UNIVERSITY OF CALIFORNIA SAN DIEGO

Singular Perturbation Method for Multiagent Coverage Control Using Time-Varying Density Functions

A Thesis submitted in partial satisfaction of the
requirements for the degree
Master of Science

in

Engineering Sciences (Aerospace Engineering)

by

Victor Gandarillas

Committee in charge:

Professor Jorge Cortés, Chair
Professor Thomas Bewley
Professor Sonia Martínez

2018

The Thesis of Victor Gandarillas is approved, and it is acceptable in quality and form for publication on microfilm:

_____

_____

_____
Chair

University of California San Diego

2018

DEDICATION

For my parents,
my friends,
and Emily.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

To my advisor, Prof. Jorge Cortés, for his guidance and mentorship. To Aaron Ma, for his help with familiarizing me with the lab and helping me out whenever I had questions. To the other members of our research group for their help and patience, especially Aamodh, Parth, and Vishal. To my friends Ricardo, Krishna, Janelle, Nima, and Sean for their help and support. To Emily for her patience, support, and all the small ways she helped me finish this thesis.

This thesis, in full, is currently being prepared for submission for publication of the material. Gandarillas, Victor; Cortés, Jorge. The thesis author is the primary investigator and author of this paper.

ABSTRACT OF THE THESIS

Singular Perturbation Method for Multiagent Coverage Control Using Time-Varying Density Functions

by

Victor Gandarillas
Master of Science in Engineering Sciences (Aerospace Engineering)
University of California San Diego, 2018
Professor Jorge Cortés, Chair

This paper presents a continuous-time coverage algorithm based on the singular perturbations method. The algorithm generates a control for a network of autonomous agents. The algorithm is distributed in the sense that the agents need only information about their neighbors. The singular perturbation method is applied as a means to update the control at a faster time scale to approach performance of a centralized approach. The paper concludes with results from simulations and experiment.

Chapter 1

Introduction

We consider the problem of influencing teams of agents by means of time-varying-density functions. The agents are tasked with reaching a target whose position is unknown. A probability density function models the target's location or some area of interest by the user. This allows the robots to use a notion of coverage based on locational optimization. This has practical applications in areas such as search and rescue operations. Additionally, the probability density function is time-varying. By maximizing coverage over the plane, the robots maximize the likelihood of any one robot finding the target. The time-varying density may also represent a human generated input, where the user commands the agents to focus some area of interest. As the density function changes over time, each agent requires information from all other agents. This presents a challenge for large decentralized systems. As a result, numerical methods become computationally expensive for large teams of agents.

1.1 Background

Gradient descent algorithms exist for agents to arrive at locally optimal solutions for time-invariant densities [1], [2], [3]. While the convergence of the Lloyd algorithm is well studied for time-invariant densities, the same convergence properties do not hold for time-varying density functions. The algorithm presented in [2] also addresses time-varying densities, but requires certain assumptions; namely,

that the cost function be an explicit function of the time and not agent positions. When considering the dependence of the locational cost as a function of position as well as time, the time varying nature of the density function presents a challenge for large decentralized systems in that each agent requires knowledge of the state of all other agents in the system. To get around the issue of increased complexity from a centralized algorithm, [4] and [5] propose distributed approximations through a von Neumann series. Additional terms improve the approximation at the cost of more computation and more communication between agents that are not neighbors.

1.2 Statement of Contributions

Our main contribution is the design of a distributed algorithm suitable for time-varying densities that does away with the types of approximations used in [4],[5]. The technical approach proposed in this paper relies on the singular perturbation method to control the system in a distributed manner, allowing for performance approaching that of the centralized solution and reduced computational cost when compared to distributed approximations of [4],[5]. The velocity of each agent is computed with limited knowledge of other agents; i.e. each agent only requires information about adjacent neighbors. The desired velocity is not modeled directly; rather, the time rate of change of the input is modeled as a boundary layer system. In other words, the desired velocity input has its own dynamics operating on a smaller time scale resulting in asymptotic convergence to the optimal control. Finally, we describe in some detail an algorithm based on modeling the system as a singular perturbation, avoiding such numerical difficulties encountered in [4],[5] and present results from simulation and experiment.

1.3 Organization

In Chapter 2, we introduce the notation used throughout the paper and review the singular perturbation method, which forms the basis of our main re-

sult. Chapter 3 defines the coverage control problem formally. Chapter 5 reviews previous approaches to solving the problem of coverage control of multiple agents with time-varying densities and introduces the singular perturbation method as a novel approach to maximizing coverage in a distributed way. Chapter 6 presents results from simulation for a large team of holonomic agents and a comparison of performance between algorithms. Chapter 7 presents results from experiment and a comparison of performance between algorithms for a small team of agents with unicycle dynamics. Chapter 8 provides concluding remarks.

Chapter 2

Notation and Preliminaries

2.1 Notation

In this section, we describe the notation used throughout the paper. We denote the set of reals by $\mathbb{R}$. We define the Euclidean 2-norm over the 2-dimensional vector space $\mathbb{R}^2$ by $\|\cdot\|_2^2$. We consider a multiagent system composed by n agents (or sensors) evolving on a plane $Q \subset \mathbb{R}^2$. The plane Q is partitioned into n disjoint cells, denoted P_i, where $\cup_i^n P_i = Q$. We let $p_i \in P_i$ denote the position of agent $i \in \{1, \ldots, n\}$. For convenience, the vector $p \in \mathbb{R}^{2n}$ contains the positions of all agents. The vector $u \in \mathbb{R}^{2n}$ contains the control inputs of each agent $u_i \in \mathbb{R}^2$. A time-varying density function $\phi : \mathbb{R}^2 \times \mathbb{R} \to (0, 1]$ provides a weighting on the plane to be covered, where $q \in Q$ is an arbitrary point in the plane.

We also make use of the Voronoi tessellation $V = (V_1, \ldots, V_n)$ to partition the plane Q. the points $(p_1, \ldots, p_n)$ generate the Voronoi tesselation V and a Voronoi cell $V_i(p)$ is defined as:

$$V_i(p) = \{q \in Q | \|q - p_i\| \leq \|q - p_j\|, \forall j \neq i\} \tag{2.1}$$

where p_i generates the Voronoi cell $V_i(p)$ for the agent i and $\cup_i^n V_i = Q$. For a comprehensive treatments on Voronoi diagrams, see [6],[7].

Since Q is a convex polygon, the boundary of each $V_i(p)$ is a convex polygon. When the two Voronoi regions $V_i(p)$ and $V_j(p)$ are adjacent, p_i is called a (Voronoi) neighbor of p_j (and vice versa).

2.2 Singular Perturbation

We now review the singular perturbation method and present a decentralized algorithm for obtaining optimal coverage with time-varying density functions. Perturbation methods generally obtain solutions of equations through a power series. This leads to problems in which a small parameter multiplies a derivative. Singularly perturbed problems are generally characterized by dynamics operating on multiple scales in which a small parameter cannot be approximated by setting the parameter value to zero. Consider the system of differential equations

$$\dot{p} = f(p, c, u, t), \qquad \varepsilon \dot{u} = g(p, c, u, t) \tag{2.2}$$

where $0 < \varepsilon \ll 1$ is a small perturbation, $f(p, c, u, t)$ represents the dynamics of the positions p of the agents and $g(p, c, u, t)$ represents the dynamics of the control input u. Taking the limit as $\varepsilon \to 0^+$ produces the reduced model

$$\dot{p} = f(p, c, u, t) \tag{2.3a}$$

$$0 = g(p, c, u, t) \tag{2.3b}$$

This neglects the highest order derivative in the system and is not a good approximation of the full system. Analyzing the system on a stretched time scale, given by $\eta = t/\varepsilon$ as $\varepsilon \to 0^+$, the dynamics are given by the boundary layer model

$$dp/d\eta = 0 \tag{2.4a}$$

$$du/d\eta = g(p, c, u, t) \tag{2.4b}$$

where (2.4b) represents the fast dynamics of the system. This only models the behavior of the highest order derivative on the stretched time scale and neglects the behavior of the lower order derivatives. This is summarized in Theorem 1 [8].

Theorem 1 (Tikhonov's theorem)**.** *If $u = h(p, c, t)$ is a stable root of the slow manifold (2.3b), then the solution of the system (2.2) approaches the solution of the reduced model (2.3) as $\varepsilon \to 0^+$. Note that there may be more than one such function h.*

Proof. The proof is provided in [8]. □

Theorem 1 forms the basis for the main result of the paper in Theorem 3

Chapter 3

Problem Definition

We seek to solve the coverage problem for time-varying densities in a distributed manner. The goal is to drive each agent within a team of agents to an optimal location that maximizes coverage over a plane Q based on a probability density function $\phi(q,t)$ which is time-varying. The time-varying density function can be thought of as the distribution of possibilities for a target's location, for example in a search-and-rescue operation, or as a human generated input where the human operator designates an area of interest where the robots should gather. The time-varying density function poses a challenge because the problem is no longer a matter of pure feedback, but involves the minimization of a locational cost that is also dependent on time. Our objective is to design a distributed control algorithm that asymptotically converges to a local optimum. The control algorithm is distributed in the sense that each agent has limited knowledge of the other agents in the system. Here we define the problem for holonomic agents. In Chapter7 we present simulation results for a team of holonomic agents as well as experimental results for agents with unicycle dynamics. The coverage control problem involves the minimization of the locational cost

$$H(p,P,t) = \sum_{i=1}^{n} \int_{P_i} \phi(q,t) \|q - p_i\|_2^2 dq \tag{3.1}$$

where $P = (P_1, \ldots, P_n)$ is the partition of the plane Q and $P_i \subset Q$ is the cell containing agent i.

At a given time t, when a configuration of agents p together with the parti-

tion P minimize (3.1), the domain is said to be optimally covered with respect to ϕ. At fixed sensors location, the optimal partition of Q is the Voronoi tessellation $V(P) = (V_1, \ldots, V_n)$ given by (2.1). With this choice of region, we remove the partition as a decision variable and the locational cost becomes

$$H(p, V, t) = \sum_{i=1}^{n} \int_{V_i} \phi(q, t) \|q - p_i\|_2^2 dq \tag{3.2}$$

In [9], it was shown that

$$\frac{\partial H}{\partial p_i} = \int_{V_i(p_i)} -2(q - p_i)^T \phi(q, t) dq \tag{3.3}$$

and since $\phi(q, t) > 0, \forall q \in Q, t \in [0, \infty)$, one can define the mass m_i and center of mass c_i of the i^{th} Voronoi cell $V_i(p)$ as

$$m_i = \int_{V_i(p)} \phi(q, t) dq, \qquad c_i = \frac{1}{m_i} \int_{V_i(p)} q\phi(q, t) dq \tag{3.4}$$

The partial derivative (3.3) can be rewritten as

$$\frac{\partial H}{\partial p_i} = -2m_i(p_i - c_i)^T \tag{3.5}$$

From this expression, we can see that a critical point of (3.5) is

$$p_i(t) = c_i(p, t), i \in \{1, 2, \ldots, n\} \tag{3.6}$$

and a minimizer to (3.3) is necessarily of this form [3]. In this state, p is a so-called centroidal Voronoi tessellation (CVT).

A critical point of the form (3.6) implies

$$\dot{p} = \dot{c} \tag{3.7}$$

This is a necessary condition for convergence to a moving CVT. In this paper, the interest is in designing algorithms that guarantee convergence to local minima with respect to time-varying density functions. No claims are made about finding the global minimum.

For time-invariant density functions, the continuous-time version of Lloyd's algorithm is a suitable control, given by

$$\dot{p}_i = -\kappa(p_i - c_i)$$

where $\kappa > 0$ is a proportional gain, which drives each agent's position to the newly generated Voronoi cell center, achieving a CVT. Note that this is not necessarily a global minimum [2], [1].

Chapter 4

Time-Varying Solution

To capture the time varying nature of the system dynamics, let $e = (p - c)$ represent position error between the agents and their centroids, and $\dot{e} = -\kappa e$ represent the rate of change in the error so that the position error is converging to zero exponentially. Then

$$\dot{p} - \dot{c} = -\kappa(p - c)$$

which represents the difference between the feedback control of Lloyd and the velocity error of the agents. Let $c(p, t) : p, t \mapsto c$ represent the centroids as a function of the agent positions p and the time t. Then

$$\left(I - \tfrac{\partial c}{\partial p}\right)\dot{p} = -\kappa(p - c) + \frac{\partial c}{\partial t} \tag{4.1}$$

Note that this is in the form of a linear system $Ax = b$. The matrix $\partial c/\partial p$ encodes the adjacency of the agents and has a sparse structure. Taking the inverse of $(I - \partial c/\partial p)$ obtains $\dot{p}$ directly, resulting in the following algorithm.

TVD-C:

$$\dot{p} = \left(I - \tfrac{\partial c}{\partial p}\right)^{-1}\left(-\kappa(p - c) + \tfrac{\partial c}{\partial t}\right) \tag{4.2}$$

As in [4], we denote this algorithm TVD-C, where TVD stands for time-varying densities, and C stands for centralized. In [10], it was shown that in the time invariant case, the inverse is well defined as long as $\phi(p)$ is a log concave function of p. Moreover, ϕ must be continuously differentiable in both arguments, and these two conditions are enough to ensure that the inverse exists.

Theorem 2. *The algorithm TVD-C converges exponentially.*

Proof. Let $V(e) = \sum \frac{1}{2}\|e\|_2^2 = \sum \frac{1}{2}\|p-c\|_2^2 > 0, \forall e = p - c \neq 0$ be a Lyapunov function. Then $\dot{V}(e) = -\sum \frac{e^2}{\kappa} = -\sum \frac{(p-c)^2}{\kappa} < 0, \forall e = p - c \neq 0$. Then the error $e = p - c$ converges exponentially. □

Figure 4.1 demonstrates Theorem 2 from simulation data with 50 holonomic agents using time-varying density functions (6.1).

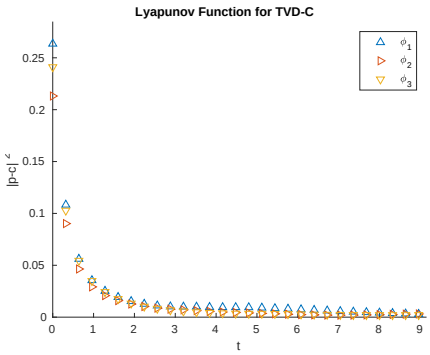

Figure 4.1: Lyapunov function of 50 holonomic agents using TVD-C with various density functions. Note that the Lyapunov function approaches zero exponentially.

Even though the matrix $I - \partial c/\partial p$ is sparse, (4.2) cannot be implemented in a distributed way because the inverse of a sparse matrix is not in general sparse. This requires each agent to have knowledge of all the other agents in the system. To get around this issue, a von Neumann series can approximate the inverse.

TVD-D$_k$:

$$\dot{p} = \sum_{l=0}^{k} \left(\frac{\partial c}{\partial p}\right)^l \left(-\kappa(p-c) + \frac{\partial c}{\partial t}\right) \tag{4.3}$$

As in [4], we denote this algorithm TVD-D$_k$, where D$_k$ stands for decentralized with k-hop adjacency information. It was shown for some time-varying density functions in [4] that TVD-D$_k$ gives similar performance to TVD-C for $k > 3$. While TVD-D$_k$ successfully leverages the sparsity structure of $\partial c/\partial p$, additional matrix multiplications are required to achieve performance approaching that of TVD-C. Furthermore, the matrix $(I - \partial c/\partial p)^{-1}$ can only be approximated

by a series expansion if the eigenvalues of $\partial c/\partial p$ lie inside the unit circle. While each agent does not require knowledge of all the other agents, increasingly accurate approximations require communication with more and more neighbors. It is also possible that the density functions change at different rates.

Chapter 5

Singular Perturbation Method

Our goal is to minimize the difference between both sides of (4.1) without the need to compute the inverse in (4.2) and without the need for each agent to communicate with all other agents in the system. This gives rise to the idea of having a dedicated algorithm that asymptotically computes the solution u. Because both the left and right hand side of (4.1) actually depend on p, we resort to the singular perturbation approach to perform this algorithm at a faster time scale than the timescale for the agents' motion. This approach is sound, in that it uses the fact that communication between the agents is much faster than the motion of the agents, and also only requires information exchange with adjacent neighbors. As each agent receives information from its neighbors, the control input updates. The control input updates faster than the motion because it updates on the communication time scale. This allows for information to propagate throughout the network without the need for each agent to obtain any knowledge aside from that of its neighbors. This leads to our main result.

Theorem 3. *Let $u = \dot{p}$. Then the control input*

TVD-SP$_\varepsilon$:

$$\dot{u} = -\frac{1}{\varepsilon}\left(I - \frac{\partial c}{\partial p}\right)^T \left(\left(I - \frac{\partial c}{\partial p}\right) u - \frac{\partial c}{\partial t} + \kappa(p - c)\right) \tag{5.1}$$

approaches the control input given by TVD-C (4.2) as $\varepsilon \to 0^+$. For $0 < \varepsilon \ll 1$, $p - c$ converges exponentially to zero.

Proof. We seek to construct the fast dynamics $\varepsilon\dot{u} = g(p, c, u, t)$ of the system defining $\dot{u}$ on a smaller/faster time scale such that a local minimum of (3.2) is achieved. Minimizing the difference of both sides of (4.1) achieves such a minimum. We use this difference to define a scalar, nonnegative, convex function $f(u)$

$$f(u) = \frac{1}{2}\left\| \left(I - \tfrac{\partial c}{\partial p}\right) u - \frac{\partial c}{\partial t} + \kappa(p - c) \right\|^2 \tag{5.2}$$

where $u = \dot{p}$. Since $f(u)$ is convex in u, there exists a unique u that minimizes (5.2). Taking the gradient of $f(u)$, we obtain

$$\nabla f(u) = \left(I - \tfrac{\partial c}{\partial p}\right)^T \left(\left(I - \tfrac{\partial c}{\partial p}\right) u - \tfrac{\partial c}{\partial t} + \kappa(p - c) \right) \tag{5.3}$$

Because $\dot{u}$ is the gradient of a quadratic function of u, the control dynamics are exponentially stable, restoring the system dynamics to the reduced model (2.3a). That is, $\dot{u} = -\nabla f(u)$ defines for u a function $h(p, c, t)$ that is a stable root of (2.3b). This in turn produces the stretched time scale dynamics

$$dp/d\eta = 0 \tag{5.4a}$$

$$du/d\eta = -\left(I - \tfrac{\partial c}{\partial p}\right)^T \left(\left(I - \tfrac{\partial c}{\partial p}\right) u - \tfrac{\partial c}{\partial t} + \kappa(p - c) \right) \tag{5.4b}$$

where $\eta = t/\varepsilon$ represents a stretched time scale as $\varepsilon \to 0^+$, (5.4a) is "frozen" in time and (5.4b) is the aforementioned boundary layer model. Since (5.4b) is linear in u, $u = h(t, p)$ must be a *unique* root of (5.4b). The control $u = h(t, p)$ is also a stable root of (5.4b). Replacing $dp/d\eta$ with $\varepsilon\dot{u}$ produces the control law (5.1). □

In order to implement (5.1), the continuous time dynamics must be converted to discrete time dynamics and the control u must be updated by $\dot{u}$. A simple integration produces

$$u[k + 1] = u[k] + \dot{u}[k]\Delta t \tag{5.5}$$

where Δt is the integration time step size, k is the time step, and $\dot{u}$ is the same as in (5.1). For low frequencies, a small value of ε does not necessarily produce a u that minimizes (5.2). In this case, the value of $\dot{u}$ may produce a change in u that is much larger than what is desired and (5.2) is never minimized. Thus, the TVD-SP$_\varepsilon$

must run at a higher frequency than TVD-C or TVD-D_k. This provides the agents with control updates on a faster time scale than the motion, leading to performance approaching that of TVD-C. The locational cost of TVD-SP_ε approaches that of TVD-C for small values of ε for very high frequencies.

Chapter 6

Simulation Results

Simulations are run to evaluate the relative performance of the algorithms presented. The algorithms are implemented in C++. We present a comparison of performance between Lloyd, TVD-C, TVD-D_k, and TVD-SP_ε for various density functions using a simulation of 50 holonomic agents.

The following functions serve as time-varying densities representing a target:

$$\begin{aligned}
\phi_1 &= e^{-\left((q_x - 2\sin(\frac{t}{\tau}))^2 + (q_y - \frac{y}{4})^2\right)} \\
\phi_2 &= e^{-\left((q_x - \sin(\frac{t}{\tau}))^2 + (q_y + \sin(\frac{2t}{\tau}))^2\right)} \\
\phi_3 &= e^{-\left((q_x - 2\cos(\frac{t}{\tau}))^2 + (q_y - 2\sin(\frac{t}{\tau}))^2\right)}
\end{aligned} \tag{6.1}$$

where $\tau = 5$. For time-varying densities, the $\frac{\partial c}{\partial p}$ and $\frac{\partial c}{\partial t}$ quantities are computed as in [5]. The measure of performance is the total cost

$$\int_0^T H(p, V, t)dt$$

Figure 6.1 shows the total cost of Lloyd, TVD-C, TVD-D_k, and TVD-SP_ε over time using density ϕ_1 in the simulations with 50 holonomic agents. As the number of terms k increases, the total cost of TVD-D_k decreases, approaching the same total cost as that of TVD-C. Likewise, as the parameter ε approaches zero, the total cost of TVD-SP_ε approaches that of TVD-C.

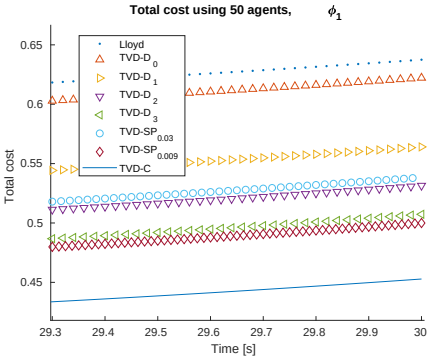

Figure 6.1: Total cost for simulations of 50 holonomic agents using density function ϕ_1

Table 6.1 shows the total cost at $t = 30$s of all algorithms presented in simulations of 50 holonomic agents using density functions (6.1) with $\tau = 5$. Using TVD-D$_k$, as the number of terms k increases, the total cost decreases, approaching the same total cost as that of TVD-C. Likewise, when using TVD-SP$_\varepsilon$, the total cost decreases as the size of the perturbation parameter ε decreases, approaching the same total cost as that of TVD-C. Selecting a small enough perturbation parameter allows for TVD-SP$_\varepsilon$ to outperform TVD-D$_k$ without any need for each agent to use information about the network beyond that of its immediate neighbors.

Table 6.1: Comparison of Total Locational Cost (50 Holonomic Agents), $\tau = 5$

Algorithm	ϕ_1	ϕ_2	ϕ_3
Lloyd	0.637	0.620	0.594
TVD-D$_0$	0.622	0.610	0.584
TVD-D$_1$	0.564	0.550	0.524
TVD-D$_2$	0.531	0.528	0.500
TVD-D$_3$	0.507	0.504	0.479
TVD-SP$_{0.01}$	0.500	0.550	0.556
TVD-SP$_{0.02}$	0.487	0.543	0.525
TVD-SP$_{0.009}$	0.475	0.533	0.461
TVD-C	0.453	0.464	0.446

Since one of the advantages of TVD-SP$_\varepsilon$ is that it updates the control on a faster time scale than the motion, allowing for information to propagate throughout the network, the perturbation parameter and the frequency must be chosen such

that the control is updated on a much faster time scale than the time-varying density, as well as the motion of the agents. In these simulations, the frequency used is $f = \max(f_b, f_b/(\tau\varepsilon))$, where f_b is a base frequency used by Lloyd, TVD-C, and TVD-D_k.

Chapter 7

Experimental Results

We present experimental results from deploying Turtlebots from Willow Garage in a rectangular space to demonstrate each algorithm's performance for agents with unicycle dynamics. The Turtlebots navigate using onboard odometry and an overhead camera. The overhead camera tracks Aruco markers on top of the Turtlebots. The algorithms are implemented in C++ using the Robot Operating System (ROS) library. All algorithms presented (Lloyd, TVD-C, TVD-D_k, and TVD-SP_ε) produce input velocities in global coordinates, which must be converted to unicycle coordinates for the Turtlebots. The input velocity is mapped to unicycle coordinates by

$$v_i = \Big[-\cos(\theta_i), \sin(\theta_i)\Big]\, u_i$$
$$\omega_i = k_s \text{atan2}\left(\frac{\Big[-\sin(\theta_i), \cos(\theta_i)\Big] u_i}{\Big[\cos(\theta_i), \sin(\theta_i)\Big] u_i}\right)$$

where v_i is the forward velocity of the Turtlebot, ω_i is the steering rate of the Turtlebot, θ_i is the heading of the Turtlebot, and k_s is a steering gain. Table 7.1 shows the total cost at $t = 30$s of all algorithms presented in simulations of 50 holonomic agents using density functions (6.1) with $\tau = 5$. Because a first order integrator updates the control, there is no guarantee that these algorithms will perform the same when applied to nonholonomic agents.

Remark 1. *Even if the series expansion converges, the performance of TVD-D_k does not necessarily approach that of TVD-C monotonically in all cases.*

Remark 2. *In the setting where agents have nonholonomic constraints, there is no guarantee that regardless of initial conditions, any distributed algorithm designed for holonomic agents will seek the same local minimum as TVD-C (except for sufficiently large values of k in the case of TVD-D_k or sufficiently small values of ε in the case of TVD-SP_ε). In some cases, TVD-D_k will not have a monotonic decrease in performance as k increases or ε decreases. The centralized algorithm may even under-perform some distributed algorithms when applied to nonholonomic agents.*

In these experiments, initial conditions are chosen so that all of the algorithms converge to the same solution. In particular, the Turtlebots are placed so that they are near a Centroidal Voronoi Tessellation and oriented such that they initially line up with their desired input velocities.

Table 7.1: Comparison of Total Locational Cost (5 Turtlebots), $\tau = 5$

Algorithm	ϕ_1	ϕ_2	ϕ_3
Lloyd	4.695	4.947	4.561
TVD-D_0	4.590	4.808	4.485
TVD-D_1	4.509	4.684	4.387
TVD-D_2	4.486	4.661	4.367
TVD-D_3	4.476	4.661	4.360
TVD-$SP_{0.02}$	4.429	4.684	4.350
TVD-$SP_{0.01}$	4.419	4.664	4.338
TVD-$SP_{0.009}$	4.410	4.653	4.337
TVD-C	4.409	4.652	4.336

In this experiment where initial conditions are selected carefully, we see the same almost the same trends in performance in Table 7.1 for TVD-D_k and TVD-SP_ε as we do for the simulations of holonomic agents in Table 6.1.

Chapter 8

Conclusion

The singular perturbation method converges to the centralized solution for sufficiently small perturbation parameter. The singular perturbation method propagates information throughout the network by updating the control input on a much smaller time scale than the motion of the agents or the density function. This allows each agent to generate a control that minimizes locational cost by obtaining information only from its Voronoi neighbors. This requires the singular perturbation method to be applied at a sufficiently high frequency for the chosen perturbation parameter. The choice becomes one of how quickly the information should propagate throughout the network as opposed to how much information to propagate at once. If for instance agents are limited in how far they can communicate, the singular perturbation method can produce a control that approaches the performance of the centralized algorithm even though information cannot propagate throughout the network at once. In this setting, the singular perturbation method is a better choice than the von Neumann approximation. The singular perturbation method is also a better option in the case where each agent has a fast processor, but very limited memory on board (and few neighbors). Because the high frequency leads to more rounds of communication, the complexity of the singular perturbation method is higher than the von Neumann approximation when these constraints are not present.

For holonomic agents, the total cost also decreases monotonically as the perturbation parameter decreases, approaching the same performance as that of the

centralized approach. For nonholonomic agents, it is necessary (but not sufficient) to choose suitable initial conditions in order for the von Neumann approximation to arrive at the same solution as the centralized algorithm. There is no guarantee that locational cost decreases monotonically except for sufficiently large number of terms in the approximation. The singular perturbation method however, does approach the same solution as the centralized algorithm and locational cost does decrease monotonically for perturbation parameters small enough for convergence.

Appendix A

Computing Jacobian Matrix

In this section, we present an alternate derivation of the Jacobian matrix $\partial c/\partial p$ to the derivation presented in [10] and [5]. To compute the partial derivative of c with respect to p, begin with the mass and centroid of the i^{th} partition.

$$m_i = \int_{V_i(p)} \phi(q,t)dq \tag{A.1}$$

$$c_i = \int_{V_i(p)} q\phi(q,t)dq/m_i \tag{A.2}$$

The following theorem is useful when applying the quotient rule

Theorem 4 (Leibniz rule)**.** *If*

$$F = \int_{\Omega(p)} f(q)dq \tag{A.3}$$

then

$$\frac{\partial F}{\partial p} = \int_{\partial\Omega(p)} f(q)\frac{\partial q}{\partial p}\cdot\hat{n}(q)dq, \forall q \in \partial\Omega \tag{A.4}$$

where $\hat{n}(q)$ represents a unit vector normal to the boundary $\partial\Omega(p)$ pointing outward.

We apply the Leibniz rule to differentiate the i^{th} centroid with respect to its j^{th} neighbor.

$$\frac{\partial c_i}{\partial p_j} =(\int_{\partial V_i(p)} q\phi(q,t)\frac{\partial q}{\partial p_j}\cdot\hat{n}(q)dq \tag{A.5}$$

$$-c_i\int_{\partial V_i(p)}\phi(q,t)\frac{\partial q}{\partial p_j}\cdot\hat{n}(q)dq)/m_i \tag{A.6}$$

where $\hat{n}(q) = (p_j - p_i)/\|p_j - p_i\|$, $j \in \mathcal{N}_{V_i}$, and $\mathcal{N}_{V_i}$ is the neighbor set of the i^{th} Voronoi cell. Each of these integrals is a line integral along a single line segment, the boundary between the i^{th} and j^{th} cell. For integrals where $i = j$, it is necessary to compute along all the segments of the i^{th} Voronoi cell boundary that is shared with a neighboring cell.

$$\frac{\partial c_i}{\partial p_i} = \sum_{j\in\mathcal{N}_{V_i}}(\int_{\partial V_i(p)} q\phi(q,t)\frac{\partial q}{\partial p_i}\cdot\hat{n}(q)dq \tag{A.7}$$

$$-c_i\int_{\partial V_i(p)}\phi(q,t)\frac{\partial q}{\partial p_i}\cdot\hat{n}(q)dq)/m_i \tag{A.8}$$

and $\hat{n}(q)$ is defined as before. Note that in the case where the boundaries of two Voronoi cells share a single point, the integral for that agent pair evaluates to zero, so j is considered not to be part of the neighbor set of i. To compute the integrals, it is necessary to express both $\partial q/_i$ and $\partial q/_j$. We derive $\partial q/_i$ and $\partial q/_j$ as follows. Let

$$q(s) = \frac{1}{2}(p_i + p_j) + \begin{bmatrix}0 & -1\\ 1 & 0\end{bmatrix}(p_j - p_i)s \tag{A.9}$$

$$\tag{A.10}$$

be a pont on the boundary between neighboring Voronoi cells. Then

$$\frac{dq}{ds} = \begin{bmatrix}0 & -1\\ 1 & 0\end{bmatrix}(p_j - p_i) \tag{A.11}$$

$$\frac{\partial q}{\partial p_i} = \frac{1}{2}I - \begin{bmatrix}0 & -1\\ 1 & 0\end{bmatrix}s \tag{A.12}$$

$$\frac{\partial q}{\partial p_j} = \frac{1}{2}I + \begin{bmatrix}0 & -1\\ 1 & 0\end{bmatrix}s \tag{A.13}$$

where $s \in [s_0, s_1]$ and $q(s_0)$ and $q(s_1)$ are the start and end points of the boundary between the i^{th} and j^{th} cells. These points are computed as

$$s_0 = \left(\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \frac{(p_j - p_i)}{\|p_j - p_i\|^2} \right)^T \left(q_0 - \tfrac{1}{2}(p_i + p_j) \right) \tag{A.14}$$

$$s_1 = \left(\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \frac{(p_j - p_i)}{\|p_j - p_i\|^2} \right)^T \left(q_1 - \tfrac{1}{2}(p_i + p_j) \right) \tag{A.15}$$

Because $\hat{n}$ is perpendicular to the boundary $\partial V_i(p)$, the following property must hold:

$$\left(q - \tfrac{1}{2}(p_i + p_j) \right) \cdot (p_j - p_i) = 0, \forall q \in \partial V_{ij}(p) \tag{A.16}$$

This produces the same result as in [10]

$$0 = (p_j - p_i)^T \left(\tfrac{\partial q}{\partial p_j} - \tfrac{1}{2} I \right) + \left(q - \tfrac{1}{2}(p_i + p_j) \right)^T \tag{A.17}$$

$$(p_j - p_i)^T \frac{\partial q}{\partial p_j} = \frac{1}{2}(p_j - p_i)^T + q^T - \frac{1}{2}(p_i + p_j)^T \tag{A.18}$$

$$= (p_j - q)^T \tag{A.19}$$

In a similar fashion,

$$0 = (p_j - p_i)^T \left(\tfrac{\partial q}{\partial p_i} - \tfrac{1}{2} I \right) - \left(q - \tfrac{1}{2}(p_i + p_j) \right)^T \tag{A.20}$$

$$(p_j - p_i)^T \frac{\partial q}{\partial p_i} = \frac{1}{2}(p_j - p_i)^T - q^T + \frac{1}{2}(p_i + p_j)^T \tag{A.21}$$

$$= (q - p_i)^T \tag{A.22}$$

To compute the line integrals, let $s = (1-\theta)s_0 + \theta s_1$, $\frac{ds}{d\theta} = s_1 - s_0$, where $\theta \in [0, 1]$. Then

$$\int_{\partial V_{ij}} f(q) dq = \int_{s_0}^{s_1} f(q(s)) \| \frac{dq}{ds} \| ds \tag{A.23}$$

$$= \int_0^1 f(q(s(\theta))) \| \frac{dq}{ds} \| \| \frac{ds}{d\theta} \| d\theta \tag{A.24}$$

Replacing $f(q(s(\theta)))$ with $\phi(q,t)$ and $q\phi(q,t)$,

$$\frac{\partial}{\partial p_j}\int_{\partial V_{ij}}\phi(q,t)dq = \int_0^1 \phi(q(s(\theta)),t)\hat{n}^T\frac{\partial q}{\partial p_j}\|p_j - p_i\||s_1 - s_0|d\theta \tag{A.25}$$

$$= \int_0^1 \phi(q(s(\theta)),t)(p_j - p_i)^T\frac{\partial q}{\partial p_j}|s_1 - s_0|d\theta \tag{A.26}$$

$$= \int_0^1 \phi(q(s(\theta)),t)(p_j - q(s(\theta)))^T|s_1 - s_0|d\theta \tag{A.27}$$

$$\frac{\partial}{\partial p_j}\int_{\partial V_{ij}}q\phi(q,t)dq = \int_0^1 q(s(\theta))\phi(q(s(\theta)),t)\hat{n}^T\frac{\partial q}{\partial p_j}\|p_j - p_i\||s_1 - s_0|d\theta \tag{A.28}$$

$$= \int_0^1 q(s(\theta))\phi(q(s(\theta)),t)(p_j - p_i)^T\frac{\partial q}{\partial p_j}|s_1 - s_0|d\theta \tag{A.29}$$

$$= \int_0^1 q(s(\theta))\phi(q(s(\theta)),t)(p_j - q(s(\theta)))^T|s_1 - s_0|d\theta \tag{A.30}$$

$$\frac{\partial}{\partial p_i}\int_{\partial V_{ik}}\phi(q,t)dq = \int_0^1 \phi(q(s(\theta)),t)\hat{n}^T\frac{\partial q}{\partial p_i}\|p_k - p_i\||s_1 - s_0|d\theta \tag{A.31}$$

$$= \int_0^1 \phi(q(s(\theta)),t)(p_k - p_i)^T\frac{\partial q}{\partial p_i}|s_1 - s_0|d\theta \tag{A.32}$$

$$= \int_0^1 \phi(q(s(\theta)),t)(q(s(\theta)) - p_i)^T|s_1 - s_0|d\theta \tag{A.33}$$

$$\frac{\partial}{\partial p_i}\int_{\partial V_{ik}}q\phi(q,t)dq = \int_0^1 q(s(\theta))\phi(q(s(\theta)),t)\hat{n}^T\frac{\partial q}{\partial p_i}\|p_k - p_i\||s_1 - s_0|d\theta \tag{A.34}$$

$$= \int_0^1 q(s(\theta))\phi(q(s(\theta)),t)(p_k - p_i)^T\frac{\partial q}{\partial p_j}|s_1 - s_0|d\theta \tag{A.35}$$

$$= \int_0^1 q(s(\theta))\phi(q(s(\theta)),t)(q(s(\theta)) - p_i)^T|s_1 - s_0|d\theta \tag{A.36}$$

The final form of the Jacobian $\partial c/\partial p$ is then

$$\frac{\partial c_i}{\partial p_j} =(\int_0^1 q\phi(q(s(\theta)),t)(p_j - q(s(\theta)))^T|s_1 - s_0|d\theta \tag{A.37}$$

$$- c_i \int_0^1 \phi(q(s(\theta)),t)(p_j - q(s(\theta)))^T|s_1 - s_0|d\theta)/m_i \tag{A.38}$$

$$\frac{\partial c_i}{\partial p_i} = \sum_{k\in\mathcal{N}_{V_i}} (\int_0^1 q\phi(q(s(\theta)),t)(q(s(\theta)) - p_i)^T|s_1^{(k)} - s_0^{(k)}|d\theta \tag{A.39}$$

$$- c_i \int_0^1 \phi(q(s(\theta)),t)(q(s(\theta)) - p_i)^T|s_1^{(k)} - s_0^{(k)}|d\theta)/m_i \tag{A.40}$$

This thesis, in full, is currently being prepared for submission for publication of the material. Gandarillas, Victor; Cortés, Jorge. The thesis author is the primary investigator and author of this paper.

Bibliography

[1] Jorge Cortes, Sonia Martinez, Timur Karatas, and Francesco Bullo. Coverage control for mobile sensing networks. *IEEE Transactions on robotics and Automation*, 20(2):243–255, 2004.

[2] Jorge Cortes, Sonia Martinez, and Francesco Bullo. Coverage Control For Mobile Sensing Networks: Variations On A Theme, July 2002.

[3] M. Emelianenko Q. Du and L. Ju. Convergence of the lloyd algorithm for computing centroidal voronoi tessellations. *SIAM J. Numer. Anal.*, 44(1):102–119, jan 2006.

[4] Sung G. Lee, Yancy Diaz-Mercado, and Magnus Egerstedt. Multirobot Control Using Time-Varying Density Functions. *IEEE Transactions on Robotics*, 31(2):489–493, April 2015.

[5] Yancy Diaz-Mercado, Sung G. Lee, and Magnus Egerstedt. Distributed dynamic density coverage for human-swarm interactions. In *American Control Conference (ACC), 2015*, pages 353–358. IEEE, 2015.

[6] M. van Kreveld M. de Berg and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer Verlag, New York, NY, 1997.

[7] K. Sugihara A. Okabe, B. Boots and S. N. Chiu. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley Series in Probability and Statistics. John Wiley & Sons, New York, NY, 2000.

[8] H. K. Khalil. *Nonlinear Systems*. Prentice-Hall, Englewood Cliffs, NJ, 2nd edition, 1996.

[9] Qiang Du, Vance Faber, and Max Gunzburger. Centroidal voronoi tessellations: Applications and algorithms. *SIAM Rev.*, 41(4):637–676, December 1999.

[10] Qiang Du and Maria Emelianenko. Acceleration schemes for computing centroidal voronoi tessellations. *Numerical Linear Algebra with Applications*, 13(2-3):173–192, 2006.