2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Reinforcement Learning-Based Optimal Multiple Waypoint Navigation

†Christos Vlachos, ‡Panagiotis Rousseas, †Charalampos P. Bechlioulis, and ††Kostas J. Kyriakopoulos

Abstract— **In this paper, a novel method based on Artificial Potential Field (APF) theory is presented, for optimal motion planning in fully-known, static workspaces, for multiple final goal configurations. Optimization is achieved through a Reinforcement Learning (RL) framework. More specifically, the parameters of the underlying potential field are adjusted through a policy gradient algorithm in order to minimize a cost function. The main novelty of the proposed scheme lies in the method that provides optimal policies for multiple final positions, in contrast to most existing methodologies that consider a single final configuration. An assessment of the optimality of our results is conducted by comparing our novel motion planning scheme against a RRT* method.**

I. Introduction

Motion planning is the problem of constructing a safe path between an initial and a goal position. A lot of effort has gone into establishing control techniques that equip robots with the ability to safely converge to the desired goal position while also ensuring optimality. Despite the existence of a plethora of tools that tackle the motion planning task, there is still room for exploration of novel solutions and improvement of existing ones. In this work, we present a novel parametric controller to tackle the objective of visiting a pre-determined set of waypoints within a known, static and bounded workspace. A robust framework for the solution of the aforementioned problem is provided through a controller that enables a robot to navigate safely towards any goal-position within a subset of the workspace. RL plays a key role in our work, in providing the optimal set of the controller's parameters.

The motivation behind our problem formulation stems from tasks such as a robot visiting different areas of a warehouse to execute tasks or performing monitoring duties in certain areas, etc. In such cases, different points of interest need to be visited over and over again throughout the robot's worklife, thus rendering indispensable a computationally efficient way of navigating between waypoints. Our goal is to design an optimal and robust motion planning scheme that tackles changes in the robot's goal position efficiently by taking advantage of the fact that while in most real-world applications only specific starting-ending point combinations are needed, the exact final position is not critical from a task-planning perspective. While the use of online approaches presents advantages with respect to computational complexity compared to traditional offline solutions, we aim at eliminating the need for excessive, on-board computational power in fully-known, static workspaces by finding a set of parametric functions w.r.t. the waypoint, that will allow the robot to navigate freely to the latter and at the same time account for possible disturbances or necessary changes, in the robot's goal position.

†The authors are with the Department of Electrical and Computer Engineering, University of Patras, Greece ‡The author is with the School of Mechanical Engineering, Control Systems Laboratory, National Technical University of Athens, Greece. ††The author is with the Center of AI & Robotics (CAIR), New York University, Abu Dhabi E-mails: {up1103869,chmpechl}@upatras.gr, prousseas@mail.ntua.gr,kk4812@nyu.edu

II. Related Work

Research on the motion planning problem has yielded many fundamentally different solutions. Discrete/graph-based methods consist of a subcategory of the formulated approaches that includes traditional algorithms (e.g., Rapidly-exploring Random Trees (RRT) algorithm [1], its variant RRT* [2], Probabilistic Roadmaps (PRM) [3]) and modern ones that rely on Machine Learning (e.g., Q-learning and policy gradient algorithms [4], RL with PRM [5]). On the other hand, the APF approach [6] involves modeling the robot as a particle, moving according to a potential field constructed over a specific workspace. By following the negative gradient of the potential field, the robot is able to safely reach the goal position. However, the method's susceptibility to local minima led to the development of Navigation Functions (NF) [7] and APFs based on harmonic functions, namely Artificial Harmonic Potential Fields (AHPFs) [8]. Regarding NF, extensive tuning is required to get rid of local minima, however, since AHPFs are free of local minima by construction, the aforementioned drawback is resolved. Optimality in APF methods has been achieved by tuning the underlying parameters through RL [9], Genetic Algorithms [10] and Evolutionary Algorithms [11]. Finally, regarding the multiple waypoint navigation problem, the latter has been treated in a variety of ways, such as the Multi-RRT* algorithm [12], a D*-Lite algorithm combined with Vector Field Histogram (VFH) based local navigation [13] and an improved A* algorithm with a dynamic window approach [14].

In this work, the aforementioned navigation problem is addressed through an AHPF-based method. A solution includes two steps, namely, the waypoint-to-waypoint navigation and the optimal visiting sequence. After assigning a safe navigation policy to each of the robot's waypoints, first, RL is employed to optimize the inter-waypoint navigation policy and subsequently, the optimal visiting order is computed by solving an Asymmetric Travelling Salesman Problem (ATSP) [15]. As opposed to existing literature, our approach deviates from established ones by providing an optimal solution for navigation between sets of final configurations within the workspace, rather than waypoints defined by a

single goal-position. To accomplish this, our method includes assigning an area around each point of interest and designing a controller, whose parameters are tuned such that the robot navigates safely and optimally towards any goal position that lies inside this area.

The AHPF-based policy, which is traditionally acquired through a weighted sum of constant weights, is further enhanced through rendering the weights functions of the goal configuration, thus generalizing the AHPF-based structure to multiple final positions. To address optimality, the optimal motion planning problem is treated as a RL one, and a Projected Gradient Descent (PGD) algorithm is formulated in order to acquire the optimal motion planning policy parameters. The projection ensures that the choice of the AHPF parameters doesn't jeopardize the robot's safety. The implementation of RL plays an important role, as it bypasses solving a very hard non-linear partial differential equation for extracting the optimal cost function.

III. Problem Formulation

Consider a point-robot[1] operating within a two-dimensional, bounded and connected set $\mathcal{G} \subset \mathbb{R}^2$, with inner distinct obstacles $\mathcal{O}_i \subset \mathcal{G},\ i = 1, \dots, M$. We define the robot's workspace as the set $\mathcal{W} = \mathcal{G} - \bigcup_{i=1}^{M} \mathcal{O}_i$ and adopt $\partial\mathcal{W}$ to denote the boundary of $\mathcal{W}$. In addition, consider a set of waypoints $\mathcal{S}_i \subset \mathcal{W} - \partial\mathcal{W},\ i = 1, \dots, J$ where S_i is a subset of the workspace. The robot's motion is dictated by the single integrator dynamics:

$$\dot{p} = u,\ p(0) = \bar{p} \in \mathcal{W}, \tag{1}$$

where $p \in \mathcal{W}$ is the robot's position, $u = u(t)$ is the input velocity and $\bar{p}$ denotes the robot's initial position. We consider the problem of developing a control policy u that minimizes the following cost function:

$$V(\bar{p}; p_d) = \int_0^\infty \left[Q(p(\tau; \bar{p}); p_d) + R(u(\tau))\right] d\tau\ \forall \bar{p} \in \mathcal{W}, \tag{2}$$

which consists of a state-related term $Q(p(\tau; \bar{p}); p_d) = \alpha \left\| p(\tau; \bar{p}) - p_d \right\|^2$ and an input related term $R(u(\tau)) = \beta \left\| u(\tau) \right\|^2$, where $p(t; \bar{p})$ is the solution of $\dot{p} = u$, from the initial position $\bar{p} = p(0)$ and p_d denotes the goal position. The R term minimizes the input energy and the Q term penalizes the robot for staying away from the desired goal position as time evolves, while $\alpha, \beta > 0$ are weighting parameters. Our goal is to find a navigation policy for each subset $\mathcal{S}_i$ that minimizes the cost function (2) for all goal-positions that lie within the aforementioned set.

IV. Methodology

In this section, we first discuss the AHPFs implemented in our work and then build the From-All-To-A-Point (FATAP) controller, which is a parametric controller that allows a robot to safely navigate to a single goal position. Furthermore, based on the FATAP controller, we construct the novel From-All-To-A-Subset (FATAS) controller, that enables a robot to navigate to any goal within a subset of its workspace. Finally, we employ RL to tune the controller's parameters so that optimality is achieved.

A. The AHPF-Controller

The AHPF that is used in this work, is derived through the panel method [8] and is defined as follows:

$$\Phi(p; p_C) = \sum_{i=0}^{K} \phi(p; p_i) w_i = w^\intercal \phi(p; p_C), \tag{3}$$

where $p_C \triangleq \{p_0, p_1, p_2, \cdots, p_K\},\ p_0 = p_d \in \mathcal{W} - \partial\mathcal{W},\ p_i \in \mathcal{W}',\ i = 1, \dots, K$ is a (K+1)-tuple containing the centers of the harmonic basis functions ϕ_i, which are placed outside of the workspace except for p_0. $w \triangleq [w_0, w_1, \dots, w_K]^\intercal$ is a vector containing the respective weights of each basis function ϕ_i. We form the basis functions' set according to [9] (see [8], [9]).

$$\phi(p; p_C) \triangleq [\phi(p; p_d), \phi(p; p_1), \cdots, \phi(p; p_K)]^\intercal : \mathcal{W} \to \mathbb{R}^{K+1}. \tag{4}$$

Hence, a parametric controller for safe navigation towards a single desired goal position $p_d \in \mathcal{W} - \partial\mathcal{W}$ is [16]:

$$u(p) = -\left\| p - p_d \right\|^2 \nabla\phi^\intercal(p; p_C) w, \tag{5}$$

where in order to obtain a safe field during navigation tasks, a set of linear inequalities w.r.t the weights of the basis functions is imposed:

$$A^\intercal w \leq 0, \tag{6}$$

where $A = [A_1, \dots, A_N]$, with $A_i = n^\intercal(z_i) \nabla\phi^\intercal(z_i; p_C)$. and N is the number of boundary points used. These conditions essentially guarantee that the velocity of the robot points inwards at the workspace boundary.

We will now present the FATAS parametric controller based on the FATAP controller. Briefly, a set of constant weights for the FATAP controller corresponds to a single point of the mapping from goal positions to harmonic-like potential fields. Our goal is to find exactly this mapping over a set contatining the aforementioned goal position. To accomplish this, consider the following parametrization for the weights of the FATAP controller:

$$w = w(p_d; p_{Cd}) = W^\intercal s(p_d; p_{Cd}). \tag{7}$$

The matrix $W \in \mathbb{R}^{K_c \times (K+1)}$ is a new set of parameters, where $p_{Cd} = \{p_{d,1}, p_{d,2}, \cdots, p_{d,K_C}\},\ p_{d,i} \in \mathbb{R}^2,\ i = 1, \dots, K_C$ are the positions of the centers of a new set of basis functions, with the latter defined as the mapping $s(p_d; p_{Cd}) : \mathcal{S} \to \mathbb{R}^{K_c}$ for a given choice of basis functions' centers. $\mathcal{S}$ denotes the subset over which our controller will provide safe navigation policies. The key insight here is that for a given weight matrix W and a constant desired position $p_d \in \mathcal{S} \subset \mathcal{W} - \partial\mathcal{W}$, Eq. (7) yields constant weights $w \in \mathbb{R}^{K+1}$ and thus the field :

$$\Phi(p; p_d; p_C) = \phi^\intercal(p; p_C) W^\intercal s(p_d; p_{Cd}),\ p_d \in \mathcal{S}, \tag{8}$$

[1] A disk robot can also be treated through inflating the boundary of the workspace inwards by a distance equal to the robot's radius. An example of such a transformation is: $T : \partial\mathcal{W} \to \partial\mathcal{W}',\ p' = T(p) = p + R \times n(p),\ \forall p \in \partial\mathcal{W}$, where R denotes the robot's radius, and $n(p)$ denotes the unitary, inwards-pointing normal vector to the boundary at each point $p \in \partial\mathcal{W}$, and $\partial\mathcal{W}'$ denotes the transformed boundary.

is harmonic, and its gradient becomes:

$$\nabla\Phi(p;p_d;p_C) = \nabla\phi^\intercal(p;p_C)W^\intercal s(p_d;p_{Cd}),\ p_d \in \mathcal{S}. \quad (9)$$

However, the field (8) is not apparently, explicitly linear w.r.t. the new parameters. Nevertheless, note that by employing the identity $\text{vec}(AXB) = (B^\intercal \otimes A)\text{vec}(X)$, equation (9) becomes:

$$\begin{aligned}&\text{vec}(\nabla\Phi(p;p_d;p_C)) = \nabla\Phi(p;p_d;p_C) = \\ &(s^\intercal(p_d;p_{C_d}) \otimes \nabla\phi^\intercal(p;p_C))\text{vec}(W^\intercal),\ p_d \in \mathcal{S}, \end{aligned} \quad (10)$$

where $s^\intercal(p_d;p_{C_d}) \otimes \nabla\phi^\intercal(p;p_C) \in \mathbb{R}^{2\times K_c(K+1)}$ and $\hat{w} = \text{vec}(W^\intercal) \in \mathbb{R}^{K_c(K+1)}$. Similarly, the safety conditions boil down to:

$$n^\intercal(z)(s^\intercal(p_d;p_{Cd})\otimes\nabla\phi^\intercal(z;p_C))\hat{w} \leq 0, \forall z \in \partial\mathcal{W}, \forall p_d \in \mathcal{S}, \quad (11)$$

which is evidently linear w.r.t. $\hat{w}$. Although the safety condition (11) involves every goal position inside the subset $\mathcal{S}$, it can be relaxed if it holds for a finite set of goals $p_G \triangleq \{p_{g,1}, p_{g,2}, \cdots, p_{g,\hat{M}}\},\ p_{g,j} \in \mathcal{S},\ j = 1,\ldots,\hat{M}$, as a system of linear inequalities w.r.t the parameters $\hat{w}$:

$$\tilde{A}^\intercal\hat{w} \leq 0, \quad (12)$$

where $\tilde{A} = [\tilde{A}_{11}, \tilde{A}_{12}, \ldots, \tilde{A}_{N\hat{M}}]$, with $\tilde{A}_{ij} = [s^\intercal(p_{gj};p_{Cd})] \otimes (n^\intercal(z_i)\nabla\phi^\intercal(z_i;p_C))]$.

B. Initial Policy

An initial safe and convergent policy comes at the expense of solving a constrained quadratic problem, with exponentially increasing complexity, in order to obtain the parameters W:

$$\min_{\hat{w}} \|\hat{w}\|^2,\ s.t.\ \tilde{A}^\intercal\hat{w} \leq -\epsilon,\ \ w_0(p_d) = W_1^\intercal s(p_d;p_{Cd}) > 0, \quad (13)$$

where $\epsilon > 0$ is a small constant and $w_0(p_d)$ is the weighted sum that is equal to the weight that corresponds to the single harmonic term assigned to the robot's goal position. To ensure that the obtained initial policies belong to the same family of solutions, the set of basis functions is extended in the following way[2]:

$$s(p_d;p_{Cd}) = [1,\ s(p_d;p_{Cd})], \quad (14)$$

while the following constraint for the first column of W is further imposed:

$$W_1^\intercal = \begin{bmatrix}1 & 0 & \ldots & 0\end{bmatrix}, \quad (15)$$

and thus:

$$W_1^\intercal s(p_d;p_{Cd}) = 1. \quad (16)$$

The solution to the quadratic problem (13) results in an initial policy, which guarantees safety and convergence for any goal position within the subset $\mathcal{S}$.

• **Choice of Basis Functions** We propose employing Radial Basis Functions (RBFs) with uniformly placed centers, owing to their approximation capabilities. A RBF is a function whose value depends only on the distance between its argument and its center. As the distance between the argument and the center increases, the value of the RBF decreases, therefore RBFs can be used to capture the local behaviour of a function close to the center. RBF approximation [17] is universally applicable to both higher and lower dimensions due to the excellent approximation properties of RBFs [18]. In this work we employ the commonly used Gaussian functions $\phi(r) = e^{-(\epsilon r)^2}$, which belong to the class of infinitely smooth RBFs, where in our case $r = \|p_d - p_{d,i}\|$ is the euclidean distance between the goal position and the i_{th} center and ϵ is a tunable shape parameter that controls the decay of the RBF. The tuning parameter ϵ is chosen in a heuristic way, so that two neighbouring RBFs with common weights overlap at a constant value c.

C. Optimal Motion Planning using Reinforcement Learning

In this subsection, we treat the optimal motion planning problem as a RL one. Based on the adopted cost function (2), we define the associated Hamiltonian as follows:

$$H(p,u,\nabla V) = \nabla V^\intercal u + \alpha\|p - p_d\|^2 + \beta\|u\|^2. \quad (17)$$

The Hamilton-Jacobi-Bellman (HJB) optimality condition is given by:

$$H(p,u^*,\nabla V^*) = 0, \quad (18)$$

where u^* denotes the optimal input and V^* the optimal cost function, while the optimal control policy can be extracted through the stationary condition on (17) $\frac{\partial H(p,u,\nabla V^*)}{\partial u}|_{u=u^*} = 0$ as follows:

$$u^* = -\frac{1}{2\beta}\nabla V^*(\bar{p};p_d). \quad (19)$$

To obtain the optimal control policy u^*, an analytic expression for the cost function $V^*(p;p_d)$ is required. Substituting the optimal control policy in the HJB optimality condition (18) results in a hard non-linear Partial Differential Equation (PDE). The difficulty in solving this PDE stems from the non-trivial solution domain. In order to circumvent solving the PDE, we employ a policy gradient optimization technique to satisfy both the stationary condition as well as the safety over the workspace boundary.

Through the RL point of view [19], the robot is interpreted as an agent interacting with its environment, i.e., the workspace $\mathcal{W}$. By sensing its state (i.e., position) $p \in \mathcal{W}$, the robot takes action (i.e., input velocity) $u \in \mathbb{R}^2$ through the parametrized policy (i.e., parametric controller) $u_{\hat{w}}(p)$. The value function, corresponds to the cost function (2), which in our case is subject to minimization. Since the robot's policy consists of a parametric controller, and due to the continuous state and action spaces, the herein treated problem is a continuous, deterministic, model-based RL one. The minimization of the cost function (2) rests in finding the optimal control policy u^*/optimal cost V^* pair that satisfies the stationary condition (19). Hence, the solution of the RL problem is reduced to finding the parameters W^* that satisfy these conditions. The policy gradient optimization algorithm used in this work is based on Policy Iteration (see Alg.

[2] This is necessary owing to the AHPF's scale invariance.

1), which is a widely used technique that solves the HJB equation [20]. The parameters W can be iteratively obtained

Algorithm 1: Off-line Policy Iteration

Given an admissible control policy $u^{(0)}$ and tolerance $\epsilon > 0$

while $\left\|u^{(i+1)} - u^{(i)}\right\| > \epsilon$ **do**

- Calculate $\nabla V^{(i)}$ based on policy $u^{(i)}$
- Update the control policy: $u^{(i+1)} = -\frac{1}{2\beta}\nabla V^{(i)}$
- $i \leftarrow i + 1$

end

by solving the following constrained quadratic problem:

$$\min_{\hat{w}} : \left\| u_{\hat{w}}^{(i+1)} + \frac{1}{2\beta}\nabla V^{(i)}(p; p_d) \right\|^2 \quad \forall p_d \in p_G \tag{20}$$
$$s.t.\ \tilde{A}^\intercal \hat{w} \leq 0,$$

where:

$$u_{\hat{w}}^{(i+1)} = \sigma^\intercal \left(W^{(i+1)}\right)^\intercal s(p_d; p_{Cd})$$
$$\sigma = -\left\|p - p_d\right\|^2 \nabla\phi(p; p_C),$$

and p_G denotes the set of goals used in the safety condition (11). We will address the calculation of the gradient of the cost function in the sequel. To avoid the computational burden of solving a constrained quadratic problem we employ a Projected Gradient Descent (PGD) algorithm, which is a policy gradient RL technique used to solve constrained optimization problems.

D. Projected Gradient Descent

First, the following objective function is defined, which should be minimized in order to acquire the best implementable policy according to the information encoded in the cost function V at each iteration:

$$M(W) = \left\| \sigma^\intercal(p) W^\intercal s(p_d; p_{Cd}) + \frac{1}{2\beta}\nabla V^{(i)}(p; p_d) \right\|^2. \tag{21}$$

The controller's parameters are then updated as follows:

$$\hat{w}^{(i+1)} = \hat{w}^{(i)} - a \cdot \mathrm{vec}\left(\nabla_W^\intercal M(W)\right), \tag{22}$$

where:

$$\nabla_W M(W) = 2(s(p_d; p_{Cd}) \otimes \sigma(p))(\sigma^\intercal(p) W^\intercal s(p_d; p_{Cd}) + \frac{1}{2\beta}\nabla V^{(i)}(p; p_d)) \in \mathbb{R}^{(K+1)\times(Kc+1)}, \tag{23}$$

is the steepest descent direction and a is the learning rate.

• **Gradient of Cost Function.** To obtain the gradient of the cost function V, first an estimate of the cost function at each iteration is necessary. Such an estimate can be obtained by executing trajectories from initial points across the boundary of the robot's workspace $\mathcal{W}$, to the goal-positions $p_d \in p_G$. The cost computation can be accomplished through an additional variable to system (1):

$$\begin{bmatrix} \dot{p}(t) \\ \dot{v}_{p_d}(t) \end{bmatrix} = \begin{bmatrix} u(p) \\ r(p,u) \end{bmatrix}, \begin{bmatrix} p(0) \\ v_{p_d}(0) \end{bmatrix} = \begin{bmatrix} \bar{p}_s \\ 0 \end{bmatrix} \ \forall p_d \in p_G, \tag{24}$$

where $r(p,u) = \alpha \left\|p - p_d\right\|^2 + \beta \left\|u\right\|^2$. The ODE (24) results in points $p_i = p(t_i)$ over the trajectories and associated cost values $\hat{V}(p(t_i); p_d) = v_{p_d}(p_d) - v_{p_d}(p(t_i))$. Repeating this process for a finite number of uniformly distributed boundary points, and for all $p_d \in p_G$, allows to obtain the cost-to-go approximation $\hat{V}(p; p_d)$ through interpolation, for all $p \in \mathcal{W}$, for all $p_d \in p_G$. With an estimate for the cost-to-go available, its gradient for $K_j, j \in \{1, \cdots, \hat{M}\}$ points in a two-dimensional grid inside the workspace can be extracted. To calculate the steepest descent direction (23), we will employ the gradient of each point $\tilde{p}$ in the chosen grid and calculate the mean value of equation (23) for all points:

$$\nabla_W M(W) = \frac{2}{\hat{M}} \sum_{m=1}^{\hat{M}} \sum_{k=1}^{K_m} \frac{1}{K_m} (s(p_{g,m}; p_{Cd}) \otimes \sigma(\tilde{p}_k)) (\sigma^\intercal(\tilde{p}_k) W^\intercal s(p_{g,m}; p_{Cd}) + \frac{1}{2\beta}\nabla V(\tilde{p}_k; p_{g,m})) \tag{25}$$

Using points from all over the workspace results in a Full-Batch training algorithm. However, executing trajectories from a large number of initial boundary points increases the computational load of the method. To remedy this, we restrict the number of trajectories to a randomly chosen subset of the boundary points for each instance of calculating the steepest descent direction. By choosing this subset so as to include only neighbouring points, an estimate of the cost function in a subset of the workspace is acquired, resulting in a Mini-Batch training algorithm.

• **Learning Rate.** The adaptive method we used to calculate the learning rate a is derived from ADADELTA [21]. Our method inherits important benefits of the ADADELTA approach, including robustness to noisy gradients and the lack of dependence on a manual learning rate with the difference that our method produces a global learning rate by using the information from the gradients of all parameters. The aforementioned algorithm is presented in Alg. (2).

• **Projection onto Convex Sets.** We employ a projection in the Gradient Descent algorithm to ensure that the updated underlying AHPF parameters satisfy the safety conditions. Each inequality $\tilde{A}_{ij}\hat{w} \leq 0$ in the safety condition (12) defines a closed half-space, and is therefore a convex set [22]. The intersection of all the closed half-spaces defines a convex polyhedral $\mathcal{Q} = \{\hat{w} \in \mathbb{R}^{(K_c+1)(K+1)} : \tilde{A}^\intercal \hat{w} \leq 0\}$. To accomplish the projection, we employ a cyclic projection algorithm [23]. The cyclic projection algorithm allows finding the orthogonal projection of a point in the intersection of a collection of convex sets by applying a sequence of projections, individually onto each set. Projection onto each individual convex set defined by the closed half-space

Algorithm 2: Computing learning rate update at iteration i

Given a decay rate ρ , a constant ϵ and initial parameters $w^{(0)} = \text{vec}\left((W^{(0)})^\intercal\right)$

- Initialize accumulation variables $E[step_w^2]_0 = 0$, $E[\Delta w^2]_0 = 0$

for $i = 1 : N$ **do**

- Compute gradient $step_w^{(i)} = \nabla_W M(W)$
- Accumulate Gradient $E[step_w^2]_i = \rho E[step_w^2]_{(i-1)} + (1-\rho) step_w^{2(i)}$
- Compute Update: $\Delta w_i = -\frac{RMS[\Delta w]_{(i-1)}}{RMS[step_w]_i} step_w^{(i)}$ where $RMS[x]_i = \sqrt{E[x^2]_i + \epsilon}$
- Accumulate Updates: $E[\Delta w^2]_i = \rho E[\Delta w^2]_{i-1} + (1-\rho)\Delta w_i^2$
- Apply Update : $w^{i+1} = w^i + \Delta w_i$

end

$\tilde{A}_{ij}\hat{w} \leq 0$, is accomplished through the projection operator:

$$\mathcal{P}_{ij} = \text{vec}\left(W^\intercal\right) - \frac{\text{vec}(W^\intercal)\hat{n_{ij}}}{\|n_{ij}\|^2}\hat{n}_{ij}, \tag{26}$$

where $\hat{n}_{ij} = \frac{\tilde{A}_{ij}}{\|\tilde{A}_{ij}\|}$ denotes the normal vector to the hyperplane $\tilde{A}_{ij}\hat{w} = 0$. If we denote $\mathcal{T}$ as the product $\mathcal{T} = \mathcal{P}_{11}\mathcal{P}_{12}...\mathcal{P}_{N\hat{M}}$ then the orthogonal projection of the weights $\hat{w}$ onto the convex polyhedral $\mathcal{Q}$ is obtained through

$$\text{vec}\left(W_{proj}^\intercal\right) = \mathcal{P}_{\mathcal{Q}}(\text{vec}\left(W^\intercal\right)) = \mathcal{T}^r\left(\text{vec}\left(W^\intercal\right)\right), as\ r \to \infty \tag{27}$$

- **Complete Algorithm.** In Alg. (3) the complete PGD algorithm is presented. Note that even though the parameters W are updated on each iteration, the gradient of the cost function V is only recalculated every k iterations. Through minimizing the objective function $M(W)$ in the i_{th} iteration, the policy $u^{(i+1)}$ is ameliorated according to the available data $-\frac{1}{2\beta}\nabla V^{(i)}$. By recalculating ∇V after performing only a single step towards the descent direction, the current data has not been fully taken advantage of. However, choosing a very large value for k results in the algorithm stagnating due to irrelevant data. A properly tuned value for k allows improving the policies faster and more accurately, laying the groundwork for faster convergence of our algorithm.

V. Results

In this section, we present the results of the proposed method in solving the multiple waypoint navigation problem. All simulations were implemented in Matlab, on a PC running Windows 10, on an intel-i5 dual-core processor, with 8GB RAM. We consider the synthetic workspace of Fig. (1a). The aim is to provide an optimal solution to the navigation problem of a robot, whose goal is to visit the four square-shaped waypoints. A unique number is also assigned to each waypoint. After obtaining a set of initial policies, the PGD algorithm was used to optimize their respective parameters. In all simulations, the cost parameters are chosen as $\alpha = 0.5$ and $\beta = 0.5$. In the PGD algorithm, the successive number of steps without recalculating the gradient of the cost function is set as $k = 5$ and finally, the overlapping parameter is selected as $c = 0.6$.

Algorithm 3: On-Policy Projected Gradient Descent

Given initial parameters vec $\left((W^{(0)})^\intercal\right)$ and number of steps $k > 0$ without recalculating $\nabla V\ \forall p_d \in p_G$

- Set $i = 0$

while *vec*$(W^\intercal)$ *has not converged* **do**

- Calculate $\nabla V^{(i)}$ using Mini-Batches
- Compute descent direction $step_{\hat{w}}^{(i)} = \nabla_W M(W)$
- Calculate learning rate $a^{(i)}$
- Update parameters vec $\left(\left(W^{(i+1)}\right)^\intercal\right) = \text{vec}\left((W^{(i)})^\intercal\right) - a^{(i)} step_{\hat{w}}^{(i)}$
- Projection onto safe set: vec $\left(\left(W_{proj}^{(i+1)}\right)^\intercal\right) = \mathcal{P}_{\mathcal{Q}}(\left(W^{(i+1)}\right)^\intercal)$: $\mathcal{Q} = \{\hat{w} \in \mathbb{R}^{(K_c+1)(K+1)} : \tilde{A}^\intercal\hat{w} \leq 0\}$
- $i \leftarrow i + 1$

if $mod(i,k)! = 0$ **then**

$\nabla V^{(i+1)} = \nabla V^{(i)}$ and skip the next ∇V calculation

end

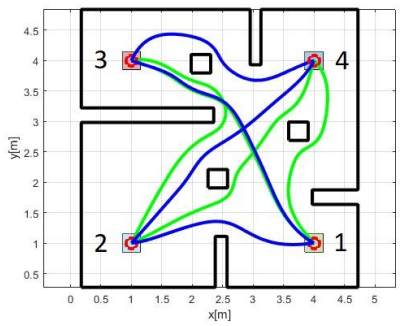
end

In Table I, the initial and final transition costs matrices are summarized, for goal positions at the center of each waypoint. The values inside the parentheses represent the final costs. Choosing an initial route for the robot consisting of the cyclic permutation of waypoints (2 3 1 4), according to Table II, the sum of the initial transition costs amounts to 214.6. After implementing our method to optimize the policies, the sum of the final transition costs becomes 145.13 Moreover, by solving an ATSP, we obtain the overall optimal solution, which consists of the cycle (4 3 1 2), with a corresponding cost of 50.37, leading to a 76.64% decrease of the total cost. In Fig. (1a), the initial (green) and final (blue) trajectories of the robot are shown.

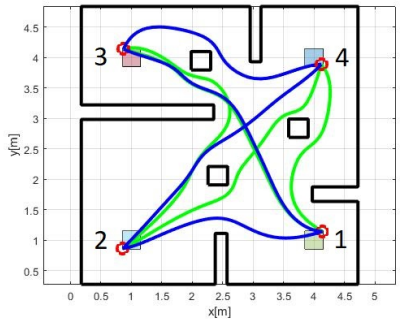
TABLE I: Initial vs Final Transition Costs (w/out Disturbance)

Start \ End	1) (4, 1)	2) (1, 1)	3) (1, 4)	4) (4, 4)
1) (4, 1)	-	7.25 (6.38)	100.68 (63.36)	7.65 (6.07)
2) (1, 1)	7.94 (5.64)	-	153.51 (112.91)	19.28 (9.99)
3) (1, 4)	31.31 (12.94)	40.20 (26.89)	-	9.71 (5.97)
4) (4, 4)	9.25 (5.83)	22.13 (13.22)	22.73 (21.06)	-

To evaluate the efficiency of the proposed method, consider the case of a disturbance to the robot's goal positions. The initial and final transition cost matrices are summarized in Table III. In addition, the comparison between the initial and the final transition costs for each cycle is depicted in Table IV. It is evident that despite changing the goal positions



(a) Trajectories w/out Disturbance



(b) Trajectories w. Disturbance

Fig. 1: Initial and Final Trajectories. The initial trajectories are depicted in green, while the final trajectories are depicted in blue.

TABLE II: Transition Costs for each Cycle (w/out Disturbance)

Cycle ╲ Sum	Initial Transition Costs	Final Transition Costs	Percentage of Decrease	RRT* Transition Costs
(4 3 1 2)	80.57	50.37	37.48%	124.84
(4 3 2 1)	78.52	59.66	24.02%	87.42
(3 4 2 1)	140.47	88.19	37.22%	124.33
(1 3 2 4)	169.41	106.08	37.38%	129.8
(4 1 2 3)	179.72	131.09	27.06%	85.43
(2 3 1 4)	214.60	145.13	32.37%	130.64

of the robot, the method leads to similar results. In Fig. (1b) a comparison between the trajectories of the initial and the final cycle is shown.

TABLE III: Initial vs Final Transition Costs (w. Disturbance)

Start ╲ End	1) (4.14, 1.14)	2)(0.86, 0.86)	3)(0.87, 4.14)	4)(4.12, 3.89)
1) (4.14, 1.14)	-	9.04 (8.19)	117.47 (77.01)	6.49 (5.41)
2) (0.86, 0.86)	11.18 (6.81)	-	283.69 (195)	21.55 (10.90)
3) (0.87, 4.14)	47.61 (15.44)	63.78 (41.87)	-	11.89 (7.03)
4) (4.12, 3.89)	9.37 (5.62)	27 (15.83)	29.62 (28.83)	-

• **Comparison with RRT*.** To assess the optimality of our results, a comparison between our method and an RRT* approach is conducted. The RRT* algorithm we used [24] was adjusted appropriately, to additionally include sampling over the input space, for the same cost function (2) with the input velocity varying linearly w.r.t. time during transition between two points of the workspace. The algorithm was

TABLE IV: Transition Costs for each Cycle (w. Disturbance)

Cycle ╲ Sum	Initial Transition Costs	Final Transition Costs	Percentage of Decrease	RRT* Transition Costs
(4 3 1 2)	107.82	63.39	41.21%	143.08
(4 3 2 1)	111.07	82.92	25.34%	101.78
(3 4 2 1)	167.54	106.68	36.33%	144.91
(1 3 2 4)	212.17	135.40	36.18%	145.52
(4 1 2 3)	313.99	215.83	31.26%	95.1
(2 3 1 4)	364.80	231.68	36.49%	144.81

executed by sampling 700 nodes on the workspace, 50 trials were carried out and the results are summarized in Tables V - VI. The values inside the parentheses represent the standard deviations of the cost calculations. Our proposed method consistently outperforms the RRT* algorithm in most waypoint transitions, however the reverse also holds true for certain transitions. It should be noted, that in our case, the use of an AHPF-based controller, results in vector fields with incompressible flows and thus, the optimized trajectories are not necessarily of minimum length, resulting in certain transitions having a lower cost by using an RRT* approach. Finally, in Tables II and IV the transition costs of each cycle for the RRT* algorithm are shown for comparison.

TABLE V: RRT* Transition Costs (w/out Disturbance)

Start ╲ End	1) (4, 1)	2) (1, 1)	3) (1, 4)	4) (4, 4)
1) (4, 1)	-	20 (10.56)	40 (22.35)	20 (12.59)
2) (1, 1)	22.3 (14.83)	-	27.2 (20.36)	44.94 (28.76)
3) (1, 4)	41.41 (30)	26.63 (15.70)	-	20 (17.47)
4) (4, 4)	18.23 (9.44)	42.03 (21.10)	18.49 (9)	-

TABLE VI: RRT* Transition Costs (w. Disturbance)

Start ╲ End	1) (4.14, 1.14)	2)(0.86, 0.86)	3)(0.87, 4.14)	4)(4.12, 3.89)
1) (4.14, 1.14)	-	23.75 (14.08)	47.59 (30.33)	17.70 (11.29)
2) (0.86, 0.86)	28.40 (20.42)	-	32.41 (14.85)	48.43 (24)
3) (0.87, 4.14)	48.96 (30.53)	33.74 (19.39)	-	23.18 (16.27)
4) (4.12, 3.89)	15.76 (8.30)	45.74 (23.75)	21.94 (11.99)	-

VI. Conclusion - Future Work

While our method shows promising results, certain limitations need to be taken into consideration. Firstly, the proposed approach is not suitable for dynamic workspaces. Moreover, by developing a controller for single integrator dynamics, stochasticity and nonlinearities are not being addressed. In addition, our method can only be applied in the framework of two-dimensional navigation. Finally, regarding the computational complexity, the necessity to solve a constrained quadratic problem to obtain our controller's parameters, becomes a hindrance in more complex environments. Solving the aforementioned issues is an interesting direction we intend to pursue, while additionally expanding the RL framework by considering the decay parameters of the RBFs and the RBF centers' positions as additional parameters subject to optimization.

REFERENCES

[1] S. M. LaValle, "Rapidly-exploring random trees : a new tool for path planning," *The annual research report*, 1998.

[2] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[3] L. Kavraki, P. Svestka, J. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *Robotics and Automation, IEEE Transactions on*, vol. 12, pp. 566 – 580, 09 1996.

[4] C. Zhou, B. Huang, and P. Fränti, "A review of motion planning algorithms for intelligent robotics," 2021.

[5] A. Francis, A. Faust, H.-T. L. Chiang, J. Hsu, J. C. Kew, M. Fiser, and T.-W. E. Lee, "Long-range indoor navigation with prm-rl," 2019. [Online]. Available: https://arxiv.org/abs/1902.09458

[6] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, vol. 2, 1985, pp. 500–505.

[7] D. E. Koditschek and E. Rimon, "Robot navigation functions on manifolds with boundary," *Advances in Applied Mathematics*, vol. 11, no. 4, pp. 412–442, 1990.

[8] J. O. Kim and P. K. Khosla, "Real-time obstacle avoidance using harmonic potential functions," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 3, pp. 338–349, 1992.

[9] P. Rousseas, C. P. Bechlioulis, and K. J. Kyriakopoulos, "Optimal motion planning in unknown workspaces using integral reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 6926–6933, 2022.

[10] J. Amiryan and M. Jamzad, "Adaptive motion planning with artificial potential fields using a prior path," in *3rd RSI International Conference on Robotics and Mechatronics*, 2015, pp. 731–736.

[11] P. Vadakkepat, K. C. Tan, and W. Ming-Liang, "Evolutionary artificial potential fields and their application in real time robot path planning," in *Proceedings of the 2000 Congress on Evolutionary Computation. CEC00*, 2000, pp. 256–263.

[12] T. Bai, Z. Fan, M. Liu, S. Zhang, and R. Zheng, "Multiple waypoints path planning for a home mobile robot," in *2018 Ninth International Conference on Intelligent Control and Information Processing (ICICIP)*, 2018, pp. 53–58.

[13] K. Balan and C. Luo, "Optimal trajectory planning for multiple waypoint path planning using tabu search," in *2018 9th IEEE Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON)*, 2018, pp. 497–501.

[14] B. Zhang, W. Jin, X. Gao, and W. Chen, "A multi-goal global dynamic path planning method for indoor mobile robot," in *2021 3rd International Symposium on Robotics Intelligent Manufacturing Technology (ISRIMT)*, 2021, pp. 97–103.

[15] G. Ausiello, V. Bonifaci, and L. Laura, "The on-line asymmetric traveling salesman problem," *Journal of Discrete Algorithms*, vol. 6, no. 2, pp. 290–298, 2008, selected papers from CompBioNets 2004.

[16] P. Rousseas, C. Bechlioulis, and K. J. Kyriakopoulos, "Harmonic-based optimal motion planning in constrained workspaces using reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2005–2011, 2021.

[17] Z. Majdisova and V. Skala, "Radial basis function approximations: comparison and applications," *Applied Mathematical Modelling*, vol. 51, pp. 728–743, 11 2017.

[18] M. D. Buhmann, "Radial basis functions," *Acta Numerica*, vol. 9, p. 1–38, 2000.

[19] S. L. Brunton and J. N. Kutz, *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, 2019.

[20] F. L. Lewis, D. Vrabie, and K. G. Vamvoudakis, "Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers," *IEEE Control Systems Magazine*, vol. 32, no. 6, pp. 76–105, 2012.

[21] M. D. Zeiler, "Adadelta: An adaptive learning rate method," 2012. [Online]. Available: https://arxiv.org/abs/1212.5701

[22] C. H. Tan, "Convex sets and convex functions," 2019.

[23] F. Deutsch and H. Hundal, "The rate of convergence for the cyclic projections algorithm i: Angles between convex sets," *Journal of Approximation Theory*, vol. 142, no. 1, pp. 36–55, 2006.

[24] S. Vermpala. 2D/3D RRT* algorithm. Retrieved August 24, 2022. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/60993-2d-3d-rrt-algorithm