_f^2. (iii) *Length scale* l. The last two *hyperparameters* can be computed by minimizing the negative log marginal likelihood with respect to the hyperparameters [16,36]. Intuitively, this works by selecting the covariance function (10) that explains best the observations.

An interesting property, which supports the use of GPR models in this work, is that the smoothness of the realizations of a GPR model can be adjusted by accurately defining the covariance function (including its parameters) [16]. We know from our previous model-based approach [33] that we expect the curve, expressing a safe stop maneuver, to be smooth.

3.2. Predictions from the GPR Model

Having built the GPR model, we can then use it to make predictions. Specifically, we are interested in the evolution of the process in terms of the predicted linear velocity v over a desired *continuous* range (or domain) of x^{obst}. The result is shown in Figure 7a and consists of two pieces of information, namely the mean function (black line) and its associated 99.7% pointwise confidence interval or $3\sigma_{y_*}$ interval (gray area), where σ_{y_*} is the standard deviation (9). The 99.7% value and, consequently, the factor 3 are arbitrary values that act interchangeably as tuning parameters—this is discussed later in Section 4. The gray area illustrates the range of velocities that the GPR model believes are acceptable, relative to the distance to the obstacle x^{obst}, and based on what it has learned of the user's behavior. The next question is how can we use this information in place of the blue curve in the shared control algorithm in Figure 5.

3.3. Personalized Shared Control

The predictions based on the GPR model in Figure 7a integrate the features of the relationship between the operator, robot and environment into the shared controller. To be able to use this information meaningfully inside our shared controller, we rely on the assumption that the protocol used to carry out the experimental data collection for building the GPR model is consistent and relevant to the real-world scenario.

Finally, we propose to design a conservative (cautious) shared controller that is guaranteed to allow for safe stop maneuvers in 99.7% of cases; 99.7% relates to a $3\sigma_{y_*}$ interval. This arbitrary value of $3\sigma_{y_*}$ can be adjusted depending on how critical it is to avoid the object. For this shared control, we need to consider the lower bound curve of the confidence interval, namely $m(x) - 3\sigma_{y_*}$ in place of the blue curve in Figure 5. As will be discussed later in Section 4, this curve requires minor adjustments at the extremities.

3.4. Testing

The circuit in Figure 6b was used to drive the vehicle with the proposed shared control enabled. By analyzing the heart rate and eye blinking as psychometric measures of cognitive load [39], we noticed their average values were different when comparing the situation with and without assistive control. Compared to the situation without any assistive control, driving was found to require less cognitive effort, or concentration, for

the operator to avoid striking obstacles. The operator was thus able to focus on other higher-level tasks, such as navigating toward a desired destination point or future mission planning. In addition to that, we noticed that fewer obstacles were struck during the experiments, at the expense of a slower moving robot, which may be the price to pay for increased safety in this framework. The paper's Supplementary Material Video S1, also available on https://youtu.be/6vkszxGEn50 (accessed on 14 September 2022), shows excerpts for data collection and testing the obstacle avoidance shared control algorithm, all carried out by the one experienced operator. A more elaborate statistical study with multiple participants (operators) will be conducted in the future to confirm and quantify the aforementioned observations, similar to the study in [33].

4. Discussion

In this section, we add some comments about the proposed learning-based shared control method.

The central section of the predictions in Figure 7a, i.e., the results between approximately $x^{\text{obst}} = 0.1$ m and $x^{\text{obst}} = 0.6$ m, are in agreement with the physical intuition and captures correctly the variability of the process. However, the predictions outside this region are not. According to the physics, the origin point $(x^{\text{obst}}, v) = (0, 0)$ has no variability, i.e., its standard deviation equals zero, as the robot is at a standstill; however, the GPR does not capture this. Instead, it assumes that there is additive noise present everywhere on the latent function [16]. In addition, the right-hand side of the prediction in Figure 7a curves downward, and eventually, the mean function converges (falls) to zero (not shown). This is a well-known and typical behavior of a GPR in unexplored areas (regions where no experimental data have been collected), where the predictions tend toward the chosen m in (4), in our case to zero [8]. Consequently, the extremities need to be corrected. In particular, we are only interested in the lower bound of the confidence interval in Figure 7a since this is the only information used in our shared control algorithm. It is adjusted around the origin by saturating it to zero, and on the far right of Figure 7a, we stop making use of predictions as soon the the lower bound of the confidence interval tends to bend downward (this happens around $x^{\text{obst}} = 0.62$ m in Figure 7a).

For comparison reasons, we included the results that were obtained using alternative techniques. First, the *simple linear regression* [34] in Figure 7b, where the predicted confidence interval suffers from the same problem as GPR at the extremities. Moreover, (i) the estimated 99.7% confidence band is too wide, in the sense that for any fixed x^{obst} value on the horizontal axis in Figure 7b, we would not expect physically to collect data points v that are so far apart as the confidence band in Figure 7b suggests (however, by gathering more experimental data, the confidence band will tend to shrink, thus better modeling the physical variability); (ii) there is no physical reason why the *true* mean would follow a straight line, which is one of the underlying assumptions of a simple linear regression model. This can potentially be overcome using, for example, polynomial regression.

We also compared the results to those obtained using an interpolation method presented in [34], in Figure 7c. This captures correctly the lack of variability for the stationary situation $(v, x^{\text{obst}}) = (0, 0)$; however, it was not able to model the variability for the other experimental data points. Finally, the results were compared to those obtained using a Bayesian neural network [40]. These results are shown in Figure 7d, where the neural network has a single hidden layer containing three nodes. The *posterior mean prediction* converges toward a finite value of approximately $v = 0.83$ m/s (not shown in the figure), which is similar to the *mean function* of the GPR model in Figure 7a, which falls toward the constant value of $v = 0$ m/s. Note that the standard deviation around the posterior mean prediction in Figure 7d can be further improved by (i) adding more data points to this automatic learning procedure, or, (ii) similar to what we did in Section 3.1 and Figure 7a, by manually specifying (fixing) the *noise variance* hyperparameter according to experimental data (observations) and identifying the rest of the hyperparameters using the optimization procedure.

To conclude, the results presented in Figure 7 show that the GPR model in Figure 7a is superior to the other three stochastic models in Figure 7b–d, as it better captures the physics of the real process.

Preliminary experiments with this shared control enabled (see Figure 6b) give the feeling of a rather slow-moving robot. To make it advance faster, loosely speaking and appealing to the reader's intuition, we need to "bend" the blue curve in Figure 5 toward the left side. This can be achieved in two ways: either by reducing the desired confidence margin considered in the shared control algorithm (say from 99.7% to 95.45% or $2\sigma_{y_*}$ interval, making the algorithm less conservative), and/or by choosing a smaller threshold value Δt_{stop}, say $\Delta t_{\text{stop}} = 1$ s. The latter would involve a new set of experiments to be performed for collecting a new training set. The downside of reducing the 99.7% value is the increased chance of colliding with obstacles. The benefit, though, is the increased level of autonomy for the operator, now able to maneuver a faster moving robot.

The obstacle avoidance shared control presented in Section 2 is suited for usage in disaster scenes where the environment cannot be mapped before the deployment of robots. In other words, it does not need any prior knowledge about already mapped environments generated, e.g., using simultaneous mapping and localization (SLAM) techniques [32]. In future works, the method we presented could be extended to integrate this type of environment information together with clusters of pre-trained environment-specific GPR-based models. This family of models can then be used in conjunction with the online identified environment in selecting the most adequate GPR model to be used by the control algorithm.

Future Work Plan

In our immediate future work plan, we aim to conduct a comprehensive study involving the use of confidence metrics within a robot's control system. In that work, we will compare the proposed GPR-based method against other stochastic methods, as well as demonstrating the benefits that stochastic techniques offer over deterministic techniques.

5. Conclusions

Gaussian process regression (GPR) models are useful for their expressive power in modeling signals and their statistical properties (mean function and confidence interval). In this article, we fitted a GPR model using experimental training data, and the novelty was to use the predicted *confidence interval* to enable a better-informed, personalized obstacle-avoidance shared-control algorithm that can cope with the stochastic nature of interactions between (i) a human operator and the robot, and (ii) the robot and its environment. A proof of concept in this setting was shown using a tele-operated ground-based robotic vehicle. In future work, we intend to test this algorithm on other types of robots, e.g., as shown in Figure 4c, a drone conducting an inspection of a tall chimney or building, possibly beyond the visual line of sight where collisions are undesirable, and in Figure 4d, a co-bot arm manipulator working in collaboration with a human on the same task, such as sorting objects in a box, where collisions between the two should be avoided.

Supplementary Materials: The accompanying video, showing excerpts for data collection and testing the obstacle avoidance shared control algorithm on the SuperDroid HD2 robot can be downloaded at: https://www.mdpi.com/article/10.3390/robotics11050102/s1, Video S1: A proof of concept.

Author Contributions: Conceptualization, C.S.T.; methodology, C.S.T.; software, C.S.T.; validation, C.S.T., K.G. and B.L.; formal analysis, C.S.T. and B.L.; investigation, C.S.T.; resources, C.S.T., K.G. and B.L.; data curation, C.S.T.; writing—original draft preparation, C.S.T., K.G. and B.L.; writing—review and editing, K.G. and B.L.; visualization, K.G.; supervision, B.L.; project administration, B.L.; funding acquisition, B.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by The Engineering and Physical Sciences Research Council (EPSRC) via the NNUF-HR project EP/T011491/1. K.G. and B.L. also acknowledge the support

provided by EPSRC through RAIN (EP/R026084/1) and RNE (EP/P01366X/1). B.L. acknowledges support from the Royal Academy of Engineering (CiET1819\13).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The experimental dataset and the code used for analysis and experimentation are freely available on https://github.com/StefanT83/SC_obst_avoid_GPR_1D.git (accessed on 14 September 2022).

Acknowledgments: C.S.T. wishes to thank Michael Hellebrand from Remote Applications in Challenging Environments (RACE) for the loan of the SuperDroid HD2 robot.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ramakrishnan, N.; Bailey-Kellogg, C. Gaussian Process Models in Spatial Data Mining. In *Encyclopedia of GIS*; Shekhar, S., Xiong, H., Eds.; Springer: New York, NY, USA, 2008; pp. 325–329. [CrossRef]
2. Guo, Y.; Yu, X.; Wang, Y.; Zhang, R.; Huang, K. State-of-Health estimation of lithium-ion batteries based on thermal characteristics mining and multi-Gaussian process regression strategy. *Energy Technol.* **2022**, *10*, 2200151. [CrossRef]
3. Ruiz, A.V.; Olariu, C. A general algorithm for exploration with Gaussian processes in complex, unknown environments. In Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 26–30 May 2015; pp. 3388–3393.
4. West, A.; Tsitsimpelis, I.; Licata, M.; Jazbec, A.; Snoj, L.; Joyce, M.J.; Lennox, B. Use of Gaussian process regression for radiation mapping of a nuclear reactor with a mobile robot. *Sci. Rep.* **2021**, *11*, 1–11. [CrossRef] [PubMed]
5. Ketu, S.; Mishra, P.K. Enhanced Gaussian process regression-based forecasting model for COVID-19 outbreak and significance of IoT for its detection. *Appl. Intell.* **2021**, *51*, 1492–1512. [CrossRef] [PubMed]
6. Wang, J.M.; Fleet, D.J.; Hertzmann, A. Gaussian process dynamical models for human motion. *IEEE Trans. Pattern Anal. Mach. Intell.* **2008**, *30*, 283–298. [CrossRef] [PubMed]
7. Hewing, L.; Wabersich, K.P.; Menner, M.; Zeilinger, M.N. Learning-Based Model Predictive Control: Toward Safe Learning in Control. *Annu. Rev. Control. Robot. Auton. Syst.* **2020**, *3*, 269–296. [CrossRef]
8. Ostafew, C.J.; Schoellig, A.P.; Barfoot, T.D.; Collier, J. Learning-based nonlinear model predictive control to improve vision-based mobile robot path tracking. *J. Field Robot.* **2016**, *33*, 133–152. [CrossRef]
9. Berkenkamp, F.; Schoellig, A.P. Safe and robust learning control with Gaussian processes. In Proceedings of the 2015 European Control Conference (ECC), Linz, Austria, 15–17 July 2015; pp. 2496–2501.
10. Deisenroth, M.P.; Fox, D.; Rasmussen, C.E. Gaussian processes for data-efficient learning in robotics and control. *IEEE Trans. Pattern Anal. Mach. Intell.* **2015**, *37*, 408–423. [CrossRef] [PubMed]
11. Matschek, J.; Gonschorek, T.; Hanses, M.; Elkmann, N.; Ortmeier, F.; Findeisen, R. Learning References with Gaussian Processes in Model Predictive Control applied to Robot Assisted Surgery. In Proceedings of the 2020 European Control Conference (ECC), Saint Petersburg, Russia, 12–15 May 2020; pp. 362–367.
12. Matschek, J.; Findeisen, R. Learning Supported Model Predictive Control for Tracking of Periodic References. In Proceedings of the 2nd Conference on Learning for Dynamics and Control, Zurich, Switzerland, 11–12 June 2020; pp. 511–520.
13. Matschek, J.; Himmel, A.; Sundmacher, K.; Findeisen, R. Constrained Gaussian process learning for model predictive control. *IFAC-Pap.* **2015**, *53*, 971–976. [CrossRef]
14. Budd, M.; Lacerda, B.; Duckworth, P.; West, A.; Lennox, B.; Hawes, N. Markov Decision Processes with Unknown State Feature Values for Safe Exploration using Gaussian Processes. In Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 25–29 October 2020; pp. 7344–7350. [CrossRef]
15. Leco, M.; Kadirkamanathan, V. A perturbation signal based data-driven Gaussian process regression model for in-process part quality prediction in robotic countersinking operations. *Robot. -Comput.-Integr. Manuf.* **2021**, *71*, 102105. [CrossRef]
16. Rasmussen, C.E.; Williams, C.K.I. *Gaussian Processes for Machine Learning*; 2nd ed.; MIT Press: Cambridge, MA, USA, 2006.
17. Khosravi, M.; Behrunani, V.; Smith, R.S.; Rupenyan, A.; Lygeros, J. Cascade control: Data-driven tuning approach based on Bayesian optimization. *IFAC-Pap.* **2020**, *53*, 382–387. [CrossRef]
18. Neal, R.M. *Bayesian Learning for Neural Networks*; Springer: Berlin/Heidelberg, Germany, 1996; Volume 118.
19. Murphy, K.P. *Probabilistic Machine Learning: An Introduction*; Adaptive Computation and Machine Learning; MIT Press: Cambridge, MA, USA, 2022.
20. Santner, T.J.; Williams, B.J.; Notz, W.I. (Eds.) *The Design and Analysis of Computer Experiments*, 2nd ed.; Springer: Berlin/Heidelberg, Germany, 2018.
21. Nabney, I.T. *NETLAB: Algorithms for Pattern Recognition*; Springer: Berlin/Heidelberg, Germany, 2002.
22. de Coulon, F. *Théorie et Traitement des Signaux*, 3rd ed.; PPUR: Lausanne, Switzerland, 1996.
23. The Mathworks. *Statistics and Machine Learning Toolbox™ User's Guide*, R2022a ed.; The Mathworks: Natick, MA, USA, 2022.

24. Mendoza-Trejo, O.; Cruz-Villar, C.A. Robust concurrent design of a 2-DOF collaborative robot (Cobot). *IEEE/ASME Trans. Mechatron.* **2021**, *26*, 347–357. [CrossRef]
25. Calanca, A.; Dimo, E.; Palazzi, E.; Luzi, L. Enhancing force controllability by mechanics in exoskeleton design. *Mechatronics* **2022**, *86*, 102867. [CrossRef]
26. Shahid, A.A.; Piga, D.; Braghin, F.; Roveda, L. Continuous control actions learning and adaptation for robotic manipulation through reinforcement learning. *Auton. Robot.* **2022**, *46*, 483–498. [CrossRef]
27. Cheah, W.; Garcia-Nathan, T.; Groves, K.; Watson, S.; Lennox, B. Path Planning for a Reconfigurable Robot in Extreme Environments. In Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA), Xi'an, China, 30 May–5 June 2021; pp. 10087–10092. [CrossRef]
28. West, A.; Wright, T.; Tsitsimpelis, I.; Groves, K.; Joyce, M.J.; Lennox, B. Real-time avoidance of ionising radiation using layered costmaps for mobile robots. *Front. Robot. AI* **2022**, *9*, 862067. [CrossRef] [PubMed]
29. Ackerman, E. Boston Dynamics' Spot Is Helping Chernobyl Move towards Safe Decommissioning. IEEE Spectrum. Available online: https://spectrum.ieee.org/boston-dynamics-spot-chernobyl (accessed on 14 June 2022).
30. Spong, M.W.; Hutchinson, S.; Vidyasagar, M. *Robot Modeling and Control*, 2nd ed.; John Wiley & Sons: Hoboken, NJ, USA, 2020.
31. Craig, J.J. *Introduction to Robotics: Mechanics and Control*, 4th ed.; Pearson: London, UK, 2021.
32. Zheng, K. ROS navigation tuning guide. In *Robot Operating System (ROS)*; Springer: Cham, Switzerland, 2021; pp. 197–226.
33. Teodorescu, C.S.; Carlson, T. AssistMe: Using policy iteration to improve shared control of a non-holonomic vehicle. In Proceedings of the International Conference on Systems, Man, and Cybernetics, Prague, Czech Republic, 9–12 October 2022.
34. Murphy, K.P. *Machine Learning: A Probabilistic Perspective*; MIT Press: Cambridge, MA, USA, 2012.
35. Luckcuck, M.; Fisher, M.; Dennis, L.; Frost, S.; White, A.; Styles, D. *Principles for the Development and Assurance of Autonomous Systems for Safe Use in Hazardous Environments*; Technical Report; Zenodo, CERN: Meyrin, Switzerland, 2021. [CrossRef]
36. Rasmussen, C.E.; Nickisch, H. Gaussian Process Regression and Classification Toolbox version 4.2 for GNU Octave 3.2.x and Matlab 7.x and higher. Available online: http://www.gaussianprocess.org/gpml/code/ (accessed on 14 September 2022).
37. Gutman, B. Linear Regression Confidence Interval. MATLAB Central File Exchange. Available online: https://www.mathworks.com/matlabcentral/fileexchange/39339-linear-regression-confidence-interval (accessed on 14 September 2022).
38. Nabney, I. Netlab. MATLAB Central File Exchange. Available online: https://www.mathworks.com/matlabcentral/fileexchange/2654-netlab (accessed on 14 September 2022).
39. Klepsch, M.; Schmitz, F.; Seufert, T. Development and validation of two instruments measuring intrinsic, extraneous, and germane cognitive load. *Front. Psychol.* **2017**, *8*, 1997. [CrossRef] [PubMed]
40. Bishop, C.M. *Pattern Recognition and Machine Learning*; Springer: Berlin/Heidelberg, Germany, 2006.