Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Σηματων, Ελεγχου και Ρομποτικης

From All To A Subset Harmonic-Based Optimal Motion Planning in Constrained Workspaces using Reinforcement Learning

Διπλωματικη Εργασια

του

ΧΡΙΣΤΟΥ Σ. ΒΛΑΧΟΥ

Επιβλέπων: Κωνσταντίνος Τζαφέστας
Αν. Καθηγητής ΕΜΠ

Αθήνα, Σεπτέμβριος 2022

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Σημάτων, Ελέγχου και Ρομποτικής

From All To A Subset Harmonic-Based Optimal Motion Planning in Constrained Workspaces using Reinforcement Learning

Διπλωματικη Εργασια

του

χριστου Σ. βλαχου

Επιβλέπων: Κωνσταντίνος Τζαφέστας
Αν. Καθηγητής ΕΜΠ

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 5η Σεπτεμβρίου 2022.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Κωνσταντίνος Τζαφέστας	Κωνσταντίνος Κυριακόπουλος	Χαράλαμπος Ψυλλάκης
Αν. Καθηγητής ΕΜΠ	Καθηγητής ΕΜΠ	Λέκτορας ΕΜΠ

Αθήνα, Σεπτέμβριος 2022

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Σημάτων, Ελέγχου και Ρομποτικής

ΔΗΛΩΣΗ ΜΗ ΛΟΓΟΚΛΟΠΗΣ ΚΑΙ ΑΝΑΛΗΨΗΣ ΠΡΟΣΩΠΙΚΗΣ ΕΥΘΥΝΗΣ

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ενυπογράφως ότι είμαι αποκλειστικός συγγραφέας της παρούσας Πτυχιακής Εργασίας, για την ολοκλήρωση της οποίας κάθε βοήθεια είναι πλήρως αναγνωρισμένη και αναφέρεται λεπτομερώς στην εργασία αυτή. Έχω αναφέρει πλήρως και με σαφείς αναφορές, όλες τις πηγές χρήσης δεδομένων, απόψεων, θέσεων και προτάσεων, ιδεών και λεκτικών αναφορών, είτε κατά κυριολεξία είτε βάσει επιστημονικής παράφρασης. Αναλαμβάνω την προσωπική και ατομική ευθύνη ότι σε περίπτωση αποτυχίας στην υλοποίηση των ανωτέρω δηλωθέντων στοιχείων, είμαι υπόλογος έναντι λογοκλοπής, γεγονός που σημαίνει αποτυχία στην Πτυχιακή μου Εργασία και κατά συνέπεια αποτυχία απόκτησης του Τίτλου Σπουδών, πέραν των λοιπών συνεπειών του νόμου περί πνευματικών δικαιωμάτων. Δηλώνω, συνεπώς, ότι αυτή η Πτυχιακή Εργασία προετοιμάστηκε και ολοκληρώθηκε από εμένα προσωπικά και αποκλειστικά και ότι, αναλαμβάνω πλήρως όλες τις συνέπειες του νόμου στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής άλλης πνευματικής ιδιοκτησίας.

(Υπογραφή)

.....
Χρίστος Βλάχος
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

5 Σεπτεμβρίου 2022

Περίληψη

Στην παρούσα διπλωματική εργασία, παρουσιάζεται μια καινοτόμος λύση για το πρόβλημα του βέλτιστου σχεδιασμού πορείας σε στατικά, πλήρως γνωστά περιβάλλοντα. Η προσέγγισή μας, επιτρέπει σε ένα ρομπότ να πλοηγηθεί με ασφάλεια προς οποιοδήποτε προορισμό εντός ενός υποσυνόλου του χώρου εργασίας του, με την χρήση ενός παραμετρικού ελεγκτή που βασίζεται στην θεωρία Τεχνητών Δυναμικών Πεδίων (ΤΔΠ). Η βελτιστότητα επιτυγχάνεται με την εφαρμογή τεχνικών Ενισχυτικής Μάθησης (ΕΜ). Πιο συγκεκριμένα, οι παράμετροι του δυναμικού πεδίου ρυθμίζονται κατάλληλα μέσω ενός αλγορίθμου κλίσης πολιτικής με σκοπό την ελαχιστοποίηση μιας συνάρτησης κόστους. Παρά τους περιορισμούς της, η προτεινόμενη μέθοδος μπορεί να αποτελέσει μια σημαντική προσθήκη στις ήδη υπάρχουσες τεχνικές βέλτιστου σχεδιασμού πορείας.

Λέξεις Κλειδιά

Βέλτιστος Σχεδιασμός Πορείας, Παραμετρικός Ελεγκτής, Τεχνητά Δυναμικά Πεδία, Ενισχυτική Μάθηση, Αλγόριθμος Κλίσης Πολιτικής

Abstract

In this thesis, a novel solution is presented for optimal motion planning in static, fully-known environments. Our approach allows a robot to safely navigate towards any destination within a subset of its workspace by using a parametric controller based on Artificial Potential Field (APF) theory. Optimization is achieved through the application of Reinforcement Learning (RL) techniques. More specifically, the parameters of the underlying potential field are adjusted appropriately with a policy gradient algorithm in order to minimize a cost function. The proposed method is not without its limitations, but can still be a valuable addition to the arsenal of established motion planning approaches.

Keywords

Optimal Motion Planning, Parametric Controller, Artificial Potential Field, Reinforcement Learning, Policy Gradient Algorithm

Ευχαριστίες

Θα ήθελα καταρχάς να εκφράσω τις θερμές μου ευχαριστίες στον καθηγητή κ. Κωνσταντίνο Κυριακόπουλο για την επίβλεψη αυτής της διπλωματικής εργασίας καθώς και για όλο το ενδιαφέρον που έδειξε καθ΄όλη τη διάρκεια της. Επίσης, θα ήθελα να πω ένα τεράστιο ευχαριστώ στον καθηγητή κ. Χαράλαμπο Μπεχλιούλη και στον υποψήφιο διδάκτορα Παναγιώτη Ρουσσέα για όλη την καθοδήγηση και τη στήριξη που μου προσέφεραν, η εκπόνηση της διπλωματικής εργασίας δεν θα ήταν δυνατή χωρίς τη βοήθειά τους. Τέλος, θα ήθελα να ευχαριστήσω την οικογένειά μου και όλους τους ανθρώπους που ήταν δίπλα μου, για τη συμπαράστασή τους όλα αυτά τα χρόνια.

Αθήνα, Σεπτέμβριος 2022

Χρίστος Βλάχος

Περιεχόμενα

Chapter 1

Εκτεταμένη Περίληψη

1.1 Εισαγωγή

Το πρόβλημα βέλτιστου σχεδιασμού πορείας (Optimal Motion Planning) ενός ρομπότ περιλαμβάνει την εύρεση μιας βέλτιστης και ασφαλούς διαδρομής μεταξύ δύο σημείων στο χώρο. Μεγάλη έμφαση έχει δοθεί στην εύρεση λύσεων του μακροχρόνιου αυτού προβλήματος, το οποίο εμφανίζεται σε ποικίλες μορφές λόγω ιδιαιτεροτήτων όπως είναι οι μη-ολόνομοι περιορισμοί και η ύπαρξη εμποδίων στο χώρο. Παρά την πληθώρα διαθέσιμων εργαλείων ικανών να αντιμετωπίσουν τέτοια προβλήματα, υπάρχει ακόμη περιθώριο για την ανάπτυξη και τη βελτίωση των ήδη υπάρχοντων μεθόδων.

Στην παρούσα εργασία, σκοπός μας είναι ο σχεδιασμός ενός καινοτόμου παραμετρικού ελεγκτή που θα χειρίζεται αλλαγές στον τελικό στόχο του ρομπότ πιο αποτελεσματικά, αφαιρώντας την ανάγκη για υπολογιστική ισχύ σε πλήρως γνωστούς, στατικούς χώρους. Με τον υπολογισμό ενός σετ παραμέτρων, το ρομπότ θα έχει τη δυνατότητα να πλοηγείται με ασφάλεια προς οποιονδήποτε τελικό στόχο που βρίσκεται εντός ενός προκαθορισμένου υποσυνόλου του περιβάλλοντός του. Η βελτιστότητα επιτυγχάνεται ρυθμίζοντας κατάλληλα τις τιμές των παραμέτρων του ελεγκτή με την χρήση Ενισχυτικής Μάθησης (Reinforcement Learning), έτσι ώστε να ελαχιστοποιείται μια συνάρτηση κόστους δίχως να διακυβεύονται η ασφάλεια κατά την πλοήγηση και η τελική σύγκλιση στο στόχο.

Μία από τις ευρέως γνωστές μεθόδους προσέγγισης προβλημάτων σχεδιασμού πορείας, πάνω στην οποία θα βασιστούμε σε αυτή την εργασία, είναι η μέθοδος Τεχνητών Δυναμικών Πεδίων (Artificial Potential Field) [1]. Στη μέθοδο αυτή, το ρομπότ μοντελοποιείται ως ένα σωματίδιο που κινείται υπό την επιρροή ενός δυναμικού πεδίου που καθορίζεται από τα εμπόδια και τον τελικό στόχο του ρομπότ. Τα εμπόδια αναπαριστώνται από ένα απωθητικό δυναμικό πεδίο, ενώ ο τελικός στόχος από ένα ελκτικό, με σκοπό το ρομπότ να φτάσει στο στόχο αποφεύγοντας τη σύγκρουση κατά τη διάρκεια της πλοήγησης. Το μεγάλο μειονέκτημα αυτής της μεθόδου είναι η ύπαρξη τοπικών ελαχίστων που έχουν ως αποτέλεσμα να παγιδεύουν το ρομπότ. Ωστόσο, λύση στο πρόβλημα αυτό δόθηκε με την χρήση δυναμικών πεδίων βασισμένων σε αρμονικές συναρτήσεις, τα Τεχνητά Αρμονικά Δυναμικά Πεδία (Artificial Harmonic Potential Fields) [2], λόγω της ιδιότητας των αρμονικών συναρτήσεων να μην παρουσιάζουν τοπικά ελάχιστα.

Στην παρούσα εργασία, επεκτείνουμε την μέθοδο των Τεχνητών Αρμονικών Δυναμικών Πεδίων συνδυάζοντάς τη με την προσέγγιση συναρτήσεων ακτινικής βάσης (Radial Basis

Function Approximation) [3]. Εκφράζοντας τις παραμέτρους του Τεχνητού Αρμονικού Δυναμικού Πεδίου ως ένα σταθμισμένο άθροισμα συναρτήσεων ακτινικής βάσης και υπολογίζοντας τα βάρη αυτά, μπορούμε να κατασκευάσουμε έναν παραμετρικό ελεγκτή που θα οδηγεί το ρομπότ με ασφάλεια σε οποιοδήποτε επιθυμητό στόχο εντός ενός υποσυνόλου του χώρου εργασίας.

Με την αναδιατύπωση του προβλήματος βέλτιστου σχεδιασμού πορείας ως ένα πρόβλημα Ενισχυτικής Μάθησης, μπορούμε να υλοποιήσουμε έναν αλγόριθμο καθοδικής κλίσης με προβολή (Projected Gradient Descent) για την επίλυσή του. Η προβολή των παραμέτρων κατά την διάρκεια της εκμάθησης γίνεται ώστε να κατοχυρώσουμε την ασφάλεια του πεδίου πλοήγησης. Τέλος, η χρήση Ενισχυτικής Μάθησης παίζει σημαντικό ρόλο καθώς μας απαλλάσσει από την επίλυση μια δύσκολης μη-γραμμικής μερικής διαφορικής εξίσωσης για τον υπολογισμό της συνάρτησης κόστους.

1.2 Διατύπωση Προβλήματος Βέλτιστου Σχεδιασμού Πορείας

Έστω ένα σημειακό ρομπότ, που πλοηγείται εντός ενός δισδιάστατου, φραγμένου, συνεκτικού συνόλου $\mathcal{G} \subset \mathbb{R}^2$, με εσωτερικά διακεκριμένα εμπόδια $\mathcal{O}_i \subset \mathcal{G}, i = 1, ..., M$. Ορίζουμε ως τον χώρο εργασίας του ρομπότ $\mathcal{W}$, το σύνολο $\mathcal{W} = \mathcal{G} - \bigcup_{i=1}^{M} \mathcal{O}_i$. Επιπλέον θεωρούμε ένα τελικό σημείο στόχο p_d. Το ρομπότ κινείται σύμφωνα με την δυναμική εξίσωση:

$$\dot{p} = u,\ p(t=0) = \bar{p} \in \mathcal{W} \tag{1.1}$$

όπου $p \in \mathcal{W}$ είναι το διάνυσμα κατάστασης (δηλ. θέση του ρομπότ) , $u = u(t) : \mathbb{R} \to \mathbb{R}^2$ είναι μια είσοδος ελέγχου (δηλ. ταχύτητα εισόδου) και με $\bar{p}$ συμβολίζουμε την αρχική θέση του ρομπότ. Η λύση του προβλήματος βέλτιστου σχεδιασμού πορείας περιλαμβάνει την εύρεση μιας πολιτικής ελέγχου u που ελαχιστοποιεί την παρακάτω συνάρτηση κόστους:

$$V(\bar{p}, p_d) = \int_0^\infty [Q(p(\tau; \bar{p}); p_d) + R(u(\tau))]\, d\tau\ \forall \bar{p} \in \mathcal{W} \tag{1.2}$$

Η συνάρτηση κόστους (1.2) περιλαμβάνει έναν όρο που σχετίζεται με το διάνυσμα κατάστασης $Q(p(\tau; \bar{p}); p_d)$ και έναν που έχει να κάνει με την είσοδο $R(u(\tau))$, όπου $p(t; \bar{p}) = \int_0^t u(p(\tau; \bar{p}))\, d\tau + \bar{p}$ είναι η λύση της (1.1). Ο όρος R ελαχιστοποιεί την ενέργεια εισόδου ενώ ο όρος Q τιμωρεί το ρομπότ για όσο περισσότερο χρόνο μένει μακρυά από τον τελικό στόχο και επιλέγονται ως εξής:

$$Q(p(\tau; \bar{p}); p_d) = \alpha \|p(\tau; \bar{p}) - p_d\|^2 \tag{1.3}$$

$$R(u(\tau)) = \beta \|u(\tau)\|^2 \tag{1.4}$$

όπου α, β είναι θετικές παράμετροι. Επομένως η συνάρτηση κόστους μπορεί να γραφτεί:

$$V(\bar{p}, p_d) = \int_0^\infty [\alpha \|p(\tau; \bar{p}) - p_d\|^2 + \beta \|u(\tau)\|^2]\, d\tau\ \forall \bar{p} \in \mathcal{W} \tag{1.5}$$

Στόχος μας στην παρούσα διπλωματική εργασία, είναι η κατασκευή ενός ελεγκτή που θα παράγει ασφαλή πεδία πλοήγησης προς οποιοδήποτε τελικό στόχο εντός ενός υποσυνόλου $\mathcal{S} \subset \mathcal{W}$ του χώρου εργασίας και θα ελαχιστοποιεί ταυτόχρονα την συνάρτηση (1.5).

1.3 Τεχνητά Δυναμικά Αρμονικά Πεδία

Το δυναμικό πεδίο που έχουμε υλοποιήσει για την ασφαλή πλοήγηση του ρομπότ εντός του χώρου εργασίας, βασίζεται στην μέθοδο των πάνελς (panel method) με αρμονικές συναρτήσεις [2] και ορίζεται ως:

$$\Phi(p; p_C) = \sum_{i=0}^{K} \phi(p; p_i) w_i = w^\intercal \phi(p; p_C) \tag{1.6}$$

όπου $p_C \triangleq \{p_0, p_1, p_2, \cdots, p_K\},\ p_0 = p_d \in \mathcal{W} - \partial\mathcal{W}, p_i \in \mathcal{W}', i = 1, \ldots, K$ είναι μια (K+1)-πλειάδα που περιέχει τα κέντρα των αρμονικών συναρτήσεων βάσεων ϕ_i, τα οποία τοποθετούνται εκτός του χώρου εργασίας $\mathcal{W}$ (με εξαίρεση το $p_0 = p_d$) και $w \triangleq [w_0, w_1, \ldots, w_K]^\intercal$ είναι ένα διάνυσμα που περιέχει το αντίστοιχο βάρος της συνάρτησης βάσης ϕ_i.

Το σύνολο των συναρτήσεων βάσεων $\phi(p; p_C)$ ορίζεται ως:

$$\phi(p; p_C) \triangleq [\phi(p; p_d), \phi(p; p_1), \cdots, \phi(p; p_K)]^\intercal : \mathcal{W} \to \mathbb{R}^{K \times 1} \tag{1.7}$$

όπου ο ακόλουθος αρμονικός όρος τοποθετείται στον τελικό στόχο p_d του ρομπότ:

$$\phi(p; p_d) = ln(\|p - p_d\|) \tag{1.8}$$

Οι υπόλοιπες συναρτήσεις βάσεις επιλέγονται σύμφωνα με το [4]. Τα αρμονικά πάνελ τοποθετούνται εκτός του χώρου εργασίας ώστε να αποφύγουμε τις ιδιομορφίες (singularities) εντός του χώρου.

1.4 From-All-To-A-Point (FATAP) Ελεγκτής

Με βάση το Τεχνητό Αρμονικό Δυναμικό Πεδίο που έχουμε ορίσει, προτείνουμε τον ακόλουθο παραμετρικό FATAP (From-All-To-A-Point) ελεγχτή, για την ασφαλή πλοήγηση του ρομπότ προς ένα μοναδικό τελικό σημείο-στόχο $p_d \in \mathcal{W} - \partial\mathcal{W}$:

$$u(p) = -\|p - p_d\|^2 \nabla\Phi(p; p_C) = -\|p - p_d\|^2 \nabla\phi^\intercal(p; p_C) w \tag{1.9}$$

όπου:

$$\nabla\phi(p; p_d) = \frac{p - p_d}{\|p - p_d\|^2} \tag{1.10}$$

Δεδομένης της ιδιομορφίας που παρουσιάζει η κλίση του πεδίου Φ στο τελικό σημείο-στόχο, χρησιμοποιούμε τον όρο $\|p - p_d\|^2$ ώστε να εξασφαλίσουμε την ευστάθεια του νόμου ελέγχου στον τελικό στόχο p_d, για $w_0 > 0$

Η ασφάλεια του πεδίου προκύπτει επιβάλλοντας την παρακάτω σχέση:

$$n^\intercal(z) u(z) \geq 0, \forall z \in \partial\mathcal{W} \tag{1.11}$$

όπου $n(z)$ είναι το κανονικό διάνυσμα που δείχνει προς το εσωτερικό του χώρου εργασίας σε κάθε σημείο του συνόρου και $u(z)$ το διάνυσμα της ταχύτητας σε κάθε ένα από αυτά τα σημεία. Παρόλο που η συνθήκη ασφαλείας (1.11) περιλαμβάνει όλα τα σημεία του συνόρου του χώρου εργασίας $\partial\mathcal{W}$, αποδεικνύεται πως η ασφάλεια του πεδίου είναι εγγυημένη ακόμη και αν η συνθήκη ισχύει για έναν πεπερασμένο αριθμό σημείων του συνόρου [5]. Η συνθήκη ασφαλείας (1.11) μπορεί να εκφραστεί ως ένα σύστημα γραμμικών ανισώσεων ως προς τα βάρη των συναρτήσεων βάσεων. Σε μορφή πίνακα γράφεται ως εξής:

$$A^\intercal w \leq 0 \tag{1.12}$$

όπου $A = [A_1, \ldots, A_N]$ και $A_i = n^\intercal(z_i)\nabla\phi^\intercal(z_i; p_C)$.

Επομένως, οι παράμετροι του αρμονικού πεδίου που εξασφαλίζουν τη σύγκλιση στο τελικό σημείο-στόχο με ασφάλεια, μπορούν να προκύψουν από την επίλυση του παρακάτω τετραγωνικού προβλήματος:

$$\min_w \|w\|^2,\ s.to\ A^\intercal w \leq -\epsilon, w_0 > 0 \tag{1.13}$$

όπου $\epsilon > 0$ ένας μικρός αριθμός ώστε να αποφύγουμε την μηδενική λύση στο τετραγωνικό πρόβλημα. Τροποποιώντας κατάλληλα το τετραγωνικό πρόβλημα, μπορούμε να θέσουμε περιορισμούς ώστε το βάρος που αντιστοιχεί στον αρμονικό όρο που τοποθετείται στον τελικό στόχο, να προκύπτει ίσο με την μονάδα ($w_0 = 1$). Με τον τρόπο αυτό εξασφαλίζουμε ότι οι αρχικές μας πολιτικές θα ανήκουν στην ίδια οικογένεια λύσεων.

1.5 From-All-To-A-Subset (FATAS) Ελεγκτής

Είμαστε έτοιμοι να παρουσιάσουμε τον καινοτόμο FATAS (From-All-To-A-Subset) ελεγκτή. Αρχικά, θεωρούμε την εξής παραμετροποίηση των βαρών του FATAP ελεγκτή:

$$w = w(p_d; p_{Cd}) = W^\intercal s(p_d; p_{Cd}) \tag{1.14}$$

Ο πίνακας $W \in \mathbb{R}^{K_c \times (K+1)}$ είναι ένα νέο σετ παραμέτρων όπου $p_{Cd} = \{p_{d,1}, p_{d,2}, \cdots, p_{d,K_C}\}$, $p_{d,i} \in \mathbb{R}^2, i = 1, \ldots, K_C$ είναι τα κέντρα των συναρτήσεων βάσεων, με το σετ των συναρτήσεων βάσεων να ορίζεται ως η αντιστοιχία $s(p_d; p_{Cd}) : (\mathcal{W} - \partial\mathcal{W}) \to \mathbb{R}^{K_c}$, για μια δεδομένη επιλογή κέντρων.

Παρόλο που η επιλογή των συναρτήσεων αυτών μπορεί να γίνει αυθαίρετα, προτείνουμε την χρήση συναρτήσεων ακτινικής βάσης (Radial Basis Functions) με ομοιόμορφα κατανεμημένα κέντρα. Η συνάρτηση ακτινικής βάσης είναι μια συνάρτηση, η τιμή της οποίας εξαρτάται μόνο από την απόσταση του ορίσματός της από ένα σταθερό σημείο που ονομάζεται κέντρο. Οι συναρτήσεις ακτινικής βάσης χρησιμοποιούνται για την προσέγγιση των χαρακτηριστικών μιας συνάρτησης κοντά στο κέντρο τους. Στην παρούσα εργασία επιλέξαμε Γκαουσιανές συναρτήσεις, που είναι από τις πιο ευρέως χρησιμοποιούμενες συναρτήσεις ακτινικής βάσης:

$$\phi(r) = e^{-(\epsilon r)^2}$$

όπου $r = \|p_d - p_{d,i}\|$ είναι η ευκλείδεια απόσταση μεταξύ του τελικού στόχου και του ι-οστού

κέντρου, και ϵ μια παράμετρος που ρυθμίζει την εξασθένιση (decay) της συνάρτησης.

Το σημείο κλειδί είναι ότι για έναν δεδομένο πίνακα W και ένα σταθερό τελικό σημείο-στόχο σε ένα υποσύνολο του χώρου εργασίας $p_d \in \mathcal{S} \subset \mathcal{W}$, η εξίσωση (1.14) δίνει σταθερά βάρη $w \in \mathbb{R}^{K+1}$ και επομένως το πεδίο:

$$\Phi(p; p_d; p_C) = \phi^\intercal(p; p_C) W^\intercal s(p_d; p_{Cd}),\ p_d \in \mathcal{S} \tag{1.15}$$

είναι αρμονικό και η κλίση του γίνεται:

$$\nabla\Phi(p; p_d; p_C) = \nabla\phi^\intercal(p; p_C) W^\intercal s(p_d; p_{Cd}),\ p_d \in \mathcal{S} \tag{1.16}$$

Για να κάνουμε το πεδίο (1.15) γραμμικό ως προς τις καινούριες παραμέτρους, χρησιμοποιούμε την παρακάτω ιδιότητα που ισχύει για έναν πίνακα A, διαστάσεων $m \times n$ και έναν πίνακα B, $r \times q$:

$$vec(AXB) = (B^\intercal \otimes A) vec(X) \tag{1.17}$$

Επομένως η σχέση (1.16) γίνεται:

$$vec(\nabla\Phi(p; p_d; p_C)) = \nabla\Phi(p; p_d; p_C) = (s^\intercal(p_d; p_{C_d}) \otimes \nabla\phi^\intercal(p; p_C)) vec(W^\intercal), p_d \in \mathcal{S} \tag{1.18}$$

όπου

$$s^\intercal(p_d; p_{C_d}) \otimes \nabla\phi^\intercal(p; p_C) \in \mathbb{R}^{2 \times K_c(K+1)}$$
$$\hat{w} = vec(W^\intercal) \in \mathbb{R}^{K_c(K+1)}$$

Η συνθήκη ασφαλείας γράφεται ως εξής:

$$n^\intercal(z)(s^\intercal(p_d; p_{Cd}) \otimes \nabla\phi^\intercal(z; p_C))\hat{w} \leq 0, \forall z \in \partial\mathcal{W}, \forall p_d \in \mathcal{S} \tag{1.19}$$

η οποία είναι γραμμική ως προς τα βάρη $\hat{w}$. Παρόλο που η συνθήκη ασφαλείας (1.19) περιέχει κάθε τελικό σημείο στόχο μέσα στο υποσύνολο $\mathcal{S}$, η ασφάλεια είναι εγγυημένη αν αυτή ισχύει για μια πλειάδα τελικών σημείων $p_G \triangleq \{p_{d1}, p_{d2}, \ldots, p_{dm}\}$. Η συνθήκη ασφαλείας μπορεί να εκφραστεί όπως και πριν, ως ένα σύστημα γραμμικών ανισώσεων ως προς τις παραμέτρους $\hat{w}$:

$$\tilde{A}^\intercal \hat{w} \leq 0 \tag{1.20}$$

όπου $\tilde{A} = [\tilde{A}_1, \ldots, \tilde{A}_N]$, με $\tilde{A}_i = [s^\intercal(p_{di}, p_{Cd})] \otimes (n^\intercal(z)\nabla\phi^\intercal(z; p_C))]$

Το σετ παραμέτρων W προκύπτει επιλύοντας το ακόλουθο τετραγωνικό πρόβλημα:

$$\min_{\hat{w}} \|\hat{w}\|^2,\ s.to\ \tilde{A}^\intercal \hat{w} \leq -\epsilon, \quad w_0(p_d) = W_1^\intercal s(p_d; p_{Cd}) > 0 \tag{1.21}$$

όπου $w_0(p_d)$ είναι το σταθμισμένο άθροισμα που αντιστοιχεί στο βάρος του αρμονικού όρου που τοποθετούμε στον τελικό στόχο του ρομπότ. Για να εξασφαλίσουμε ότι οι αρχικές πολιτικές ανήκουν στην ίδια οικογένεια λύσεων, επεκτείνουμε τις συναρτήσεις βάσεις με τον ακόλουθο

τρόπο:

$$\tilde{s}(p_d; p_{Cd}) = [1 \; s(p_d; p_{Cd})] \tag{1.22}$$

Επομένως το καινούριο σετ παραμέτρων γίνεται $\tilde{W} \in \mathbb{R}^{(K_c+1)\times(K+1)}$ και:

$$w = \tilde{w}(p_d; p_{Cd}) = \tilde{W}^\intercal \tilde{s}(p_d; p_{Cd}) \tag{1.23}$$

$$\hat{\tilde{w}} = vec(\tilde{W}^\intercal) \tag{1.24}$$

Το τετραγωνικό πρόβλημα (1.21) γίνεται:

$$\min_{\hat{\tilde{w}}} \|\hat{\tilde{w}}\|^2, \; s.to \; \hat{\tilde{A}}\hat{\tilde{w}} \leq -\epsilon, \;\; A_{eq}\hat{\tilde{w}} = B_{eq} \tag{1.25}$$

όπου με κατάλληλους περιορισμούς ισότητας μπορούμε να επιβάλλουμε τον περιορισμό:

$$\tilde{W}_1^\intercal = \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix} \tag{1.26}$$

και επομένως:

$$\tilde{W}_1^\intercal \tilde{s}(p_d; p_{cd}) = 1 \tag{1.27}$$

Η λύση στο τετραγωνικό πρόβλημα (1.25) αποτελεί την αρχική μας πολιτική, που εγγυάται την ασφάλεια και την σύγκλιση προς οποιοδήποτε τελικό στόχο εντός του υποσυνόλου $\mathcal{S}$. Αυτή η πολιτική θα βελτιωθεί με την χρήση Ενισχυτικής Μάθησης ώστε να ελαχιστοποιεί την συνάρτηση κόστους (1.5) για κάθε τελικό στόχο $p_d \in S$.

1.6 Βελτιστοποίηση του FATAS ελεγκτή με χρήση Ενισχυτικής Μάθησης

Όπως ορίσαμε στην ενότητα 1.2, το πρόβλημα βέλτιστου σχεδιασμού πορείας ανάγεται στην εύρεση μιας πολιτικής ελέγχου u που ελαχιστοποιεί τη συνάρτηση κόστους:

$$V(\bar{p}, p_d) = \int_0^\infty [\alpha \|p(\tau; \bar{p}) - p_d\|^2 + \beta \|u(\tau)\|^2] \, d\tau \; \forall \bar{p} \in \mathcal{W} \tag{1.28}$$

Ορίζουμε την ακόλουθη Χαμιλτονιανή συνάρτηση με βάση τη συνάρτηση κόστους (1.28):

$$H(p, u, \nabla V) = \nabla V^\intercal u + \alpha \|p - p_d\|^2 + \beta \|u\|^2 \tag{1.29}$$

Η συνθήκη βελτιστότητας Hamilton-Jacobi-Bellman (HJB) δίνεται από:

$$H(p, u^*, \nabla V^*) = 0 \tag{1.30}$$

και η βέλτιστη πολιτική ελέγχου προκύπτει από τη συνθήκη στασιμότητας $\frac{\partial H(p,u,\nabla V^*)}{\partial u}|_{u=u^*} = 0$ ως εξής:

$$u^* = -\frac{1}{2\beta} \nabla V^* \tag{1.31}$$

Για να βρούμε τη βέλτιστη πολιτική ελέγχου u^* χρειαζόμαστε μια αναλυτική έκφραση για

τη συνάρτηση κόστους $V^*(p; p_d)$ η οποία μπορεί να προκύψει αντικαθιστώντας τη βέλτιστη πολιτική ελέγχου στη συνθήκη βελτιστότητας HJB (1.30) και επιλύοντας μια πολύ δύσκολη μη-γραμμική μερική διαφορική εξίσωση. Για να το αποφύγουμε αυτό, θα αναδιατυπώσουμε το πρόβλημα μας σαν ένα πρόβλημα Ενισχυτικής Μάθησης και θα υλοποιήσουμε έναν αλγόριθμο κλίσης πολιτικής (policy gradient) για να βρούμε την βέλτιστη πολιτική ελέγχου.

Η ελαχιστοποίηση της συνάρτησης κόστους (1.28) συμπίπτει με την εύρεση της βέλτιστης πολιτικής ελέγχου u^* και του κόστους V^* που ικανοποιούν την συνθήκη στασιμότητας (1.31). Αυτό ανάγεται στην εύρεση των παραμέτρων $\tilde{W}^*$ του ελεγκτή, που οδηγούν σε βέλτιστο πεδίο πλοήγησης. Ο αλγόριθμος κλίσης πολιτικής που έχουμε χρησιμοποιήσει για την βελτιστοποίηση των παραμέτρων, είναι βασισμένος στον αλγόριθμο βελτίωσης πολιτικών (Policy Iteration), η οποία είναι μια ευρέως γνωστή τεχνική που λύνει την εξίσωση HJB [6].

Algorithm 1.1: *Off-line Policy Iteration*

Require: Any admissible control policy $u^{(0)}$ and tolerance $\epsilon > 0$
  while $\|u^{(i+1)} - u^{(i)}\| > \epsilon$ **do**
    1) Calculate $\nabla V^{(i)}$ based on policy $u^{(i)}$
    2) Update the control policy: $u^{(i+1)} = -\frac{1}{2\beta}\nabla V^{(i)}$
    3) $i \leftarrow i + 1$
  end while

Έχοντας ως στόχο να βρούμε τις βέλτιστες παραμέτρους $\tilde{W}^*$ του FATAS ελεγχτή, γράφουμε:

$$u_{\hat{\tilde{w}}}^{(i+1)} = \sigma^\intercal \tilde{W}^{\intercal(i+1)} \tilde{s}(p_d; p_{Cd}) \tag{1.32}$$

όπου:

$$\sigma = -\|p - p_d\|^2 \nabla\phi(p; p_C)$$

Επομένως, οι παράμετροι $\tilde{W}$ σε κάθε επανάληψη (iteration) του αλγορίθμου, δίνονται από την επίλυση του τετραγωνικού προβλήματος (1.33). Το σύνολο p_G είναι το σύνολο των τελικών στόχων του ρομπότ που χρησιμοποιήσαμε στην συνθήκη ασφαλείας για να αποκτήσουμε την αρχική μας πολιτική.

$$\min_{\hat{\tilde{w}}=vec(\tilde{W}^\intercal)} : \|\sigma^\intercal \tilde{W}^{\intercal(i+1)} \tilde{s}(p_d; p_{Cd}) + \frac{1}{2\beta}\nabla V^{(i)}(p_d)\|^2 \;\; \forall p_d \in p_G \tag{1.33}$$

$$subject\ to\ \tilde{A}^\intercal \hat{\tilde{w}} \leq 0$$

Για να εφαρμόσουμε τον αλγόριθμο καθοδικής κλίσης με προβολή (Projected Gradient Descent) για τις παραμέτρους του ελεγκτή, πρώτα ορίζουμε την αντικειμενική συνάρτηση (objective function) (1.34).

$$M(\tilde{W}^\intercal) = \|\sigma^\intercal(p)\tilde{W}^\intercal \tilde{s}(p_d; p_{cd}) + \frac{1}{2\beta}\nabla V(p, p_d)\|^2 \tag{1.34}$$

Οι παράμετροι του ελεγκτή ενημερώνονται με τον ακόλουθο τρόπο:

$$\hat{\tilde{w}}^{(i+1)} = \hat{\tilde{w}}^{(i)} - a\nabla_{\tilde{W}^\intercal} M(\tilde{W}^\intercal) \tag{1.35}$$

όπου:

$$\nabla_{\tilde{W}^\intercal} M(\tilde{W}^\intercal) = 2(\tilde{s}(p_d; p_{Cd}) \otimes \sigma(p))(\sigma^\intercal(p)\tilde{W}^\intercal \tilde{s}(p_d; p_{Cd}) + \frac{1}{2\beta}\nabla V(p, p_d)) \tag{1.36}$$

είναι η κατεύθυνση καθόδου (descent direction) ($\nabla_{\tilde{W}^\intercal} M(\tilde{W}^\intercal) \in \mathbb{R}^{(K+1)(Kc+1)}$) και α είναι ο ρυθμός εκμάθησης.

Για τον υπολογισμό της κατεύθυνσης καθόδου χρειάζεται να υπολογίσουμε την κλίση της συνάρτησης κόστους V για όλους τους τελικούς στόχους $p_d \in p_G$. Για την μείωση της υπολογιστικής πολυπλοκότητας, σε κάθε επανάληψη του αλγορίθμου, όταν υπολογίζουμε την κλίση της συνάρτησης κόστους χρησιμοποιούμε μόνο N σημεία, τυχαία επιλεγμένα από το σύνολο p_G, όπου $N < m$ και m είναι ο αριθμός των σημείων του συνόλου p_G. Στη συνέχεια, για κάθε p_{di}, όπου $1 \leq i \leq N$ θα υπολογίσουμε μια εκτίμηση της κλίσης $\nabla V(p, p_{di})$. Για να γίνει αυτό, θα χρειαστεί πρώτα να υπολογίσουμε μια εκτίμηση της συνάρτησης κόστους για κάθε τελικό στόχο p_{di}, επιλύοντας ένα σύστημα διαφορικών εξισώσεων. Για περαιτέρω αντιμετώπιση της υπολογιστικής επιβάρυνσης, η εκτίμηση της συνάρτησης κόστους, και κατά συνέπεια η εκτίμηση της κλίσης, θα γίνουν για ένα υποσύνολο του χώρου εργασίας με αποτέλεσμα ο αλγόριθμός μας να αποτελεί έναν Mini-Batch αλγόριθμο καθοδικής κλίσης. Τέλος, για να αποκτήσουμε την κατεύθυνση καθόδου θα υπολογίσουμε την μέση τιμή της σχέσης (1.36) για όλες τις παρτίδες (batches) και για όλους τους τελικούς στόχους p_{di}:

$$\nabla_{\tilde{W}^\intercal} M(\tilde{W}^\intercal) = \frac{2}{N}\sum_{i=1}^{N}\sum_{j=1}^{K_i}\frac{1}{K_i}(\tilde{s}(p_{di}; p_{Cd}) \otimes \sigma(p_j))(\sigma^\intercal(p_j)\tilde{W}^\intercal \tilde{s}(p_{di}; p_{Cd}) + \frac{1}{2\beta}\nabla V(p_j, p_{di})) \tag{1.37}$$

Ο ρυθμός εκμάθησης (learning rate) α υπολογίζεται με την χρήση ενός προσαρμοστικού αλγορίθμου, που πηγάζει από τον αλγόριθμο ADADELTA [7]. Η διαφορά μεταξύ των δύο αλγορίθμων είναι ότι ενώ ο τελευταίος υπολογίζει ένα διαφορετικό ρυθμό εκμάθησης για κάθε παράμετρο, ο αλγόριθμός μας υπολογίζει έναν καθολικό (global) ρυθμό εκμάθησης χρησιμοποιώντας πληροφορίες από τις κλίσεις όλων των παραμέτρων. Η τροποποίηση αυτή έγινε γιατί στις προσομοιώσεις, ένας καθολικός ρυθμός εκμάθησης αποδείχτηκε πιο αποτελεσματικός. Τα βήματα του αλγορίθμου παρουσιάζονται στον αλγόριθμο (1.2).

ALGORITHM 1.2: *Computing learning rate update at iteration i*

Require: Decay rate ρ , Constant ϵ
Require: Initial parameters $w^{(0)} = vec(W^{\intercal(0)})$
Require: Initialization of accumulation variables $E[step_w^2]_0 = 0,\ E[\Delta w^2]_0 = 0$
for $i = 1 : N$ **do**
1) Compute gradient $step_w^{(i)} = \nabla_{W^\intercal} M(W^\intercal)$
2) Accumulate Gradient $E[step_w^2]_i = \rho E[step_w^2]_{(i-1)} + (1-\rho) step_w^{2(i)}$
3) Compute Update: $\Delta w_i = -\frac{RMS[\Delta w]_{(i-1)}}{RMS[step_w]_i} step_w^{(i)}$
where $RMS[x]_i = \sqrt{E[x^2]_i + \epsilon}$
4) Accumulate Updates: $E[\Delta w^2]_i = \rho E[\Delta w^2]_{i-1} + (1-\rho)\Delta w_i^2$
5) Apply Update : $w^{i+1} = w^i + \Delta w_i$
end for

Τέλος, για να εξασφαλίσουμε ότι ενημερώνοντας τις παραμέτρους του FATAS ελεγκτή δεν θα παραβιαστούν οι περιορισμοί της συνθήκης ασφαλείας (1.20), μετά από κάθε επανάληψη του αλγορίθμου, εάν η συνθήκη έχει παραβιαστεί, υπολογίζουμε την ορθογώνια προβολή των ενημερωμένων παραμέτρων πάνω στο κυρτό πολύεδρο $\tilde{\mathcal{Q}}$ [8] που ορίζεται από:

$$\tilde{\mathcal{Q}} = \{\hat{\tilde{w}} \in \mathbb{R}^{(K_c+1)(K+1)} : \hat{\tilde{A}}^{\intercal}\hat{\tilde{w}} \leq 0\} \tag{1.38}$$

Ο τελεστής προβολής $\mathcal{P}_{\tilde{\mathcal{Q}}}$ ορίζεται ως εξής:

$$\mathcal{P}_{\tilde{\mathcal{Q}}}(\hat{\tilde{w}}^{(i)}) = \arg\min_{\hat{\tilde{w}}\in\tilde{\mathcal{Q}}} \frac{1}{2}\|\hat{\tilde{w}} - \hat{\tilde{w}}^{(i)}\|_2^2 \tag{1.39}$$

και αποτελεί ένα πρόβλημα βελτιστοποίησης, η λύση του οποίου είναι μοναδική, λόγω της κυρτότητας του πολυέδρου $\tilde{\mathcal{Q}}$. Για να αποφύγουμε την επίλυση ενός τετραγωνικού προβλήματος, χρησιμοποιούμε έναν αλγόριθμο κυκλικών προβολών (cyclic projection algorithm) [9], ο οποίος μας επιτρέπει να βρίσκουμε την ορθογώνια προβολή ενός σημείου επάνω στην τομή κυρτών συνόλων, εκτελώντας μια ακολουθία από προβολές, ξεχωριστά σε κάθε κυρτό σύνολο που ορίζεται από τον ημιχώρο $\tilde{A}_k\hat{\tilde{w}} \leq 0$. Παρόλο που η σύγκλιση του αλγορίθμου είναι αργή για μεγάλο αριθμό κυρτών συνόλων, είναι μια πολύ σημαντική βελτίωση έναντι της επίλυσης ενός τετραγωνικού προβλήματος.

Η προβολή σε ένα κυρτό σύνολο που ορίζεται από τον ημιχώρο $\tilde{A}_k\hat{\tilde{w}} \leq 0$ γίνεται με χρήση του ακόλουθου τελεστή:

$$\mathcal{P}_k = vec(\tilde{W}^{\intercal(i+1)}) - \frac{vec(\tilde{W}^{\intercal(i+1)}) * \hat{n_k}}{\|n_k\|^2} * \hat{n}_k \tag{1.40}$$

όπου n_k είναι το κανονικό διάνυσμα στο υπερεπίπεδο που ορίζεται από την ισότητα $\hat{\tilde{A}}_k^{\intercal}\hat{\tilde{w}} = 0$. Αν συμβολίσουμε $\mathcal{T}$ το γινόμενο $\mathcal{T} = \mathcal{P}_1\mathcal{P}_2...\mathcal{P}_k$ τότε έχουμε:

$$vec(\tilde{W}_{proj}^{\intercal(i)}) = \mathcal{P}_{\tilde{\mathcal{Q}}}(vec(\tilde{W}^{\intercal(i)})) = \mathcal{T}^r(\tilde{W}^{\intercal(i)}) \; as \; r \to \infty \tag{1.41}$$

Στον αλγόριθμο (1.3) περιγράφεται ο πλήρης αλγόριθμος καθοδικής κλίσης με προβολή, που χρησιμοποιήθηκε για την βελτιστοποίηση των παραμέτρων του FATAS ελεγκτή. Είναι σημαντικό να αναφέρουμε ότι παρόλο που οι παράμετροι $\hat{\tilde{w}}$ ενημερώνονται σε κάθε επανάληψη του αλγορίθμου, η κλίση της συνάρτησης κόστους επανυπολογίζεται κάθε k επαναλήψεις. Όταν επιδιώκουμε την ελαχιστοποίηση της αντικειμενικής συνάρτησης (1.34) στην i-οστή επανάληψη του αλγορίθμου, σκοπός μας είναι να φέρουμε την τρέχουσα πολιτική ελέγχου $u^{(i+1)}$ όσο πιο κοντά γίνεται στην διαθέσιμη πληροφορία που έχουμε για την ποσότητα $-\frac{1}{2\beta}\nabla V^{(i)}$. Επανυπολογίζοντας την κλίση ∇V ύστερα από ένα μόνο βήμα προς την κατεύθυνση καθόδου, δεν αξιοποιούμε πλήρως την τρέχουσα πληροφορία για την βελτιστοποίηση της πολιτικής μας. Ωστόσο, η επιλογή μεγάλου k έχει ως αποτέλεσμα, τελικά, την μετακίνηση ως προς μια κατεύθυνση βασισμένη σε πληροφορία που δεν είναι πλέον έγκυρη. Ένα κατάλληλο k μας επιτρέπει να βελτιώνουμε τις πολιτικές μας έγκυρα και γρήγορα, βάζοντας τις βάσεις για γρηγορότερη σύγκλιση του αλγορίθμου μας.

ALGORITHM 1.3: *On-Policy Projected Gradient Descent*

Require: Initial parameters $vec(\tilde{W}^{\intercal(0)})$ (safe), $i = 0$
Require: Number of steps $k > 0$ without recalculating $\nabla V \ \forall p_d$
while vec$(\tilde{W}^{\intercal})$ has not converged **do**
1) Calculate $\nabla V^{(i)} \ \forall p_d$ using Mini-Batch Gradient Descent
2) Compute descent direction $step_{\hat{\tilde{w}}}^{(i)} = \nabla_{\tilde{W}^{\intercal}} M(\tilde{W}^{\intercal})$
3) Calculate learning rate $a^{(i)}$
4) Update parameters $vec(\tilde{W}^{\intercal(i+1)}) = vec(\tilde{W}^{\intercal(i)}) - a^{(i)} step_{\hat{\tilde{w}}}^{(i)}$
5) Projection onto safe set: $vec(\tilde{W}_{proj}^{\intercal(i+1)}) = \mathcal{P}_{\tilde{\mathcal{Q}}}(\tilde{W}^{\intercal(i+1)}) :\ \tilde{\mathcal{Q}} = \{\hat{\tilde{w}} \in \mathbb{R}^{(K_c+1)(K+1)} : \tilde{A}^{\intercal}\hat{\tilde{w}} \leq 0\}$
6) $i \leftarrow i + 1$
if $mod(i, k)! = 0$ **then**
$\nabla V^{(i+1)} = \nabla V^{(i)} \ \forall p_d$ and skip 1)
end if
end while

1.7 Αποτελέσματα Προσομοίωσης

Στην ενότητα αυτή θα παρουσιάσουμε τα αποτελέσματα του παραμετρικού μας ελεγκτή σε ένα τεχνητό περιβάλλον σχήματος ταφ. Όλες οι προσομοιώσεις έγιναν στο προγραμματιστικό περιβάλλον του Matlab. Στην εικόνα (1.1), παρουσιάζεται ο χώρος εργασίας του ρομπότ όπου το επιλεγμένο υποσύνολο του χώρου για το οποίο θέλουμε ο ελεγκτής μας να παράγει ασφαλή πεδία πλοήγησης, είναι το σκιασμένο πολύγωνο. Τα μπλε σημεία αναπαριστούν τα κέντρα p_{Cd} των Γκαουσιανών συναρτήσεων ακτινικής βάσης, ενώ τα μαύρα σημεία τους τελικούς στόχους $p_d \in p_G$ που χρησιμοποιήθηκαν στην συνθήκη ασφαλείας (1.19).

Με χρήση του αλγόριθμου καθοδικής κλίσης με προβολή, η αρχική πολιτική που προέκυψε από το τετραγωνικό πρόβλημα (1.25), ρυθμίστηκε κατάλληλα ώστε να ελαχιστοποιεί την συνάρτηση κόστους (1.28) για κάθε τελικό σημείο-στόχο εντός του επιλεγμένου υποσυνόλου. Οι παράμετροι της συνάρτησης κόστους επιλέχτηκαν ως $\alpha = 0.5$, $\beta = 0.5$ και επιπλέον, στον αλγόριθμο καθοδικής κλίσης, ο αριθμός των διαδοχικών επαναλήψεων για τις οποίες δεν γίνεται επανυπολογισμός της κλίσης της συνάρτησης κόστους ορίστηκε ως $k = 5$.

Στην εικόνα (1.2), παρουσιάζεται η σύγκλιση των παραμέτρων του ελεγκτή μαζί με τον ρυθμό εκμάθησης. Επιπλέον, στις εικόνες (1.3) εώς (1.6) απεικονίζονται τα κανονικοποιημένα διανυσματικά πεδία της αρχικής και της τελικής πολιτικής για τέσσερις τελικούς στόχους, καθώς και η βελτίωση της αντίστοιχης συνάρτησης κόστους. Αξίζει να σημειωθεί ότι η χρήση αρμονικών πολιτικών έχει ως αποτέλεσμα την ασυμπίεστη ροή των παραγόμενων πεδίων, επομένως οι βέλτιστες τροχιές που προκύπτουν δεν είναι απαραίτητα ελαχίστου μήκους.

1.8 Συμπεράσματα

Η μέθοδός μας, αν και πολλά υποσχόμενη, δεν είναι κατάλληλη να χειριστεί δυναμικά περιβάλλοντα. Επίσης, περιορίζεται από την επίλυση ενός δύσκολου τετραγωνικού προβλήματος. Σε πιο περίπλοκα περιβάλλοντα, όπως ένας λαβύρινθος ή ένας χώρος με πολλά εμπόδια, οι

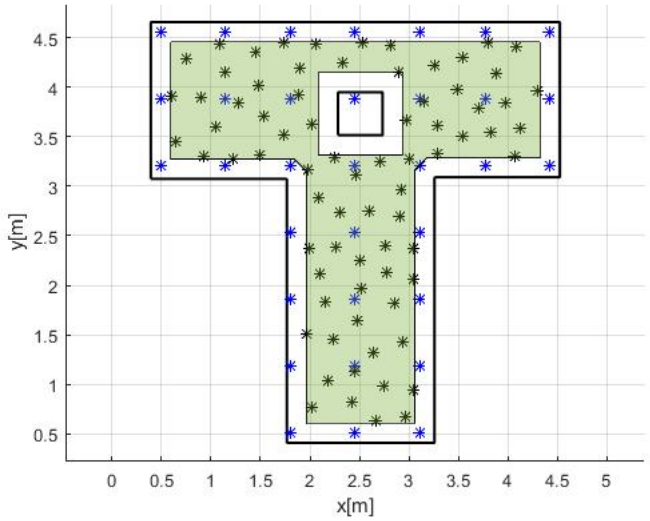
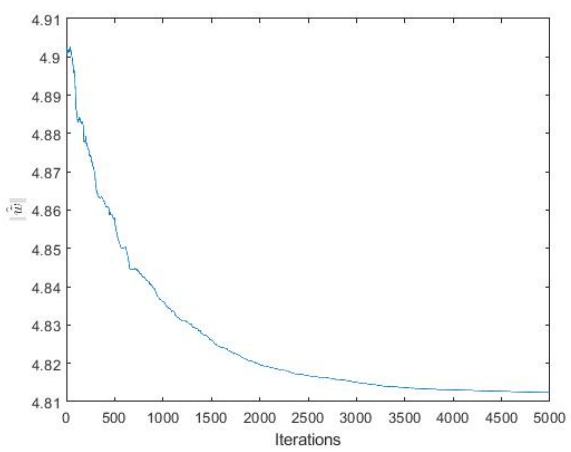
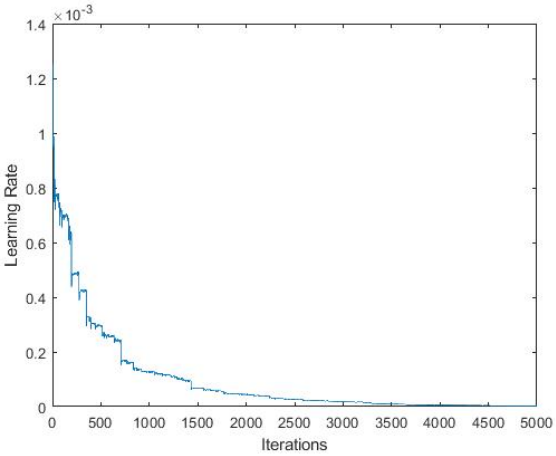


Figure 1.1: *T-Shaped Artificial Workspace*

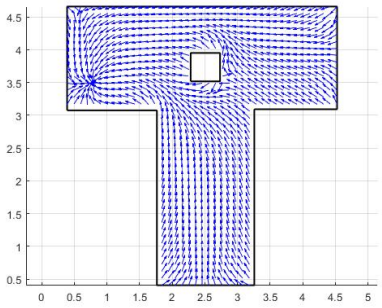


(a) *Convergence of* $\|\hat{\tilde{w}}\|$

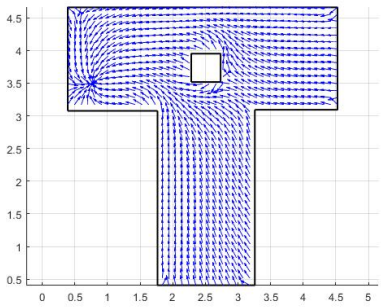


(b) *Adaptive learning rate*

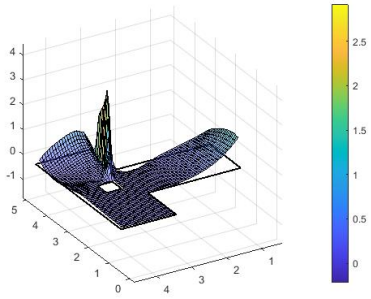
Figure 1.2: *Convergence of controller's parameters and learning rate a*



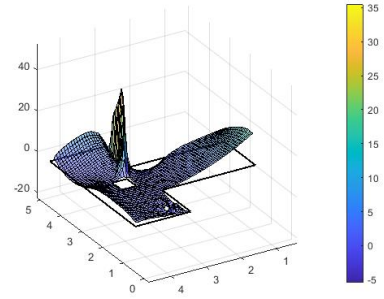
(a) *Initial Vector Field*



(b) *Final Vector Field*

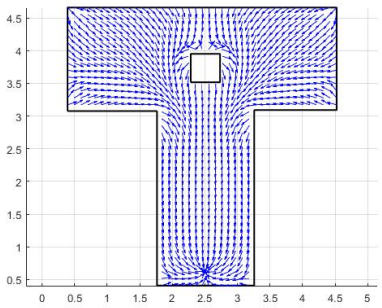


(c) *Cost Improvement*

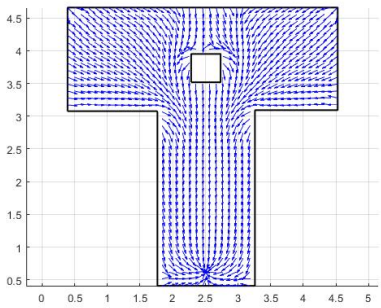


(d) *Cost Improvement (% Percentage)*

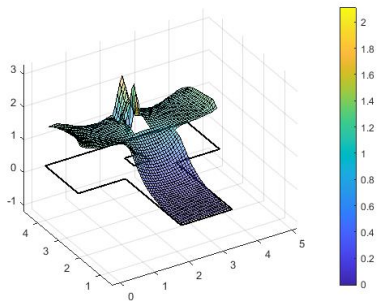
Figure 1.3: *Initial Vs Final Vector Field for (0.8,3.5)*



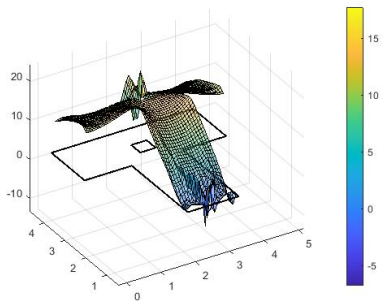
(a) *Initial Vector Field*



(b) *Final Vector Field*

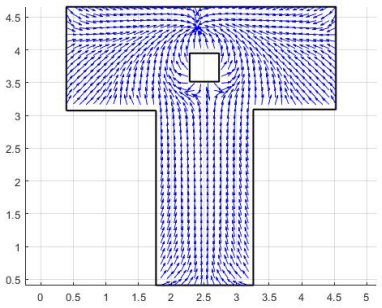


(c) *Cost Improvement*

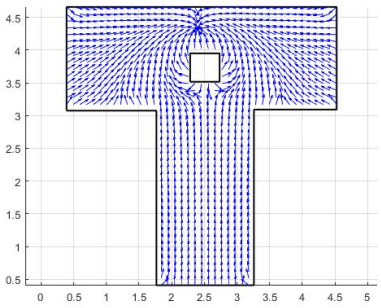


(d) *Cost Improvement (% Percentage)*

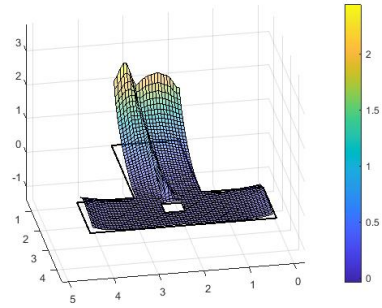
Figure 1.4: *Initial Vs Final Vector Field for (2.5,0.65)*



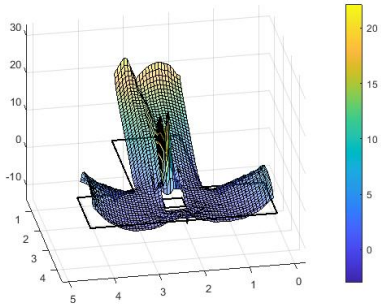
(a) *Initial Vector Field*



(b) *Final Vector Field*

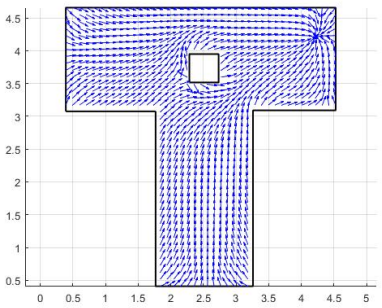


(c) *Cost Improvement*

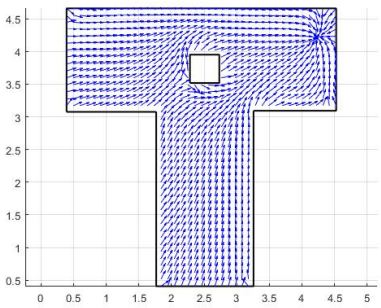


(d) *Cost Improvement (% Percentage)*

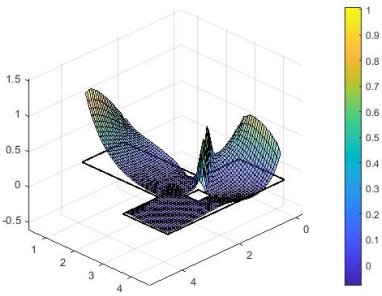
Figure 1.5: *Initial Vs Final Vector Field for (2.4,4.3)*



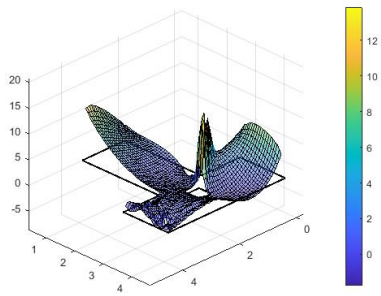
(a) *Initial Vector Field*



(b) *Final Vector Field*



(c) *Cost Improvement*



(d) *Cost Improvement (% Percentage)*

Figure 1.6: *Initial Vs Final Vector Field for (4.2,4.2)*

απαιτήσεις μνήμης κλιμακώνονται, ειδικά σε περιπτώσεις που θέλουμε ο FATAS ελεγκτής να καλύπτει ένα σημαντικό κομμάτι του χώρου εργασίας. Παρ΄ όλα αυτά, η μέθοδός μας, με το να προσφέρει ασφαλή πεδία πλοήγησης προς ένα υποσύνολο του χώρου εργασίας, μπορεί να αποτελέσει ένα σπουδαίο εργαλείο για ποικίλες εφαρμογές όπως η εκτέλεση διάφορων εργασιών σε αποθήκες προϊόντων ή η πραγματοποίηση καθηκόντων επίβλεψης.

Σε μελλοντική δουλειά, η μέθοδός μας μπορεί να επεκταθεί για περισσότερες από δύο διαστάσεις, ενσωματώνοντας στοχαστικά στοιχεία στην δυναμική του ελεγκτή. Επιπλέον, όσον αφορά τη βέλτιστη λύση στο πρόβλημα σχεδιασμού πορείας, οι παράμετροι που καθορίζουν την εξασθένιση των συναρτήσεων ακτινικής βάσης, μπορούν να θεωρηθούν ως επιπλέον παράμετροι υποκείμενοι σε βελτιστοποίηση. Τέλος, σημαντική είναι η ανάπτυξη ενός αποδεδειγμένα βέλτιστου τρόπου τοποθέτησης των κέντρων των συναρτήσεων βάσεων καθώς και των τελικών στόχων στην συνθήκη ασφαλείας των παραμέτρων του FATAS ελεγκτή, με σκοπό την ελάφρυνση του υπολογιστικού κόστους.

Chapter 2

Introduction

2.1 The Motion Planning Problem

The motion planning problem is defined as the problem of finding a safe path between two points while satisfying a set of constraints. While the task of motion planning plays a vital role in robotics and more generally in the field of automation, it has several applications in other fields, such as virtual environments, computer-aided design and computational biology. More specifically, robot motion planning is a long-standing problem that comes in a variety of forms due to peculiarities such as non-holonomic constraints or the existence of obstacles in the robot's path.

A lot of effort has gone into establishing control techniques that equip the robot with the ability to safely converge to the desired goal position while ensuring the optimality of its motion in the workspace. Additionally, suggested methods need to also take into consideration the computational efficiency. 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 method for optimal reactive motion planning from everywhere to a subset of a workspace, using a parametric controller for position-state feedback. By applying Reinforcement Learning (RL) methods, we aim to optimize every possible safe path that leads to a goal position within a subset of a two-dimensional, constrained, but fully known workspace with internal fixed obstacles. More specifically, a policy gradient technique is implemented, for the adjustment of the controller's parameters, to achieve optimality of the robot's motion with respect to a specific cost function.

2.2 Motivation

The goal of this thesis is to present a novel solution to the motion planning problem that tackles changes in the robot's target position efficiently. It is true that in most real-world applications only specific starting-ending point combinations are needed, and while the use of online approaches can offer advantages with respect to computational complexity compared to traditional offline solutions, the source of motivation behind this work is to eliminate the necessity for computational power in fully known workspaces by finding a set of parameters that allow the robot to navigate freely to any destination that lies within a

predesignated part of its environment.

2.3 Related Work

Research on the motion planning problem has yielded many fundamentally different solutions. The formulated approaches can be classified as roadmap methods (visibility graph method, Voronoi diagram) [10] , methods based on cell decomposition [11] and potential field approaches [1]. While decomposition-based methods and roadmaps can be used in many practical applications, they suffer from computational burden in highly-complex environments due to reliance on explicit representation of the obstacles in the configuration space. These struggles paved the way for the development of sampling-based algorithms such as Probabilistic Roadmaps [12] and Rapidly Exploring Random Trees [13].

The Artificial Potential Field (APF) method involves modeling the robot as a particle moving under the influence of a potential field that is determined by the set of obstacles and the desired goal-position. In this approach, the obstacles to be avoided are represented by a repulsive artificial potential and the goal is represented by an attractive potential so that the robot reaches the goal without colliding with obstacles. The method's major drawback is the existence of local minima. Navigation Functions (NF) [14] are a subclass of APFs that contributed towards broader implementation of these methods, but the avoidance of local minima required extensive tuning. These issues were resolved by the introduction of a class of APFs, namely the Artificial Harmonic Potential Fields (AHPF) [2]. AHPFs are artificial potentials based on harmonic functions. The most important property of harmonic functions is that they are free from local minima.

In this work, we expand upon AHPFs by employing a Radial Basis Function (RBF) approximation [3]. The underlying AHPF parameters take the form of a weighted sum of RBFs. Therefore, by calculating these weights, we can build a parametrized controller based on APF theory, that allows the robot to converge safely to any desired goal position within a subset of the workspace.

Optimality of the robot's motion in the workspace is achieved through RL. Optimization with RL has been successfully used in combination with harmonic-based motion planning in the past [5]. In our case, we reformulate the optimal motion planning problem as a RL one and we employ a Projected Gradient Descent (PGD) algorithm, which is a policy gradient optimization technique, in order to solve it. Projection is implemented to ensure that the parameters of the AHPF, that are subject to learning, don't jeopardize the robot's safety. The implementation of RL plays an important role, as it enables the ability to avoid solving a very hard non-linear partial differential equation for calculating the cost function.

2.4 Chapter Overview

In the following chapter we will present the optimal motion planning problem, along with preliminary background on AHPFs, Optimal Control and RL. Afterwards, in chapter 4, we will design two parametric controllers. The first one allows the robot to travel safely to a single desired goal position, and the second one is our novel proposed controller, that lets

the robot perform navigation tasks whose endpoints lie within a subset of the workspace. Then, we will present our proposed controller's results, not taking the optimality of our solution into consideration, along with the limitations of our approach. Subsequently, the implementation of RL for the optimization of the robot's movement in the workspace along with results will be presented in chapter 5. Finally, in chapter 6 we will present an efficient and robust method for optimal navigation between a predetermined sequence of waypoints in a workspace, with the use of our novel proposed controller.

Chapter 3

Problem Formulation and Preliminaries

3.1 Problem Formulation

Consider a point-robot, that navigates within a two-dimensional bounded and connected set $\mathcal{G} \subset \mathbb{R}^2$, with inner distinct obstacles $\mathcal{O}_i \subset \mathcal{G}, i = 1, ..., M$. We define the robot's workspace $\mathcal{W}$, the set $\mathcal{W} = \mathcal{G} - \bigcup_{i=1}^{M} \mathcal{O}_i$. In addition, consider a goal-position $p_d \in \mathcal{W} - \partial\mathcal{W}$. The robot moves in accordance with the single integrator dynamics:

$$\dot{p} = u,\ p(t = 0) = \bar{p} \in \mathcal{W} \tag{3.1}$$

where $p \in \mathcal{W}$ is the state vector (i.e., robot's position) , $u = u(t) : \mathbb{R} \to \mathbb{R}^2$ is a control input (i.e., input velocity) and $\bar{p}$ denotes the robot's initial position.

We consider the optimal motion planning problem, the problem of developing a control policy u that minimizes the following cost function :

$$V(\bar{p}, p_d) = \int_0^\infty [Q(p(\tau; \bar{p}); p_d) + R(u(\tau))]\, d\tau\ \forall \bar{p} \in \mathcal{W} \tag{3.2}$$

The cost function consists of a state-related term $Q(p(\tau; \bar{p}); p_d)$ and a control input related term $R(u(\tau))$, where $p(t; \bar{p}) = \int_0^t u(p(\tau; \bar{p}))\, d\tau + \bar{p}$ is the solution of $\dot{p} = u$, where $\bar{p}$ is the initial state of the system $\bar{p} = p(0)$ and p_d denotes the goal position. Our goal in this work is to present a novel parametric controller for the dynamics of Eq.(3.1), that produces optimal and safe vector fields with convergence to any desired goal position in a subset $\mathcal{S} \subset \mathcal{W}$ of the workspace.

3.2 Preliminaries

This chapter aims at providing the main tools and background that are used in this thesis. First, we will discuss the AHPFs implemented in this thesis. Next, we will present some background in Optimal Control. Finally, we make a brief introduction to RL, as we aspire to make the reformulation of the optimal motion planning problem as a RL one, more transparent.

3.2.1 Artificial Harmonic Potential Fields

The potential field implemented in this thesis, that allows the robot to safely navigate within the workspace, is derived by using the panel method with harmonic functions [2] and is defined as follows:

$$\Phi(p; p_C) = \sum_{i=0}^{K} \phi(p; p_i) w_i = w^\intercal \phi(p; p_C) \tag{3.3}$$

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

We form the basis functions' set according to [4] :

$$\phi(p; p_C) \triangleq [\phi(p; p_d), \phi(p; p_1), \cdots, \phi(p; p_K)]^\intercal : \mathcal{W} \to \mathbb{R}^{K \times 1} \tag{3.4}$$

More specifically, a single harmonic term is assigned to the robot's goal position p_d as follows:

$$\phi(p; p_d) = ln(\|p - p_d\|) \tag{3.5}$$

and the rest of the basis functions are selected following the panel method as in [4]:

$$\phi(p; p_i) = \int_{\frac{-L_i}{2}}^{\frac{L_i}{2}} ln(r_d(p, p_i, l)) dl \tag{3.6}$$

$$r_d(p, p_i, l) = \|p - (p_i + l[cos(\theta_i), sin(\theta_i)]^\intercal)\| \tag{3.7}$$

where the integral runs over the length $L_i \in \mathbb{R}_+, i = 1, \ldots, K$ of the respective panel centred at $p_i \in \mathcal{W}'$ and $\theta_i \in [0, 2\pi), i = 1, \ldots, K$ denotes the angle of the linear panel w.r.t the coordinate system over which the integration takes place. All harmonic panels are placed outside the workspace $\mathcal{W}$ to refrain from having singularities in the interior of the workspace.

3.2.2 Optimal Control

Optimal control theory is an essential tool for many fields of engineering and especially robotics. Consider the dynamical system governed by the equation $\dot{x} = f(x, u)$ and the associated cost function:

$$V(x(t)) = \int_t^\infty L(x(\tau), u(\tau)) d\tau \tag{3.8}$$

with state $x(t) \in \mathbb{R}^n$ and control input $u(t) \in \mathbb{R}^m$. Our interest lies in the determination of a control input $u^*(t) \in \mathbb{R}^m$ that drives our initial state x_0 to a desired final state x_f while minimizing the cost function.

By splitting the time interval into $[t, t + \Delta t]$ and $[t + \Delta t, \infty)$ the cost function (3.8) can

be written as:

$$V(x(t)) = \int_t^{t+\Delta t} L(x,u)d\tau + V(x+\Delta x) \tag{3.9}$$

Equation (3.9) describes all possible costs-to-go from time t. Let us define V^* as the optimal cost. Then, according to Bellman's principle of optimality, the optimal cost-to-go is:

$$V^*(x(t)) = \min_{u(\tau), t\leq\tau\leq t+\Delta t} \{\int_t^{t+\Delta t} L(x,u)d\tau + V^*(x+\Delta x)\} \tag{3.10}$$

From equation (3.10) it is clear that if we know the optimal control from time $t+\Delta t$ onwards, we only need to determine the current control $u(t)$ on the interval $[t, t+\Delta t]$.

Studying the case where $\Delta t \to 0$ we can rewrite equation (3.10) as a partial differential equation for the optimal cost $V^*(x(t))$ which is known as the Hamilton-Jacobi-Bellman (HJB) equation:

$$\min_{u(t)}(L(x,u) + \nabla V^\intercal f(x,u)) = 0 \tag{3.11}$$

where ∇V denotes the gradient of the cost function with respect to x.

The Hamiltonian function is defined as:

$$H(x,u,\lambda) = L(x,u) + \lambda^\intercal f(x,u) \tag{3.12}$$

Hence, the HJB equation can be written as

$$H(x,u^*,\nabla V^*) = 0 \tag{3.13}$$

Lastly, the optimal control policy u^* is given by the stationary condition.

$$\frac{\partial H(x,u,\nabla V^*)}{\partial u}|_{u=u^*} = 0 \tag{3.14}$$

3.2.3 Reinforcement Learning

Reinforcement Learning [15] is an area of Machine Learning that is concerned with capitalizing on data accumulated over the behavior of an agent to learn optimal control laws and policies. More specifically, an agent, can learn an appropriate sequence of actions that maximize a reward, through interacting with its environment. The agent interacts with its environment through sensory equipment, while concurrently taking actions by following a policy, while each action presents a corresponding reward. The agent's policy is optimized with the goal of maximizing future rewards.

Given a deterministic policy we can write:

$$\alpha = \pi(s):\ \mathcal{S} \to \mathcal{A} \tag{3.15}$$

to represent the action taken.

The state space $\mathcal{S}$ is composed of all the possible states that the agent can transition to and the action space $\mathcal{A}$ is the set of all actions the agent can act out in the environment. The agent arrives at a sequence of different states $s_k \in \mathcal{S}$ by performing actions $\alpha_k \in \mathcal{A}$ through

the policy π. Each of these actions results in a positive or negative reward $r_k : \mathcal{S} \times \mathcal{A} \to \mathbb{R}_+$. In many cases the policy π can be written as a parametrized policy:

$$\alpha = \pi_\theta(s) : \mathcal{S} \to \mathcal{A} \tag{3.16}$$

where θ is the parameter vector.

A policy π is evaluated by a quantity called the value function V_π, which expresses the "value" of implementing a policy π while in the state s. The value function consists of the sum over the immediate reward r_0 along with the rewards of all subsequent states:

$$V_\pi(s) = \sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s \tag{3.17}$$

where γ is the discount rate that favors immediate over long term rewards.

Our aim in RL lies in determining a policy that maximizes the total reward acquired by following the actions dictated by the policy. The techniques used to achieve this can be categorized into model-based and model-free techniques [15], depending on whether a model for the agent's environment is known or not.

In the case of a parametrized policy, policy gradient optimization is one of the most powerful tools we can employ. By directly optimizing the parameters θ of the policy, through gradient based methods like gradient ascent [16], we aim to find the parameters that maximize the expected future reward. Hence, policy gradient methods can be formulated as a maximization problem with the objective function being the expected future reward:

$$J(\theta) = V_{\pi_\theta}(s) = \sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s \tag{3.18}$$

The policy parameters in that case are updated as follows:

$$\theta_{k+1} = \theta_k + \alpha \nabla_\theta J(\theta) \tag{3.19}$$

Chapter 4

From All To A Subset Motion Planning

In this chapter, we will firstly present the From-All-To-A-Point (FATAP) controller, which is a parametric controller that allows a robot to safely navigate to a desired goal position. Subsequently, based on the FATAP controller, we will construct the From-All-To-A-Subset (FATAS) controller, that enables a robot to navigate to any goal within a subset of its workspace.

4.1 From-All-To-A-Point (FATAP) Controller

By employing AHPF theory, we propose the following parametric controller for safe navigation towards a single desired goal position $p_d \in \mathcal{W} - \partial\mathcal{W}$:

$$u(p) = -\|p - p_d\|^2 \nabla\Phi(p; p_C) = -\|p - p_d\|^2 \nabla\phi^\intercal(p; p_C)w \tag{4.1}$$

where

$$\nabla\phi(p; p_d) = \frac{p - p_d}{\|p - p_d\|^2} \tag{4.2}$$

Since the gradient of the field Φ exhibits a singularity at the goal position, the term $\|p - p_d\|^2$ is employed to render the singularity at $p = p_d$ equal to zero, and the control law stabilizing at p_d for $w_0 > 0$.

In order to obtain a safe field during navigation tasks, the following condition is imposed:

$$n^\intercal(z)u(z) \geq 0, \forall z \in \partial\mathcal{W} \tag{4.3}$$

where $u(z)$ is the underlying vector field dictating the robot's motion and $n(z)$ denotes the normal vector at each point of the boundary pointing inwards. Although the safety condition involves the whole set of points over the boundary $\partial\mathcal{W}$ of the workspace, this property can be relaxed if the above inequality holds for a finite set of boundary points [5].

Theorem 4.1 (Theorem 1 of [5])**.** *Consider the boundary $\partial\mathcal{W}$ of the workspace as well as a finite number of uniformly distributed points $p_j \in \partial\mathcal{W},\ j = 1\ldots, N$ along with their respective normal vectors $n_j = n(p_j),\ j = 1\ldots, N$ pointing inwards the workspace. There exists a number $N_0 \in \mathbb{N}$ such that*

$$n_j^\intercal u_j > 0\ \forall j = 1, \ldots, N \text{ with } N \geq N_0 \tag{4.4}$$

guarantees safety over the whole boundary $\partial\mathcal{W}$ as described by (4.3).

The safety condition (4.3) can be expressed as a set of linear inequalities w.r.t the weights of the basis functions. In matrix form, we express the safety condition as:

$$A^\intercal w \leq 0 \tag{4.5}$$

where $A = [A_1, \ldots, A_N]$, with $A_i = n^\intercal(z_i)\nabla\phi^\intercal(z_i; p_C)$.

Thus, the AHPF parameters that guarantee convergence and safety can be obtained by solving the following constrained quadratic problem:

$$\min_w \|w\|^2, \ s.to \ A^\intercal w \leq 0, w_0 > 0 \tag{4.6}$$

We should note here that in practice, to avoid the zero vector being the solution to the quadratic problem (4.6) we employ a small constant $\epsilon > 0$ and solve the following problem:

$$\min_w \|w\|^2, \ s.to \ A^\intercal w \leq -\epsilon, w_0 > 0 \tag{4.7}$$

Due to the scale invariance of the potential field, we can modify the quadratic problem (4.7) appropriately so that the restriction $w_0 = 1$ applies. By doing so we guarantee that the obtained initial policies belong in the same family of solutions. Through RL, our initial policy can be later improved upon to achieve not only the aforementioned convergence and safety but also optimality with respect to a specific cost function.

4.2 From-All-To-A-Subset (FATAS) Controller

We will now construct the FATAS parametric controller based on the FATAP controller. First, consider the following parametrization for the weights of the FATAP controller:

$$w = w(p_d; p_{Cd}) = W^\intercal s(p_d; p_{Cd}) \tag{4.8}$$

The matrix $W \in \mathbb{R}^{K_c \times (K+1)}$ is a new set of parameters , where $p_{Cd} = \{p_{d,1}, p_{d,2}, \cdots, p_{d,K_C}\}, p_{d,i} \in \mathbb{R}^2, i = 1, \ldots, K_C$ are the positions of the centers of the basis functions, with the set of basis functions defined as the mapping $s(p_d; p_{Cd}) : (\mathcal{W} - \partial\mathcal{W}) \to \mathbb{R}^{K_c}$ for a given choice of basis functions' centers. In the next section we will discuss our choice of basis functions. The key insight here is that for a given weight matrix W and a constant desired position in a subset of the workspace $p_d \in \mathcal{S} \subset \mathcal{W}$, equation (4.8) yields constant weights $w \in \mathbb{R}^{K+1}$ and thus the field :

$$\Phi(p; p_d; p_C) = \phi^\intercal(p; p_C) W^\intercal s(p_d; p_{Cd}), \ p_d \in \mathcal{S} \tag{4.9}$$

is harmonic, and its gradient becomes:

$$\nabla\Phi(p; p_d; p_C) = \nabla\phi^\intercal(p; p_C) W^\intercal s(p_d; p_{Cd}), \ p_d \in \mathcal{S} \tag{4.10}$$

However, the field (4.9) is not evidently linear w.r.t. the new parameters. We remedy this

by employing the following identity for an $m \times n$ matrix A and $r \times q$ matrix B:

$$vec(AXB) = (B^\intercal \otimes A)vec(X) \tag{4.11}$$

Thus equation (4.10) becomes:

$$vec(\nabla\Phi(p; p_d; p_C)) = \nabla\Phi(p; p_d; p_C) = (s^\intercal(p_d; p_{C_d}) \otimes \nabla\phi^\intercal(p; p_C))vec(W^\intercal), p_d \in \mathcal{S} \tag{4.12}$$

where

$$s^\intercal(p_d; p_{C_d}) \otimes \nabla\phi^\intercal(p; p_C) \in \mathbb{R}^{2\times K_c(K+1)}$$
$$\hat{w} = vec(W^\intercal) \in \mathbb{R}^{K_c(K+1)}$$

Finally, the safety condition boils down to:

$$n^\intercal(z)(s^\intercal(p_d; p_{Cd}) \otimes \nabla\phi^\intercal(z; p_C))\hat{w} \leq 0, \forall z \in \partial\mathcal{W}, \forall p_d \in \mathcal{S} \tag{4.13}$$

which is evidently linear w.r.t. $\hat{w}$. Although the safety condition (4.13) involves every goal position inside the subset $\mathcal{S}$, it can be relaxed if it holds for a finite set of goals $p_G \triangleq \{p_{d1}, p_{d2}, \cdots, p_{dm}\}$, due to the nature of Radial Basis Function (RBF) approximation, which we will discuss shortly.

The safety condition can be expressed like before, as a system of linear inequalities w.r.t the parameters $\hat{w}$:

$$\tilde{A}^\intercal\hat{w} \leq 0 \tag{4.14}$$

where $\tilde{A} = [\tilde{A}_1, \ldots, \tilde{A}_N]$, with $\tilde{A}_i = [s^\intercal(p_{di}, p_{Cd})] \otimes (n^\intercal(z)\nabla\phi^\intercal(z; p_C))]$

Similarly to the FATAP controller, the parameters W can be obtained through a constrained quadratic problem:

$$\min_{\hat{w}} \|\hat{w}\|^2, \ s.to \ \tilde{A}^\intercal\hat{w} \leq -\epsilon, \quad w_0(p_d) = W_1^\intercal s(p_d; p_{Cd}) > 0 \tag{4.15}$$

where $w_0(p_d)$ 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, first, we extend our basis function in the following way:

$$\tilde{s}(p_d; p_{Cd}) = [1 \ s(p_d; p_{Cd})] \tag{4.16}$$

Thus, our new set of parameters becomes $\tilde{W} \in \mathbb{R}^{(K_c+1)\times(K+1)}$ and:

$$w = \tilde{w}(p_d; p_{Cd}) = \tilde{W}^\intercal\tilde{s}(p_d; p_{Cd}) \tag{4.17}$$
$$\hat{\tilde{w}} = vec(\tilde{W}^\intercal) \tag{4.18}$$

Furthermore, the constrained quadratic problem (4.15) becomes:

$$\min_{\hat{\tilde{w}}} \|\hat{\tilde{w}}\|^2, \ s.to \ \hat{\tilde{A}}\hat{\tilde{w}} \leq -\epsilon, \ \ A_{eq}\hat{\tilde{w}} = B_{eq} \tag{4.19}$$

where with proper equality constraints we can impose the following constraint:

$$\tilde{W}_1^\intercal = \begin{bmatrix} 1 & 0 & \dots & 0 \end{bmatrix} \tag{4.20}$$

and thus:

$$\tilde{W}_1^\intercal \tilde{s}(p_d; p_{cd}) = 1 \tag{4.21}$$

The solution to the quadratic problem (4.19) consists of our initial policy's parameters, which guarantee safety and convergence for any goal position within the subset $\mathcal{S}$. These parameters will be later improved upon with the use of RL to achieve optimality with respect to a specific cost function.

Summarizing, our proposed controller is given by:

$$u(p, p_d) = -\|p - p_d\|^2 \nabla \phi^\intercal(p; p_C) \tilde{W}^\intercal \tilde{s}(p_d; p_{Cd}) \tag{4.22}$$

Through this set of parameters $\tilde{W}$, we can express the weights of the AHPF for any desired goal position within a subset of the workspace as a weighted sum of the basis functions. We are now ready to discuss in the next section our choice of these functions.

4.3 Choice of Basis Functions

Although the basis functions may be chosen arbitrarily, we propose employing Radial Basis Functions (RBFs) with uniformly placed centers. RBFs are functions whose value depends only on the distance between the input and a fixed point, called the center. As the distance between the point and the center increases, the value of the RBF gets smaller, therefore RBFs can be used to approximate the local characteristics of a function close to the center. RBF approximation [3] is universally applicable to both higher and lower dimensions due to the excellent approximation properties of RBFs [17]. The centers of the RBFs are placed not only within, but also outside the subset for which we want to obtain safe navigation policies. The reason for this is that it helps us better approximate the vector field parameters for goal positions near the boundary of our chosen subset.

In this work we employ Gaussian functions, which belong to the class of infinitely smooth RBFs and are among the most commonly used RBFs:

$$\phi(r) = e^{-(\epsilon r)^2}$$

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 overlap at 0.6:

$$s_i(p^*; p_{d,i}) = s_{i+1}(p^*; p_{d,i+1}) = 0.6 \tag{4.23}$$

where p^* is the point in the middle of two neighbouring centers.

4.4 Simulation Results

In this section we will present the results of our parametric controller for different goal positions in two artificial workspaces. All simulations in this thesis were implemented in Matlab on a PC running Windows 10, on an intel-i5 dual-core processor, with 8GB RAM.

In Fig.(4.1), we illustrate the first artificial workspace used to test our parametric controller, which consists of a T-shaped outer boundary and one inner square-shaped obstacle. The chosen subset for which we want our controller to produce safe navigation policies to, is the highlighted polygon. The blue points represent the centers of the Gaussian RBFs p_{Cd}, used to parametrize the AHPF weights and the black ones, the set of goal positions p_G involved in the safety condition (4.13). Similarly, in Fig.(4.2), we have chosen a subset within a more complex, office-like workspace. In Fig.(4.3), we present four normalized vector fields that resulted for various goal positions inside the chosen subset of the T-shaped workspace. All of them exhibit both safety and convergence to the goal position. Furthermore, Fig.(4.4) contains four more normalized vector fields produced from the FATAS controller for points within the designated subset of the workspace illustrated in Fig.(4.2).

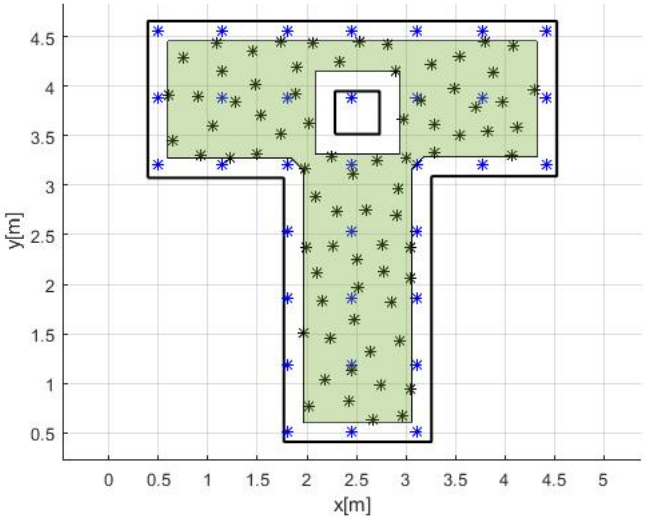


Figure 4.1: *T-Shaped Artificial Workspace*

4.5 Discussion and Limitations

We shall now address our proposed approach's limitations. First and foremost, the constrained quadratic problem that has to be solved in order to obtain our controller's parameters, is computationally very expensive. More complex environments, such as mazes, or a workspace with many obstacles, require a higher amount of harmonic panels and more normal vectors placed at the boundary to derive a vector field that guarantees safety. As a result, we also need an increased number of basis functions and goal positions in the safety condition (4.13). This results in a problem that is too memory intensive, especially in cases where we want our subset to cover a large area of the workspace. Furthermore,

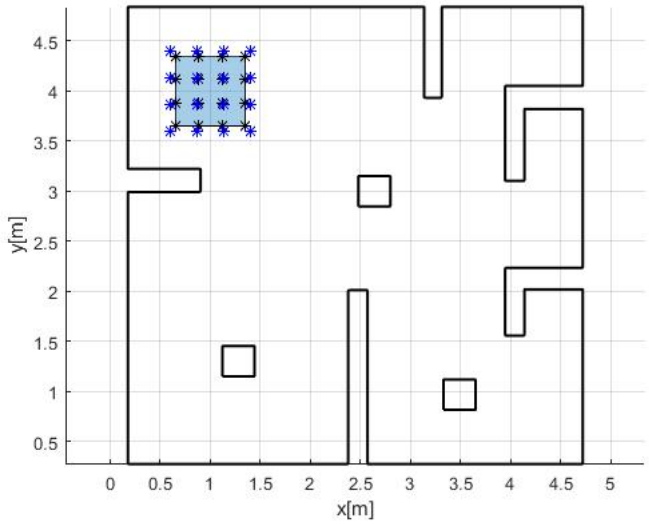
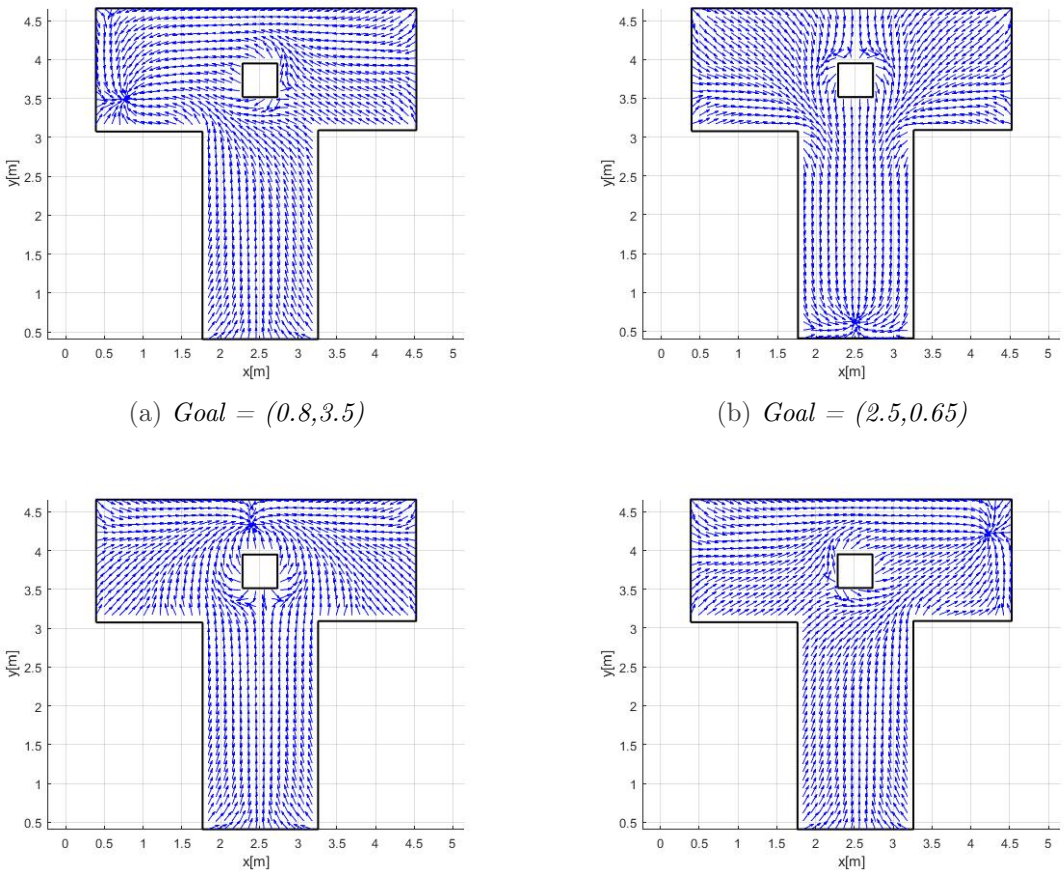


Figure 4.2: *Office-Like Artificial Workspace*

employing goal positions near the boundary of the workspace in the safety condition, results in even greater computational issues, making our controller unfit for navigation tasks near the boundary of a workspace.

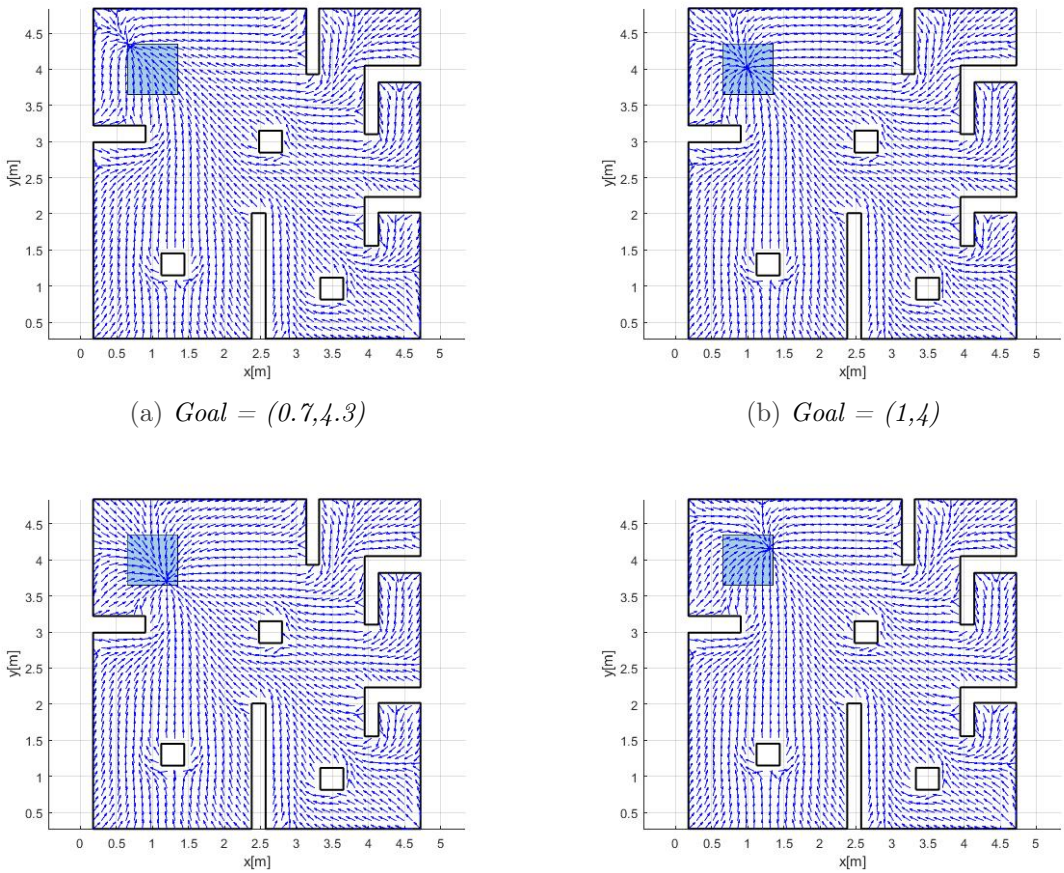
Despite of our approach's weaknesses, we can easily employ our method in simpler environments for a subset that covers a large area of the workspace such as the T-shaped workspace in Fig.(4.1). Also, for more complex environments, it is important to note that the FATAS controller can still prove to be highly useful. It is not unreasonable for a robot in a real-life scenario to be required to visit only a pre-determined area throughout its work-life. In chapter 6 we will discuss an application of our novel controller for the aforementioned scenario.



(a) *Goal = (0.8,3.5)* (b) *Goal = (2.5,0.65)*

(c) *Goal = (2.4,4.3)* (d) *Goal = (4.2,4.2)*

Figure 4.3: *Four different goal positions and their respective normalized vector fields.*



(a) *Goal = (0.7,4.3)*

(b) *Goal = (1,4)*

(c) *Goal = (1.2,3.7)*

(d) *Goal = (1.3,4.15)*

Figure 4.4: *Four different goal positions and their respective normalized vector fields.*

Chapter 5

Optimal Motion Planning Using Reinforcement Learning

In this chapter, we tackle the optimal motion planning problem with the use of RL. First, we reformulate our problem as a RL one and provide a detailed description of the implemented Projected Gradient Descent (PGD) algorithm that adjusts the parameters of the FATAP controller for the minimization of our cost function. Afterwards, we extend the algorithm for the FATAS parametric controller and in the last section, we present the results of our novel controller's optimization.

5.1 Problem Formulation

As we have established in chapter 2, we consider the optimal motion planning problem, the problem of developing a control policy u that minimizes the following cost function:

$$V(\bar{p}, p_d) = \int_0^\infty [Q(p(\tau; \bar{p}); p_d) + R(u(\tau))]\, d\tau \;\forall \bar{p} \in \mathcal{W} \tag{5.1}$$

The cost function consists of a state-related term $Q(p(\tau; \bar{p}); p_d)$ and a control input related term $R(u(\tau))$ where:

$$Q(p(\tau; \bar{p}); p_d) = \alpha \|p(\tau; \bar{p}) - p_d\|^2 \tag{5.2}$$

$$R(u(\tau)) = \beta \|u(\tau)\|^2 \tag{5.3}$$

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. Thus, the cost function subject to minimization can be written as:

$$V(\bar{p}, p_d) = \int_0^\infty [\alpha \|p(\tau; \bar{p}) - p_d\|^2 + \beta \|u(\tau)\|^2]\, d\tau \;\forall \bar{p} \in \mathcal{W} \tag{5.4}$$

We will now define the Hamiltonian function based on the adopted cost function (5.4):

$$H(p, u, \nabla V) = \nabla V^\intercal u + \alpha \|p - p_d\|^2 + \beta \|u\|^2 \tag{5.5}$$

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

$$H(p, u^*, \nabla V^*) = 0 \tag{5.6}$$

while the optimal control policy is given by the stationary condition $\frac{\partial H(p,u,\nabla V^*)}{\partial u}|_{u=u^*} = 0$ as:

$$u^* = -\frac{1}{2\beta}\nabla V^* \tag{5.7}$$

To obtain the optimal control policy u^*, an analytic expression for the cost function $V^*(p; p_d)$ is required. By substituting the optimal control policy in the HJB optimality condition (5.6) we can obtain such an expression at the cost of solving a hard non-linear partial differential equation. The difficulty increases if we take into consideration the safety conditions that need to be satisfied on the boundary of the workspace. Instead, by applying a policy gradient optimization technique we can adjust the weights of our parametric controller appropriately so as to satisfy the stationary condition as well as the safety over the workspace boundary. In the next section, we shall discuss the reformulation of our problem as a RL one, in order to apply the aforementioned policy gradient optimization technique.

5.2 Optimal Motion Planning as a Reinforcement Learning Problem

Looking at the optimal motion planning problem through the prism of RL, we can interpret the robot as an agent interacting with its environment, 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) $u_w(p)$. The value function, corresponds to the cost function (5.4), which in our case is subject to minimization. Since the robot's policy consists of our parametric controller, and due to the nature of the robot's state and action space, our problem is a continuous, deterministic and model-based RL problem.

The minimization of the cost function (5.4) coincides with finding the optimal control policy u^* and the optimal cost V^* that satisfy the stationary condition (5.7). Thus, the solution of our RL problem is reduced to finding the optimal parameters w^* of our controller.

The policy gradient optimization algorithm used in this work is based on Policy Iteration which is a widely used technique that solves the HJB equation [6]. We will discuss how we can obtain the gradient of the cost function V later. Since what we aim to find are the optimal values for the weights w, we write:

$$u_w^{(i+1)} = \sigma^\intercal w^{(i+1)} \tag{5.8}$$

where:

$$\sigma = -\|p - p_d\|^2 \nabla\phi(p; p_C)$$

ALGORITHM 5.1: *Off-line Policy Iteration*

Require: Any admissible control policy $u^{(0)}$ and tolerance $\epsilon > 0$
  while $\|u^{(i+1)} - u^{(i)}\| > \epsilon$ **do**
    1) Calculate $\nabla V^{(i)}$ based on policy $u^{(i)}$
    2) Update the control policy: $u^{(i+1)} = -\frac{1}{2\beta}\nabla V^{(i)}$
    3) $i \leftarrow i + 1$
  end while

Thus, the weights w on each iteration can be obtained by solving the following constrained quadratic problem:

$$\min_{w} : \|\sigma^{\intercal} w^{(i+1)} + \frac{1}{2\beta}\nabla V^{(i)}\|^2 \tag{5.9}$$

$$\textit{subject to } A^{\intercal} w \leq 0$$

Since we want to avoid the computational burden of solving a constrained quadratic problem we will employ a PGD algorithm, which is is a policy gradient RL technique used to solve constrained optimization problems. In the next section we will present the steps of our algorithm and address any remaining issues.

5.3 Projected Gradient Descent

First, let's define the following objective function:

$$M(w) = \|\sigma^{\intercal}(p) w + \frac{1}{2\beta}\nabla V(p)\|^2 \tag{5.10}$$

Our controller's parameters are then updated as follows:

$$w^{(i+1)} = w^{(i)} - a\nabla_w M(w) \tag{5.11}$$

where

$$\nabla_w M(w) = 2\sigma(p)(\sigma^{\intercal}(p) w + \frac{1}{2\beta}\nabla V(p))$$

is the descent direction and a is a tuning parameter called the learning rate.

We shall now address the following remaining issues. First, we will tackle the calculation of the gradient of the cost function V. Subsequently, we will present the adaptive algorithm used to calculate the learning rate a. Finally, we will discuss the implemented projection algorithm.

5.3.1 Gradient of Cost Function

To obtain the gradient of the cost function V, first, we need an estimation for the cost function. Such an estimation can be obtained by executing trajectories from initial points across the boundary of the robot's workspace $\mathcal{W}$ and solving a system of differential

equations for each initial point.

$$\begin{bmatrix} \dot{p}(t) \\ \dot{\bar{V}}(p;p_d) \end{bmatrix} = \begin{bmatrix} u(p) \\ r(p,u) \end{bmatrix} \quad \begin{bmatrix} p(0) \\ \bar{V}(\bar{p}_j;p_d) \end{bmatrix} = \begin{bmatrix} \bar{p}_j \\ 0 \end{bmatrix} \tag{5.12}$$

$$\text{where } r(p,u) = \alpha\|p-p_d\|^2 + \beta\|u\|^2 \tag{5.13}$$

Solving (5.12) gives us the points p_i in the robot's trajectory that starts from a boundary point $\bar{p}_j$ and ends on the robot's goal position p_d. Also, for each point in the trajectory, we obtain an estimation for the accumulated cost $\bar{V}(p_i;p_d)$. Thus, we acquire an estimation for the cost-to-go for each point in the trajectory in the following way:

$$\hat{V}(p_i;p_d) = \bar{V}(p_d;p_d) - \bar{V}(p_i;p_d) \tag{5.14}$$

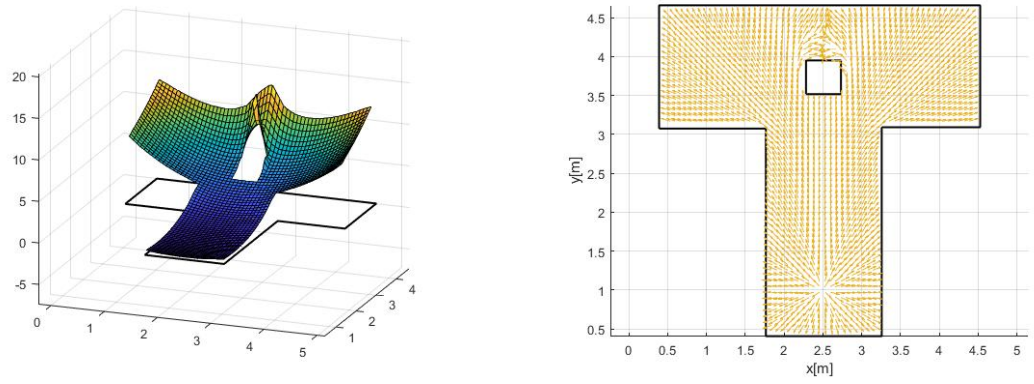
Repeating this process for a finite number of uniformly distributed boundary points allows us, through interpolation, to easily get the cost-to-go $V(p)$ for all $p \in \mathcal{W}$. With an estimation for the cost-to-go available, it is now easy to obtain its gradient for points in a two-dimensional grid inside the workspace. To calculate the descent direction, we will use the gradient of each point in the chosen grid and calculate the mean value of equation (5.15) for all points.

$$\nabla_w M(w) = \frac{2}{K}\sum_{j=1}^{K}\sigma(p_j)(\sigma^\intercal(p_j)w + \frac{1}{2\beta}\nabla V(p_j)) \tag{5.15}$$

Using points from all over the workspace will result in a Full-Batch training algorithm. However, executing trajectories from a large number of initial boundary points increases the computational complexity. To remedy this, we restrict ourselves to a randomly chosen subset of the boundary points each time we need to calculate the descent direction. If we choose this subset to include only neighbouring points, we are able to get an estimation for the cost function in a small area of the workspace. Extracting the gradient and substituting it in equation (5.15) to calculate the descent direction results in what is known as a Mini-Batch training algorithm. In Fig.(5.1), we illustrate the difference between the two methods of acquiring the gradient.

5.3.2 Learning Rate

The adaptive method used in this work to calculate the learning rate is derived from ADADELTA [7]. This method inherits important benefits of the ADADELTA approach, including robustness to noisy gradients and the lack of dependence on a manual learning rate. The main difference is that ADADELTA has a separate dynamic learning rate per-dimension, while our method produces a global learning rate by using the info from the gradients of all parameters. The reason for this is that in practice, a global learning rate proved to be more suitable for our problem. The steps of the algorithm are presented in Alg.(5.2).



(a) *Cost Function* (b) *Gradient (Normalized)*

(c) *Cost Function (Partial)* (d) *Gradient*

Figure 5.1: *Full-Batch vs Mini-Batch Training*

ALGORITHM 5.2: *Computing learning rate update at iteration i*

Require: Decay rate ρ , Constant ϵ
Require: Initial parameters $w^{(0)}$
Require: Initialization of accumulation variables $E[step_w^2]_0 = 0,\ E[\Delta w^2]_0 = 0$
for $i = 1 : N$ **do**
1) Compute gradient $step_w^{(i)} = \nabla_w M(w)$
2) Accumulate Gradient $E[step_w^2]_i = \rho E[step_w^2]_{(i-1)} + (1-\rho) step_w^{2(i)}$
3) Compute Update: $\Delta w_i = -\frac{RMS[\Delta w]_{(i-1)}}{RMS[step_w]_i} step_w^{(i)}$
where $RMS[x]_i = \sqrt{E[x^2]_i + \epsilon}$
4) Accumulate Updates: $E[\Delta w^2]_i = \rho E[\Delta w^2]_{i-1} + (1-\rho)\Delta w_i^2$
5) Apply Update : $w^{i+1} = w^i + \Delta w_i$
end for

5.3.3 Projection Onto Convex Sets

We use projection in the Gradient Descent algorithm to ensure that the updated underlying AHPF parameters satisfy the safety conditions.

As we established in chapter 3, the safety condition can be expressed as a set of linear inequalities w.r.t the weights w. Each inequality $A_i w \leq 0$ defines a closed half-space, and thus is a convex set [8]. The intersection of all the closed half-spaces defines a convex polyhedral $\mathcal{Q}$:

$$\mathcal{Q} = \{w \in \mathbb{R}^{(K+1)} : A^\intercal w \leq 0\} \tag{5.16}$$

The projection operator $\mathcal{P}_\mathcal{Q}$ is defined as:

$$\mathcal{P}_\mathcal{Q}(w^{(i)}) = \arg\min_{w \in \mathcal{Q}} \frac{1}{2}\|w - w^{(i)}\|_2^2 \tag{5.17}$$

which itself is an optimization problem. Since $\mathcal{Q}$ is convex, the problem has a unique solution. However, since we want to avoid solving a constrained quadratic problem, we employ a cyclic projection algorithm [9]. The cyclic projection algorithm allows us to find 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. Although convergence is slow for a large number of convex sets, it is a drastic improvement over solving a quadratic problem. We can project onto each individual convex set defined by the closed half $A_k w \leq 0$, with the use of the following projection operator:

$$\mathcal{P}_k = w^{(i)} - \frac{w^{(i)} * \hat{n_k}}{\|n_k\|^2} * \hat{n}_k \tag{5.18}$$

where $\hat{n}_k = \frac{A_k}{\|A_k\|}$ denotes the normal vector to the hyperplane $A_k w = 0$.

If we denote $\mathcal{T}$ as the product $\mathcal{T} = \mathcal{P}_1 \mathcal{P}_2 ... \mathcal{P}_k$ then we can get the orthogonal projection of the weights w onto the convex polyhedral $\mathcal{Q}$ in the following way:

$$w_{proj}^{(i)} = \mathcal{P}_\mathcal{Q}(w^{(i)}) = \mathcal{T}^r(w^{(i)}) \; as \; r \to \infty \tag{5.19}$$

5.3.4 Complete Algorithm

We will now present the complete Projected Gradient Descent algorithm. Since in our algorithm we calculate the gradient of the cost function using the policy subject to optimization, it is considered an on-policy algorithm. It is important to mention that even though we update the parameters w on each iteration, the gradient of the cost function V is only recalculated every k iterations. When we attempt to minimize the objective function $M(w)$ in the i_{th} iteration, what we are aiming for is to make our current policy $u^{(i+1)}$ get closer to the available data $-\frac{1}{2\beta}\nabla V^{(i)}$. By recalculating ∇V after performing only a single step towards the descent direction, we have not fully taken advantage of the currently available data to optimize our policy. However, choosing a very large value for k results in, eventually, moving in a direction based on data that is no longer valid for our policy. A properly tuned value for k allows us to improve our policies faster and more

accurately, laying the groundwork for faster convergence of our algorithm.

ALGORITHM 5.3: *On-Policy Projected Gradient Descent*

Require: Initial parameters $w^{(0)}$ (safe), $i = 0$
Require: Number of steps $k > 0$ without recalculating ∇V
while weights have not converged **do**
1) Calculate $\nabla V^{(i)}$ using Mini-Batch-Gradient Descent
2) Compute descent direction $step_w^{(i)} = \nabla_w M(w)$
3) Calculate learning rate $a^{(i)}$
4) Update weights $w^{(i+1)} = w^{(i)} - a^{(i)} step_w^{(i)}$
5) Projection onto safe set:$w_{proj}^{(i+1)} = \mathcal{P}_{\mathcal{Q}}(w^{(i+1)}) :\ \mathcal{Q} = \{w \in \mathbb{R}^{(K+1)} : A^\intercal w \leq 0\}$
6) $i \leftarrow i + 1$
if $mod(i, k)! = 0$ **then**
$\nabla V^{(i+1)} = \nabla V^{(i)}$ and skip 1)
end if
end while

5.4 Optimization of the FATAS Controller

In this section, we will present the Projected Gradient Descent algorithm that adjusts the parameters of the FATAS controller so that the cost function (5.4) is minimized for all $p_d \in \mathcal{S}$, where $\mathcal{S}$ is the predesignated subset for which the controller can provide safe navigation policies to.

Similarly to the FATAP controller, based on the Policy Iteration algorithm we can obtain the policy u at each iteration by solving the following constrained quadratic problem:

$$\min_u : \|u^{(i+1)}(p_d) + \frac{1}{2\beta}\nabla V^{(i)}(p_d)\|^2 \ \forall p_d \in p_G \tag{5.20}$$

$$subject\ to\ \hat{\tilde{A}}^\intercal \hat{\tilde{w}} \leq 0 \tag{5.21}$$

Since what we are aiming to find is the parameters $\tilde{W}^*$ of the FATAS controller, we write:

$$u_{\hat{\tilde{w}}}^{(i+1)} = \sigma^\intercal \tilde{W}^{\intercal(i+1)} \tilde{s}(p_d; p_{Cd}) \tag{5.22}$$

where:

$$\sigma = -\|p - p_d\|^2 \nabla \phi(p; p_c)$$

Thus, the parameters $\tilde{W}$ on each iteration can be obtained by solving the constrained quadratic problem (5.23). The set p_G denotes the set of goal positions used in the safety condition from which we derived our controller's initial parameters.

$$\min_{\hat{\tilde{w}}=vec(\tilde{W}^\intercal)} : \|\sigma^\intercal \tilde{W}^{\intercal(i+1)} \tilde{s}(p_d; p_{Cd}) + \frac{1}{2\beta}\nabla V^{(i)}(p_d)\|^2 \ \ \forall p_d \in p_G \tag{5.23}$$

$$subject\ to\ \tilde{A}^\intercal \hat{\tilde{w}} \leq 0$$

To apply the Projected Gradient Descent algorithm for the FATAS controller's parameters, first we define the objective function:

$$M(\tilde{W}^\intercal) = \|\sigma^\intercal(p)\tilde{W}^\intercal\tilde{s}(p_d; p_{cd}) + \frac{1}{2\beta}\nabla V(p, p_d)\|^2 \tag{5.24}$$

Our controller's parameters are updated as follows:

$$\hat{\tilde{w}}^{(i+1)} = \hat{\tilde{w}}^{(i)} - a\nabla_{\tilde{W}^\intercal} M(\tilde{W}^\intercal) \tag{5.25}$$

where

$$\nabla_{\tilde{W}^\intercal} M(\tilde{W}^\intercal) = 2(\tilde{s}(p_d; p_{Cd}) \otimes \sigma(p))(\sigma^\intercal(p)\tilde{W}^\intercal\tilde{s}(p_d; p_{Cd}) + \frac{1}{2\beta}\nabla V(p, p_d)) \tag{5.26}$$

is the descent direction ($\nabla_{\tilde{W}^\intercal} M(\tilde{W}^\intercal) \in \mathbb{R}^{(K+1)(Kc+1)}$) and a is the learning rate.

As in the case of the FATAP controller, to compute the descent direction we need to calculate the gradient of the cost function V, this time for all $p_d \in p_G$. However, for computational efficiency we shall use only N points, chosen randomly from the set p_G each time the descent direction is computed, where $N < m$ and m is the number of points in the set p_G. Then, for each p_{di}, where $1 \leq i \leq N$, we will calculate an approximation of the true gradient $\nabla V(p, p_{di})$ by using mini-batches. Finally, we obtain the descent direction by calculating the mean value of equation (5.26) for all mini-batches of the chosen N goal positions:

$$\nabla_{\tilde{W}^\intercal} M(\tilde{W}^\intercal) = \frac{2}{N}\sum_{i=1}^{N}\sum_{j=1}^{K_i}\frac{1}{K_i}(\tilde{s}(p_{di}; p_{Cd}) \otimes \sigma(p_j))(\sigma^\intercal(p_j)\tilde{W}^\intercal\tilde{s}(p_{di}; p_{Cd}) + \frac{1}{2\beta}\nabla V(p_j, p_{di})) \tag{5.27}$$

The learning rate α is calculated with the same adaptive method we described in the previous section which provides us with a global learning rate based on the descent direction of all parameters.

Finally, to ensure the safety of our policies, if any of the constraints is violated during the learning process, we need to find the orthogonal projection of our parameters onto the convex polyhedral $\tilde{\mathcal{Q}}$, defined by the safety condition:

$$\tilde{\mathcal{Q}} = \{\hat{\tilde{w}} \in \mathbb{R}^{(K_c+1)(K+1)} : \hat{\tilde{A}}^\intercal\hat{\tilde{w}} \leq 0\} \tag{5.28}$$

The cyclic projection algorithm allows us to find the orthogonal projection onto the convex polyhedral $\tilde{\mathcal{Q}}$ by performing a sequence of projections individually onto each convex set defined by $\hat{\tilde{A}}_k^\intercal\hat{\tilde{w}} \leq 0$.

The projection operator onto the k_{th} convex set is given by:

$$\mathcal{P}_k = vec(\tilde{W}^{\intercal(i+1)}) - \frac{vec(\tilde{W}^{\intercal(i+1)}) * \hat{n_k}}{\|n_k\|^2} * \hat{n}_k \tag{5.29}$$

where n_k denotes the normal vector to the hyperplane defined by $\hat{\tilde{A}}_k^\intercal\hat{\tilde{w}} = 0$. If we denote

$\mathcal{T}$ as the product $\mathcal{T} = \mathcal{P}_1\mathcal{P}_2...\mathcal{P}_k$ then we can write:

$$vec(\tilde{W}^{\intercal(i)}_{proj}) = \mathcal{P}_{\tilde{\mathcal{Q}}}(vec(\tilde{W}^{\intercal(i)})) = \mathcal{T}^r(\tilde{W}^{\intercal(i)}) \; as \; r \to \infty \tag{5.30}$$

In Alg. (5.4), the complete Projected Gradient Descent algorithm for the optimization of the FATAS controller's parameters is presented.

ALGORITHM 5.4: *On-Policy Projected Gradient Descent*

Require: Initial parameters $vec(\tilde{W}^{\intercal(0)})$ (safe), $i = 0$
Require: Number of steps $k > 0$ without recalculating $\nabla V \; \forall p_d$
while vec$(\tilde{W}^{\intercal})$ has not converged **do**
1) Calculate $\nabla V^{(i)} \; \forall p_d$ using Mini-Batch Gradient Descent
2) Compute descent direction $step^{(i)}_{\hat{\tilde{w}}} = \nabla_{\tilde{W}^{\intercal}} M(\tilde{W}^{\intercal})$
3) Calculate learning rate $a^{(i)}$
4) Update parameters $vec(\tilde{W}^{\intercal(i+1)}) = vec(\tilde{W}^{\intercal(i)}) - a^{(i)} step^{(i)}_{\hat{\tilde{w}}}$
5) Projection onto safe set: $vec(\tilde{W}^{\intercal(i+1)}_{proj}) = \mathcal{P}_{\tilde{\mathcal{Q}}}(\tilde{W}^{\intercal(i+1)}) : \; \tilde{\mathcal{Q}} = \{\hat{\tilde{w}} \in \mathbb{R}^{(K_c+1)(K+1)} : \tilde{A}^{\intercal}\hat{\tilde{w}} \leq 0\}$
6) $i \leftarrow i + 1$
if $mod(i, k)! = 0$ **then**
$\nabla V^{(i+1)} = \nabla V^{(i)} \; \forall p_d$ and skip 1)
end if
end while

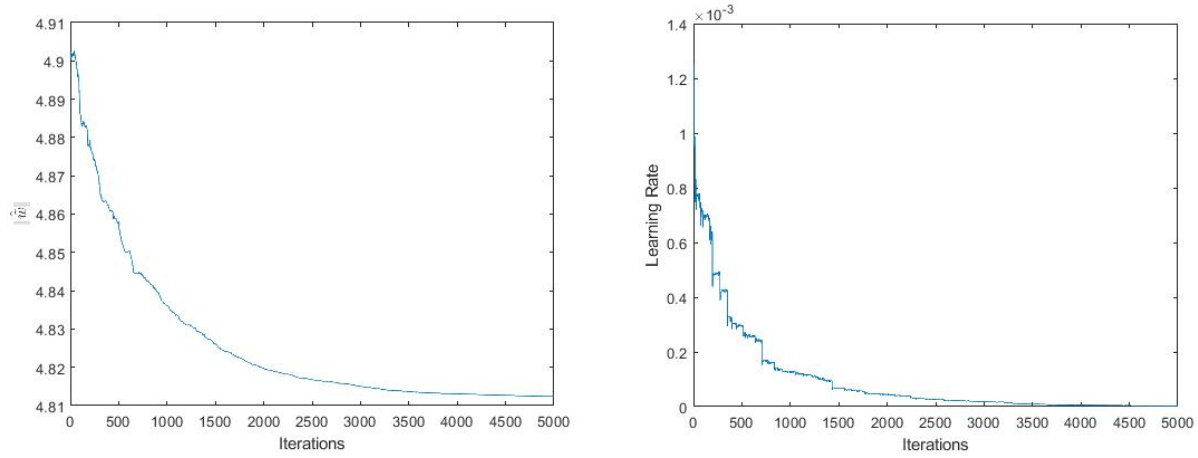
5.5 Simulation Results

In this section we will present the results of our implemented Projected Gradient Descent algorithm, by comparing the value of the adopted cost function as well as the policies produced by our method, prior to and after optimizing the parameters of the proposed policy. The parameters in our adopted cost function were chosen as $\alpha = 0.5$, $\beta = 0.5$ for all simulations. In addition, the successive number of steps without recalculating the gradient of the cost function in the PGD algorithm is set as $k = 5$.

In Fig.(5.2) the convergence of our controller's parameters, along with the adaptive learning rate a, are presented for the case of the FATAS controller of the T-shaped workspace described in Fig.(4.1) of chapter 4. Furthermore, in Fig.(5.4) a comparison between the initial and the final, normalized vector fields for the goal position $p_d = (0.8, 3.5)$ is depicted, along with the cost function's improvement both as a difference and as a percentage of decrease. The inherent simplicity of this specific workspace leaves not much room for improvement. Also it should be noted, that 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. In Fig.(5.5) through (5.7), the same comparison is done for three more goal positions of the T-shaped workspace.

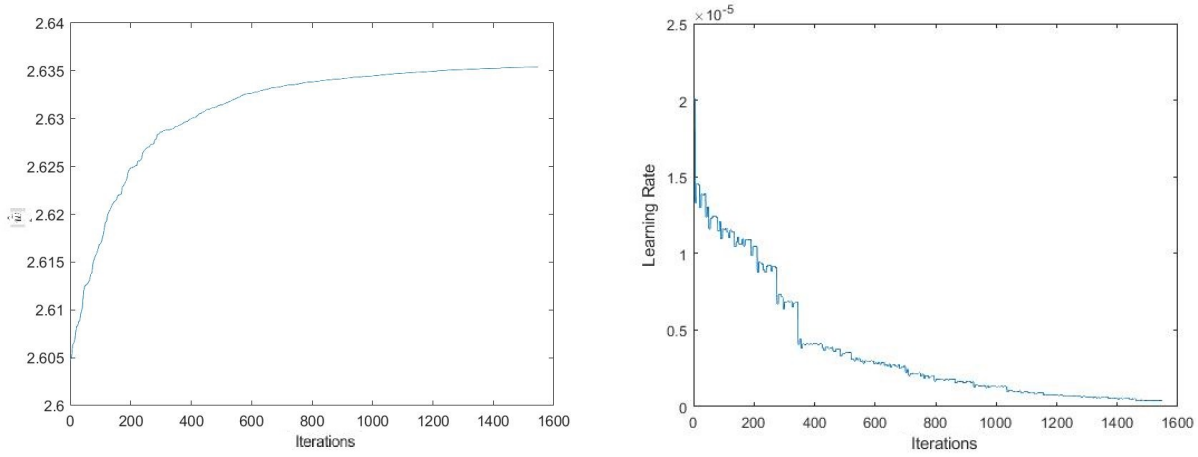
Additionally, similar results are presented regarding the FATAS controller designed for the more complex office-like workspace depicted in Fig.(4.2) of chapter 4, in Fig.(5.3)

and Fig.(5.8) through (5.11). The percentage of decrease clearly doesn't give enough insight, due to its ill-behavior for initial values close to zero. For example, in Fig.(5.11d), a percentage decrease of -5.66% is attributed to an increase of the initial cost from 4.18 to 4.42. To gain a better perspective, one has to also take the difference between the initial and final values of the cost function in Fig.(5.11c) into consideration as well.



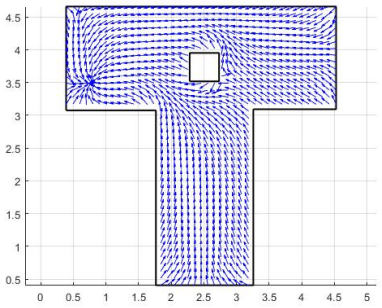
(a) *Convergence of* $\|\hat{\tilde{w}}\|$ (b) *Adaptive learning rate*

Figure 5.2: *Convergence of controller's parameters and learning rate a (T-Shaped Workspace)*

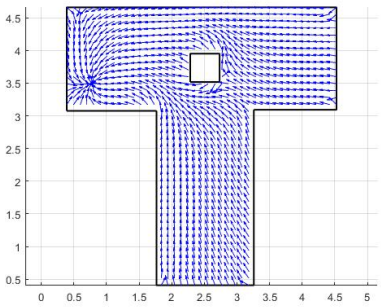


(a) *Convergence of* $\|\hat{\tilde{w}}\|$ (b) *Adaptive learning rate*

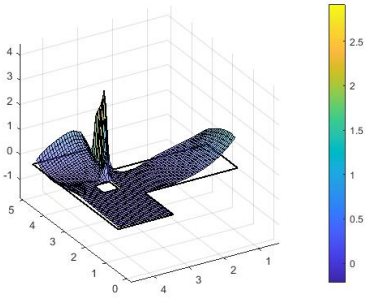
Figure 5.3: *Convergence of controller's parameters and learning rate a (Office-Like Workspace)*



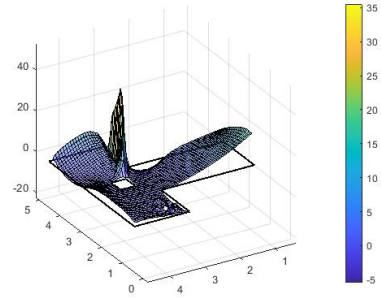
(a) *Initial Vector Field*



(b) *Final Vector Field*

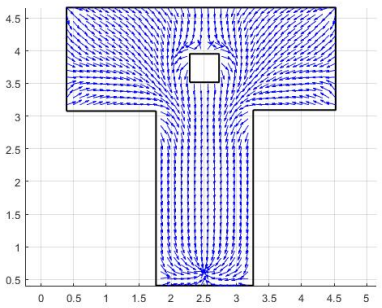


(c) *Cost Improvement*

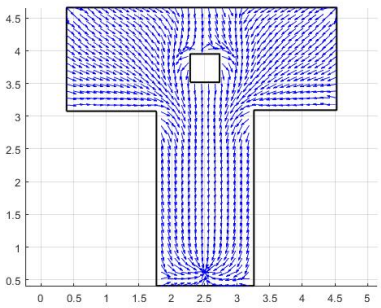


(d) *Cost Improvement (% Percentage)*

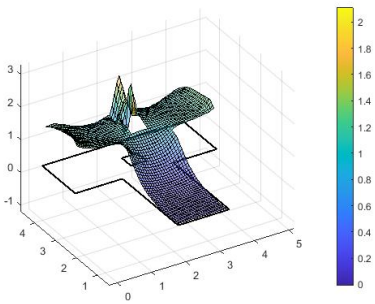
Figure 5.4: *Initial Vs Final Vector Field for (0.8,3.5)*



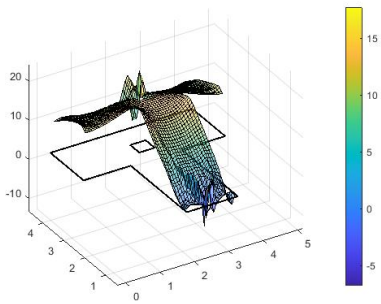
(a) *Initial Vector Field*



(b) *Final Vector Field*

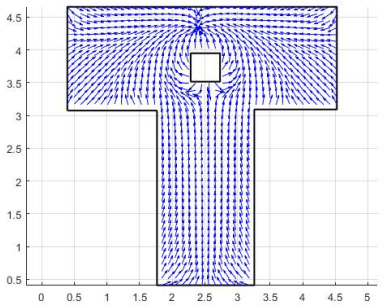


(c) *Cost Improvement*

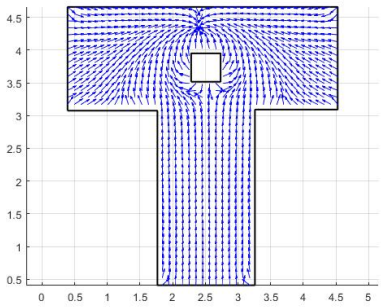


(d) *Cost Improvement (% Percentage)*

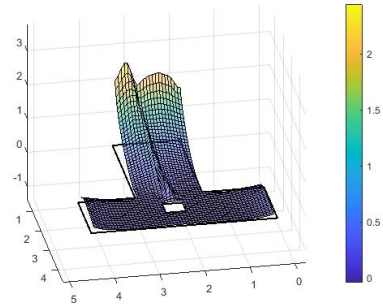
Figure 5.5: *Initial Vs Final Vector Field for (2.5,0.65)*



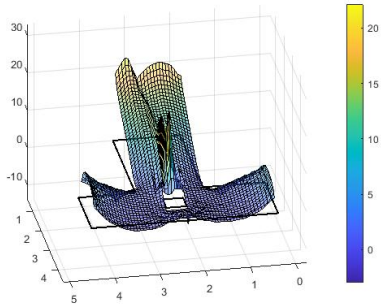
(a) *Initial Vector Field*



(b) *Final Vector Field*

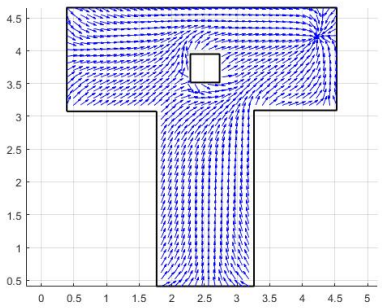


(c) *Cost Improvement*

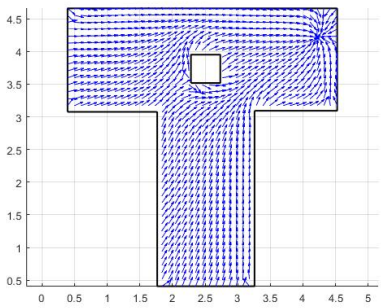


(d) *Cost Improvement (% Percentage)*

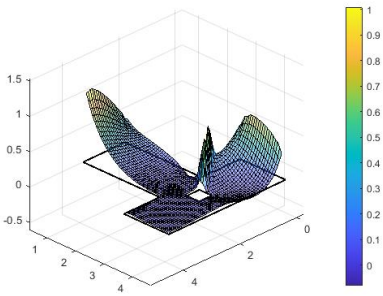
Figure 5.6: *Initial Vs Final Vector Field for (2.4,4.3)*



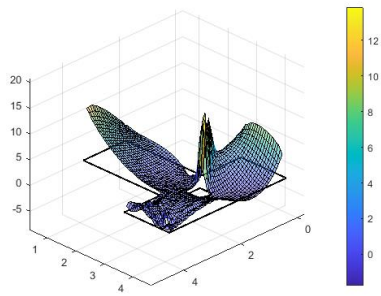
(a) *Initial Vector Field*



(b) *Final Vector Field*

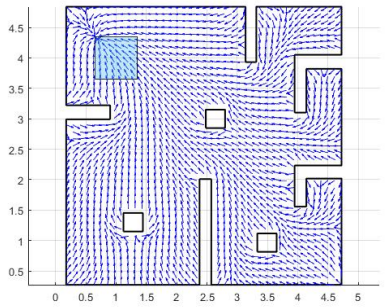


(c) *Cost Improvement*

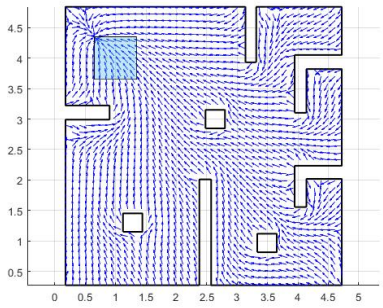


(d) *Cost Improvement (% Percentage)*

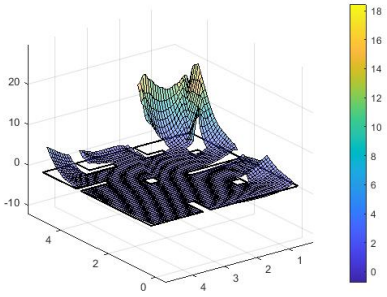
Figure 5.7: *Initial Vs Final Vector Field for (4.2,4.2)*



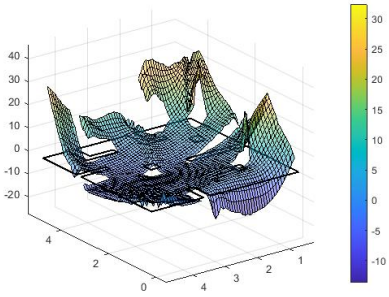
(a) *Initial Vector Field*



(b) *Final Vector Field*

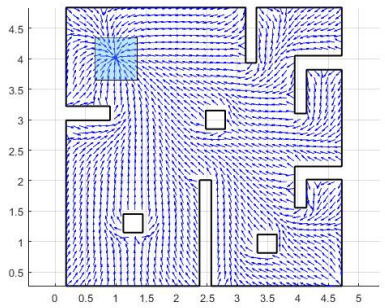


(c) *Cost Improvement*

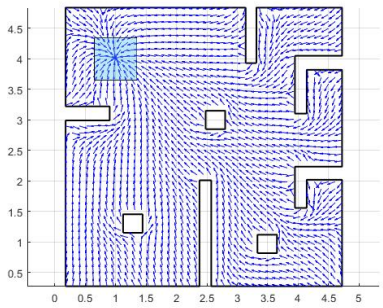


(d) *Cost Improvement (% Percentage)*

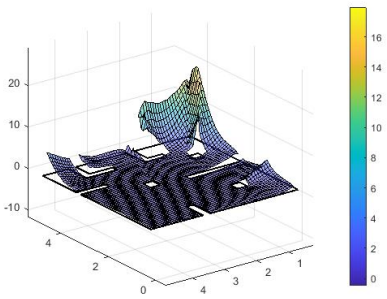
Figure 5.8: *Initial Vs Final Vector Field for (0.7,4.3)*



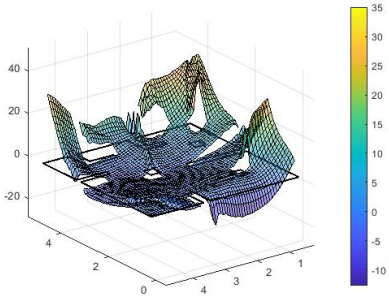
(a) *Initial Vector Field*



(b) *Final Vector Field*

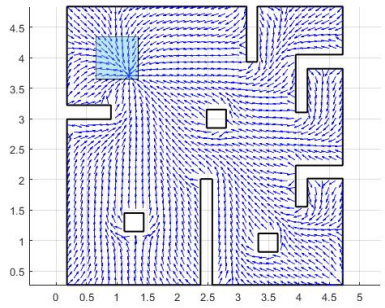


(c) *Cost Improvement*

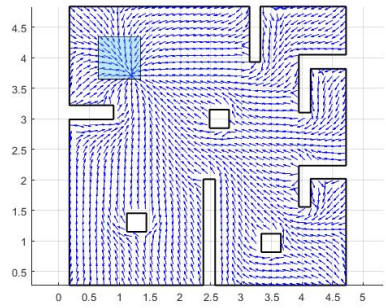


(d) *Cost Improvement (% Percentage)*

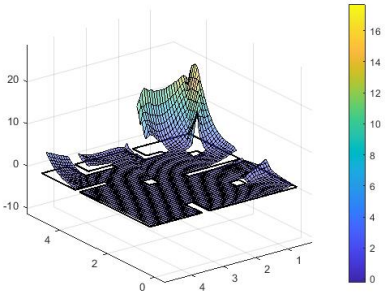
Figure 5.9: *Initial Vs Final Vector Field for (1,4)*



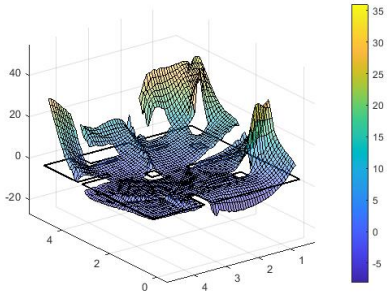
(a) *Initial Vector Field*



(b) *Final Vector Field*

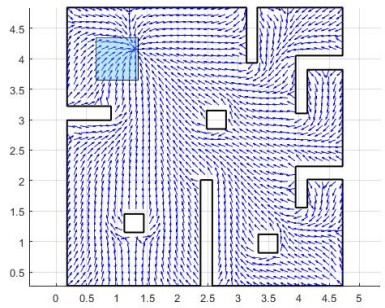


(c) *Cost Improvement*

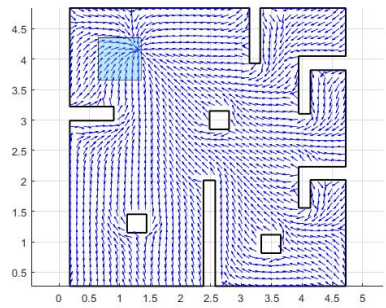


(d) *Cost Improvement (% Percentage)*

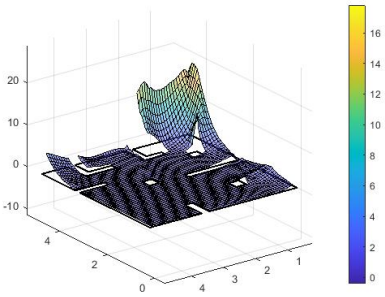
Figure 5.10: *Initial Vs Final Vector Field for (1.2,3.7)*



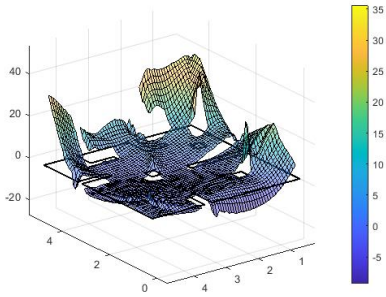
(a) *Initial Vector Field*



(b) *Final Vector Field*



(c) *Cost Improvement*



(d) *Cost Improvement (% Percentage)*

Figure 5.11: *Initial Vs Final Vector Field for (1.3,4.15)*

Chapter 6

Multiple Waypoint Navigation using the FATAS Controller

6.1 Motivation and Proposed Method

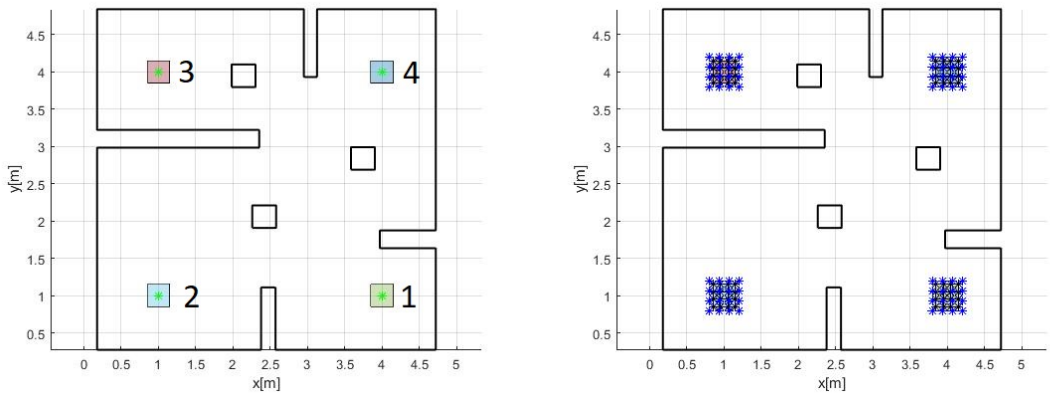
In this chapter, we propose a robust solution to the navigation problem of a robot, whose objective is to visit a pre-determined set of waypoints within a completely known and static workspace, with the use of our novel parametric controller. Our work is motivated by a plethora of real-life scenarios such as a robot visiting different rooms of a warehouse to execute tasks, or performing monitoring duties in certain areas.

When considering the aforementioned scenarios, a solution includes two steps, namely, the waypoint-to-waypoint navigation and the optimal visiting sequence. After we assign a safe navigation policy to each of the robot's waypoints, first, we employ Reinforcement Learning to turn those policies into optimal ones and in the next step, we find the optimal order of visitation by solving an Asymmetric Travelling Salesman Problem (ATSP) [18] so that the sum of all transition costs is minimized.

It is true that one can consider shrinking the waypoints down to individual points and use the FATAP, instead of the FATAS parametric controller. Such an approach could be adopted, for example, in a monitoring task. However, in that case, even a slight change in the robot's input goal position will render this approach unusable. Thus, by employing the FATAS parametric controller along with the policy gradient optimization technique we presented, not only we achieve safety and optimality, but also robustness w.r.t the goal positions assigned to the robot.

6.2 Simulation Results

We consider the artificial workspace illustrated in Fig.(6.1). Our aim is to provide an optimal solution to the navigation problem of a robot, whose goal throughout its work-life is to visit the four depicted, green, star-shaped waypoints. Also, for convenience, a unique number is assigned to each waypoint. To implement the FATAS controller for the aforementioned problem, a square area is chosen around each waypoint. Then, for each one, an initial policy is obtained with the steps we described in chapter 4. The centers of the Radial Basis Functions p_{Cd} (blue points) and the goal positions $p_{di} \in p_G$ (black points)



(a) *Waypoint Numbering* (b) *Placement of P_{Cd} and p_{di}*

Figure 6.1: *Artifical Workspace with Four Waypoints*

used for the safety condition, are depicted in Fig.(6.1b). After obtaining our initial policies, the Projected Gradient Descent algorithm we discussed on chapter 5, was implemented to convert those policies into optimal ones.

In Table 6.1, the initial transition cost matrix between each waypoint is summarized. Furthermore, the final transition costs are shown in Table 6.2. Supposing one would pick an initial route for the robot consisting of the cyclic permutation of waypoints (2 3 1 4), according to Table 6.3, the sum of the initial transition costs amounts to 214.6. After employing Reinforcement Learning to optimize our policies, the sum of the final transition costs becomes 153.8. 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 53.34, leading to a 75.14% decrease of the total cost. In Fig.(6.2), the initial and final trajectories of the robot are shown. While Fig.(6.2a) depicts the trajectories of the robot in the initial cycle (2 3 1 4), in Fig.(6.2b) the trajectories of the final cycle (4 3 1 2) are shown.

Start ╲ End	1) $(4,1)$	2) $(1,1)$	3) $(1,4)$	4) $(4,4)$
1) $(4,1)$	-	7.25	100.68	7.65
2) $(1,1)$	7.94	-	153.51	19.28
3) $(1,4)$	31.31	40.20	-	9.71
4) $(4,4)$	9.25	22.13	22.73	-

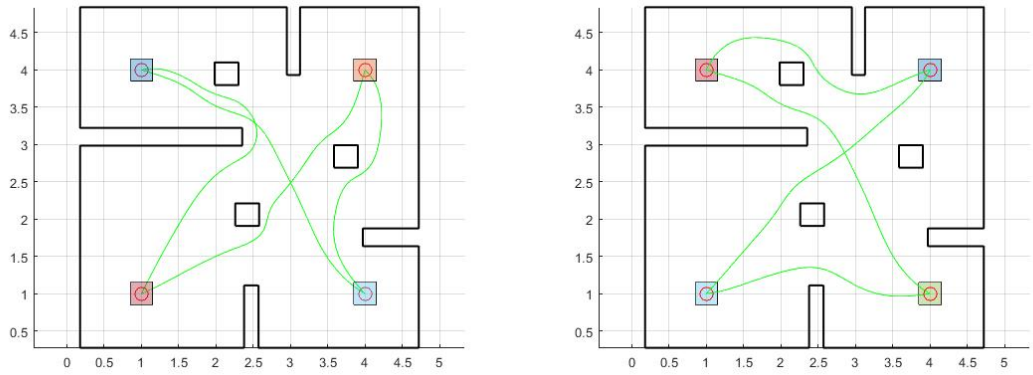
Table 6.1: *Initial Transition Costs*

Start ╲ End	1) $(4,1)$	2) $(1,1)$	3) $(1,4)$	4) $(4,4)$
1) $(4,1)$	-	6.22	68.24	6.16
2) $(1,1)$	5.76	-	120.50	10.43
3) $(1,4)$	14.77	25.07	-	6.30
4) $(4,4)$	6.50	12.37	21.94	-

Table 6.2: *Final Transition Costs*

Cycle ╲ Sum	Initial Transition Costs	Final Transition Costs	Percentage of Decrease
(4 3 1 2)	80.57	53.34	33.80%
(4 3 2 1)	78.52	58.93	24.95%
(3 4 2 1)	140.47	92.67	34.03%
(1 3 2 4)	169.41	110.23	34.93%
(4 1 2 3)	179.72	139.52	22.37%
(2 3 1 4)	214.60	153.80	28.33%

Table 6.3: *Sum of Transition Costs for each Cycle*



(a) *Initial Cycle* (b) *Final Cycle*

Figure 6.2: *Comparison of trajectories between Initial and Final Cycle*

To evaluate the robustness of our proposed method, first, let us consider the case of a small disturbance when the positions of the four waypoints are passed to the robot. The initial and final transition cost matrices for our controller are summarized in Table 6.4 and Table 6.5 respectively. In addition, the comparison between the initial and the final transition costs is depicted in Table 6.6. We can see that despite the change in the goal positions of the robot, we managed to obtain similar results, maintaining a 77.39% decrease in the total cost between the initial cycle with the highest sum of transition costs and the final cycle that consists of the overall optimal solution. In Fig.(6.3) a comparison between the trajectories of the initial and the final cycle is shown.

Start ╲ End	1) $(3.99, 0.99)$	2) $(1.02, 1.03)$	3) $(0.98, 4.05)$	4) $(4.01, 3.97)$
1) $(3.99, 0.99)$	-	7.14	111.94	7.56
2) $(1.02, 1.03)$	7.66	-	177.92	18.66
3) $(0.98, 4.05)$	32.72	43.90	-	10.10
4) $(4.01, 3.97)$	8.97	21.99	23.54	-

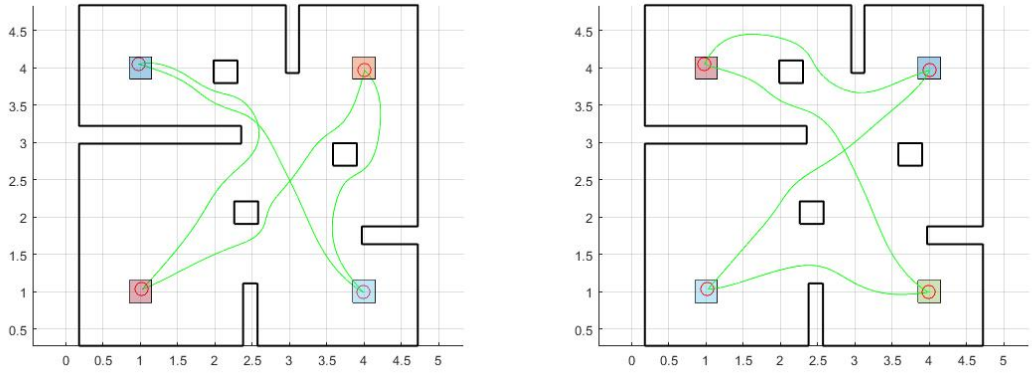
Table 6.4: *Initial Transition Costs (Small Disturbance)*

Start ╲ End	1) $(3.99, 0.99)$	2) $(1.02, 1.03)$	3) $(0.98, 4.05)$	4) $(4.01, 3.97)$
1) $(3.99, 0.99)$	-	6.04	73.50	6.07
2) $(1.02, 1.03)$	5.63	-	134.32	10.16
3) $(0.98, 4.05)$	15.29	26.06	-	6.46
4) $(4.01, 3.97)$	6.36	12.11	22.82	-

Table 6.5: *Final Transition Costs (Small Disturbance)*

Cycle ╲ Sum	Initial Transition Costs	Final Transition Costs	Percentage of Decrease
(4 3 1 2)	82.06	54.31	33.82%
(4 3 2 1)	82.66	60.59	26.70%
(3 4 2 1)	151.69	97.70	35.59%
(1 3 2 4)	183.46	116.08	36.73%
(4 1 2 3)	204.12	153.17	24.96%
(2 3 1 4)	240.19	167.79	30.14%

Table 6.6: *Sum of Transition Costs for each Cycle (Small Disturbance)*



(a) *Initial Cycle* (b) *Final Cycle*

Figure 6.3: *Comparison of trajectories between Initial and Final Cycle (Small Disturbance)*

Finally, let us consider a large disturbance regarding the positions of the waypoints. The initial and final transition cost matrices for our method are summarized in Table 6.7 and Table 6.8 respectively. The comparison between the initial and the final transition costs for the case of the large disturbance is depicted in Table 6.9. The decrease of the overall cost between the two cycles of interest further increases to a total of 81.4%. Finally, in Fig.(6.4) we present a comparison between the trajectories of the initial and the final cycle for the case of the large 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	117.47	6.49
2) (0.86, 0.86)	11.18	-	283.69	21.55
3) (0.87, 4.14)	47.61	63.78	-	11.89
4) (4.12, 3.89)	9.37	27	29.62	-

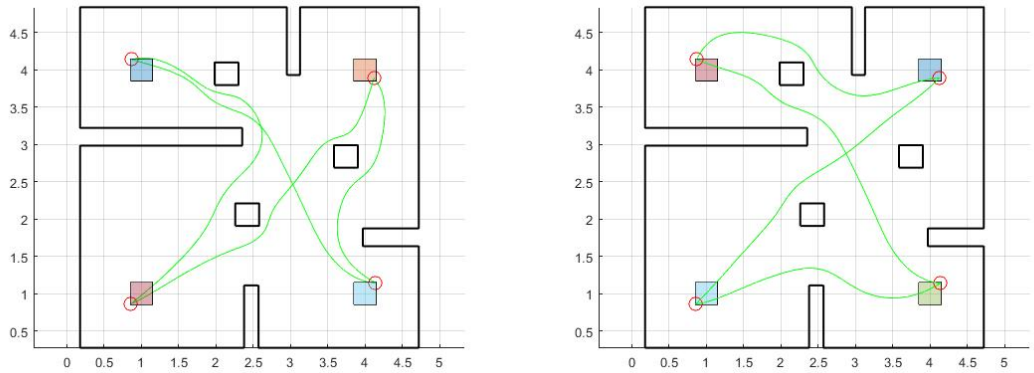
Table 6.7: *Initial Transition Costs (Large Disturbance)*

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)	-	8.72	85.95	5.46
2) (0.86, 0.86)	7.12	-	209.72	11.28
3) (0.87, 4.14)	17.83	49.93	-	7.40
4) (4.12, 3.89)	6.48	18.86	30.04	-

Table 6.8: *Final Transition Costs (Large Disturbance)*

Cycle ╲ Sum	Initial Transition Costs	Final Transition Costs	Percentage of Decrease
(4 3 1 2)	107.82	67.87	37.05%
(4 3 2 1)	111.07	92.56	16.67%
(3 4 2 1)	167.54	119.33	28.78%
(1 3 2 4)	212.17	153.64	27.59%
(4 1 2 3)	313.99	232.32	26.01%
(2 3 1 4)	364.80	251.87	30.96%

Table 6.9: *Sum of Transition Costs for each Cycle (Large Disturbance)*



(a) *Initial Cycle* (b) *Final Cycle*

Figure 6.4: *Comparison of trajectories between Initial and Final Cycle (Large Disturbance)*

Chapter 7

Conclusions and Future Work

While our method shows promising results, certain limitations need to be taken into consideration. Firstly, our novel 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 hard constrained quadratic problem to obtain our controller's parameters, becomes a hindrance in more complex environments. Despite the aforementioned issues, we believe that in future work, our proposed method has the potential to grow into a valuable addition to the arsenal of established motion planning approaches.

In forthcoming work, our method may be expanded for higher dimensional cases while incorporating noise and stochastic elements in the dynamics of the controller as well. Furthermore, regarding our solution to the motion planning problem, one can also consider the decay parameters of the Radial Basis Functions, as additional parameters subject to optimization. Ultimately, it is important to mention that the development of a provably optimal way to distribute the centers of the basis functions, as well as the goal positions in the safety condition of the FATAS controller's parameters, can be worked upon in order to avoid unnecessary computational expense.

Bibliography

[1] 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.

[2] 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.

[3] Z. Majdisova and V. Skala, "Radial basis function approximations: comparison and applications," *Applied Mathematical Modelling*, vol. 51, pp. 728–743, 11 2017. [Online]. Available: https://doi.org/10.1016%2Fj.apm.2017.07.033

[4] 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.

[5] 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.

[6] 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.

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

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

[9] 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. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0021904506000463

[10] J.-C. Latombe, *Roadmap Methods*. Boston, MA: Springer US, 1991, pp. 153–199. [Online]. Available: https://doi.org/10.1007/978-1-4615-4022-9_4

[11] ——, *Exact Cell Decomposition*. Boston, MA: Springer US, 1991, pp. 200–247. [Online]. Available: https://doi.org/10.1007/978-1-4615-4022-9_5

[12] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[13] Z. Kingston, M. Moll, and L. E. Kavraki, "Sampling-based methods for motion planning with constraints," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, no. 1, pp. 159–185, 2018. [Online]. Available: https://doi.org/10.1146/annurev-control-060117-105226

[14] 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. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0196885890090017S

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

[16] S. H. Chong, Edwin K. P.; Żak, *Gradient Methods.* Hoboken: Wiley, 2013, pp. 131–160.

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

[18] 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. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1570866707000172