sensors

MDPI

Article

Deep Learning for Efficient and Optimal Motion Planning for AUVs with Disturbances

Juan Parras *, Patricia A. Apellániz and Santiago Zazo

Information Processing and Telecommunications Center, E.T.S. Ingenieros de Telecomunicación, Universidad Politécnica de Madrid, 28040 Madrid, Spain; patricia.alonsod@upm.es (P.A.A.); santiago.zazo@upm.es (S.Z.)
* Correspondence: j.parras@upm.es

Abstract: We use the recent advances in Deep Learning to solve an underwater motion planning problem by making use of optimal control tools—namely, we propose using the Deep Galerkin Method (DGM) to approximate the Hamilton–Jacobi–Bellman PDE that can be used to solve continuous time and state optimal control problems. In order to make our approach more realistic, we consider that there are disturbances in the underwater medium that affect the trajectory of the autonomous vehicle. After adapting DGM by making use of a surrogate approach, our results show that our method is able to efficiently solve the proposed problem, providing large improvements over a baseline control in terms of costs, especially in the case in which the disturbances effects are more significant.

Keywords: optimal control; Hamilton–Jacobi–Bellman equation; autonomous underwater vehicle; Deep Galerkin Method

Citation: Parras, J.; Apellániz, P.A.; Zazo, S. Deep Learning for Efficient and Optimal Motion Planning for AUVs with Disturbances. *Sensors* **2021**, *21*, 5011. https://doi.org/10.3390/s21155011

Academic Editors: Joaquin Aranda and Ernesto Aranda-Escolastico

Received: 15 June 2021
Accepted: 20 July 2021
Published: 23 July 2021

1. Introduction

The problem of motion planning for Autonomous Underwater Vehicles (AUVs) is to choose the best trajectory that satisfies a set of constraints, such as the maximum acceleration and velocity of the AUV. Among all trajectories that satisfy these constrains, the best trajectory is the one that optimizes a certain metric, such as time to reach a certain target or computation time. Traditionally, there have been two ways to address the problem of motion planning for AUVs: the field of robotics, which pays attention specially to computational issues and real-time control, and the field of optimal control, which emphasizes the optimization of the trajectory regarding a certain performance measure [1].

Even though the optimal control approach is very attractive, as it is able to return the best trajectory that fulfills the constrains regarding the performance metric chosen, it suffers from the so called "curse of dimensionality": as the dimensionality of the problem grows (i.e., the number of state variables of the AUV), the computational resources required to use optimal control tools become too high. Even though there are exceptional cases in which the solution is known, as the case of linear quadratic problems [2], the use of optimal control tools in general leads to an intractable problem. As optimal control applies to many problems of interest in different fields, a lot of efforts have been addressed to find efficient ways to solve, in an exact or approximate way, optimal control problems, such as [3–8]. Some of these works are still unable to scale up to practical problems, and others require a specific structure of the problem. However, the recent advances in the deep learning field have brought a novel approach to solve optimal control problems, based on the Deep Galerkin Method (DGM, [9]) that can be used to solve optimal control problems without being subject to the "curse of dimensionality" [10] and is becoming popular due to its competitive results [11–13].

In this work, we make use of DGM to develop a method that is able to address motion planning problems for AUVs using optimal control tools. The use of neural networks to address motion planning problems for ocean navigation is not new, as [14] already makes use of such tools to approximate the solution of a Kalman filter for a surface ship navigation

problem. In [15], the authors use optimal control tools for underwater navigation, but they have to use a restricted mesh of discrete states to avoid the curse of dimensionality (as in [16] for surface vehicles). A recent work close to ours is [17], where the authors use optimal control tools for mine detection missions. They use autonomous vehicles that move in the ocean surface, and seek to obtain optimal trajectories in the sense that they maximize the mine detection probability using different detection sensor models. However, they use coverage path planning, which consists of solving the optimal control problem in a discrete environment, and then check whether the discrete solution can be applied to a continuous system. A similar procedure is described in [18], where the authors first find a discrete solution and then interpolate it. In contrast to these works, which rely on discretization of the states, we solve the problem in the continuous domain, and also use a different target function, as we intend to minimize the time to reach a certain target. Another popular approach is based on using Model Predictive Control (MPC), which can be used to approximate nonlinear control problems in an online or offline fashion [19–22]. MPC differs from our approach in that we obtain the optimal solution by obtaining first the value function and also our approach may deal with continuous time without having to use discretization.

A problem that arises in the underwater medium is localization: as GPS does not reach the ocean depths, it is necessary to develop other location systems. This topic is subject to intense research, as shown in [23], as we need accurate localization mechanisms to implement optimal trajectories. However, this paper is focused on obtaining optimal trajectories, and hence, we will consider that there is a localization mechanism in the AUV such that its position and velocities are known (i.e., fully observable).

The main contributions of our work are the following:

1. We solve a motion planning problem for an AUV using computationally efficient tools, namely, DGM, which solves a continuous time and states nonlinear optimal control problem. This tool will prove efficient computationally, and departs from numerous approaches that rely on state discretization;
2. The optimal control problem we focus on consists of reaching a target position as soon as possible. As this problem cannot be solved directly using DGM, as DGM required a fixed time horizon, we develop a surrogate problem that can be solved using DGM and returns an equivalent control to our original control problem. This supposes an advance, as it extends the range of uses of DGM to problems with an unknown time horizon;
3. We take into account the effect of several disturbances in the computation of our optimal trajectory. The presence of disturbances is frequent in the ocean, where currents and swirls affect the motion of the AUV. We model some disturbances and show that DGM is able to obtain optimal trajectories that take into account the effect of the disturbance.

In short, we make use of the recent advances in deep learning to efficiently solve the optimal control problem applied to AUV motion planning in the presence of disturbances. The rest of this paper goes as follows: Section 2 presents our setup and the disturbances that we use in this work. Then, Section 3 presents the optimal control tools that we use, and shows how it is possible to use DGM to solve a surrogate optimal control problem for motion planning efficiently. Our ideas are validated via simulations in Section 4, and finally, we draw some conclusions in Section 5.

2. Setup Description

2.1. Underwater Navigation Model

We consider an AUV that moves in the plane (i.e., constant depth), where the AUV position is the vector (x, y), its velocity is (v_x, v_y), and we consider that the control variable is the acceleration angle θ. If we assume that the disturbances affect the acceleration and are modeled by the vector (p_x, p_y), and if we model the friction by a term that depends on

the velocity and a parameter k_f, the motion is controlled by the next Ordinary Differential Equation (ODE) system:

$$\begin{aligned} \dot{x}(t) &= v_x(t) \\ \dot{y}(t) &= v_y(t) \\ \dot{v_x}(t) &= \cos(\theta(t)) + p_x(x(t), y(t)) - k_f \cdot v_x(t) \\ \dot{v_y}(t) &= \sin(\theta(t)) + p_y(x(t), y(t)) - k_f \cdot v_y(t) \end{aligned} \tag{1}$$

where $\dot{x}$ represents the derivative with respect to time of the variable x. Note that k_f limits the maximum velocity that the AUV can reach (see the Isotropic Rocket problem [24]). Additionally, it can be observed that we consider that the disturbances depend only on the AUV position. Finally, note that we choose to solve a planar problem because we can plot the results in a meaningful way, but all our developments can be easily extended to deal with 3-D models as well. It is important to note that the procedure we describe in this work could be applied to a different modeling of the problem; for instance, if the friction came from pressure drag, it would be proportional to the square of velocity.

2.2. Disturbance Models

We model three disturbances that appear in the underwater environment: swirls, currents and constant fields. The constant field is a fixed disturbance model that uses two scalar parameters $a \in \mathbb{R}$ and $\alpha \in [0, 2\pi)$, where the absolute value of a controls the strength of the field and α its orientation as:

$$\begin{aligned} p_x &= a \cdot \cos(\alpha) \\ p_y &= a \cdot \sin(\alpha) \end{aligned} \tag{2}$$

A swirl vector field can be modeled using three scalar parameters, $b \in \mathbb{R}$, $x_0 \in \mathbb{R}$ and $y_0 \in \mathbb{R}$, as:

$$\begin{aligned} p_x &= b \cdot \frac{y - y_0}{\sqrt{(x - x_0)^2 + (y - y_0)^2}} \\ p_y &= b \cdot \frac{-x + x_0}{\sqrt{(x - x_0)^2 + (y - y_0)^2}} \end{aligned} \tag{3}$$

where x_0 and y_0 control the location of the vortex of the swirl, the absolute value of b controls the strength of the swirl and the sign of b controls whether the swirl rotates clock or counterclockwise.

In this work, for simplicity, we consider only horizontal currents, which we model using three scalar parameters, $c \in \mathbb{R}$, $d \in \mathbb{R}$ and $y_0 \in \mathbb{R}$, as follows:

$$\begin{aligned} p_x &= c \cdot e^{-\frac{(y - y_0)^2}{d^2}} \\ p_y &= 0 \end{aligned} \tag{4}$$

where y_0 locates the maximum current strength y-coordinate, the absolute value of d indicates the breadth of the current, the absolute value of c controls the strength of the current and the sign of c controls whether the current direction is positive or negative.

3. Optimal Control Motion Planning

An Optimal Control Problem (OCP) is an extension of the optimization problem to the case in which time is involved, and we want to obtain a trajectory that is optimal in a certain sense. In our case, a typical OCP is to determine the control trajectory (i.e., the value of the acceleration as a function of time) that minimizes the time to reach a certain position goal. There are several tools that could be used to solve such an OCP, depending

on whether the time is discrete or continuous, and the literature on the topic is extensive (see [2,25–27] and their references). In this work, we will work using continuoustime OCPs.

3.1. Continuous Time Optimal Control

Let us assume the following OCP formulation [2] p. 104:

$$\begin{aligned} \min J(s(t), u(t)) &= K\Big((s_{t_f}), t_f\Big) + \int_0^{t_f} L(s(t), u(t), t)dt \\ s.t. \quad & \frac{ds}{dt} = f(s(t), u(t), t) \\ & s(0) = s_0 \\ & \Psi(s(t_f), t_f) = 0 \\ & u(t) \in U, \quad \forall t \in \Big[t_0, t_f\Big] \end{aligned} \tag{5}$$

where we have that:

- $t \in [0, t_f], \quad t \in \mathbb{R}$ is the time, where t_f is the final time;
- $s(t) \in \mathbb{R}^m$ is the state trajectory, where $s(t)$ is the state at time t;
- $u(t) \in \mathbb{R}^l$ is the control trajectory, where $u(t)$ is the control at time t. The controls belong to the set of admissible controls U;
- $J : \mathbb{R}^m \times \mathbb{R}^l \times \mathbb{R} \to \mathbb{R}$ is the cost functional to be minimized, that is formed by a terminal cost functional $K : \mathbb{R}^m \times \mathbb{R} \to \mathbb{R}$ and a running cost functional $L : \mathbb{R}^m \times \mathbb{R}^l \times \mathbb{R} \to \mathbb{R}$;
- The transition function $f : \mathbb{R}^m \times \mathbb{R}^l \times \mathbb{R} \to \mathbb{R}^m$ controls the state evolution as a function of t, $s(t)$ and $u(t)$;
- $s_0 \in \mathbb{R}^m$ is the initial state;
- $\Psi : \mathbb{R}^m \times \mathbb{R} \to \mathbb{R}^r$ is the final condition that must hold at t_f.

Since we consider that the transition function f is deterministic, if we have the initial conditions and a control trajectory, we can obtain the state trajectory and the total cost by integrating (5). Note that we consider that time is a continuous variable, thus the name of continuous time OCP; it is also possible to work using a discrete time OCP, where some important changes from (5) are that integrals are replaced by sums, and differential equations by difference equations, as shown in [2].

3.1.1. Dynamic Programming Methods for Continuous Time

There are two classical approaches to solve the OCP (5): the minimum principle and dynamic programming methods [2], where the former provides necessary conditions and the latter, sufficient conditions. We use a dynamic programming approach, which is based on the concept of value function $V(s, t)$, which is a continuous function that returns the optimal cost that can be obtained in state s at time t. The function V is obtained by solving a nonlinear first order Partial Derivative Equation (PDE) as Theorem 1 states, Section 3.3.5 in [2]:

Theorem 1 (Dynamic Programming: HJB)**.** *The sufficient conditions that a continuous and unique function $V(s, t)$ has to satisfy in order to be the optimal solution to the control problem* (5) *are the following two expressions, known as the Hamilton–Jacobi–Bellman (HJB) equation:*

$$\begin{aligned} 0 &= \frac{\partial V}{\partial t} + \min_{u \in U} \left[\frac{\partial V}{\partial s}^T f(s, u, t) + L(s, u, t) \right] \\ V\Big(s(t_f), t_f\Big) &= K\Big(s(t_f), t_f\Big) + \nu^T \Psi\Big(s(t_f), t_f\Big) \end{aligned} \tag{6}$$

Unfortunately, solving (5) using the HJB Equation (6) is, in many cases, intractable (a notable exception being the linear quadratic case, which has a closed solution based on the Ricatti equations, Section 3.5 in [2]). First, note that (6) involves solving a nonlinear PDE

due to the minimization operator which need not have a classic solution, i.e., an everywhere continuously differentiable V function. As noted by [28], we may think of admitting weaker solutions, that is, V functions that are continuous but not everywhere differentiable.

Based on this idea, Crandall and Lions proposed the viscosity solutions [29]. The key idea is that a viscosity solution to the HJB equation is a continuous but not everywhere differentiable V function. The derivative of V at points in which the viscosity solution is not differentiable is the derivative of a smooth function which touches V. Some important characteristics of the viscosity solutions are [28]: (1) the viscosity method allows selecting a single weak solution and (2) the optimal value function is the single viscosity solution to the HJB equation. Hence, finding the viscosity solution to the HJB equation allows obtaining V, and thus, it is no surprise that a large research has been addressed to these viscosity solutions, as [28–35], to mention some.

Another key advance consists of using numerical approximations that are based on discretizing the state space that, in the limit, converge to the viscosity solution of the HJB [28]. The seminal paper in this area is [36], where the authors conclude that any approximation method that satisfies the properties of monotonicity, stability and consistency is guaranteed to converge in the limit to the viscosity solution. A popular approach that is based on this framework is the finite-difference upwind method [3,4], which is based on the following ideas:

- The state space is discretized. Note that this means that we may face the curse of dimensionality if n is large;
- The value function is estimated iteratively by approximating the derivative with respect to the state by using finite differences;
- Depending on the state drift, for each state, the derivative is approximated using the backwards or the forward finite difference approximation.

This method is shown to satisfy the three properties of monotonicity, stability and consistency in [5], and hence, it converges to the viscosity solution in the limit. Two useful sources to implement these methods are [4], where several schemes and examples are described, and [5], where the upwind scheme algorithm is thoroughly described. The main problem that the upwind method faces is that it suffers from the curse of dimensionality, and hence this method is generally not useful to solve problems with a state dimension higher than 3, as the problem (1).

There are several other methods proposed to solve the HJB equation in Theorem 1, such as the ones based on level-set method and semi-Lagrangian schemes [6], which, however, are also based on a discrete grid and, hence, subject to the curse of dimensionality. Several methods have been proposed to alleviate this problem and scale to large state spaces, such as [7,8], which require obtaining mathematical expressions that link the original state space with a reduced state space, where the HJB equation is solved. Finally, a very promising method makes use of the recent deep learning advances in order to solve the HJB approximately without being subject to the curse of dimensionality: the method is known as deep Galerkin method (DGM) [9,10], which we now proceed to explain.

3.1.2. Deep Galerkin Method

Let us first assume that we know the optimal control u^* that minimizes the second term of the first equation from (6). Hence, in this case, the PDE we have to solve is:

$$\begin{aligned} 0 &= \frac{\partial V}{\partial t} + \frac{\partial V}{\partial s}^T f(s, u^*, t) + L(s, u^*, t) \\ V\Big(s(t_f), t_f\Big) &= K\Big(s(t_f), t_f\Big) + v^T \Psi\Big(s(t_f), t_f\Big) \end{aligned} \tag{7}$$

DGM approximates $V(s,t)$ using a deep neural network (DNN), whose inputs are the state vector s and the scalar t. By means of using the backpropagation algorithm, it is possible to obtain the exact gradients $\frac{\partial V}{\partial s}$ and $\frac{\partial V}{\partial t}$. The DGM takes advantage of this to approximate the PDE using batches of samples: given the state and time space, we sample

two sets of points: a set of interior points (s,t) that belong to the interior of the state and time space and a set of terminal points (s,t_f). The set of interior points is used to minimize the following training loss:

$$\left(\frac{\partial V}{\partial t}+\frac{\partial V}{\partial s}^T f(s,u^*,t)+L(s,u^*,t)\right)^2 \tag{8}$$

where we note that (8) is the squared error of the first term from (7): as (8) approaches 0, the first term of (7) is approximated. Then, the set of terminal points is used to train the NN in order to fulfil the final condition from (7) as:

$$\left(V\left(s(t_f),t_f\right)-K\left(s(t_f),t_f\right)+\nu^T\Psi\left(s(t_f),t_f\right)\right)^2 \tag{9}$$

where again, as (9) approaches 0, the final condition from (7) is approximated. Thus, the DGM approximately provides a solution for (7) by minimizing the sum of the loss terms (8) and (9). As the loss sum approaches 0, the DNN is the approximation of the optimal value function. Note that a very important property of the DGM is that is a meshless method: we do not train the DNN using a mesh over the state space, but samples, and let the DNN generalize to the states that have never been seen before. As shown by [9], the DGM is able to solve PDEs in very high dimensional spaces without being subject to the curse of dimensionality; due to this significant advantage, we used the DGM in this work.

However, the original DGM method, proposed in [9], assumed that the optimal control u^* is known, which need not be the case in all control problems. In order to overcome this problem, ref. [10] proposed a modification of the DGM known as DGM-PI (Deep Galerkin Method—Policy Iteration), in which a second DNN is used to approximate the optimal policy function. In our problem, however, we do know the optimal control u^*, so we can use the standard DGM.

3.2. Continuous Time Surrogate Control

The continuous time OCP that we want to solve is the problem of minimizing the total time that it takes to an AUV to reach the origin. Hence, in our OCP, $K=0$, $L=1$, the final position must be the origin, the transition model is (1) and the control is the acceleration angle; thus, the OCP we want to solve is:

$$\begin{aligned}
&\min J = t_f \\
&s.t. \quad \dot{x}(t)=v_x(t), \quad \dot{y}(t)=v_y(t) \\
&\qquad \dot{v_x}(t)=\cos(\theta(t))+p_x(x(t),y(t))-k_f\cdot v_x(t) \\
&\qquad \dot{v_y}(t)=\sin(\theta(t))+p_y(x(t),y(t))-k_f\cdot v_y(t) \\
&\qquad s(0)=\left(x(0),y(0),v_x(0),v_y(0)\right) \\
&\qquad x(t_f)=y(t_f)=0, \quad \theta(t)\in[0,2\cdot\pi), \quad \forall t\in\left[0,t_f\right]
\end{aligned} \tag{10}$$

and the HJB Equation (6) for this OCP is:

$$\begin{aligned}
0=\frac{\partial V}{\partial t}+\min_\theta\Bigg[&\frac{\partial V}{\partial x}\cdot v_x(t)+\frac{\partial V}{\partial y}\cdot v_y(t) \\
&+\frac{\partial V}{\partial v_x}\cdot\Big(\cos(\theta)+p_x(x(t),y(t))-k_f\cdot v_x(t)\Big) \\
&+\frac{\partial V}{\partial v_y}\cdot\Big(\sin(\theta)+p_y(x(t),y(t))-k_f\cdot v_y(t)\Big)+1\Bigg] \\
V\left(x,y,v_x,v_y,t_f\right)=0&
\end{aligned} \tag{11}$$

3.3. Surrogate OCP for Continuous Time

However, we cannot solve (10) using DGM directly, as DGM assumes a fixed final time t_f, and in (10), t_f is the minimization objective, and hence it is not fixed but free. Instead, we propose the following surrogate OCP

$$\begin{aligned} \min J = & \int_0^{t_f} \tanh\left(\sqrt{x(t)^2 + y(t)^2}\right) dt \\ s.t. \quad & \dot{x}(t) = v_x(t), \quad \dot{y}(t) = v_y(t) \\ & \dot{v_x}(t) = \cos(\theta(t)) + p_x(x(t), y(t)) - k_f \cdot v_x(t) \\ & \dot{v_y}(t) = \sin(\theta(t)) + p_y(x(t), y(t)) - k_f \cdot v_y(t) \\ & s(0) = \left(x(0), y(0), v_x(0), v_y(0)\right) \\ & \theta(t) \in [0, 2\pi), \quad \forall t \in \left[0, t_f\right] \end{aligned} \tag{12}$$

where the main changes are the following ones:

- The problem now has a fixed terminal time t_f, which is required by the DGM;
- As t_f is fixed, we must change the functional J. Recall that our target was to move the AUV as close as possible to the origin; we achieve this by using, as a running cost, a hyperbolic tangent function that depends on the distance of the AUV to the origin at each time, which is $\sqrt{x(t)^2 + y(t)^2}$. Thus, note that $L \to 1$ when the AUV is far from the origin, and as the AUV approaches the origin, $L \to -1$. In other words, our surrogate cost functional penalizes for being far from the origin, and rewards positions of the AUV that are as close as possible to the origin.

  Thus, the HJB Equation (6) for the surrogated OCP is:

$$\begin{aligned} 0 = & \frac{\partial V}{\partial t} + \min_\theta \left[\frac{\partial V}{\partial x} \cdot v_x(t) + \frac{\partial V}{\partial y} \cdot v_y(t) \right. \\ & + \frac{\partial V}{\partial v_x} \cdot \left(\cos(\theta) + p_x(x(t), y(t)) - k_f \cdot v_x(t)\right) \\ & + \frac{\partial V}{\partial v_y} \cdot \left(\sin(\theta) + p_y(x(t), y(t)) - k_f \cdot v_y(t)\right) \\ & \left. + \tanh\left(\sqrt{x(t)^2 + y(t)^2}\right) \right] \\ V & \left(x, y, v_x, v_y, t_f\right) = 0 \end{aligned} \tag{13}$$

As we have mentioned in Section 3.1.2, in order to use DGM to solve (13), we need to solve the minimization problem in the PDE. Note that the minimization problem is the following, which we obtain by dropping all terms that do not depend on θ

$$\min_\theta \left[\frac{\partial V}{\partial v_x} \cdot \cos(\theta) + \frac{\partial V}{\partial v_y} \cdot \sin(\theta) \right] \tag{14}$$

By obtaining the derivative of (14) and equalling to zero, we obtain the candidate points to be a minimum:

$$\theta^*_{DGM} = \left\{ \arctan\left(\frac{\frac{\partial V}{\partial v_x}}{\frac{\partial V}{\partial v_y}} \right), \arctan\left(\frac{-\frac{\partial V}{\partial v_x}}{-\frac{\partial V}{\partial v_y}} \right) \right\} \tag{15}$$

where the sign of each fraction determine the quadrant of the angle. Note that there are two candidate angles: the first one corresponds to the minimization solution, in which the

AUV travels towards the origin, and the second corresponds to the maximization solution, in which the AUV tries to separate from the origin. Also, observe that we constrain the controls to lie in the set $\theta \in [0, 2\pi)$ (as shown in (10) and (12)), and this constrain is naturally enforced by using the arctan operation. Thus, we now can solve the surrogate OCP (12), by making use of DGM to approximate the value function that satisfies the HJB PDE (13).

4. Empirical Simulation

In this section, we extensively study the results obtained by making use of the OCP tools explained in Section 3 applied to the setup described in Section 2. The steps we follow are:

- First, we explain the setup we use for our simulations in Section 4.1;
- Then, we train DGM and obtain the optimal control for the surrogate problem. We study the value function and control functions obtained, as well as the training convergence, in Section 4.2;
- Afterwards, we study the performance of DGM in the surrogate problem (12) in Section 4.3;
- Finally, we intensively study how the control obtained in the surrogate problem applies to our original problem, i.e., the time minimization problem (10), in Section 4.4.

4.1. Simulation Setup

As mentioned, we validate our ideas by solving the surrogate OCP (12) using the DGM. For all our simulations, we use the following discretized version of (1):

$$\begin{aligned} x_{n+1} &= x_n + \Delta \cdot v_{x,n} \\ y_{n+1} &= y_n + \Delta \cdot v_{y,n} \\ v_{x,n+1} &= v_{x,n} + \Delta \cdot \Big(\cos(\theta_n) + p_x(x_n, y_n) - k_f \cdot v_{x,n} \Big) \\ v_{y,n+1} &= v_{y,n} + \Delta \cdot \Big(\sin(\theta_n) + p_y(x_n, y_n) - k_f \cdot v_{y,n} \Big) \end{aligned} \tag{16}$$

where n denotes time indexes, $\Delta = 0.01$ s is the time step, and we set the friction parameter $k_f = 0.5$. Note that we solve the OCP using continuous time tools, but we simulate using a discrete time setup: for values small enough of Δ, the error introduced by the discretization will be negligible.

As disturbances, we used the three models introduced in Section 2.2: a constant field disturbance following (2), with parameters $(a, \alpha) = (1/2, \pi/4)$; a swirl following (3), with parameters $(b, x_0, y_0) = (1/2, 5, 1)$; and a current following (4), with parameters $(c, y_0, d) = (1, 5, 3)$.

Finally, we compared the results obtained by DGM with the following baseline control $\theta_b(t)$, which consists of always accelerating towards the origin:

$$\theta_b(t) = \arctan\left(\frac{-y(t)}{-x(t)} \right) \tag{17}$$

Note that the baseline is an intuitive control, which does not take into account the velocities nor the disturbances. Yet, if the velocities have a small magnitude and the the disturbances have a small effect, this baseline will provide good results, which we try to improve by using the DGM.

4.2. Training Results

We proceeded to train the DGM to solve the OCP (12), that is, we used the DGM to solve the HJB (13). We followed the DGM implementation used in [10]. For each of the three disturbances we studied, we used a three layer DGM neural network, where each layer has 50 nodes. The architecture can be seen in Figure 1. We trained the DGM during 10^4 epochs; in each epoch, we randomly sampled 10^4 interior points to minimize the loss (8) and other

10^4 terminal points to minimize (9). Then, these points were used to minimize both losses during 10 iterations, and then, we started another epoch, where we sampled new points. As optimization algorithm, we used Adam [37], as it is a widely used minimization algorithm to train neural networks. Each time that we sampled state points, we followed a uniform distribution where $x(t), y(t) \in [-10, 10]$ m, and $v_x(t), v_y(t) \in [-2, 2]$ m/s. Finally, note that we set $t_f = 10$ s.

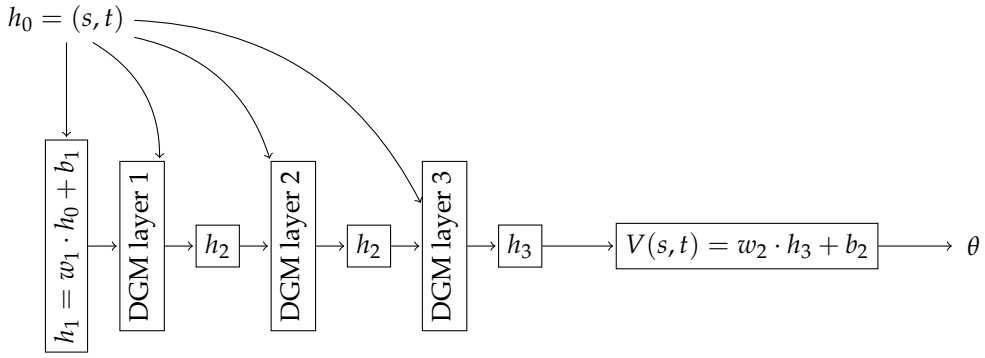
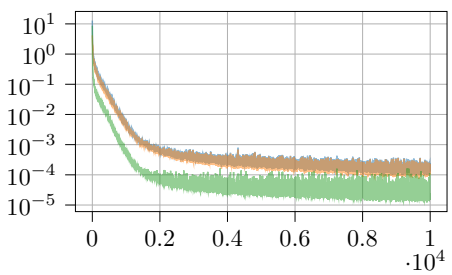
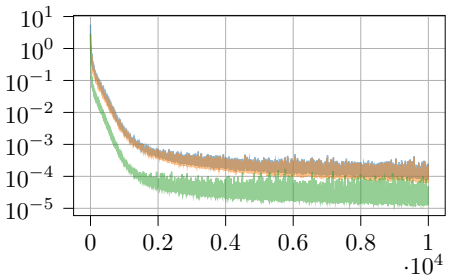


Figure 1. Block diagram showing the DGM architecture used in this work. Each DGM layer follows the implementation detailed in [10], where w_i and b_i denote weights and biases of the initial and final feed-forward layers. Each DGM layer is similar to an LSTM layer, as detailed in [10], and we used a dimension of 50 nodes for each of the 3 DGM layers. Note that the input is the pair (s, t) and the output of the neural network is the value function $V(s, t)$. We obtained the control θ using (15). After convergence, both the value function and the control approximated the optimal ones.

With these parameters, DGM converges, as shown in Figure 2, where we can see that both losses are minimized during training and reach low values. This means that we obtained a neural network that estimates both the value function $V(s, t)$ and the optimal control $\theta^*(t)$, and both can be observed in Figures 3 and 4 for the concrete cases when $v_x = v_y = 0$. First, in Figure 3, we observed the value function as a function of $(x, y, v_x = 0, v_y = 0, t = 0)$, where we note that we have to particularize the values of v_x, v_y and t to obtain a 3-D plot. As expected, each value function is different, reflecting the influence of the disturbance. For instance, by comparing the value function of the constant disturbance in Figure 3 and the disturbance effect from Figure 5, we can observe how the cost is smaller as x and y decrease, as the disturbance pushes the AUV towards the origin, while the cost increases as x and y increase, as the disturbance pushes the AUV far from the origin.

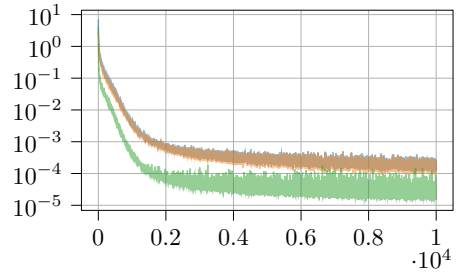


(**a**) Swirl



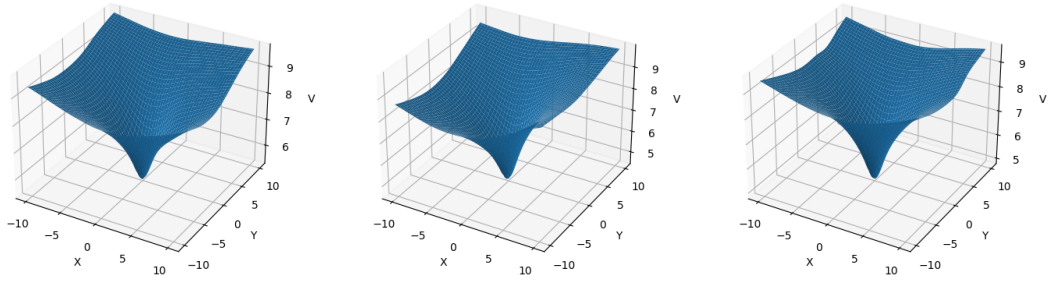
(**b**) Constant

Figure 2. *Cont.*



(**c**) Horizontal current

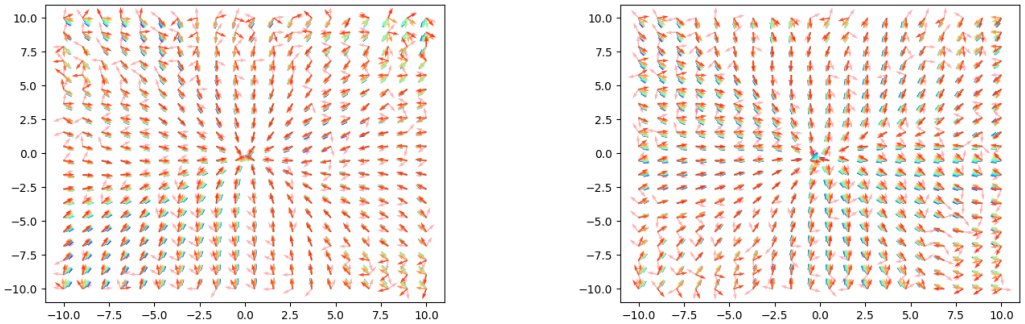
Figure 2. Training losses of DGM for the surrogated OCP solution (13): the horizontal axis represents the training epoch, while the vertical axis represents the losses (lower is better). We represent the total loss, which is the sum of the interior loss (8) and the terminal loss (9): as the latter is around one order of magnitude smaller than the former, the total loss is dominated by the interior loss. Note how all losses are minimized as the training advances, which means that DGM converges to a solution of the HJB equation.



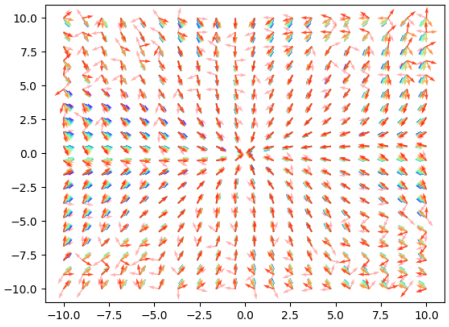
(**a**) Swirl. (**b**) Constant. (**c**) Current.

Figure 3. Value function obtained by DGM for each disturbance, where the horizontal axes represent x and y, respectively, and considering that $v_x = v_y = 0$. The value plot represent the value function $V(s,t)$ that DGM estimates as the solution to the HJB Equation (13) at $t = 0$; note that each disturbance model yields a different value function, where we can see the shape of the disturbance (compare with the disturbances shown in Figure 5).

Figure 4 shows the optimal control obtained using the DGM, where the control is obtained by following (15). Note that the optimal control depends on the derivative of the value function; hence, it may suffer from noise if the value function estimate is not good. We observe several interesting points in this figure: First, note how the optimal control obtained is very similar to the baseline control (17), as the control tends to be pointing towards the origin. Second, note how there are variations in the control depending on the disturbance; this is expected, as the DGM is able to obtain optimal control functions for each concrete disturbance. Finally, note that when we are far from the origin, the quality of the control obtained by DGM worsens, as it may indicate to accelerate in directions that separate the AUV from the origin. This is due to the fact that the cost functional J in (12) depends on the distance to the origin, and note that we sample uniformly in x and y, which in turn means that the distribution of points used to train DGM is not constant with the distance, as lower distances will have more data. Hence, what we observe is reasonable: larger distances present worse estimations, as they have appeared less frequently during training.



(**a**) Swirl. (**b**) Constant.



(**c**) Current.

Figure 4. Control obtained by DGM for each disturbance, where the horizontal and vertical axes represent x and y, respectively, and considering that $v_x = v_y = 0$. Note that these plots represent the optimal control obtained using (15), which depends on the gradient of the value function from Figure 3. We represent the control for different values of t, where cold colors represent $t \to 0$ and warm colors $t \to t_f$. We observe that the quality of the control obtained by DGM worsens slightly as the distance from the origin increases, as mentioned.

4.3. Surrogate Problem Results

Now, we proceed to compare the results obtained by the control obtained by the DGM and the baseline control. In order to benchmark the performance of both controls, we set 100 initial states $s_k(0)$, where $k \in \{1, 2, 3, \ldots, 100\}$, where each initial state is formed by an initial velocity $v_x(0), v_y(0) \in [-2, 2]$ m/s, and an initial position $x(0), y(0) \in [-8, 8]$ m. Note that the initial velocity is sampled in the same limits that were used to train the DGM, but the initial position is sampled using reduced limits to avoid the points in which DGM obtained a bad estimation of the control, as explained in the previous section.

We define the cost error as follows:

$$e = \sum_{k=1}^{100} e_k = \sum_{k=1}^{100} (J(s_k(0), \theta_{DGM}(t)) - J(s_k(0), \theta_b(t))) \tag{18}$$

where e is the error between the total costs of the DGM control $\theta_{DGM}(t)$ and the baseline control $\theta_b(t)$, averaged using the 100 different initial states $s_k(0)$. Note that as we are dealing with a minimization problem, $e \leq 0$ means that the DGM obtains better results than the baseline (i.e., a lower cost—that is, the DGM is better), and $e > 0$ means that the baseline returns a trajectory with smaller cost (i.e., the baseline is better). Note that we use the cost functional J from (12).

The results obtained can be seen in Table 1, where we represent both the error e following (18) and the proportion of trajectories in which the DGM provides a lower

cost than the baseline, i.e., the improvement proportion of trajectories. Note that in both metrics, the DGM provides considerably better results than our baseline: the error is always negative, which means that the DGM is better than the baseline, and then, the improvement proportion of DGM is higher than 80% across all disturbances. Note that this is an important result, as we have trained the same architecture of DGM and it has obtained good results for all the disturbances tested; this implies that the DGM may generalize and work with many other types of disturbances, which makes it a very powerful and flexible approach.

Table 1. Results for the surrogate OCP (12), comparing DGM and the baseline control. DGM provides better results, not only in terms of lower cost and average error (remember that negative error values mean that DGM is better), but also in that more than 80% of the trajectories present a lower cost with DGM.

Disturbance	Swirl	Current	Constant
Average J DGM	5.26	4.26	5.12
Average J baseline	6.97	4.64	6.95
e	−1.70	−0.38	−1.83
Improvement proportion	0.84	0.81	0.83

4.4. Original Problem Results

The results from the previous section were expected, as we trained the DGM to approximate the solution of the HJB (13). Now, it is time to evaluate whether our surrogate problem is effective in solving our original OCP (10). In order to evaluate this, we used the same 100 initial conditions from the previous section to evaluate the cost that yield both the baseline and DGM controls. Note that the key difference with the previous section is that now, instead of evaluating the cost using the cost functional from (12), we use the cost functional J from (10). Additionally, the original problem finished when the origin was reached; in our case, as we implemented the discrete version (16), we were satisfied that the origin was reached when the distance to the origin was smaller than 0.5 m.

The results obtained are in Table 2 and Figure 5. First, in Table 2, we represent both the error e following (18) and the proportion of trajectories in which the DGM provides a lower cost than the baseline, i.e., the improvement proportion of trajectories. Note that, in both metrics, the DGM provides considerably better results than the baseline: first, the error is always negative, which means that the DGM is better than the baseline, and then, the improvement proportion of DGM is of 87% for all disturbances tested.

Table 2. Results for the original OCP (10), comparing DGM and the baseline control. The DGM provides better results, not only in terms of lower cost and average error (remember that negative error values mean that the DGM is better), but also in that more than 85% of the trajectories present a lower cost with the DGM.

Disturbance	Swirl	Current	Constant
Average J DGM	5.45	4.75	5.63
Average J baseline	7.68	5.55	7.81
e	−2.25	−0.80	−2.17
Improvement proportion	0.87	0.87	0.87

Finally, in Figure 5, we plot several trajectories for each disturbance and control law. Observe that the subtle differences in controls between the baseline (17) and the control obtained by DGM (see Figure 4) are translated into significant differences in terms of trajectories, and hence, in costs. DGM is able to adapt to the disturbance, and specially in regions where the effect of the disturbance is strong, as close to the swirl or current centers, its advantage over the baseline is clearly seen, as it is able to obtain a control law that

minimizes the time to reach the origin. As we mentioned, in regions where the disturbance effect is small, the baseline control is good, and the DGM provides similar values to it.

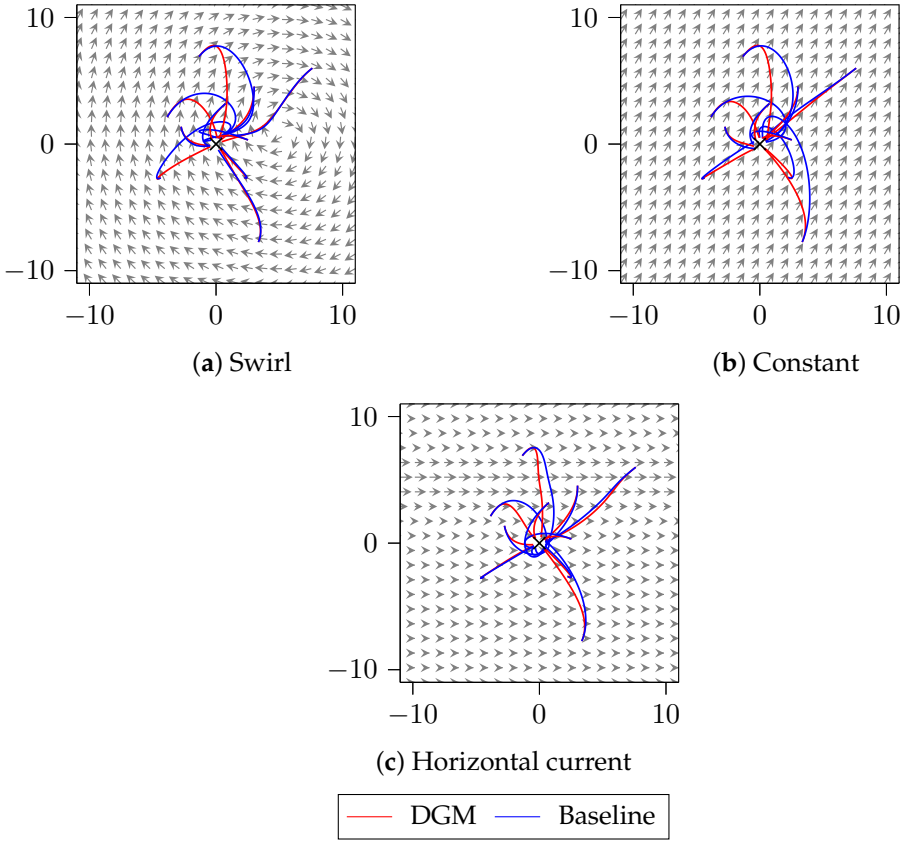


Figure 5. Example trajectories for each control law and disturbance, where the horizontal axis denotes the x coordinate, and the vertical axis is y. The vector field represents the disturbance, the red trajectories are obtained using DGM and the blue trajectories are obtained with the baseline control (17). Observe how DGM trajectories differ the most as the disturbance effect increases, thus giving a strong advantage to DGM over the baseline when the disturbance effect is not negligible.

5. Conclusions

In this work, we propose solving the OCP for navigation by means of using the DGM, an efficient meshless method that allows approximating the solution of the HJB equation making use of the recent advances in deep learning in an efficient way. Since optimal navigation problems generally are formulated in terms of minimizing the time in which a certain target is reached, we propose a novel approach in which we use a surrogate problem to transform this problem into a fixed-horizon problem, which we can solve using the DGM. When we test in the presence of disturbances, we note that DGM offers strong results compared to the baseline consisting in accelerating towards the origin, as it provides significant improvements in terms of cost, specially when the effect of the disturbances is not negligible.

There are several future lines of work that may arise from this work. First, it would be interesting checking how the DGM behaves when using different disturbances. Even though in this work we have tested using three different disturbance models, which are swirls, currents and constant disturbances, it would be interesting checking whether the DGM works on a wider set of disturbances and disturbances parameters. Another line of work could be trying to address the border effects that appeared in the control in Figure 4; one possibility would be sampling during training using a uniform distribution taken from Polar coordinates rather than Cartesian, so that the DGM is trained uniformly on the distances to the origin. Another possible line would consist of evaluating the

effects of the uncertainties in localization that arise in the underwater medium [23] in our proposed method.

Another line of work would consist of adding more constrains to the problem we are solving, to make it more realistic in different aspects. One of them could be using a noisy observation as input, instead of the actual state, as underwater location mechanisms are noisy. Another would be adding constrains to the control that reflect the actuators limitations. Note that (5) can deal with admissible controls, and in (10) we limit our controls to lie in the set $\theta \in [0, 2\pi)$ (which is enforced by the arctan operation in (15)). However, different control constrains could be enforced in our formulation depending on the AUV actuators. Additionally, our approach could also be tested using obstacles, by means of modifying the cost functional to have an additional term that increases the cost as the distance to an obstacle increases.

It would also be possible to add uncertainty about the disturbance. Note that, in this work, we have trained a neural network for each disturbance, and hence, our method requires knowing the disturbance in advance, as well as its parameters. A possible way to address this would be to extend the state to include the disturbance parameters as inputs: the negative counterpart would be an increase in the computational load as a result of the growth of the state space.

Finally, it would also be interesting checking the performance of the DGM compared to other methods used to approximately solve OCPs, such as deep reinforcement learning ones, which have already been applied to navigation ([38,39]) and offer an interesting alternative to compare the DGM with.

Author Contributions: Conceptualization, J.P., P.A.A. and S.Z.; methodology, J.P., P.A.A. and S.Z.; software, J.P.; validation, J.P., P.A.A. and S.Z.; formal analysis, J.P., P.A.A. and S.Z.; investigation, J.P., P.A.A. and S.Z.; resources, J.P., P.A.A. and S.Z.; writing—original draft preparation, J.P.; writing—review and editing, J.P., P.A.A. and S.Z.; visualization, J.P. and P.A.A.; supervision, S.Z.; project administration, S.Z.; funding acquisition, S.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Spanish Ministry of Science and Innovation under the grant TEC2016-76038-C3-1-R (HERAKLES).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data sharing is not applicable.

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

References

1. Goerzen, C.; Kong, Z.; Mettler, B. A survey of motion planning algorithms from the perspective of autonomous UAV guidance. *J. Intell. Robot. Syst.* **2010**, *57*, 65. [CrossRef]
2. Friesz, T.L. *Dynamic Optimization and Differential Games*; Springer: Boston, MA, USA, 2010; Volume 135.
3. Wang, S.; Gao, F.; Teo, K.L. An upwind finite-difference method for the approximation of viscosity solutions to Hamilton-Jacobi-Bellman equations. *IMA J. Math. Control Inf.* **2000**, *17*, 167–178. [CrossRef]
4. Tourin, A. An Introduction to Finite Diffference Methods for PDEs in Finance. In *Book Chapter: Nizar Touzi, Optimal Stochastic Target problems, and Backward SDE, Fields Institute Monographs*; Springer: New York, NY, USA, 2011; Volume 29, pp. 201–212.
5. Sun, B.; Guo, B.Z. Convergence of an upwind finite-difference scheme for Hamilton–Jacobi–Bellman equation in optimal control. *IEEE Trans. Autom. Control* **2015**, *60*, 3012–3017. [CrossRef]
6. Falcone, M.; Ferretti, R. Numerical methods for hamilton–jacobi type equations. In *Handbook of Numerical Methods for Hyperbolic Problems, Handbook of Numerical Analysis*; Elsevier: Amsterdam, The Netherlands, 2016; Volume 17, pp. 603–626.
7. Gombao, S. Approximation of optimal controls for semilinear parabolic PDE by solving Hamilton-Jacobi-Bellman equations. In Proceedings of the 15th International Symposium on the Mathematical Theory of Networks and Systems, Notre Dame, IN, USA, 12–16 August 2002.
8. McEneaney, W.M. A curse-of-dimensionality-free numerical method for solution of certain HJB PDEs. *SIAM J. Control Optim.* **2007**, *46*, 1239–1276. [CrossRef]

9. Sirignano, J.; Spiliopoulos, K. DGM: A deep learning algorithm for solving partial differential equations. *J. Comput. Phys.* **2018**, *375*, 1339–1364. [CrossRef]
10. Al-Aradi, A.; Correia, A.; Naiff, D.d.F.; Jardim, G.; Saporito, Y. Applications of the Deep Galerkin Method to Solving Partial Integro-Differential and Hamilton-Jacobi-Bellman Equations. *arXiv* **2019**, arXiv:1912.01455.
11. Chen, J.; Du, R.; Wu, K. A Comparison Study of Deep Galerkin Method and Deep Ritz Method for Elliptic Problems with Different Boundary Conditions. *arXiv* **2020**, arXiv:2005.04554.
12. Wang, J.; Xu, Z.Q.J.; Zhang, J.; Zhang, Y. Implicit bias with Ritz-Galerkin method in understanding deep learning for solving PDEs. *arXiv* **2020**, arXiv:2002.07989.
13. Li, J.; Yue, J.; Zhang, W.; Duan, W. The Deep Learning Galerkin Method for the General Stokes Equations. *arXiv* **2020**, arXiv:2009.11701.
14. Burns, R.S. The use of artificial neural networks for the intelligent optimal control of surface ships. *IEEE J. Ocean. Eng.* **1995**, *20*, 65–72. [CrossRef]
15. Lolla, T.; Lermusiaux, P.F.; Ueckermann, M.P.; Haley, P.J. Time-optimal path planning in dynamic flows using level set equations: theory and schemes. *Ocean. Dyn.* **2014**, *64*, 1373–1397. [CrossRef]
16. Takei, R.; Tsai, R.; Shen, H.; Landa, Y. A practical path-planning algorithm for a simple car: A Hamilton-Jacobi approach. In Proceedings of the 2010 American Control Conference, Baltimore, MD, USA, 30 June–2 July 2010; pp. 6175–6180.
17. Kragelund, S.; Walton, C.; Kaminer, I.; Dobrokhodov, V. Generalized Optimal Control for Autonomous Mine Countermeasures Missions. *IEEE J. Ocean. Eng.* **2020**, *46*, 466–496. [CrossRef]
18. Li, G.; Hildre, H.P.; Zhang, H. Toward time-optimal trajectory planning for autonomous ship maneuvering in close-range encounters. *IEEE J. Ocean. Eng.* **2019**, *45*, 1219–1234. [CrossRef]
19. Shen, C.; Shi, Y.; Buckham, B. Nonlinear model predictive control for trajectory tracking of an AUV: A distributed implementation. In Proceedings of the 2016 IEEE 55th Conference on Decision and Control (CDC), Las Vegas, NV, USA, 12–14 December 2016; pp. 5998–6003.
20. Shen, C.; Shi, Y. Distributed implementation of nonlinear model predictive control for AUV trajectory tracking. *Automatica* **2020**, *115*, 108863. [CrossRef]
21. Bemporad, A.; Morari, M.; Dua, V.; Pistikopoulos, E.N. The explicit linear quadratic regulator for constrained systems. *Automatica* **2002**, *38*, 3–20. [CrossRef]
22. Karg, B.; Lucia, S. Efficient representation and approximation of model predictive control laws via deep learning. *IEEE Trans. Cybern.* **2020**, *50*, 3866–3878. [CrossRef] [PubMed]
23. Paull, L.; Saeedi, S.; Seto, M.; Li, H. AUV navigation and localization: A review. *IEEE J. Ocean. Eng.* **2013**, *39*, 131–149. [CrossRef]
24. Isaacs, R. *Differential Games: A Mathematical Theory with Applications to Warfare and Pursuit, Control and Optimization*; Wiley: New York, NY, USA, 1965.
25. Bertsekas, D.P. *Dynamic Programming and Optimal Control*; Athena Scientific: Nashua, NH, USA, 2005; Volume 1.
26. Bertsekas, D.P. *Dynamic Programming and Optimal Control*; Athena Scientific: Nashua, NH, USA, 2007; Volume 2.
27. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press Cambridge: Cambridge, MA, USA, 1998.
28. Todorov, E. Optimal control theory. In *Bayesian Brain: Probabilistic Approaches to Neural Coding*; MIT Press: Cambridge, MA, USA, 2006; pp. 269–298.
29. Crandall, M.G.; Lions, P.L. Viscosity solutions of Hamilton-Jacobi equations. *Trans. Am. Math. Soc.* **1983**, *277*, 1–42. [CrossRef]
30. Lions, P.L. Hamilton-Jacobi-Bellman equations and the optimal control of stochastic systems. In Proceedings of the International Congress of Mathematicians, Warsaw, Poland, 16–24 August 1983; Volume 1, p. 2.
31. Crandall, M.G. Viscosity solutions: A primer. In *Viscosity Solutions and Applications*; Springer: Berlin/Heidelberg, Germany, 1997; pp. 1–43.
32. Bressan, A. Viscosity solutions of Hamilton-Jacobi equations and optimal control problems. *Lect. Notes* **2011**. Available online: http://personal.psu.edu/axb62/PSPDF/hj.pdf (accessed on 15 June 2021)
33. Crandall, M.G.; Ishii, H.; Lions, P.L. User's guide to viscosity solutions of second order partial differential equations. *Bull. Am. Math. Soc.* **1992**, *27*, 1–67. [CrossRef]
34. Bardi, M.; Capuzzo-Dolcetta, I. *Optimal Control and Viscosity Solutions of Hamilton-Jacobi-Bellman Equations*; Birkhäuser: Boston, MA, USA, 1997.
35. Fleming, W.H.; Soner, H.M. *Controlled Markov Processes and Viscosity Solutions*; Springer: New York, NY, USA, 2006; Volume 25.
36. Barles, G.; Souganidis, P.E. Convergence of approximation schemes for fully nonlinear second order equations. *Asymptot. Anal.* **1991**, *4*, 271–283. [CrossRef]
37. Kingma, D.P.; Ba, J.L. Adam: A method for stochastic gradient descent. In Proceedings of the ICLR: International Conference on Learning Representations, San Diego, CA, USA, 7–9 May 2015.
38. Sun, Y.; Cheng, J.; Zhang, G.; Xu, H. Mapless motion planning system for an autonomous underwater vehicle using policy gradient-based deep reinforcement learning. *J. Intell. Robot. Syst.* **2019**, *96*, 591–601. [CrossRef]
39. Cao, X.; Sun, C.; Yan, M. Target Search Control of AUV in Underwater Environment With Deep Reinforcement Learning. *IEEE Access* **2019**, *7*, 96549–96559. [CrossRef]