sensors

MDPI

Article

Real-Time Path Planning Based on Harmonic Functions under a Proper Generalized Decomposition-Based Framework

Nicolas Montés [1,*], Francisco Chinesta [2], Marta C. Mora [3], Antonio Falcó [1], Lucia Hilario [1], Nuria Rosillo [1] and Enrique Nadal [4]

[1] Department of Mathematics, Physics and Technological Sciences, University CEU Cardenal Herrera, C/San Bartolome 55, CP Alfara del Patriarca, 46115 Valencia, Spain; afalco@uchceu.es (A.F.); luciah@uchceu.es (L.H.); nrosillo@uchceu.es (N.R.)
[2] PIMM Lab, ESI Group Chair at ENSAM Institute of Technology, 151 Boulevard de L'Hôpital, 75013 Paris, France; francisco.chinesta@ensam.eu
[3] Department of Mechanical Engineering and Construction, Universitat Jaume I, 12071 Castellón, Spain; mmora@uji.es
[4] Centro de Investigación en Ingeniería Mecánica, Universitat Politécnica de València, 46022 Valencia, Spain; ennaso@upvnet.upv.es
* Correspondence: nicolas.montes@uchceu.es

Abstract: This paper presents a real-time global path planning method for mobile robots using harmonic functions, such as the Poisson equation, based on the Proper Generalized Decomposition (PGD) of these functions. The main property of the proposed technique is that the computational cost is negligible in real-time, even if the robot is disturbed or the goal is changed. The main idea of the method is the off-line generation, for a given environment, of the whole set of paths from any start and goal configurations of a mobile robot, namely the computational vademecum, derived from a harmonic potential field in order to use it on-line for decision-making purposes. Up until now, the resolution of the Laplace or Poisson equations has been based on traditional numerical techniques unfeasible for real-time calculation. This drawback has prevented the extensive use of harmonic functions in autonomous navigation, despite their powerful properties. The numerical technique that reverses this situation is the Proper Generalized Decomposition. To demonstrate and validate the properties of the PGD-vademecum in a potential-guided path planning framework, both real and simulated implementations have been developed. Simulated scenarios, such as an L-Shaped corridor and a benchmark bug trap, are used, and a real navigation of a LEGO®MINDSTORMS robot running in static environments with variable start and goal configurations is shown. This device has been selected due to its computational and memory-restricted capabilities, and it is a good example of how its properties could help the development of social robots.

Keywords: path planning; potential fields; harmonic functions; Proper Generalized Decomposition; Poisson equation

Citation: Montés, N.; Chinesta, F.; Mora, M.C.; Falcó, A.; Hilario, L.; Rosillo, N.; Nadal, E. Real-Time Path Planning Based on Harmonic Functions under a Proper Generalized Decomposition-Based Framework. *Sensors* **2021**, *21*, 3943. https://doi.org/10.3390/s21123943

Academic Editors: Pedro Núñez Trujillo, Luis Manso Fernández-Argüéllez and Luis Vicente Calderita

Received: 21 May 2021
Accepted: 4 June 2021
Published: 8 June 2021

1. Introduction

A fundamental robotic task is to plan collision-free motions among a set of static and known obstacles from a start to a goal position. The geometric construction of this planning strategy is computationally hard and hence unfeasible for its use in real-time (RT) applications [1] where the environment changes, as is the case of social robotics. This motion planning (or the piano mover's) problem has motivated many works in the field of robotics, and its complexities are well known in the literature. The problem cannot be solved in closed form and, therefore, approximations and simplifications have been developed to partially solve the problem. Some researchers have been devoted to study some sub-classes of the general problem while other researchers have considered alternative and simplified planning paradigms such as sampling-based planners, grid-based and interval-based

searches, combinatorial methods, and potential-field-based techniques [2,3]—all of them attempting to find a trade-off between completeness and computational burden.

In the above context, one of the most popular algorithms is the so-called Artificial Potential Field technique (APF) [1,4,5]. This method introduces an artificial potential field that produces a set of paths from a start to a goal configuration. It allows the computation of a unique trajectory from the start to the goal. This technique is very fast for RT applications, except when the vehicle is trapped in a deadlock (a local minimum of the potential function). For this reason, this technique is currently being investigated in the fields of mobile robotics [6], intelligent vehicles [7,8] and social robotics [9–11].

The solution to this problem lies in the use of harmonic functions to generate the potential field [12]. These functions, initially proposed in [13], appear as the solutions of the Laplace equation on a specific domain. They exhibit very elegant and interesting properties for path planning, as described in [14]. First, harmonic functions satisfy the min–max principle and, therefore, the appearance of deadlocks is not possible. Second, under certain assumptions, a path planning scheme using harmonic functions is complete. Third, solutions to Laplace's equation obey the principle of superposition, which can be used to enhance the vehicle behaviour in certain areas of the workspace (near the obstacles). Fourth, the gradient of a harmonic function can be used as a velocity reference for the mobile robot navigation. Finally, harmonic-based navigation allows dealing with errors in the model and in the mobile robot position, i.e., with uncertainty.

Despite their attractive properties, path planning based on harmonic functions has not been widely adopted because these functions cannot be computed in closed form and discrete approximations imply such a computational burden that has prevented their extensive use, as indicated in [15]. In fact, standard finite difference methods (such as Jacobi, Gauss–Seidel, SOR) are typically used to solve the Laplace equation [14,16], implying the use of static environments, i.e., static obstacles as well as fixed start and goal configurations. Any new configuration requires the recalculation of the harmonic function and the computation of the streamlines over the entire region. Only if everything remains static, path generation can proceed very quickly since it only involves the evaluation of the gradient of a precomputed potential function. Additionally, as explained in [17], the solution must be numerically obtained in a discrete mesh, which entails a computational cost that increases exponentially with the mesh resolution. For instance, in [18], a scanned environment composed of 1500 triangles was used that implied a computational cost of 19.2 s in an Intel Core 2 Processor (64-bit dual-core commercial CPU) for every re-computation of the streamlines. Although recent techniques have accelerated this calculation [19,20], the computational time is still high for RT navigation, with 646s for a 512×512 node environment using the EGSOR algorithm in the last work. Thus, its use for RT navigation applications seems unrealistic.

Recently, a novel approach called the Proper Generalized Decomposition (PGD) has appeared to approximate the solutions of non-linear convex variational problems [21]. It is a new paradigm for solving classical problems in high-dimensional spaces [22,23]. In the PGD framework, the resulting model is solved once in order to obtain all the solutions for every possible value of the parameters defined in a specific domain, that is, a sort of computational vademecum. It could also be seen as a handbook containing all the possible solutions of a variational equation for any value of the parameters. Many challenging problems can be efficiently cast into this framework, and it opens new possibilities to solve problems with strategies not envisioned until now.

In the field of path planning, the goodness of having a precomputed solution when the obstacles and the goal are fixed was already stated in [16]. In the 1990s, the computation of the streamline maps for all the possible combinations of start/goal positions was not feasible with the available tools. Fortunately, this situation has now changed with the appearance of the PGD.

The goal of the present paper is to experimentally validate the PGD as a new alternative for RT mobile robot navigation using potential harmonic functions. Previous work

was presented in [24] with local minima still present in the approximate solution due to the interaction of start and target positions, modelled as Gaussian functions. In the present work, cell-centred and staggered meshes are used to overcome this problem and a free-of-deadlocks solution is reached. To demonstrate the capabilities of the proposed approach, a harmonic potential field based on the flow theory is used in simulated complex environments as well as in RT navigation experiments with a LEGO®MINDSTORMS robot in environments with static obstacles and variable start and goal robot configurations. A LEGO®MINDSTORMS robot was also selected as the experimental platform because it is a commonly used device in teaching social robotics interactions [9]. This paper is organized as follows: Section 2 first details the potential flow theory to obtain harmonic functions by means of the Poisson equation. Additionally, the PGD-Vademecum solution for the mobile robot path planning introduced in our previous work [24] is revisited, and the restrictions required for the matrices to guarantee a free-of-deadlocks solution are defined. Section 3 presents the application of the PGD-Vademecum in simulated complex environments such as an L-Shaped corridor and a bug trap. Section 4 shows real experiments using a LEGO®MINDSTORMS robot in a square map with and without static obstacles. In Section 5, the computational properties of the PGD-based framework are disclosed. Finally, Section 6 draws conclusions and future works.

2. Previous Knowledge

2.1. Potential Flow Theory

Path planning based on the potential flow theory has been used in the literature over the last few years [12,14,16–20,25–32], focused mainly on the resolution of the Poisson equation. First of all, let us introduce the underlying mathematical model that describes a potential flow, where the velocity $\boldsymbol{v}$ verifies

$$\bigtriangledown \times \boldsymbol{v} = \mathbf{0}, \tag{1}$$

and hence it is the gradient of a scalar potential function u, i.e., $\boldsymbol{v} = -\nabla u$. Then, assuming incompressible flow, i.e.,

$$\nabla \cdot \boldsymbol{v} = 0, \tag{2}$$

it results

$$\Delta u = 0. \tag{3}$$

Let us introduce a localized source (respectively, sink) modelled by a Dirac term δ_S (respectively, $-\delta_T$) on the right-hand side of (3). The velocity of the fluid is now the solution of the Poisson equation with source term $f = \delta_S - \delta_T$, that is,

$$\Delta u = f. \tag{4}$$

Equation (4) has an infinite number of solutions. In order to solve it, appropriate boundary conditions must be introduced.

In what follows, Neuman boundary conditions compatible with the incompressibility constraint are enforced on the boundary Γ

$$\nabla u \cdot \mathbf{n} = q. \tag{5}$$

The resolution of the Poisson equation under these conditions produces a potential field from the *starting* point S to the *target* point T, without deadlocks [18]. Unfortunately, there is no analytical solution for this equation in the general case, and numerical techniques must be used. In this sense, the PGD technique, introduced in the next section, overcomes some of the drawbacks of common numerical methods allowing real-time performance.

2.2. Source Term Definition

Consider the functions $g_S : \Omega_{\underline{X}} \times \Omega_{\underline{S}} \to \mathbb{R}$ and $g_T : \Omega_{\underline{X}} \times \Omega_{\underline{T}} \to \mathbb{R}$ as 2D Gaussian density distributions centred in the start $\underline{S} = (s_1, s_2) \in \Omega_{\underline{S}}$ and target configurations $\underline{T} = (t_1, t_2) \in \Omega_{\underline{T}}$, respectively. Both functions are assumed to have equal variance given by parameter $r > 0, r \in \mathbb{R}$. More precisely, we can write $g_S = g_S((x, y); (s_1, s_2), r) = (2\pi r)^{-1} e^{-\frac{1}{2r}((x-s_1)^2+(y-s_2)^2)}$, $g_T = g_T((x, y); (t_1, t_2), r) = (2\pi r)^{-1} e^{-\frac{1}{2r}((x-t_1)^2+(y-t_2)^2)}$ and hence $\Omega_{\underline{X}} = \Omega_x \times \Omega_y$, $\Omega_{\underline{S}} = \Omega_s \times \Omega_r$ and $\Omega_{\underline{T}} = \Omega_t \times \Omega_r$. Here, $\Omega_{\underline{X}} = \Omega_{\underline{S}} = \Omega_{\underline{T}} \subset \mathbb{R}^2$.

In the context of path planning, the physical meaning of this model relies on the fact that uncertainty is always present in the vehicle position computation in the form of measurement noise and/or process noise. Both types of noise are usually modelled by Gaussian functions. Geometrically, in 2D, the standard deviations of the Gaussian functions define the radii of the uncertainty ellipse around the robot position. If the robot is holonomic (equal variance in X and Y positions), the ellipse turns into a circumference. If spherical bounding volumes are used to account for the vehicle size, in the case of an omnidirectional vehicle, these volumes can be transformed into an increase in the variance of these functions and an uncertainty bounding volume can be defined, as in [33], including the uncertainty area due to noise as well as the robot volume (radius of the bounding sphere). Additionally, a security radius can also be considered in this model in the context of minimum distance from walls.

Let us assume that the source term f in Equation (4) is non-uniform, that is, $f = g_S - g_T$ when $(x, y) \in \Omega_{\underline{X}}$ and zero otherwise. Then, the Poisson equation is now

$$-\Delta u = g_S - g_T$$

for given functions g_S and g_T. Once discretized, a separated form of this equation is obtained in (6) as a function of the map, the start and target configurations:

$$-\Delta u(\underline{X}, \underline{S}, \underline{T}) \quad = f(\underline{X}, \underline{S}, \underline{T}) \tag{6}$$

2.3. A PGD-Vademecum Solution

The PGD-vademecum is generated considering that the solution of the potential field u solving (4) and (5) can be constructed as a finite sum of terms, each one consisting of the product of three functions: a function R of the environment $\underline{X}$, a function W of the start configuration $\underline{S}$ and a function K of the target or goal configuration $\underline{T}$:

$$u^{n-1}(\underline{X}, \underline{S}, \underline{T}) = \sum_{i=1}^{n-1} R_i(\underline{X}) \cdot W_i(\underline{S}) \cdot K_i(\underline{T}) \tag{7}$$

and where the enrichment step is given by

$$u^n = u^{n-1} + R(\underline{X}) \cdot W(\underline{S}) \cdot K(\underline{T}). \tag{8}$$

The key point is to find a rank-one function

$$\begin{array}{c} R(\underline{X}) \cdot W(\underline{S}) \cdot K(\underline{T}) := \\ R(x, y) \cdot W(s_1, s_2) \cdot K(t_1, t_2) \end{array} \tag{9}$$

satisfying

$$\int_{\Omega_{\underline{X}} \times \Omega_{\underline{S}} \times \Omega_{\underline{T}}} u^* \cdot (\Delta u^n - f)\, d\Omega_{\underline{X},\underline{S},\underline{T}} = 0 \tag{10}$$

for all test functions u^* in the linear space of functions

$$\begin{array}{c} R^*(\underline{X}) \cdot W(\underline{S}) \cdot K(\underline{T}) + R(\underline{X}) \cdot W^*(\underline{S}) \cdot K(\underline{T}) \\ + R(\underline{X}) \cdot W(\underline{S}) \cdot K^*(\underline{T}). \end{array} \tag{11}$$

There are some techniques to measure the error in the approximation versus the number of PGD terms (n). One of the most appropriate error estimators is the $L^2(\Omega_{\underline{X}} \times \Omega_{\underline{S}} \times \Omega_{\underline{T}})$-residual $R(n)$ obtained by inserting the PGD-vademecum approximation into the Poisson Equation (6) and calculating the residual (10), that is

$$R(n) = \int_{\Omega_{\underline{X}} \times \Omega_{\underline{S}} \times \Omega_{\underline{T}}} (\Delta u^n - f) \cdot (\Delta u^n - f)\, d\Omega_{\underline{X},\underline{S},\underline{T}}. \tag{12}$$

One of the most important properties of the PGD is that the first terms store more relevant information than the last terms, see [24].

2.4. Meshing Constraints to Guarantee Free-of-Deadlocks Solutions

In our previous work [24], the PGD was first introduced as an approach capable of using potential fields methods based on harmonic functions for real-time navigation. However, since the PGD is an approximation method, it did not guarantee that the resulting approximate solution was free of local minima. In this work, this problem has been solved and the resulting PGD solution is free of deadlocks as explained in [34] and summarized here:

- The mesh of $\underline{X}$ must be cell-centred with the $\underline{S}$ and $\underline{T}$ meshes. This means that the physical positions of the nodes $\underline{S}$ and $\underline{T}$ must also exist in the $\underline{X}$ mesh. It is obvious as all the robot origins and destinations must appear on the map. In addition, the dimensions of $\underline{S}$ and $\underline{T}$ can be much smaller than the dimension of $\underline{X}$, since all the start (and goal) positions could be separated a moderated distance (0.5 m, for example) while the navigation map must be accurate (with separation between nodes of a few cm).
- The mesh of $\underline{S}$ must be staggered with the $\underline{T}$ mesh.

3. Simulated Validation. Construction of the PGD-Vademecum in Complex and Real Environments

In this section, the construction of a PGD-vademecum for path planning using the Poisson equation is explained for two complex and real environments depicted in Figure 1. The environment on the left is an L-shaped corridor of a plant with two zones (in pink) that should be avoided during the robot navigation. The picture on the right represents a bug trap planning environment used for benchmarking in robot motion planning. In both cases, the walls have been modelled with Neumann boundary conditions. In the case of the L-shaped corridor, the dangerous zones have been modelled with stronger Neumann boundary conditions than those employed for the walls. In the first example, the corridor is 6 m long, the $\underline{X}$ mesh has $N_x = N_y = 60$ nodes while the $\underline{S}$ and $\underline{T}$ meshes have a dimension of $N_x = N_y = 6$ nodes. In the second example, the environment is a 6×6 m square where the $\underline{X}$ mesh has $N_x = N_y = 120$ nodes, and the $\underline{S}$ and $\underline{T}$ meshes have $N_x = N_y = 12$ nodes. Figure 2 represents the PGD reconstructions for both cases using two arbitrary start and goal positions with $n = 300$ PGD terms. These simulations were developed in MATLAB using an HP laptop with a CPU Intel CORE i5, 4 GB RAM memory.

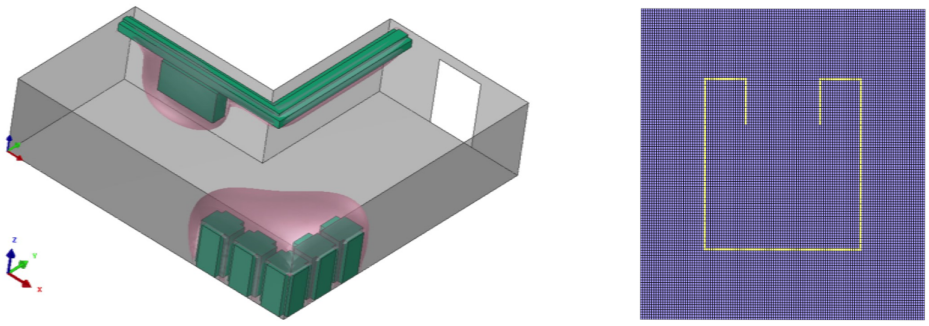

Figure 1. Complex and real environments for navigation: (**left**) L-shaped corridor in a plant; (**right**) bug trap planning environment.

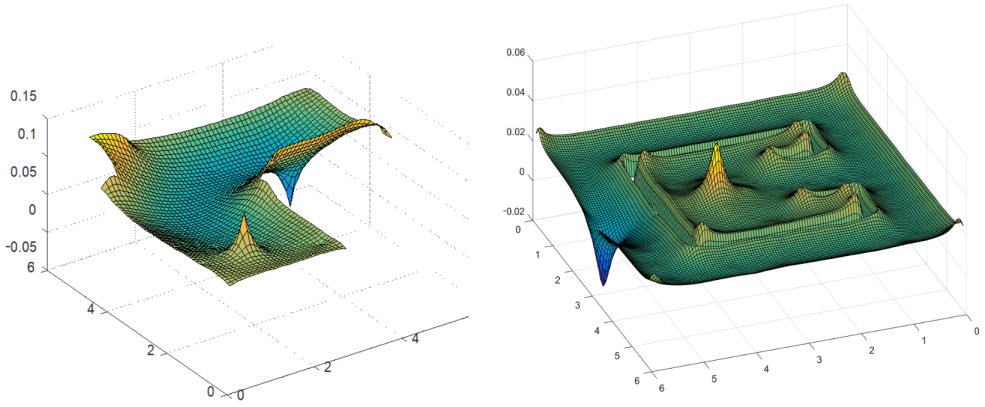

Figure 2. PGD reconstruction (potential field mesh) for two arbitrary start and goal locations in a (**left**) L-shaped corridor and (**right**) a bug trap planning environment.

4. Experimental Validation. PGD-Vademecum in a Lego ®Mindstorms

In order to test the benefits of the PGD framework, the PGD-vademecum must be programmed in a real mobile robot with restricted computational and memory capabilities. In this sense, a LEGO®MINDSTORMS Education EV3 Core Set has been selected. The Core Set includes 541 elements usually employed for teaching science, technology, engineering, mathematics, and computer science to children. The Core Set comes in a sturdy storage bin with a sorting tray for easy classroom management and includes three servomotors and a computer-in-a-brick that makes it possible to control motors and collect data from sensors. The brick has a LINUX operating system with an ARM controller that runs at 300 MHz with a 64 MB RAM and 16 MB flash memory. These computational and memory capabilities are far from those of the recent CPUs. An omnidirectional robot has been built with the EV3 Brick, two servomotors and some common LEGO®MINDSTORMS pieces. The robot is shown in Figure 3.

Figure 3. Experimental LEGO robot.

The EV3 Brick can be directly programmed in C++ but a Matlab® compiler with the LEGO®EV3 library for Simulink® is used instead. The wireless communication between the EV3 Brick and the PC is established by Wi-Fi. The block diagram of the experimental setup is depicted in Figure 4. A webcam with 1280×720 pixels is used to sense the 1.5×1.5 m map where the vehicle moves. The camera is connected to a desktop PC, in particular, an HP laptop with Intel CORE i5, 4 GB RAM memory. The PC computes, on the one hand, the actual *pose* (x, y, θ) of the LEGO® vehicle from the position of three blue squares attached to the top of the vehicle and, on the other hand, the *target* position (X_g, Y_g) indicated by a red plastic disk. This information is sent to the LEGO® brick in real-time. Detailed explanations about the implementation setup can be found in the authors' previous work [35].

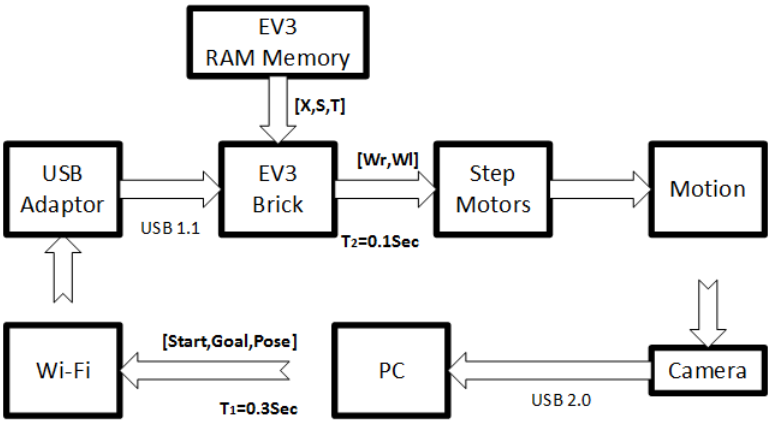

Figure 4. Block diagram of the Lego® vehicle experimental setup.

The EV3 Brick stores the compiled code that controls the vehicle as well as the PGD-Vademecum matrices $(\underline{X}, \underline{S}, \underline{T})$, denoted by (X, S, T) in the RAM memory. Each of the above matrices has a particular $N_x \times N_y$-size depending on the size of the x and y meshes of $\underline{X}$, $\underline{S}$ and $\underline{T}$.

Given the desired wheels velocities, a PID is used to control the vehicle trajectory. These velocities are computed as explained in the following subsections.

4.1. PGD-Vademecum to Compute the Wheels Velocities

One of the advantages of the PGD-vademecum in contrast to FEM simulation techniques used in [18] is that it allows the reconstruction of single mesh nodes in each algorithm execution k and for any kind of parameter combination. In the case of a robotics application, this property implies the possibility of computing the mobile robot path in a small portion of the map for any combination of the start and goal configurations $\underline{S}$, $\underline{T}$. The so-called Region Of Interest (ROI) is composed of the four neighbouring nodes of a particular vehicle position $\underline{X} = (x, y)$ and can be described by a 2×2 block matrix $ROI(\underline{X})$ as

$$ROI(\underline{X}) = \begin{bmatrix} \underline{X}_{11} & \underline{X}_{12} \\ \underline{X}_{21} & \underline{X}_{22} \end{bmatrix} \tag{13}$$

where $\underline{X}_{11} = (x_1, y_1)$, $\underline{X}_{12} = (x_1, y_2)$, $\underline{X}_{21} = (x_2, y_1)$ and $\underline{X}_{22} = (x_2, y_2)$ are the coordinates of the neighbouring nodes of $\underline{X}$, represented as red rhombuses in Figure 5.

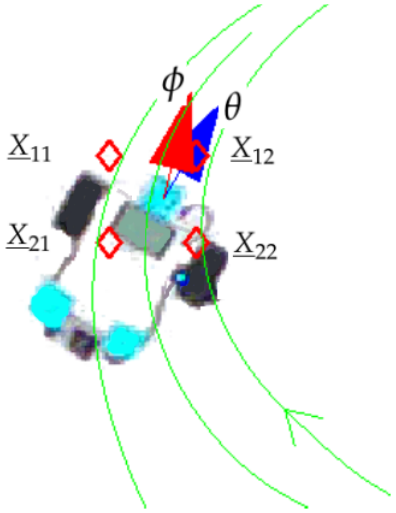

Figure 5. Coordinates of neighbour nodes in a particular mobile robot position.

Selecting $\underline{S} = (x_S, y_S)$ as the start node position and $\underline{T} = (x_T, y_T)$ as the goal node position, the ROI-Poisson approximate solution at $\underline{X}$ can be simply computed using the PGD-vademecum solution u as

$$u_{ROI}(\underline{X}, \underline{S}, \underline{T}) = \begin{bmatrix} u(\underline{X}_{11}, \underline{S}, \underline{T}) & u(\underline{X}_{12}, \underline{S}, \underline{T}) \\ u(\underline{X}_{21}, \underline{S}, \underline{T}) & u(\underline{X}_{22}, \underline{S}, \underline{T}) \end{bmatrix}. \tag{14}$$

The ROI velocity field matrix can be computed as indicated in Equation (15), where $V_{x_{ROI}}, V_{y_{ROI}}$ are 2×2 matrices with the velocity field for each node.

$$\begin{aligned} V_{x_{ROI}} = \tfrac{\partial u_{ROI}}{\partial x}(\underline{X}, \underline{S}, \underline{T}) &= \begin{bmatrix} \frac{\partial u}{\partial x}(\underline{X}_{11}, \underline{S}, \underline{T}) & \frac{\partial u}{\partial x}(\underline{X}_{12}, \underline{S}, \underline{T}) \\ \frac{\partial u}{\partial x}(\underline{X}_{21}, \underline{S}, \underline{T}) & \frac{\partial u}{\partial x}(\underline{X}_{22}, \underline{S}, \underline{T}) \end{bmatrix} \\ V_{y_{ROI}} = \tfrac{\partial u_{ROI}}{\partial y}(\underline{X}, \underline{S}, \underline{T}) &= \begin{bmatrix} \frac{\partial u}{\partial y}(\underline{X}_{11}, \underline{S}, \underline{T}) & \frac{\partial u}{\partial y}(\underline{X}_{12}, \underline{S}, \underline{T}) \\ \frac{\partial u}{\partial y}(\underline{X}_{21}, \underline{S}, \underline{T}) & \frac{\partial u}{\partial y}(\underline{X}_{22}, \underline{S}, \underline{T}) \end{bmatrix} \end{aligned} \tag{15}$$

The velocity vector in the mobile robot position (x, y) at step k can be computed by means of a bi-linear interpolation technique as

$$v_x(\underline{X}) = \begin{pmatrix} 1 - x^* & x^* \end{pmatrix} V_{x_{ROI}} \begin{pmatrix} 1 - y^* \\ y^* \end{pmatrix} \tag{16}$$

$$v_y(\underline{X}) = \begin{pmatrix} 1 - x^* & x^* \end{pmatrix} V_{y_{ROI}} \begin{pmatrix} 1 - y^* \\ y^* \end{pmatrix} \tag{17}$$

where (x^*, y^*) are the normalized distances:

$$x^* = \frac{x_2 - x}{x_2 - x_1}, y^* = \frac{y_2 - y}{y_2 - y_1} \tag{18}$$

The resulting vector must be re-scaled to guarantee that the vehicle follows the streamline at constant velocity by taking

$$\begin{aligned} \phi &= \arctan\left(\tfrac{v_y(\underline{X})}{v_x(\underline{X})}\right) \\ v &= v_0, \end{aligned} \tag{19}$$

where v_0 represents the constant mobile robot's velocity. Assuming that $\underline{X}$ is the vehicle position at time instant t, i.e., $(x(t), y(t)) = \underline{X}$, then its position at $t + \Delta t$ (previously fixed time step Δt), denoted by $(x(t + \Delta t), y(t + \Delta t))$, is given by

$$x(t + \Delta t) = x(t) + v_0 \cdot \cos(\phi)\Delta t, \tag{20}$$

$$y(t + \Delta t) = y(t) + v_0 \cdot \sin(\phi)\Delta t, \tag{21}$$

and hence the displacement on each axis between instants t and $t + \Delta$ is given by:

$$\Delta x = v_0 \cdot \cos(\phi) \cdot \Delta t, \tag{22}$$

$$\Delta y = v_0 \cdot \sin(\phi) \cdot \Delta t. \tag{23}$$

Finally, the angular displacement on the wheels $(\Delta\alpha_r, \Delta\alpha_l)$ between t and $t + \Delta$ are computed, as usual, by means of inverse kinematics:

$$\begin{pmatrix} \Delta\alpha_r & \Delta\alpha_l \end{pmatrix} = \begin{pmatrix} \frac{\cos(\phi)}{R} & \frac{\sin(\phi)}{R} & \frac{L}{R} \\ \frac{\cos(\phi)}{R} & \frac{\sin(\phi)}{R} & -\frac{L}{R} \end{pmatrix} \cdot \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta\phi \end{pmatrix} \tag{24}$$

where R is the wheel radius, L is the wheel separation and $\Delta\phi = \phi - \theta$. Figure 5 displays a snapshot of a particular robot pose (x, y, θ). The blue arrow shows the robot orientation θ, and the red arrow shows the computed reference for the next step, $k+1$, by means of the ROI. Green curves are the continuous streamlines derived from the potential field.

4.2. Experimental Tests

Three tests have been developed with the LEGO® mobile robot in the 1.5×1.5 m square environment and are explained in the following paragraphs. In them, a red plastic disk is used as a random goal.

Test 1: Point to Point.
This test shows that the PGD-vademecum could cover all the possible start and goal combinations $(N_x \times N_y)^2$ for the vehicle position. The test procedure is as follows: the red disk is thrown to the scene, falling into an arbitrary position, and the vehicle must reach the red disk (goal); once the robot reaches the goal, the red disk is thrown to another arbitrary position.

Test 2: Dynamic Goal.
This test demonstrates that the robot does not need additional computation when the goal changes before the robot reaches it, unlike in [18] where, if the goal changes while the robot is following a particular streamline, a new FEM simulation must be executed and the robot has to wait for the new FEM simulation solution. The test procedure is the following: the red disk is thrown to the scene falling into an arbitrary position; the red disk (goal) changes in real-time (due to external causes) forcing the robot to adjust its target while navigating the environment.

Test 3: Perturbations.
This tests demonstrates the robustness of the potential field approach for each particular PGD-vademecum solution. For that purpose, the robot is perturbed, changing its pose before reaching the goal. The procedure of this test is as follows: the red disk is thrown to the scene falling into an arbitrary position; the robot pose is manually changed to another arbitrary pose, forcing it to recompute its trajectory in real-time.

These three tests are performed in two modalities:

Modality A: Without static obstacles.
In this modality, the robot navigates a square environment of 1.5×1.5 m discretized with 20×20 nodes. The precomputed PGD-Vademecum has the following parameters: $r = 0.7$, $n = 200$ and $q = 0.06$.

Modality B: With a static obstacle.
In this modality, the robot navigates a 1.5 m $\times$ 1.5 m square with a square static obstacle of 0.3×0.3 m located at the centre of the environment. This would be a common situation for an autonomous robot moving in a house, a parking lot, a field with trees, etc., where the map contains static obstacles, and the robot has to skirt them. The environment is also discretized with 20×20 nodes. The precomputed PGD-vademecum has the following parameters: $r = 0.7$ and $n = 200$ and $q = 1$. The generation of static obstacles only involves the assignation of boundary conditions to the nodes belonging to each static obstacle. It allows the construction of any configuration map.

Figure 6 shows some snapshots of the experiments carried out for tests 1A, 2A, 3A. Figure 7 shows some snapshots of the experiments performed for tests 1B, 2B, 3B. The robot velocity in all the tests was $v = 0.1$ m/s. In [36], a video with all the tests is shown.

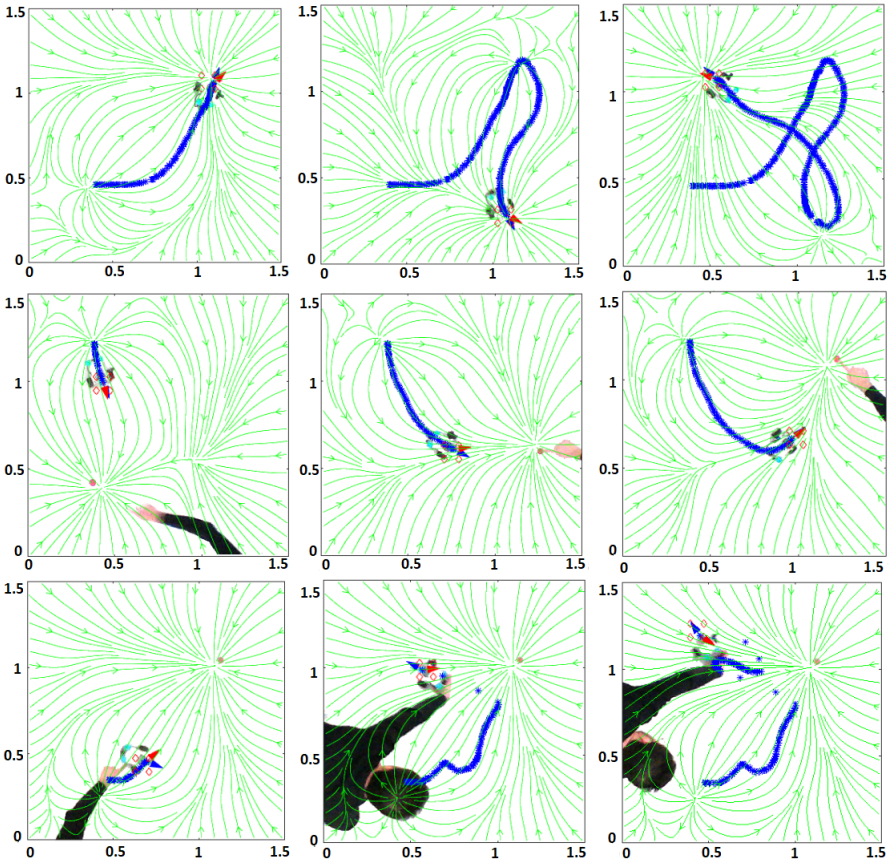

Figure 6. Experimental tests with a LEGO robot. Rows 1, 2, 3 for tests 1A, 2A, 3A.

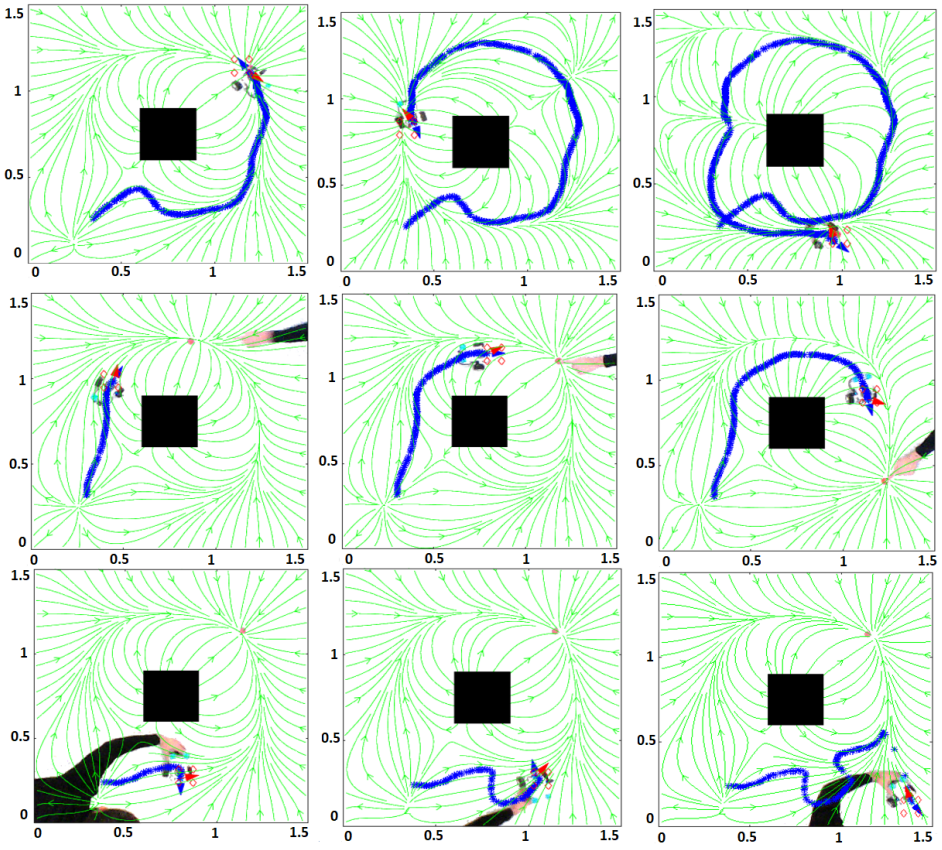

Figure 7. Experimental tests with a LEGO robot. Rows 1, 2, 3 for tests 1B, 2B, 3B.

5. Time and Memory Complexity of the Method

The method presented in this work consists of two phases: an off-line phase and an on-line phase, as depicted in the flowchart of Figure 8. In the off-line phase, the aim is to obtain an approximation of the Poisson equation through the PGD method. This stage provides a solution in the form of a system of uncoupled matrices. These matrices can be easily combined in the on-line phase in order to reconstruct the required solution in an extremely efficient manner, as it only implies the computation of a set of sums and products. The implementation presented in this paper uses the fixed point iteration method to calculate the uncoupled matrices X, S and T. In the numerical example presented in Figure 9, the off-line computational cost of calculating each of the PGD terms is 1.3 s on a Dell Notebook, with an ®Intel Core i7-8550U processor at 1.80 Ghz and 16 GB RAM. The space occupied in memory by the resulting matrices using $N_x = N_y = 50$ for matrix X and $N_x = N_y = 5$ for matrices S, T is 2.9 MB using simple precision floating-point numbers. An extrapolation of the results of B type tests can be made for the map of a real small village, such as 1000×1000 m. In this case, selecting a resolution of 0.5 m, the entire map would have a size of 5.8 GB.

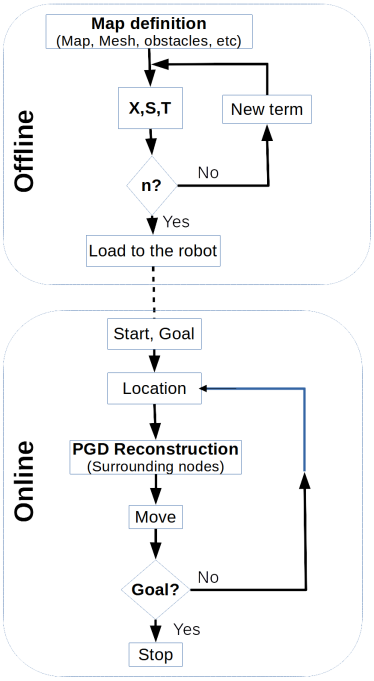

Figure 8. Flowchart of the two phases (off-line and on-line) involved in the PGD-based path planning method.

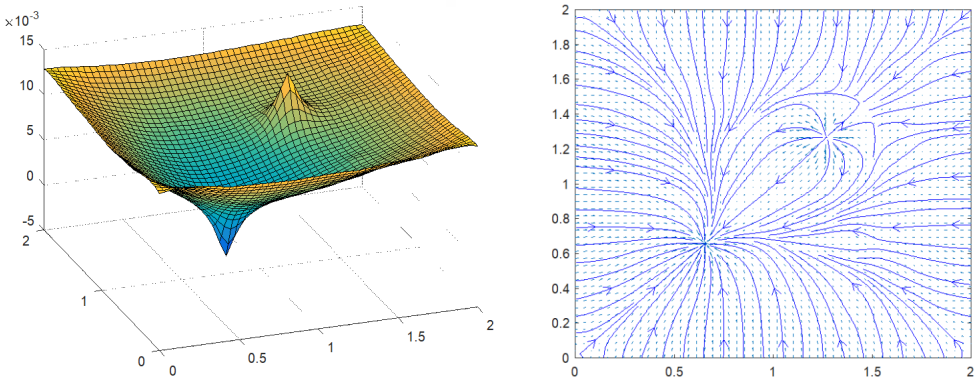

Figure 9. Numerical examples for two arbitrary $\underline{S}$, $\underline{T}$ locations in two complex environments.

Once the off-line process has finished, the resulting X, S and T matrices are stored in the robot's memory for it to navigate the map using the approximate solution obtained through the PGD. The process is performed as shown in Figure 8: given an initial vehicle position and any destination within the map, in every algorithm iteration only a portion of the path around the vehicle location is reconstructed, the ROI, composed by the surrounding nodes of the vehicle position (Figure 5). This calculation is modelled in Equation (13), taking into account the tensor representation of u by (25), and entails the realization of eight products and n sums. If the number of PGD terms is, for instance, $n = 300$, it would imply a total of 2400 floating-point operations. In a modest desktop computer with a Pentium 4 or Athlon 64-type processor, which typically operates faster than 3 GHz and has a computational performance in the range of a few GFLOPS, this on-line calculation would take 2.3 ms considering one GFLOPS performance. Furthermore, if an Intel ®Core i7 microprocessor were used and taking into account that it operates in the range of TFLOPS, the on-line phase would take a few µs, a negligible processing time. Therefore, the computational time of the path reconstruction using the PGD solution is really low, nearly constant $O(1)$ and invariant with respect to the size of the map and the selected map resolution.

$$u(\underline{X}, \underline{S}, \underline{T}) = \sum_{i=1}^{n} R_i(\underline{X}) W_i(\underline{S}) K_i(\underline{T}) \tag{25}$$

Regarding the off-line phase, specific details about the method of convergence, computational time, complexity, etc., can be found in the previous authors' work [37], where numerical examples are provided in order to describe the relationship between the PGD (also called the Greedy Rank-One Update algorithm) and the FEM for high-dimensional partial differential equations based on the tensor product of one-dimensional bases. A comparison between the PGD method and the standard linear system solver of Matlab-Octave `A\b` for the resolution of the Poisson equation is provided in that paper and is also depicted in Figure 10, where the evolution of the CPU time in seconds for both algorithms as a function of the number of nodes used in the discretization of the Poisson equation is shown for a high-performance computer ®Intel Xeon Gold 6130 CPU at 2.1 GHz and 128 GB RAM (64-bit 16-core x86 multi-socket high performance server microprocessor). This illustration helps to understand at a glance the importance of this method as a numerical solver. It is worth noting that with the traditional method, `A\b` it is necessary to solve the equation for the whole map in real-time in every algorithm execution and, therefore, the curse of dimensionality appears when the number of nodes increases (big maps). Nonetheless, with the PGD method, each term and node are decoupled, and it is possible to reconstruct only a simple node or a region of interest as a sum of multiplications, with the computational time independent of the size and resolution of the map.

Finally, Figure 10 allows the comparison between the proposed method and any other grid-based planning method for different grid sizes. For instance, in [38], the A* algorithm is used for real-time motion planning on a 30 × 45 m map discretized using a high resolution grid of 150 × 225 nodes. The entire path planned by the motion planning algorithm contains 2–3 s of commands and assumes an 80 ms processing time (although it can exceed 450 ms if the goal is unachievable). Using the PGD-based framework with a similar but square environment of 150 × 150 nodes (22,500 nodes) and according to Figure 10, the off-line phase would take 14 h in a high-performance computer resulting in the solution of the equation for every start and goal vehicle position. The on-line phase would take negligible time, as the reconstruction is carried out at every time cycle with the same ROI size.

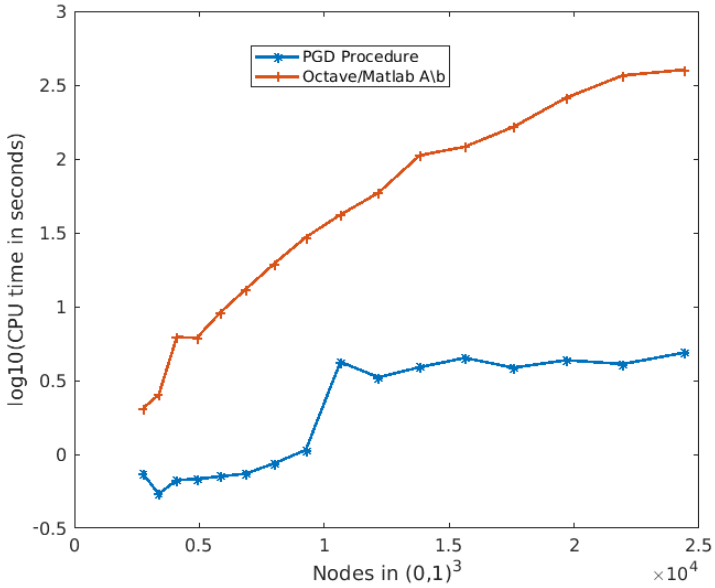

Figure 10. Comparison made in [37] in terms of CPU time between the PGD method and the traditional `A\b` Matlab-Octave method for the resolution of the Poisson equation as a function of the number of nodes in the discretization of the equation.

6. Conclusions and Future Work

The present paper validates the applicability of the numerical technique known as PGD-vademecum to global path planning for mobile robots using harmonic functions. This method produces, for a predefined map, a vademecum containing all the possible vehicle paths for any combination of the start and goal configurations within the map. The PGD-vademecum is computed off-line and reconstructed on-line for any particular combination of the start and goal configurations. It is really fast as its formulation is a simple sum of products. In fact, in RT applications, only the surrounding nodes of the vehicle position need to be reconstructed every execution cycle. As a consequence, the computational costs are nearly negligible, which allows its implementation even in robots with restricted computational capabilities, such as a LEGO®EV3 toy. The resulting paths are based on the Laplace/Poisson equation and, therefore, are free of deadlocks. This property makes it a promising technique to solve the piano mover's problem for any type of robots, such as social robots, because many parameters can be introduced in the PGD-Vademecum as extra coordinates, for instance, parameters related to different kinematic or dynamic robot models or even dynamic obstacles. In this sense, the immediate future work is related to the inclusion of mobile obstacles in the environment. There are various possibilities to take dynamic obstacles into account, such as the one studied in [39], where the disturbance in conductivity is used to model the presence of a moving obstacle, and the location of that conductivity disturbance is included as an additional parameter in the PGD formulation. Other future works imply the consideration of kinematic models for non-holonomic robots in the PGD framework and its use in solving the motion planning problem for unmanned aerial vehicles.

Author Contributions: Conceptualization, N.M., F.C., M.C.M.; methodology, N.M., F.C., A.F., E.N.; software, N.M., N.R., E.N.; validation, N.M., N.R., M.C.M.; formal analysis, N.M., F.C., A.F., L.H.; investigation, N.M., F.C., L.H., M.C.M., E.N., A.F.; resources, N.M., A.F., F.C.; data curation, L.H., M.C.M., N.R.; writing—original draft preparation, N.M., A.F., M.C.M., F.C.; writing—review and editing, N.Mo., A.F., M.C.M., F.C.; visualization, N.M.; supervision, N.M., A.F., M.C.M., F.C., E.N.; project administration, L.H.; funding acquisition, A.F., F.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by the grant number INDI20/13 from Universidad CEU Cardenal Herrera.

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

References

1. Reif, J.H. Complexity of the mover's problem and generalizations. In Proceedings of the 20th Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 29–31 October 1976; pp. 421–427.
2. Kavraki, L.E.; LaValle, S. Chapter 5. Motion Planning. In *Handbook of Robotics;* Siciliano, K., Ed.; Springer: Berlin/Heidelberg, Germany, 2008.
3. González, D.; Pérez, J.; Milanés, V.; Nashashibi, F. A Review of Motion Planning Techniques for Automated Vehicles. *IEEE Trans. Intell. Transp. Syst.* **2016**, *17*, 1135–1145. [CrossRef]
4. Khatib, O. Real-time obstacle avoidance for manipulators and mobile robots. *Int. J. Robot Res.* **1986**, *5*, 90–98. [CrossRef]
5. Rimon, E.; Koditschek, D. Exact robot navigation using artificial potential functions. *IEEE Trans. Robot. Autom.* **1992**, *8*, 501–518. [CrossRef]
6. Lazarowska, A. Discrete Artificial Potential Field Approach to Mobile Robot Path Planning. *IFAC-PapersOnLine* **2019**, *52*, 277–282. [CrossRef]
7. ; Pengwei, W.; Song, W.; Liang, L.; Binbin, S.; Shuo, C. Obstacle Avoidance Path Planning Design for Autonomous Driving Vehicles Based on an Improved Artificial Potential Field Algorithm. *Energeies* **2019**, *12*, 2342.
8. Xinping, G.; Mengxin, H.; Weishuai, Z.; Gang, X.; Guohua, Z.; Yunpeng, H. Intelligent Vehicle Path Planning Based on Improved Artificial Potential Field Algorithm. In Proceedings of the IEEE International Conference on High Performance Big Data and Intelligent Systems (HPBD&IS), Shenzhen, China, 9–11 May 2019; pp. 104–109.
9. Kipp, A.; Schneider, S. Applied Social Robotics—Building Interactive Robots with LEGO Mindstorms. In *Robotics in Education: Advances in Intelligent Systems and Computing*; Merdan, M., Lepuschitz, W., Koppensteiner, G., Balogh, R., Eds.; Springer: Cham, Switzerland, 2017; Volume 457, pp. 29–40.
10. Chen, W.; Zhang, T.; Yanbiao, Z. Mobile robot path planning based on social interaction space in social environment. *Int. J. Adv. Robot. Syst.* **2018**, *15*, 1–10. [CrossRef]
11. Calderita, L.V.; Vega, A.; Bustos, P.; Nuñez, P. Social Robot Navigation adapted to Time-dependent Affordance Spaces: A Use Case for Caregiving Centers. In Proceedings of the IEEE International Workshop on Robot and Human Communication (ROMAN), Naples, Italy, 31 August–4 September 2020; Volume 15, pp. 944–949.
12. Kim, J.; Khosla, P. Real-time obstacle avoidance using harmonic potencial functions. *IEEE Trans. Robot. Autom.* **1992**, *8*, 338–349. [CrossRef]
13. Zhachmanoglou, E.; Thoe, D.W. *Introduction to Partial Differential Equations with Applications*; Dover Publications, Inc.: New York, NY, USA, 1986.
14. Connolly, C.I.; Grupen, R. The Application of Harmonic functions to Robotics. *J. Robot. Syst.* **1993**, *10*, 931–946. [CrossRef]
15. Garrido, S.; Moreno, L.; Blanco, D.; Martín Monar, F. Robotic Motion Using Harmonic Functions and Finite Elements. *J. Intell. Robot. Syst.* **2010**, *59*, 57–73. [CrossRef]
16. Connolly, C.I.; Burns, J.B.; Weiss, R. Path planning using Laplace's equation. In Proceedings of the IEEE International Conference on Robotics and Automation, Cincinnati, OH, USA, 13–18 May 1990; pp. 2102–2106.
17. Waydo, S.; Murray, R.M. Vehicle motion planning using stream functions. In Proceedings of the 2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422), Taipei, Taiwan, 14–19 September 2003; Volume 2, pp. 2484–2491.
18. Gingras, D.; Dupuis, E.; Payre, G.; Lafontaine, J. Path Planning Based on Fluid mechanics for mobile robots used Unstructured Terrain models. In Proceedings of the IEEE International Conference on Robotics and Automation, Anchorage, AK, USA, 3–7 May 2010.
19. Saudi, A.; Sulaiman, J. Path Planing for mobile robots using 4EGSOR via Nine-Point Laplacian (4EGSOR9L) Iterative method. *Int. J. Comput. Appl.* **2012**, *53*, 38–42.
20. Saudi, A.; Sulaiman, J.; Ahmad Hijazi, M.H. Robot Path Planing with EGSOR Iterative Method using Laplacian Behaviour-Based Control (LBBC). In Proceedings of the 5th International Conference on Intelligent Systems, Modelling and Simulation, Langkawi, Malaysia, 27–29 January 2014; pp. 87–91.
21. Falcó, A.; Nouy, A. Proper Generalized Decomposition for Nonlinear Convex Problems in Tensor Banach Spaces. *Numer. Math.* **2012**, *121*, 503–530. [CrossRef]
22. Chinesta, F.; Leygue, A.; Bordeu, F.; Aguado, J.V.; Cueto, E.; Gonzalez, D.; Alfaro, I.; Ammar, A.; Huerta, A. PGD-Based Computational Vademecum for Efficient Design, Optimization and Control. *Arch. Comput. Methods Eng.* **2013**, *20*, 31–49. [CrossRef]
23. Chinesta, F.; Keunings, R.; Leygue, A. *The Proper Generalized Decomposition for Advanced Numerical Simulations: A Primer*; Springer Briefs in Applied Science and Technology; Springer: Berlin/Heidelberg, Germany, 2014.
24. Montés, N.; Chinesta, F.; Falco, A.; Mora, M.C.; Hilario, L.; Nadal, E.; Duval, J.L. Towards a PGD-based Computational Vademecum for robot path planning. In *Informatics in Control, Automation and Robotics, Proceedings of the ICINCO 2019, Prague, Czech Republic, 29–31 July 2019;* Lecture Notes in Electrical Engineering; Gusikhin, O., Madani, K., Zaytoon, J., Eds.; Springer: Cham, Switzerland, 2020; Volume 720, pp. 1–15.
25. Akishita, S.; Kawamura, S.; Hayashi, K. New navigation function utilizing hydrodynamic potential for mobile robot. In Proceedings of the IEEE International Workshop on Intelligent Motion Control, Istanbul, Turkey, 20–22 August 1990; Volume 2, pp. 413–417.

26. Akishita, S.; Hisanobu, T.; Kawamura, S. Fast path planning available for moving obstacle avoidance by use of Laplace potential. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Yokohama, Japan, 26–30 July 1993; Volume 1, pp. 673–678.
27. Guldner, J.; Utkin, V.I.; Hashimoto, H. Robot obstacle avoidance in n-dimensional space using planar harmonic artificial fields. *J. Dyn. Syst. Meas. Control* **1997**, *119*, 160–166. [CrossRef]
28. Keymeulen, D.; Decuyper, J. The fluid dynamics applied to mobile robot motion: The stream field method. In Proceedings of the IEEE International Conference on Robotics and Automation, San Diego, CA, USA, 8–13 May 1994; pp. 378–385.
29. Li, Z.X.; Bui, T.D. Robot path planning using Fluid Model. *J. Intell. Robot. Syst.* **1998**, *21*, 29–50. [CrossRef]
30. Rosell, J.; Iniguez, P.A. Hierarchical and dynamic method to compute harmonic functions for constrained motion planning. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Lausanne, Switzerland, 30 September–4 October 2002; Volume 3, pp. 2335–2340.
31. Sato, K. Deadlock-free motion planning using the Laplace potential field. *Adv. Robot.* **1993**, *7*, 449–461. [CrossRef]
32. Sullivan, J.; Waydo, S.; Campbell, M. Using stream functions for complex behavior and path generation. In Proceedings of the AIAA Guidance, Navigation, and Control Conference, Austin, TX, USA, 11–14 August 2003.
33. Mora, M.C.; Tornero, J. Predictive and Multirate Sensor-Based Planning Under Uncertainty. *IEEE Trans. Intell. Transp. Syst.* **2015**, *16*, 1493–1504. [CrossRef]
34. Kashiwa, B.; Lee, W.H. Comparisons between the cell-centered and staggered mesh Lagrangian hydrodynamics. In *Advances in the Free-Lagrange Method Including Contributions on Adaptive Gridding and the Smooth Particle Hydrodynamics Method, Proceedings of the Next Free-Lagrange Conference Held at Jackson Lake Lodge, Moran, WY, USA, 3–7 June 1990;* Lecture Notes in Physics; Trease, H.E., Fritts, M.F., Crowley, W.P., Eds.; Springer: Berlin/Heidelberg, Germany, 1990; Volume 395.
35. Montes, N.; Rosillo, N.; Mora, M.C.; Hilario, L. A Novel Real-Time MATLAB/Simulink/LEGO EV3 Platform for Academic Use in Robotics and Computer Science. *Sensors* **2021**, *21*, 1006. [CrossRef] [PubMed]
36. Montés, N.; Chinesta, F.; Falco, A.; Mora, M.C.; Hilario, L.; Rosillo, N. Embedded PGD-Vademecum Tests in a LEGO Mindstorms EV3. 2017. Available online: https://www.youtube.com/watch?v=LC_kFZPmOH0 (accessed on 7 June 2021).
37. Ammar, A.; Chinesta, F.; Falco, A. On the convergence of a Greedy Rank-one update algorithm for a class of linear systems. *Arch. Comput. Methods Eng.* **2010**, *17*, 473–486. [CrossRef]
38. Bacha, A.; Bauman, C.; Faruque, R.; Fleming, M.; Terwelp, C.; Reinholtz, C.; Hong, D.; Wicks, A.; Alberi, T.; Anderson, D.; et al. Odin: Team VictorTango's entry in the DARPA Urban Challenge. *J. Field Robot.* **2008**, *25*, 467–492. [CrossRef]
39. Falco, A.; Hilario, L.; Montes, N.; CMora, M.; Nadal, E. A Path Planning Algorithm for a Dynamic Environment Based on Proper Generalized Decomposition. *Mathematics* **2020**, *8*, 2245. [CrossRef]