Optimal Estimation and Control of Large Collaborative Swarms using Random Finite Set Theory

A DISSERTATION
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Bryce Doerr

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Richard Linares, Adviser

September 2019

Acknowledgements

There are many people I would like to thank for their comments, ideas, and support to make this work possible. I have been supported by so many who have greatly contributed to my personal and technical growth. These individuals have allowed me to progress to be the best person I can be both in and out of the field.

I would like to thank my adviser, Dr. Richard Linares, who introduced me to the world of research. Dr. Linares has allowed me to make my mark in the research world with his vast knowledge of the details, and he has provided me with ample room to independently develop my own research. With the support of Dr. Linares, I was able to find a unique foothold to solve problems and learn to grow my interests into passions. I am grateful for the opportunity for growth that Dr. Linares has provided.

I would like to thank my committee, Dr. Peter Seiler, Dr. Demoz Gebre-Egziabher, and Dr. Andrew Lamperski for the wisdom they have imparted to me. Their classes piqued my interest into the world of control and estimation, and their guidance throughout my graduate school journey has motivated me to express the same type of commitment to others who are on a similar journey. I also thank the committee for reading and reviewing my dissertation.

I would like to thank Daniel Miller, Lauren Schlenker, Jordan Larson, and Maria Formiller for reviewing my dissertation drafts.

I would like to acknowledge support from the National Aeronautics and Space Administration under Contract Number NNX16CP45P issued through the NASA STTR Program by the Jet Propulsion Laboratory (JPL) and led by Amir Rahmani at JPL.

There is also a multitude of friends, peers, and mentors I would like to thank. I am grateful for my friends on the UMN Rocket Team including Zach Fox, Hannah Weiher, Chad Serba, Paul Gross, and many others who have helped me become a team player, leader, and an innovator to make our team renown throughout collegiate competitions and industry. I am also grateful for my colleagues and friends from the UMN UAV Lab including Brian Taylor, Chris Regan, Patipan Pipatpinyopong, Alex Hrovat, Adhika Lie, and many others who introduced me to the world of flight control systems and UAVs. I would like to thank my mentors and friends from NASA Armstrong including Albion Bowers, Oscar Murillo, Manuel Castro, Kelley Hashemi, Joe Lorenzetti, Jack Toth, Nancy Reyes, Brian Plank, Kassidy McLaughlin, Lizxandra Flores-Rivera, William Alfano, Alexandra Ocasio, Dhvani Patel, Keenan Albee, Alexander Chen, and many others who have made my internship experiences educational, productive, and most importantly fun. I would like to thank my mentors and friends from AFRL including Christopher Petersen, Andrew Sinclair, Andrew Harris, Brandi McPherson, Benjamin Grace, and many others who have inspired me to become a world class researcher in space systems. I would like to thank my mentors and friends at NASA Goddard including Jim O'Donnell, Paul Mason, Nikesha Davis, Eric Stoneking, Matt Heron, Asher Smith, and many others who have made my dream job into an actual dream career. I would like to thank my graduate student friends including Kerry Sun, Sally Ann Keyes, Siew Peng Mun, Andy Akerson, Jordan Hoyt, and many others who had to deal with my craziness throughout my journey. I have made lasting connections with you all, and I am very thankful.

I would like to thank my closest family and friends, for without, I would have not succeeded. I would like to thank the Chang, Formiller, Nam, and Dale families who have always supported me through thick and thin. Finally, I would like to thank my parents, cousins, aunts, uncles, grandparents, and my brother who have pushed me to be the best person I can be. For all of you, I am forever grateful.

Dedication

To my brother, Ryan

and my parents, Todd and Kyong Doerr

Abstract

Controlling large swarms of robotic agents presents many challenges including, but not limited to, computational complexity due to a large number of agents, uncertainty in the functionality of each agent in the swarm, and uncertainty in the swarm's configuration. The contributions of this work is to form the Random Finite Set (RFS) control for large collaborative swarms, decentralize RFS control for individual agents, and form RFS control using other multi-agent RFS filters.

The state representation of the large swarms with an unknown number of agents is generalized as an RFS where an RFS is a collection of agent states with no ordering between individual agents that can randomly change through time. The novelty of this idea is to generalize the notion of distance using RFS-based distance measures and "close-the-loop" between an estimating and controlling a swarm RFS. Specifically, multi-target estimation is determined using the Gaussian Mixture Probability Hypothesis Density (GM-PHD) filter which processes measurements from an unknown number of agents with defined spawn, birth, and death rates. RFS control is then compared for each distributional distance-based cost studied including the Cauchy-Schwarz, L_2^2, and a modified L_2^2 divergence using a model predictive control (MPC) based Quasi-Newton optimization. Next, RFS control and estimation is extended to MPC via iterative linear quadratic regulator (a variant of differential dynamic programming) for spacecraft swarms. The swarm is estimated in both cardinality (number of agents) and state using the GM-PHD filter which provides the estimates for RFS control. RFS control through ILQR approximates a quadratic value function from the distributional distance-based cost (i.e. the modified L_2^2 divergence) to find an optimal control solution. This results in an implicit proof for RFS control of large collaborative swarms.

The RFS control formulation assumes that the topology underlying the swarm control is complete and uses the complete graph in a centralized manner. To generalize the control topology in a localized or decentralized manner, sparse LQR is used to sparsify the RFS control gain matrix obtained using ILQR. This allows agents to use information of agents near each other (localized topology) or only the agent's own information (decentralized topology) to make a control decision. Sparsity and performance for decentralized RFS control are compared for different degrees of localization in feedback control gains which show that the stability and performance compared to centralized control do not degrade significantly in providing RFS control for large collaborative swarms.

The GM-PHD filter is the most basic RFS-based filters used for estimation. Other RFS-based filters can improve the estimate or provide additional tracking information for RFS control by using either the Cardinalized Probability Hypothesis Density (CPHD) filter or the Generalized labeled Multi-Bernoulli (GLMB) filter, respectively. The CPHD filter generalizes the GM-PHD filter by jointly propagating a generalized cardinality distribution as well as the RFS to produce better estimates at high cardinality. The GLMB filter incorporates labels into the RFS, thus the GLMB filter is able to track individual trajectories of agents through time. Both these filters are propagated in feedback with RFS control for the spacecraft relative motion problem. Specifically, the MPC-based ILQR is implemented to provide swarm control in a centralized manner. By using the CPHD and GLMB filters, the cardinality and state estimates become more accurate for RFS control for large collaborative swarms.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Control of large robotic networks or swarms is currently an area of great interest in controls research. A swarm network is typically comprised of tiny robots programmed with limited actuators that perform specific tasks in the network formation. For example, the swarm can use its combined effort to grasp or move in the environment which can offer better way to meet a goal compared to the abilities of a single agent (Kube and Zhang, 1993). Specifically, in space applications, swarm control of satellites and rovers can be used for the exploration of asteroids and other celestial bodies of interest (Vassev et al., 2008). UAV swarms have also proven to be widely useful in military applications such as border patrol, search and rescue missions, surveillance, communication relaying, and the mapping of hostile territory (Ryan et al., 2004). In reviewing these applications, swarm control of large groups of agents is required.

1.0.1 Challenges

A key challenge that is presented with controlling swarms is the increase in computational complexity with an increasing number of agents (Rubenstein et al., 2014). As the state vector size increases in dimensionality for each additional agent added, the computational complexity increases for the vector. Thus, it can take many hours to meet the control objectives for swarms consisting of one thousand agents when using traditional approaches. Rubenstein et al. (2014) observed that it took 12 hours for 1024 Kilobots to move into a desired formation. These results identify the controller computation time as a function of the number of agents in the swarm as an important problem to consider.

Another challenge is the uncertainty of each agent within the system (Lunze, 1992). Swarm systems that involve low-cost individual agents are not expected to function properly together during the period of control. The behavior of each swarm agent cannot be accurately described without explicitly analyzing and modeling all uncertainty. It is difficult to compute heuristically, and it may be computationally expensive to incorporate. Even under circumstances in which the uncertainty is correctly described, there may be times that the model is not valid during the swarm operation. With all these factors, it becomes increasingly difficult to accurately control swarms while considering the uncertainty of each low-cost agent.

Transmission of information in swarms is an area of concern, especially for low-cost agents (Lunze, 1992). Every agent may have complete measurement data on its system, yet it may possess limited knowledge of the other agents in the swarm. Complications of the overall swarm control objective may occur if the agents have contradictory goals since each agent may not know the overall state completely. To mitigate the information limitation, information structure constraints are imposed to achieve simplicity in the control strategies. Unfortunately, adding information constraints may lead to more complex modeling and analysis. For example, in swarm UAVs, aerial surveillance, tracking, and collision avoidance are all control objectives the swarm may want to achieve (Ryan et al., 2004). These objectives rely on more robust data transmission sensors. Depending on the individual UAV's size, the hardware of the swarm agent is limited on what information can be transmitted between agents. Thus, adding information constraints might be a necessary task to meet the control objectives.

1.0.2 Previous Work

Changing how the model for the representation and behavior for a swarm state in space and time has been shown to alleviate the computational complexity of control methods and solutions (Foderaro et al., 2014; Rudd et al., 2013; Foderaro et al., 2018; Ferrari et al., 2016; Huang et al., 2006). Previously, the swarm/potential model using the random finite set (RFS) formalism was used to describe the temporal evolution of the probabilistic description of the robotic swarm to promote decentralized coordination (Pace et al., 2013). By using a measure-value recursion of the RFS formalism for the swarm agents, the swarm dynamics can be determined with computational efficiency.

Several control techniques have been implemented on swarms to date. With centralized control, one agent in the swarm computes the overall swarm control and manages the control execution for individual agents allowing it to oversee the other agents' system processes. (Sommerville, 2016). This type of control is the easiest to implement on a robotic swarm. Unfortunately, centralized control suffers from two main problems. As the number of agents in the swarm increases, the computational workload becomes more expensive. This is especially true when the swarm agents are low-cost. Additionally, centralized control is not robust against individual agent failures (Mondada et al., 2005). With a thousand low-cost agents present in a swarm, communication, actuation, and sensing are performed with less reliability. Thus, centralized control may not be a viable option for these systems.

Another approach that has been studied is the use of decentralized control which breaks down the centralized control problem into smaller manageable subproblems which are weakly dependent or independent from each other (Bakule, 2008). An early method for decentralizing control of swarms is the use of abstracting on the shape manifold and using Lie groups in the configuration space (Belta and Kumar, 2004). This was to promote control of cooperative robots using limited communication and sensing of the entire group. Unfortunately, this method is not optimal in trajectory generation for individual agents and scaling up the number of agents in a varying environment has not been fully considered.

One improvement to decentralized control methods for swarms that move into different formations can be achieved through the use of artificial potential functions (Kim et al., 2006). By varying the potential functions, the authors were able to develop decentralized control strategies that allow for attractive and repulsive properties for group behavior and motion planning in the swarm. That is, a control solution was found that avoided local minimums from interactions between varying goals, obstacles, or other agents in the system. Although this decentralized control using artificial potential functions is able to converge to different formations, manual tuning and modifications of the artificial potential functions is necessary for the varying goals and obstacles in the system. Additionally, the complexity of solving control problems using artificial potential functions can become more computationally complex as the number of agents increase in the swarm. A method to alleviate this problem is to use bifurcation theory in conjunction with artificial potential fields to control different swarm configurations into formation (Sun and Chen, 2018). Computational complexity using a large number of agents and artificial potential fields is decreased by adjusting a bifurcation parameter for the equilibrium states.

Probabilistic swarm guidance has also been used to enable swarms to converge to target distributions (Bandyopadhyay et al., 2014). Probabilistic swarm guidance controls a swarm density distribution through distributed control so that each agent determines its own trajectory while the swarm converges to the target distribution. Distributed control is defined as the reformulation of a control problem as a set of interdependent subproblems and solving these subproblems (Lunze, 1992). Probabilistic swarm guidance solves issues that involve a large number of agents, also identified as "computationally complex", by controlling the swarm density distribution of the agents (Bandyopadhyay et al., 2014). Using optimal transport, convergence is achieved more rapidly compared to the results of a homogeneous Markov chain approach. Additionally, the cost function is minimized (Bandyopadhyay et al., 2014; Açikmeşe and Bayard, 2012; Chattopadhyay and Ray, 2009; Açıkmeşe and Bayard, 2014; Demir et al., 2015). The inhomogeneous Markov chain can be used as an alternative method (Bandyopadhyay et al., 2013; Hadaegh et al., 2016). Similarly to the homogeneous Markov chain, this method allows agents to move in a decentralized fashion. However, the algorithm also allows for communication between each agent to settle at the target destination. Because the algorithms themselves use the swarm density distribution, these methods are robust to external disturbances. Velocity field generation for swarm control is a non-optimal decentralized control method for swarms that synthesizes smooth velocity fields as a function of time and position (Eren and Açıkmeşe, 2017). With a designated target distribution, the swarm follows the velocity field using the heat equation to convergence by using local agent position information to estimate its local density. The advantage of this method is that the agents facilitate collision avoidance and move in a smoother manner than the previously mentioned Markov chain approaches (Açikmeşe and Bayard, 2012). Unfortunately, the use of the heat equation diffuses the agents in a locally uniform manner to the target density. Therefore, this is a non-optimal method of controlling the swarm to achieve a target distribution.

The distributed optimal control method is a method that controls multi-agent systems by modeling the agents as Gaussian mixtures and using an integral cost function that is optimized to the advection equation (a partial differential equation that contains the dynamical constraint) (Foderaro et al., 2014; Rudd et al., 2013; Foderaro et al., 2018). The control laws themselves are determined using potential functions that attract the agent distributions to the desired state and repel the distribution from obstacles (Reif and Wang, 1999). By minimizing the objective function based on distributions using the necessary conditions of optimality, the optimal control law is found using the potential function. The distributed optimal control method is expanded to use the Kullback-Leibler divergence metric using distributions in the objective function for the use of path planning (Ferrari et al., 2016). This provides a discovery to a whole class of divergence measures of distributions that can provide converging optimal control solutions to multi-agent systems.

Model predictive control (MPC) has been heavily studied for nonlinear systems (Garcia et al., 1989; Mayne et al., 2000) and for applications including spacecraft maneuvering and attitude control (Camacho et al., 1999; Di Cairano et al., 2012; Manikonda et al., 1999). Decentralized MPC has also been studied for thousands of low-cost spacecraft with limited capabilities (Morgan et al., 2013). This method computes the control input by solving a finite horizon problem. By optimizing over the present time to some time in the future, a control input is found that accounts for future consequences. The benefit of this solution is that it decentralizes the computation and communication required for the swarm system (Morgan et al., 2013). Consequently, swarms of more than a thousand units can be controlled without being computationally expensive. This method also decreases the run-time of the finite horizon optimal control problem that uses convex programming because it reduces the horizon for the optimization to take place and allows for constraints on each swarm agent (Morgan et al., 2013). By using this decentralized configuration approach, any measurement and process uncertainties in the trajectories are accounted for within the algorithm. Thus, this provides robustness for the swarm while pushing the initial swarm to its desired state.

Another decentralized approach for controlling swarms is using sequential convex programming (Morgan et al., 2012). Sequential convex programming uses multiple iterations to maintain the accuracy of the convex approximations of the constraints which create more efficient trajectories. Uncertainties in the trajectories are accounted for when the algorithm is tuned. Using sequential convex programming in combination with MPC in real time provides robustness for the swarm while pushing the agents to the designated targets. The same authors also used sequential convex programming to do target assignment (mapping of agents to targets) and trajectory generation for varying swarm sizes through time (Morgan et al., 2015). This method is viable for swarms, but as it will be discussed with RFS control, target assignment is not necessary.

1.0.3 Contributions

Although these approaches provide a method to control swarm agents, RFS control for large, low-cost swarms mitigates problems such as computational complexity due to a large number of agents, uncertainty in the functionality of each agent in the swarm, and uncertainty in the swarm's configuration for control (Doerr and Linares, 2018). The contributions of this work is to form the RFS control for large collaborative swarms, decentralize RFS control for individual agents, and form RFS control using other multi-agent RFS filters.

RFS Control

The first contribution presented is to generalize the state representation of the control problem to account for large swarms with an unknown number of the agents (Doerr and Linares, 2018). This is done by representing the swarm as an RFS, where RFS is a collection of agent states, with no ordering between individual agents, that can randomly change through time (Mahler, 2003). Figure 1.1 shows the concept of the contributed work, where the first moment of the RFS is used as the state, ν, and the desired RFS swarm configuration is defined by its first moment, ν_{des}. The novelty of this work is to generalize the notion of distance using RFS-based distance measures and to “close-the-loop” by processing measurements from an unknown number of agents with defined spawn (B), birth (Γ), and death (D) rates. This multi-target estimation is determined using the Gaussian Mixture Probability Hypothesis Density (GM-PHD) filter. Initially, the RFS control is compared for each distributional distance-based cost using model predictive control (MPC) using a Quasi-Newton optimization. Next, RFS control and estimation is extended to MPC via iterative LQR (ILQR), a variant of differential dynamic programming (DDP), for spacecraft swarms. In this example, the topology underlying the swarm control is complete and uses the complete graph in a centralized manner. To obtain a complete graph for RFS control, the swarm is estimated in both cardinality (number of agents) and state using the GM-PHD filter. RFS control through ILQR approximates a quadratic value function from the distributional distance-based cost, and iterates to determine an optimal control solution. The results combining the PHD filter and ILQR using the RFS formalism provide implicit proof for RFS control of large collaborative swarms.

Decentralized RFS Control

The second contribution is to generalize the control (information) topology in a localized or decentralized manner using sparsity matrices in control. In the original RFS control problem, the control topology is assumed to be complete using all the state information obtained from the GM-PHD filter.

This is centralized control in which the swarm computes the overall swarm control and manages the control execution for individual agents allowing it to oversee the other agents’ control processes. In this contribution, the decentralized or localized RFS control is realized using sparse LQR to sparsify the RFS control gain matrix obtained using ILQR. This allows agents to use local information topology (information of agents near each other) or a fully decentralized topology (information of the agent’s own information) to make a control decision. Sparse LQR allows for more stability

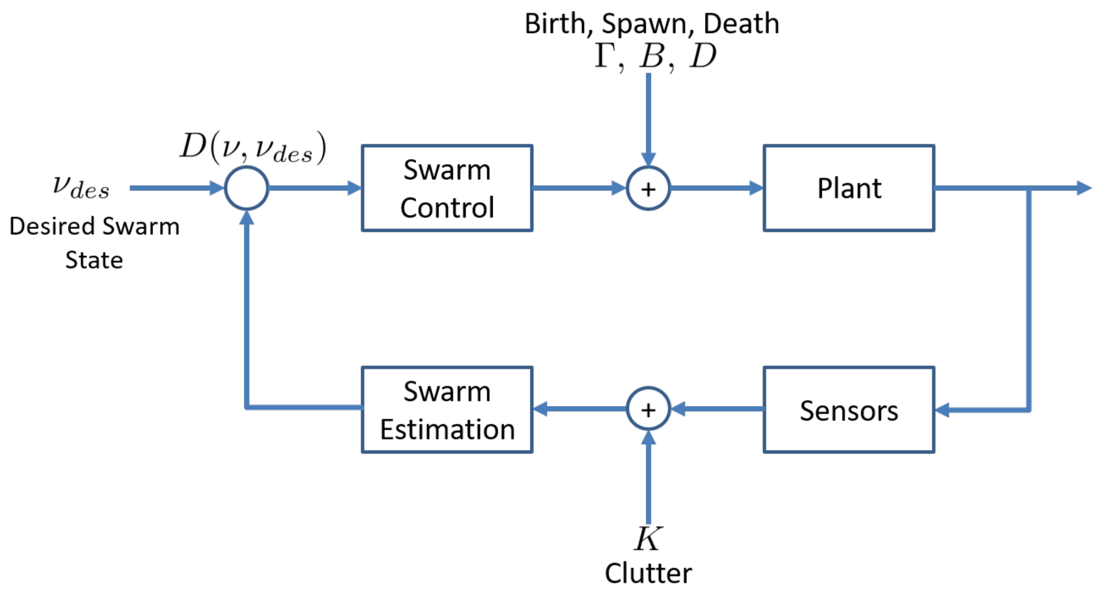


Figure 1.1: A block diagram of the RFS control and estimation architecture in a closed-loop.

and less performance degradation than truncating a centralized control matrix may provide. Sparsity and performance for decentralized RFS control are compared for different degrees of localization in the feedback control gains which show the viability for decentralized control for large collaborative swarms.

Other multi-Target Filters for RFS Control of Large Collaborative Swarms

The third contribution to RFS control is the use of multi-agent filters that improve the RFS estimate of the swarm or provide additional tracking information using either the Cardinalized Probability Hypothesis Density (CPHD) filter or the Generalized labeled Multi-Bernoulli (GLMB) filter, respectively.

In the GM-PHD filter, the RFS is assumed to be Poisson distributed. Thus, the mean and the variance are the same. The mean of the RFS is the total number of agents. So, as the number of agents increase, the more varied the estimate becomes.

The CPHD filter generalizes the GM-PHD filter by jointly propagating the cardinality distribution as well as the RFS intensity to produce reliable estimates at high cardinality.

Another problem with the GM-PHD filter is that it cannot label agents it estimates through time (i.e. the filter cannot differentiate between estimated agents as they evolve through time). Thus, it makes it difficult to correspond an individual agent's trajectory with estimated time-history data from the filter. The GLMB filter alleviates this problem by incorporating labels into the RFS. Thus, the GLMB filter is able to track individual trajectories through time.

For both the CPHD and the GLMB filter, the RFS estimated is propagated with the RFS control in feedback for the spacecraft relative motion problem. Specifically, the MPC-based ILQR is implemented to provide swarm control in a centralized manner. By using the CPHD and GLMB filters, the cardinality and state estimates become more accurate for RFS control for large collaborative swarms.

Chapter 2

RFS Control

2.1 Random Finite Set Control Problem Formulation

The framework to control swarming agents is to set up a multiple-agent filtering problem using RFS theory (Mahler, 2003; Vo and Ma, 2006). To formulate the multiple-agent filtering problem, the single-agent filtering problem is first discussed.

2.1.1 Single-Agent Filtering

When estimating the dynamical system for a single agent, it is usually assumed that the state space follows a Markov process with a transition density,

$$f_{k|k-1}\left(\mathbf{x}_k|\mathbf{x}_{k-1}\right), \tag{2.1}$$

to move discretely from the previous state $\mathbf{x}_{k-1}$ to the next state at $\mathbf{x}_k$. Note that $\mathbf{x}_k$ is for a single agent. For generality, the dynamical system is partially observed as a likelihood function given by

$$g_k\left(\mathbf{z}_k|\mathbf{x}_k\right), \tag{2.2}$$

where the likelihood function is a probability density of observing the system by obtaining measurements, $\mathbf{z}_k$. By using the observation information from $\mathbf{z}_{1:k} = (\mathbf{z}_1, \cdots, \mathbf{z}_k)$, the posterior density estimate at a time k is determined using the Bayesian recursion given by

$$p_{k|k-1}\left(\mathbf{x}_k|\mathbf{z}_{1:k-1}\right) = \int f_{k|k-1}\left(\mathbf{x}_k|\mathbf{x}_{k-1}\right) p_{k-1}\left(\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1}\right) d\mathbf{x}_{k-1}, \tag{2.3a}$$

$$p_k\left(\mathbf{x}_k|\mathbf{z}_{1:k}\right) = \frac{g_k\left(\mathbf{z}_k|\mathbf{x}_k\right) p_{k|k-1}\left(\mathbf{x}_k|\mathbf{z}_{1:k-1}\right)}{\int g_k\left(\mathbf{z}_k|\mathbf{x}_k\right) p_{k|k-1}\left(\mathbf{x}_k|\mathbf{z}_{1:k-1}\right) dx_k}. \tag{2.3b}$$

The posterior density contains the measurement update, and the estimate for this single-agent system can be found using a minimum mean squared error method.

2.1.2 RFS Formulation

The multi-agent problem considers the Bayesian recursion through a RFS formulation with discrete-time dynamics (Vo and Ma, 2006). This theory addresses the decentralized formulation for each agent in the formation. Each agent has the challenge of estimating its local formation configuration and designing a control policy to achieve some local configuration. It is assumed that each agent within the swarm is identical and that using unique identifiers on each agent is unnecessary. Using RFS theory, the number of agents and their states is determined from measurements. The agents in the field may die, survive and move into the next state through dynamics, or appear by spawning or birthing. The number of agents in the field is denoted by $N_{\text{total}}(t)$ and may be randomly varying at each time-step by the union of the birth ($\Gamma_k : \emptyset \rightarrow \{\mathbf{x}_{i,k}, \mathbf{x}_{i+1,k}, \cdots, \mathbf{x}_{i+N_{birth},k}\}$), spawn ($B_{k|k-1}(\zeta) : \mathbf{x}_{i,k-1} \rightarrow \left\{\mathbf{x}_{i,k}, \mathbf{x}_{i+1,k}, \cdots, \mathbf{x}_{i+N_{spawn},k}\right\}$), and surviving ($S_{k|k-1}(\zeta) : \mathbf{x}_{i,k-1} \rightarrow \mathbf{x}_{i,k}$) agents. Death is denoted by $D_k(\zeta) : \mathbf{x}_{i,k-1} \rightarrow \emptyset$. Note that $\mathbf{x}_{i,k}$ is for the ith swarm agent's state. This is described by a RFS, X_k, given by

$$X_k = \left[\bigcup_{\zeta \in X_{k-1}} S_{k|k-1}(\zeta) \right] \cup \left[\bigcup_{\zeta \in X_{k-1}} B_{k|k-1}(\zeta) \right] \cup \Gamma_k. \tag{2.4}$$

$X_k = \left\{ \mathbf{x}_{1,k}, \mathbf{x}_{2,k}, \cdots, \mathbf{x}_{N_{total(k)},k} \right\}$ denotes a realization of the RFS distribution for agents. The individual RFSs in Eq. (2.4) are assumed to be independent from each other. For example, any births that occur at any time-step are independent from any surviving agents. At any time, k, the RFS probability density function can be written as

$$p(X_k = \{\mathbf{x}_{1,k}, \mathbf{x}_{2,k}, \cdots, \mathbf{x}_{n,k}\}) = p(|X_k| = n) p(\{\mathbf{x}_{1,k}, \mathbf{x}_{2,k}, \cdots, \mathbf{x}_{n,k}\} \mid |X_k| = n). \tag{2.5}$$

For a generalized observation process, the agents are either detected $(\Theta_k(\mathbf{x}_k) : \mathbf{x}_{i,k} \rightarrow \mathbf{y}_{i,k})$, or not detected $(F_k(\mathbf{x}_k) : \mathbf{x}_{i,k} \rightarrow \emptyset)$. Clutter or false alarms $(K_k : \emptyset \rightarrow \{\mathbf{y}_{1,k}, \mathbf{y}_{2,k}, \cdots, \mathbf{y}_{N_{clutter},k}\})$, defined as measurements that do not belong to any agents, are also present in the set of observations. Note that $\mathbf{y}_{i,k}$ is for the ith swarm agent's measurement. Therefore, RFS of measurements is described by

$$Z_k = K_k \cup \left[\bigcup_{\mathbf{x_k} \in X_k} \Theta_k(\mathbf{x}_k) \right], \tag{2.6}$$

where the origins of each measurement are not known and unique identifiers are not necessary. Again, the individual RFSs in Eq. (2.6) are independent of each other, so measurements and clutter are obtained independently from each other. Single-agent filtering cannot be applied because measurements cannot be associated with the agent that generated it. By using the RFS formulation, measurements can be associated to individual agents in the swarm.

On a similar note, the control sequence is also defined by a RFS in the form

$U_k = \left\{\mathbf{u}_{1,k}, \mathbf{u}_{2,k}, \cdots, \mathbf{u}_{N_{total(k)},k}\right\}$ and a RFS probability density given by

$$p(U_k = \{\mathbf{u}_{1,k}, \mathbf{u}_{2,k}, \cdots, \mathbf{u}_{n,k}\}) = p(|U_k| = n)p(\{\mathbf{u}_{1,k}, \mathbf{u}_{2,k}, \cdots, \mathbf{u}_{n,k}\} \mid |U_k| = n), \quad (2.7)$$

since the number of the agents in the field to be controlled is also varying.

The random finite set formulation of describing multi-agent states and observations can be described very similarly to Eq. (2.1) and (2.2), but the RFS states (X_k) and observations (Z_k) are used instead. To determine the multi-agent posterior density, a multi-agent Bayes recursion is used given by

$$p_{k|k-1}\left(X_k|Z_{1:k-1}\right) = \int f_{k|k-1}\left(X_k|X_{k-1}\right) p_{k-1}\left(X_{k-1}|Z_{1:k-1}\right) \mu_s(dX_{k-1}), \quad (2.8a)$$

$$p_k\left(X_k|Z_{1:k}\right) = \frac{g_k\left(Z_k|X_k\right) p_{k|k-1}\left(X_k|Z_{1:k-1}\right)}{\int g_k\left(Z_k|X_k\right) p_{k|k-1}\left(X_k|Z_{1:k-1}\right) \mu_s(dX_k)}, \quad (2.8b)$$

where μ_s is a reference measure on some function $F(X)$. The recursion is computationally expensive due to multiple integrals, but solutions have been found for a small number of targets using sequential Monte Carlo (Ma et al., 2006). Fortunately, a PHD filter approximation provides computational tractability for a larger number of agents.

2.1.3 Probability Hypothesis Density (PHD) Filter

Instead of propagating the multi-agent posterior density through a multi-agent Bayes recursion, the Probability Hypothesis Density (PHD) filter propagates the posterior intensity function. The nonnegative intensity function, $v(\mathbf{x})$, is a first-order statistical moment of the RFS state that represents the probability of finding an agent in a region of state space S. The expected number of agents in the region S is the integral of the intensity function given by

$$\mathbb{E}(|X \cap S|) = \int |X \cap S| P(dX) = \int_S v(\mathbf{x}) d\mathbf{x}, \tag{2.9}$$

where the expectation represents a RFS X intersecting a region S with a probability distribution P dependent on X. This gives the total mass or the expected number of agents of RFS X in a region S. The local maximum in intensity $v(\mathbf{x})$ shows the highest concentration of expected number of agents which can be used to determine an estimate for the agents in X at a time-step.

Poisson RFS are fully characterized by their intensities. By assuming the RFS X is Poisson of the form $p(|X| = n)$ and $p(\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\} \mid |X| = n)$, approximate solutions can be determined by the PHD filter. Propagation of the PHD can be determined if the agents are assumed to be independent and identically distributed with the cardinality of the agent set that is Poisson distributed (Vo and Ma, 2006). Secondly, it is assumed that the agents' motion and measurements are independent of each other. Thirdly, clutter and birth RFSs are assumed to be Poisson RFSs and clutter is independent from the measurement RFS. Lastly, the time-update multi-target density $p_{k|k-1}$ is Poisson, but if there is no spawning and the surviving and birth RFSs are Poisson, then this assumption is satisfied. It is noted that the assumptions made by the PHD filter are strong assumptions for swarming robotics. However, this is a good starting point for an initial proof-of-concept study. The PHD recursion for a general intensity function, $v_t(\mathbf{x})$, is given by

$$\bar{v}_t(\mathbf{x}) = b(\mathbf{x}) + \int p_s(\zeta) f(\mathbf{x}|\zeta) v(\zeta) d\zeta + \int \beta(\mathbf{x}|\zeta) v(\zeta) d\zeta, \tag{2.10a}$$

where $b(\mathbf{x})$, $p_s(\zeta)$, and $\beta(\mathbf{x}|\zeta)$ are the agents' birth, survival, and spawn intensity, $f(\mathbf{x}|\zeta)$ is the target motion model, and ζ is the previous state respectively (Vo and Ma, 2006). The bar on $\bar{v}_t(\mathbf{x})$ denotes that the PHD has been time-updated. For the measurement update, the equation is given by

$$v_t(\mathbf{x}) = (1 - p_d(\mathbf{x}))\bar{v}_t(\mathbf{x}) + \sum_{\mathbf{z}\in Z_t} \frac{p_d(\mathbf{x})g(\mathbf{z}_t|\mathbf{x})\bar{v}_t(\mathbf{x})}{c(\mathbf{z}) + \int p_d(\zeta)g(\mathbf{z}_t|\zeta)\bar{v}_t(\zeta)d\zeta}, \tag{2.10b}$$

where $p_d(\mathbf{x})$, $g(\mathbf{z}_t|\mathbf{x})$, and $c(\mathbf{z})$ are the probability of detection, likelihood function, and clutter model of the sensor respectively (Vo and Ma, 2006). Note that the PHD filter given by Eq. (2.10) is more thoroughly derived using finite set statistics in Appendix A. By using this recursion, the swarm probabilistic description can be updated. The recursion itself avoids computations that arise from the unknown relation between agents and its measurements, and that the posterior intensity is a function of a single agent's state space. Unfortunately, Eq. (2.10) does not contain a closed-form solution and the numerical integration suffers from higher computational time as the state increases due to an increasing number of agents.

2.1.4 Gaussian Mixture Model and Control Formulation

Fortunately, a closed-form solution exists if it is assumed that the survival and detection probabilities are state independent (i.e. $p_s(\mathbf{x}) = p_s$ and $p_d(\mathbf{x}) = p_d$), and the intensities of the birth and spawn RFSs are Gaussian mixtures. To formulate the optimal control problem, the current and desired intensities are

$$\bar{\nu}(\mathbf{x}, k) \triangleq \sum_{i=1}^{N_f} w_f^{(i)} \mathcal{N}\left(\mathbf{x}; \mathbf{m}_f^i, P_f^i\right) \triangleq \nu_b(\mathbf{x}, k) + \nu_{p_s}(\mathbf{x}, k) + \nu_\beta(\mathbf{x}, k), \tag{2.11}$$

$$\nu_{des}(\mathbf{x}, k) \triangleq g(\mathbf{x}) \triangleq \sum_{i=1}^{N_g} w_g^{(i)} \mathcal{N}\left(\mathbf{x}; \mathbf{m}_g^i, P_g^i\right), \tag{2.12}$$

where $w^{(i)}$ are the weights and $\mathcal{N}\left(\mathbf{x}; \mathbf{m}^i, P^i\right)$ is the probability density function of a ith multivariate Gaussian distribution with a mean and covariance corresponding to the peaks and spread of the intensity respectively. N_f and N_g are the total number of multivariate Gaussian distributions in the current and desired intensities, respectively. It is assumed that the desired Gaussian mixture intensity, $\nu_{des}(\mathbf{x}, k)$, is known. Eq. (2.11) includes the summation of the individual birth ($\nu_b(\mathbf{x}, k)$), spawn (ν_β), and survival ($\nu_{p_s}(\mathbf{x}, k)$) Gaussian mixture intensities which simplify to another Gaussian mixture. Note that closed form solutions using Gaussian mixtures exist for cases without the state independent assumption. Additionally, $\sum_{i=1}^{N_f} w_f^{(i)} = N_{\text{total}}(t)$ and $\sum_{i=1}^{N_g} w_g^{(i)} = \bar{N}_{\text{total}}(t)$ where $\bar{N}_{\text{total}}(t)$ is the desired number of agents. The intensity function $\nu(\mathbf{x}, t)$ is in terms of the swarm state while $\nu_{des}(\mathbf{x}, t)$ is in terms of the desired state. The swarm intensity function can be propagated through updates on the mean and covariance of the Gaussian mixtures as given by

$$\mathbf{m}_{f,k+1}^i = A_k \mathbf{m}_{f,k}^i + B_k \mathbf{u}_{f,k}^i, \tag{2.13}$$

$$P_{f,k+1}^i = A_k P_{f,k}^i A_k^T + Q_k, \tag{2.14}$$

where Q_k is process noise. The agents' states $\mathbf{x}$ are incorporated in the mean and covariance of the Gaussian mixture intensity. Then given the Gaussian mixture intensities assumption, a control variable is calculated for each component $\mathbf{u}_{f,k}^i$. Additionally, each Gaussian mixture component represents many agents since the intensity function integrates to the total number of agents. Note that although linear dynamics are used, the dynamics can be modeled as a nonlinear function of the state.

Additionally, the measurement update is also closed form given by the intensity

$$\nu_k(\mathbf{x}, k) = f(\mathbf{x}) = (1 - p_d(\mathbf{x}))\bar{\nu}_k(\mathbf{x}) + \sum_{\mathbf{z} \in Z_k} \sum_{j=1}^{N_f} w_k^{(j)} \mathcal{N}\left(\mathbf{x}; \mathbf{m}_{k|k}^{(j)}(\mathbf{z}), P_{k|k}^{(j)}\right), \tag{2.15}$$

where

$$w_k^{(j)} = \frac{p_d(\mathbf{x}) w_f^{(j)} q^{(j)}(\mathbf{z})}{K(\mathbf{z}) + p_d(\mathbf{x}) \sum_{l=1}^{N_f} w_f^{(l)} q^{(l)}(\mathbf{z})}, \tag{2.16a}$$

$$\mathbf{m}_{k|k}^{(j)}(\mathbf{z}) = \mathbf{m}_f^{(j)} + K^{(j)} \left(\mathbf{z} - H_k \mathbf{m}_f^{(j)}\right), \tag{2.16b}$$

$$P_{k|k}^{(j)} = \left(I - K^{(j)} H_k\right) P_f^i, \tag{2.16c}$$

$$K^{(j)} = P_f^i H_k^T \left(H_k P_f^i H_k^T + R_k\right)^{-1}, \tag{2.16d}$$

$$q_k^{(j)}(\mathbf{z}) = \mathcal{N}\left(\mathbf{z}; H_k \mathbf{m}_f^{(j)}, R_k + H_k P_f^i H_k^T\right), \tag{2.16e}$$

which follow closely to the Kalman filter measurement update equations.

Each individual swarm agent runs a local PHD observer to estimate the state of the swarm by modeling the swarm as a distribution. Thus, using RFS theory, it is assumed that the individual swarm agents form an intensity function that is a Gaussian mixture intensity in which the means and covariances of the Gaussian mixture are propagated and controlled. An optimal control problem is set up that tracks a desired swarm formation by minimizing its control effort in the following objective function

$$J(\mathbf{u}) = \int_0^T \mathbf{u}(t)^T R \mathbf{u}(t) + D(\nu(\mathbf{x}, t), \nu_{des}(\mathbf{x}, t)) dt, \tag{2.17}$$

where $\nu_{des}(\mathbf{x}, t)$ is the desired formation, R is the positive definite control weight matrix, and $\mathbf{u}(t)$ is the control effort for the Gaussian mixture intensities shown in Eq. (2.13). Both $\nu(\mathbf{x}, t)$ and $\nu_{des}(\mathbf{x}, t)$ are defined over the complete state space which include position and velocity parameters. $D(\cdot, \cdot)$ is the distance between Gaussian mixtures which has several closed-form solutions, and it has been used previously to define an objective function for path planning of multi-agent systems (Ferrari et al., 2016).

The key features for the RFS control problem is that it can allow for a unified representation for swarming systems. This unified representation is achieved by minimizing the RFS objective function, Eq. (2.17), about the swarm intensity statistics given by Eq. (2.13) and (2.14). Thus, it can handle multi-fidelity swarm localization and control. The swarm is treated probabilistically and the bulk motion is modeled which allows the theory to handle large numbers of indistinguishable units with unknown swarm size. This reduces the dimensionality of the state while enabling complex behavior. Naturally, the RFS control problem is formulated to enable complex decision making through RFS theory.

2.2 Distributional Distance Based-Cost

The control objective for the RFS formulation of agents with an unknown distance between the intensities is provided by Eq. (2.17). The distance metric can be defined using several closed-form solutions for Gaussian mixtures. Then, the corresponding optimal control problem is formulated using several closed-form methods discussed in the next section.

2.2.1 Cauchy-Schwarz Divergence

The Cauchy-Schwarz divergence is based on the Cauchy-Schwarz inequality for inner products of RFS, and it is defined for two RFS with intensities f and g given by

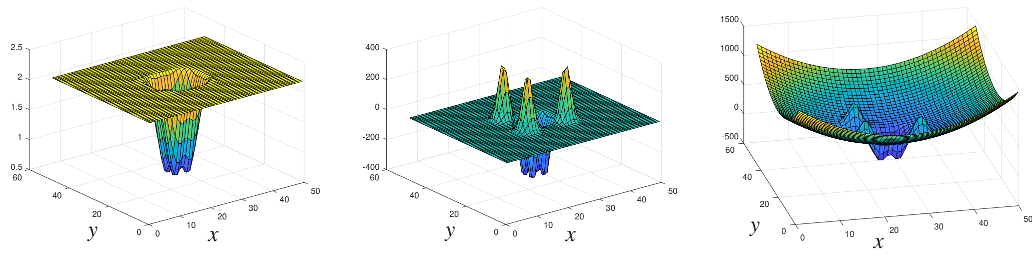
$$D_{CS}(f,g) = -\ln\left(\frac{\langle f,g\rangle}{\|f\|\|g\|}\right), \tag{2.18}$$

where $\langle\cdot,\cdot\rangle$ is the L_2^2 inner product over the RFS intensities. The argument of the logarithm is non-negative because probability densities are non-negative, and it does not exceed one by the Cauchy-Schwarz inequality. The Cauchy-Schwarz divergence can be interpreted as an approximation to the Kullback-Leibler divergence but has a closed-form expression for Gaussian mixtures (Hoang et al., 2015). This is useful for calculating the distance between two-point processes represented by intensity functions. By substituting the intensities from Eq. (2.15) and Eq. (2.12) for f and g respectively, the Cauchy-Schwarz divergence between two Poisson point processes with Gaussian mixture intensities, $D_{CS}(f,g)$, is simplified to

$$\begin{aligned} D_{CS}(f,g) = &\frac{1}{2}\ln\left(\sum_{j=1}^{N_f}\sum_{i=1}^{N_f} w_f^{(j)} w_f^{(i)} \mathcal{N}(\mathbf{m}_f^j; \mathbf{m}_f^i, P_f^i + P_f^j)\right) \\ &+ \frac{1}{2}\ln\left(\sum_{j=1}^{N_g}\sum_{i=1}^{N_g} w_g^{(j)} w_g^{(i)} \mathcal{N}(\mathbf{m}_g^j; \mathbf{m}_g^i, P_g^i + P_g^j)\right) \\ &- \ln\left(\sum_{j=1}^{N_g}\sum_{i=1}^{N_f} w_g^{(j)} w_f^{(i)} \mathcal{N}(\mathbf{m}_g^j; \mathbf{m}_f^i, P_g^i + P_f^j)\right). \end{aligned} \tag{2.19}$$

Note that in the control formulation used, only $\nu(\mathbf{x}, t)$ is assumed to depend on the control $\mathbf{u}$. Therefore, the term that depends only on $\nu_{des}(\mathbf{x}, t)$ is omitted from the objective function since $\nu_{des}(\mathbf{x}, t)$ does not depend on $\mathbf{u}$.

Figure 2.1a shows the surface plot using the Cauchy-Schwarz divergence for four Gaussian mixtures in the swarm at an initial time instance which designates the distributional distance-based cost of the objective function. The four Gaussian mixtures start with initial conditions of (±3,±3) in a square grid. The desired intensity is set as (±1,±1) in a square grid. From the surface plot, each initial intensity has hills while the desired intensity has valleys. The goal is to minimize the objective function, thus, an optimization method (e.g. the Quasi-Newton method) determines a control solution which minimizes the objective. Since the desired intensity in Figure 2.1a is located at a minimum in the objective surface plot, the optimization method finds a



(a) Cauchy-Schwarz Divergence (b) L_2^2 Distance (c) L_2^2 + Quadratic Term

Figure 2.1: Surface Plots of the objective function with three distributional distance-based costs. The current intensity and desired intensity are initialed at (±3,±3) and (±1,±1) in a square grid, respectively.

control input to move towards that point. The opposite occurs with the hills (current intensity). The minimization finds a control solution that moves away from the hills, and thus gives individual current Gaussian mixtures collision avoidance attributes. Therefore in the minimization of the objective function, each Gaussian mixture will repel each other while moving towards the desired Gaussian mixtures through time. Although the Cauchy-Schwarz divergence has a repelling effect, collision avoidance is not guaranteed, but the distance does encourage collision-reducing trajectory solutions. If the initial intensity is too large compared to the desired intensity, it will take longer for the four Gaussian mixtures to converge to the desired values or diverge due to the optimization getting stuck in local minima (the flat plane). Also, the repelling effect due to the hills are relatively small. Thus, the Cauchy-Schwarz divergence may not be the fastest converging solution for the objective function minimization.

2.2.2 L_2^2 Distance

Alternatively, the distance between two Poisson point processes with Gaussian mixture intensities can be determined by using the L_2^2 distance between the intensities.

The L_2^2 is given by

$$D_{L_2^2}(f,g) = \int (f-g)^2 \, d\mathbf{x} = ||f-g||^2, \tag{2.20}$$

where the close-form solution for Gaussian mixture intensities is simplified to

$$\begin{aligned} D_{L_2^2}(f,g) &= \sum_{j=1}^{N_f}\sum_{i=1}^{N_f} w_f^{(j)} w_f^{(i)} \mathcal{N}\left(\mathbf{m}_f^j; \mathbf{m}_f^i, P_f^i + P_f^j\right) \\ &+ \sum_{j=1}^{N_g}\sum_{i=1}^{N_g} w_g^{(j)} w_g^{(i)} \mathcal{N}\left(\mathbf{m}_g^j; \mathbf{m}_g^i, P_g^i + P_g^j\right) \\ &- 2\sum_{j=1}^{N_g}\sum_{i=1}^{N_f} w_g^{(j)} w_f^{(i)} \mathcal{N}\left(\mathbf{m}_g^j; \mathbf{m}_f^i, P_g^i + P_f^j\right). \end{aligned} \tag{2.21}$$

The L_2^2 distance is stationary, i.e. gradients are zero, when intensities f and g are equal. That is, the cost is minimum when the target g is reached from any intensity f.

The L_2^2 distance follows the property of the Bregman divergence which has an additional property of convexity (Banerjee et al., 2005). The distance, given by

$$D_F(f,g) = F(f) - F(g) - \langle \nabla F(g), f-g \rangle, \tag{2.22}$$

is convex if $F(\cdot)$ is strictly convex and continuously differentiable on a closed convex set (Banerjee et al., 2005). A list of strictly convex functions are listed in (Banerjee et al., 2005). For this work, the squared Euclidean distance $F(f) = f^2$ was used to generate the Bregman divergence given by

$$D_F(f,g) = \langle f,f \rangle + \langle g,g \rangle - 2\langle f,g \rangle, \tag{2.23}$$

which is in the same exact form of Eq. (2.21). Figure 2.1b shows the surface plot using the L_2^2 distance for a 4 Gaussian mixture swarm for the same example as the Cauchy-Schwarz divergence. The initial intensity has more defined hills compared to the Cauchy-Schwarz divergence. Thus, the initial Gaussian mixtures have a stronger repelling effect upon one another. Also, the desired Gaussian mixtures have large valleys that create a large attraction effect for each initial Gaussian mixture to move to. Thus, the optimization solution will be faster in the L_2^2 distance case. Unfortunately, the L_2^2 distance suffers from a similar issue to the Cauchy-Schwarz divergence. If the initial conditions increase farther away from the desired intensity, the optimization may take much longer or get stuck in local minima due to a flat surface away from the desired intensity.

2.2.3 L_2^2 Distance with Quadratic Term

The issue of convergence remains for the L_2^2 distance when the initial states are farther away from the desired intensity. To achieve faster convergence, an additional term is added to the L_2^2 distance to shape the gradient descent through a quadratic term as given by

$$D_{L_2^2 mod}(f,g) = D_{L_2^2}(f,g) - \alpha \sum_{j=1}^{N_g} \sum_{i=1}^{N_f} w_g^{(j)} w_f^{(i)} \ln \left(\mathcal{N}(\mathbf{m}_g^j; \mathbf{m}_f^i, P_g^i + P_f^j) \right), \tag{2.24}$$

where α is a fixed or changing parameter. Unfortunately, adding the quadratic term to the L_2^2 distance does not make the objective function stationary at $f = g$. To alleviate this issue, the α parameter is included with the quadratic term to relax the contribution of the gradient to the L_2^2 stationary point. By substituting Eq. (2.21) into Eq. (2.24), the equation becomes

$$\begin{aligned}
D_{L_2^2 mod}(f,g) = & \sum_{j=1}^{N_f}\sum_{i=1}^{N_f} w_f^{(j)} w_f^{(i)} \mathcal{N}(\mathbf{m}_f^j; \mathbf{m}_f^i, P_f^i + P_f^j) \\
& + \sum_{j=1}^{N_g}\sum_{i=1}^{N_g} w_g^{(j)} w_g^{(i)} \mathcal{N}(\mathbf{m}_g^j; \mathbf{m}_g^i, P_g^i + P_g^j) \\
& - 2\sum_{j=1}^{N_g}\sum_{i=1}^{N_f} w_g^{(j)} w_f^{(i)} \mathcal{N}(\mathbf{m}_g^j; \mathbf{m}_f^i, P_g^i + P_f^j) \\
& - \alpha\sum_{j=1}^{N_g}\sum_{i=1}^{N_f} w_g^{(j)} w_f^{(i)} \ln\left(\mathcal{N}(\mathbf{m}_g^j; \mathbf{m}_f^i, P_g^i + P_f^j)\right) .
\end{aligned} \tag{2.25}$$

Note that this term is referred as quadratic, although it may be more appropriate to call it quadratic-like. Figure 2.1c shows the surface plot using Eq. (2.25) for the same 4 Gaussian mixture swarm used in the Cauchy-Schwarz divergence. Compared to the L_2^2 distance, the initial and desired intensities provide the hills and valleys necessary to obtain convergence. However, as the initial intensity move outwards, the surface map decreases in a quadratic fashion instead of staying flat. This prevents the optimization from converging to a local minima. Instead, the additional quadratic term allows convergence to the desired intensity (global minima). Thus, the optimization can occur at any point to reach convergence.

Traditional LQR based solutions are not applicable to the minimization of the objective function, Eq. (2.25), since the L_2^2 terms are nonquadratic (Todorov and Li, 2005). The minimization of the objective function in discrete time is

$$\begin{aligned}\min_{\mathbf{u}_k, k=1,\ldots,T} J(\mathbf{u}) = \sum_{k=1}^{T} \mathbf{u}_k^T R \mathbf{u}_k &+ \sum_{j=1}^{N_f}\sum_{i=1}^{N_f} w_{f,k}^{(j)} w_{f,k}^{(i)} \mathcal{N}(\mathbf{m}_{f,k}^j; \mathbf{m}_{f,k}^i, P_{f,k}^i + P_{f,k}^j) \\ &+ \sum_{j=1}^{N_g}\sum_{i=1}^{N_g} w_{g,k}^{(j)} w_{g,k}^{(i)} \mathcal{N}(\mathbf{m}_{g,k}^j; \mathbf{m}_{g,k}^i, P_{g,k}^i + P_{g,k}^j) \\ &- 2\sum_{j=1}^{N_g}\sum_{i=1}^{N_f} w_{g,k}^{(j)} w_{f,k}^{(i)} \mathcal{N}(\mathbf{m}_{g,k}^j; \mathbf{m}_{f,k}^i, P_{g,k}^i + P_{f,k}^j) \\ &- \alpha\sum_{j=1}^{N_g}\sum_{i=1}^{N_f} w_{g,k}^{(j)} w_{f,k}^{(i)} \ln\left(\mathcal{N}(\mathbf{m}_{g,k}^j; \mathbf{m}_{f,k}^i, P_{g,k}^i + P_{f,k}^j)\right),\end{aligned} \tag{2.26}$$

$$\begin{aligned}\text{Subject to}: \mathbf{m}_{f,k+1}^i &= A_k \mathbf{m}_{f,k}^i + B_k \mathbf{u}_{f,k}^i, \\ P_{f,k+1}^i &= A_k P_{f,k}^i A_k^T + Q_k,\end{aligned} \tag{2.27}$$

where $\mathbf{u}_k = [(\mathbf{u}_{f,k}^1)^T, \cdots, (\mathbf{u}_{f,k}^{N_f})^T]^T$ is the collection of all control variables. Therefore, control solutions are found by either using DDP where the objective function is quadratized by taking a Taylor series approximation about a nominal trajectory or using optimization techniques (e.g. the Quasi-Newton method) where the nonquadratic objective function is used directly to find an optimal control solution.

2.3 Differential Dynamic Programming

The LQR finite-horizon optimal control is first discussed in Section 2.3.1 in order to provide the necessary background to approach DDP control for general nonlinear system dynamics and a nonquadratic objective function discussed in Section 2.3.2.

2.3.1 LQR Finite-Horizon Optimal Control Problem

The linear quadratic regulator problem is defined by a discrete time-varying system given by

$$\mathbf{x}_{k+1} = A_k\mathbf{x}_k + B_k\mathbf{u}_k + \mathbf{g}_k, \tag{2.28}$$

where $\mathbf{g}_k$ is Brownian process noise. For the finite horizon N, the total cost is calculated from an initial state $\mathbf{x}_0$ and using the control sequence $U = [\mathbf{u}_k, \mathbf{u}_{k+1}, \cdots, \mathbf{u}_{N-1}]$ applied to the dynamics given by

$$J(\mathbf{x}_0, U) = \sum_{k=0}^{N-1} l(\mathbf{x}_k, \mathbf{u}_k) + l_f(\mathbf{x}_N), \tag{2.29}$$

where $l(\mathbf{x}_k, \mathbf{u}_k)$ is the running cost and $l_f(\mathbf{x}_N)$ is the terminal cost. The LQR costs are quadratic given by

$$l(\mathbf{x}_k, \mathbf{u}_k) = \frac{1}{2}\begin{bmatrix} 1 \\ \mathbf{x}_k \\ \mathbf{u}_k \end{bmatrix}^T \begin{bmatrix} 0 & \mathbf{q}_k^T & \mathbf{r}_k^T \\ \mathbf{q}_k & Q_k & P_k \\ \mathbf{r}_k & P_k & R_k \end{bmatrix} \begin{bmatrix} 1 \\ \mathbf{x}_k \\ \mathbf{u}_k \end{bmatrix}, \quad l_f(\mathbf{x}_N) = \frac{1}{2}\mathbf{x}_N^T Q_N \mathbf{x}_N + \mathbf{x}_N^T \mathbf{q}_N, \tag{2.30}$$

where $\mathbf{q}_k$, $\mathbf{r}_k$, Q_k, R_k, and P_k are the running weights (coefficients), and Q_N and $\mathbf{q}_N$ are the terminal weights. The weight matrices, Q_k and R_k, are positive definite and the block matrix $\begin{bmatrix} Q_k & P_k \\ P_k & R_k \end{bmatrix}$ is positive-semidefinite (Inaba and Corke, 2016). The running and terminal costs are substituted into Eq. (2.29), and due to the symmetry in the weight matrices, the total cost is simplified to

$$J(\mathbf{x}_0, U) = \sum_{k=0}^{N-1} \mathbf{x}_k^T \mathbf{q}_k + \mathbf{u}_k^T \mathbf{r}_k + \frac{1}{2}\mathbf{x}_k^T Q_k \mathbf{x}_k + \frac{1}{2}\mathbf{u}_k^T R_k \mathbf{u}_k + \mathbf{u}_k^T P_k \mathbf{x}_k + \frac{1}{2}\mathbf{x}_N^T Q_N \mathbf{x}_N + \mathbf{x}_N^T \mathbf{q}_N. \tag{2.31}$$

The optimal control solution is based on minimizing the cost function in terms of the control sequence which is given by

$$U^*(\mathbf{x}_0) = \arg\min_U J(\mathbf{x}_0, U). \tag{2.32}$$

To solve for the optimal control solution given by Eq. (2.32), a value iteration method is used. Value iteration is a method that determines the optimal cost-to-go (value) starting at the final time-step and moving backwards in time minimizing the control sequence. The cost-to-go is defined as

$$J(\mathbf{x}_k, U_k) = \sum_{k}^{N-1} l(\mathbf{x}_k, \mathbf{u}_k) + l_f(\mathbf{x}_N), \tag{2.33}$$

where $U_k = [\mathbf{u}_k, \mathbf{u}_{k+1}, \cdots, \mathbf{u}_{N-1}]$. This is very similar to Eq. (2.29), but the only difference is that the cost starts from time-step k instead of $k = 0$. The optimal cost-to-go is calculated similar to Eq.(2.32) which is

$$V(\mathbf{x}_k) = \min_{U_k} J(\mathbf{x}_k, U_k). \tag{2.34}$$

At a time-step k, the optimal cost-to-go function is a quadratic function given by

$$V(\mathbf{x}_k) = \frac{1}{2}\mathbf{x}_k^T S_k \mathbf{x}_k + \mathbf{x}_k^T \mathbf{s}_k + c_k, \tag{2.35}$$

where S_k, $\mathbf{s}_k$, and c_k are computed backwards in time using the value iteration method. First, the final conditions $S_N = Q_N$, $\mathbf{s}_N = \mathbf{q}_N$, and $c_N = c$ are set. This reduces the minimization of the entire control sequence to just a minimization over a control input at a time-step which is the principle of optimality (Bellman et al., 1954). To find the optimal cost-to-go, the Riccati equations are used to propagate the final conditions backwards in time given by

$$S_k = A_k^T S_{k+1} A_k + Q_k - \left(B_k^T S_{k+1} A_k + P_k^T\right)^T \left(B_k^T S_{k+1} B_k + R_k\right)^{-1} \left(B_k^T S_{k+1} A_k + P_k^T\right), \tag{2.36a}$$

$$\begin{aligned} \mathbf{s}_k &= \mathbf{q}_k + A_k^T \mathbf{s}_{k+1} + A_k^T S_{k+1} \mathbf{g}_k \\ &\quad - \left(B_k^T S_{k+1} A_k + P_k^T\right)^T \left(B_k^\mathsf{T} S_{k+1} B_k + R_k\right)^{-1} \left(B_k^T S_{k+1} \mathbf{g}_k + B_k^T \mathbf{s}_{k+1} + \mathbf{r}_k\right), \end{aligned} \tag{2.36b}$$

$$\begin{aligned} c_k &= \mathbf{g}_k^T S_{k+1} \mathbf{g}_k + 2\mathbf{s}_{k+1}^T \mathbf{g}_k + c_{k+1} \\ &\quad - \left(B_k^T S_{k+1} \mathbf{g}_k + B_k^T \mathbf{s}_{k+1} + \mathbf{r}_k\right)^T \left(B_k^\mathsf{T} S_{k+1} B_k + R_k\right)^{-1} \left(B_k^T S_{k+1} \mathbf{g}_k + B_k^T \mathbf{s}_{k+1} + \mathbf{r}_k\right). \end{aligned} \tag{2.36c}$$

Using the Ricatti solution, the optimal control policy is in the affine form

$$\mathbf{u}_k(\mathbf{x}_k) = K_k \mathbf{x}_k + \mathbf{l}_k, \tag{2.37}$$

where K_k is the controller given by

$$K_k = -(R_k + B_k^T S_{k+1} B_k)^{-1} (B_k^T S_{k+1} A_k + P_k^T), \tag{2.38}$$

and l_k is the controller offset given by

$$\mathbf{l}_k = -(R_k + B_k^T S_{k+1} B_k)^{-1}(B_k^T S_{k+1}\mathbf{g}_k + B_k^T \mathbf{s}_{k+1} + \mathbf{r}_k). \tag{2.39}$$

This optimal solution to the LQR problem works for linear dynamics and quadratic cost functions, but unfortunately, the objective function specified for the swarm problem is nonquadratic. Fortunately, differential dynamic programming can be used for nonlinear dynamics and nonquadratic local cost functions.

2.3.2 The Differential Dynamic Programming Problem

The DDP approach to solving nonlinear and nonquadratic equations uses a similar process as the LQR solution, but a second-order approximation of the dynamics and objective function are obtained for value iteration and the solution is iterated to increasingly get better approximations of the optimal trajectory of the system. Note that if linear dynamics are used, the iterative linear quadratic regulator (ILQR) formulation is obtained (Tassa et al., 2014; Todorov and Li, 2005). Since the results are produced by a linear system, both the DDP and ILQR terms can be used interchangeably. The following discussion on DDP follows closely to that of Tassa et al. (2014, 2012). The general nonlinear discrete-time dynamics is given by

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k), \tag{2.40}$$

where the state at the next time-step, $\mathbf{x}_{k+1}$, is a function of the current state, $\mathbf{x}_k$, and control input $\mathbf{u}_k$. The cost function is in the form of Eq. (2.29), but the costs are nonquadratic. The solution to the optimal control problem is Eq. (2.32). Similarly, the cost-to-go and the optimal cost-to-go function are defined by Eq. (2.33) and Eq. (2.34) respectively. By setting the terminal condition $V(\mathbf{x}_N) = l_f(\mathbf{x}_N)$, the priniciple of optimality is used to minimize over the control at a time-step given by

$$V(\mathbf{x}_k) = \min_{\mathbf{u}_k} \left(l(\mathbf{x}_k, \mathbf{u}_k) + V(\mathbf{x}_{k+1}) \right), \tag{2.41}$$

and solved through time by a backwards pass (value iteration).

Backward Pass

The first step in the backward pass (value iteration) is to determine a value function that is quadratic. The argument in Eq. (2.41) is taken as a function of small perturbations around the state ($\delta\mathbf{x}_k$) and control input ($\delta\mathbf{u}_k$), and it is quadratized through a second order Taylor series expansion given by

$$\begin{aligned} Q(\delta\mathbf{x}, \delta\mathbf{u}) &= l(\mathbf{x}_k + \delta\mathbf{x}_k, \mathbf{u}_k + \delta\mathbf{u})_k - l(\mathbf{x}, \mathbf{u}) + V(\mathbf{x}_{k+1} + \delta\mathbf{x}_{k+1}) - V(\mathbf{x}_{k+1}), \\ &\approx \frac{1}{2} \begin{bmatrix} 1 \\ \delta\mathbf{x}_k \\ \delta\mathbf{u}_k \end{bmatrix}^T \begin{bmatrix} 0 & Q_x^T & Q_u^T \\ Q_x & Q_{xx} & Q_{xu} \\ Q_u & Q_{ux} & Q_{uu} \end{bmatrix} \begin{bmatrix} 1 \\ \delta\mathbf{x}_k \\ \delta\mathbf{u}_k \end{bmatrix}, \end{aligned} \tag{2.42}$$

where Q_x, Q_u, Q_{xx}, Q_{xu}, and Q_{uu} are the running coefficients (weights) of the quadratized value function at a certain time-step. Note, in the standard formulation, the time-step k is dropped for these equations. Any primes denote the next time-step. The equations for the running weights are given by

$$Q_x = l_x + f_x^T V_x', \tag{2.43a}$$

$$Q_u = l_u + f_u^T V_x', \tag{2.43b}$$

$$Q_{xx} = l_{xx} + f_x^T V_{xx}' f_x + V_x' f_{xx}, \tag{2.43c}$$

$$Q_{uu} = l_{uu} + f_u^T V'_{xx} f_u + V'_x f_{uu}, \tag{2.43d}$$

$$Q_{ux} = l_{ux} + f_u^T V'_{xx} f_x + V'_x f_{ux}, \tag{2.43e}$$

where l_x, l_u, l_{xx}, l_{uu}, and l_{ux} are the gradients and Hessians of the cost function, f_x, f_u, f_{xx}, f_{uu} f_{ux} are the gradients and Hessians of the nonlinear dynamics, and V'_x, and V'_{xx} are the gradient and Hessian of the value function. For the ILQR formulation, the gradients and Hessians for an LTV system (which is the model used for RFS control in Section 2.5) are trivial, but for the DDP formulation, the gradients and Hessians for the nonlinear dynamics must be computed. By using this quadratic approximation, the minimum in terms of $\delta\mathbf{u}$ is found using

$$\delta\mathbf{u} = \arg\min_{\delta\mathbf{u}} Q\left(\delta\mathbf{x}, \delta\mathbf{u}\right) = -Q_{uu}^{-1}\left(Q_u + Q_{ux}\delta\mathbf{x}\right), \tag{2.44}$$

which provides local feedback and feed-forward gains of

$$K = -Q_{uu}^{-1} Q_{ux}, \tag{2.45a}$$

$$\mathbf{k} = -Q_{uu}^{-1} Q_u, \tag{2.45b}$$

respectively. The locally optimal controller is substituted back into Eq. (2.42) to get the optimal value at a time-step k given by

$$\Delta V = -\frac{1}{2}\mathbf{k}^T Q_{uu}\mathbf{k}, \tag{2.46a}$$

$$V_x = Q_x - K^T Q_{uu}\mathbf{k}, \tag{2.46b}$$

$$V_{xx} = Q_{xx} - K^T Q_{uu} K, \tag{2.46c}$$

so the value can be propagated backwards in time to find new locally optimal solutions to the value function.

Forward Pass

By continually computing the quadratic approximations in Eq. (2.43), local controller in Eq. (2.45), and the new values in Eq. (2.46) backwards in time from the terminal condition $V(\mathbf{x}_N) = l_f(\mathbf{x}_N)$, the updated trajectory can be found through a forward pass given by

$$\hat{\mathbf{x}}_0 = \mathbf{x}_0, \tag{2.47a}$$

$$\hat{\mathbf{u}}_k = \mathbf{u}_k + \mathbf{k}_k + K_k \left(\hat{\mathbf{x}}_k - \mathbf{x}_k \right), \tag{2.47b}$$

$$\hat{\mathbf{x}}_{k+1} = f(\hat{\mathbf{x}}_k, \hat{\mathbf{u}}_k). \tag{2.47c}$$

where $\hat{\mathbf{x}}_k$ and $\hat{\mathbf{u}}_k$ consists of the state and control input at a time-step of the new trajectory $\left(\hat{\mathbf{X}}, \hat{\mathbf{U}}\right)$. This composes one iteration of DDP. If the cost of the new trajectory, $\left(\hat{\mathbf{X}}, \hat{\mathbf{U}}\right)$, is less than the cost of the old trajectory, $(\mathbf{X}, \mathbf{U})$, then $\mathbf{X} = \hat{\mathbf{X}}$ and $\mathbf{U} = \hat{\mathbf{U}}$ are set, and the algorithm is ran again until a convergence threshold is met between the old and new costs.

Regularization via the Levenberg-Marquardt Heuristic

If the cost of the new trajectory is greater than the cost of the old trajectory, the iteration has not provided a better solution. To circumvent this issue, the Hessian is regularized. This is called the Levenberg-Marquardt heuristic. The control sequence that is calculated in DDP is computed like a Newton optimization which uses second order information (curvature information) on top of the first order information (gradient information) (Liao and Shoemaker, 1992). By including second order curvature to the update, optimization can occur faster, but this relies on the fact that the Hessian is positive definite and an accurate quadratic model. If the control update is not improving (for a non-positive definite Hessian and inaccurate quadratic model), the Levenberg-Marquardt heuristic uses less curvature information and more on the gradient information. This regularization is added to the Hessian of the control cost given by

$$\tilde{Q}_{uu} = Q_{uu} + \mu I_m, \tag{2.48}$$

where $\tilde{Q}_{uu}$ is the regularized control cost Hessian, μ is the Levenberg-Marquardt parameter, and I_m is the identity matrix that is the size of the control input vector (Liao and Shoemaker, 1991). This allows for the increase or decrease of curvature information in the optimization by adding a quadratic cost around the current control input. Unfortunately, adding this regularization term can have different effects at different time-steps using the same control perturbation based on a changing f_u in the linearized dynamics. By increasing $\mu \to \infty$, the $\mathbf{k}$ and K gains become very small due to the $\tilde{Q}_{uu}^{-1}$ term. Therefore, the regularization term is improved by penalizing the states instead of the control inputs which are given by

$$\tilde{Q}_{uu} = l_{uu} + f_u^T \left(V'_{xx} + \mu I_n\right) f_u + V'_x f_{uu}, \tag{2.49a}$$

$$\tilde{Q}_{ux} = l_{ux} + f_u^T \left(V'_{xx} + \mu I_n\right) f_x + V'_x f_{ux}, \tag{2.49b}$$

$$K = -\tilde{Q}_{uu}^{-1} \tilde{Q}_{ux}, \tag{2.49c}$$

$$\mathbf{k} = -\tilde{Q}_{uu}^{-1} Q_u, \tag{2.49d}$$

where I_n is the identity matrix that is the size of the state vector. The μ parameter is placed on the state instead of the control input. For this method, the regularization term is directly incorporated with f_u, and the feedback gains, $\mathbf{k}$ and K, do not disappear as $\mu \to \infty$. Instead, the new $\mathbf{k}$ and K values bring the new trajectory closer to the old one. For the implementation of the μ term, three requirements should be followed. If reaching the minimum is accurate, the μ should become zero in order to obtain faster convergence due to the second order optimization term. If a non-positive definite $\tilde{Q}_{uu}$ is found, the backward pass should be restarted with a larger μ. The last requirement is that when a $\mu > 0$ is needed, the smallest μ should be used that allows the $\tilde{Q}_{uu}$ to be positive definite. Therefore, more of the second order information can be used to provide faster convergence than using gradient descent. The specific algorithm is found in (Tassa et al., 2012).

Eq. (2.46) must also be modified based on regularization added in Eq. (2.49a) (Todorov and Li, 2005). Eq. (2.46) was originally derived using Eqs. (2.42) and (2.44), but using the new regularized terms in Eq. (2.49a) creates error. Therefore, the modified values at a time-step k are

$$\Delta V = \frac{1}{2}\mathbf{k}^T Q_{uu}\mathbf{k} + \mathbf{k}^T Q_u, \tag{2.50a}$$

$$V_x = Q_x + K^T Q_{uu}\mathbf{k} + K^T Q_u + Q_{ux}^T\mathbf{k}, \tag{2.50b}$$

$$V_{xx} = Q_{xx} + K^T Q_{uu} K + K^T Q_{ux} + Q_{ux}^T K. \tag{2.50c}$$

The regularization terms create a faster and more accurate solution to the backwards pass of the DDP solution.

Forward Pass Line Search

Regularization of the forward pass can improve convergence and performance of the DDP algorithm. For linear time-varying systems, one iteration provides a minimal solution after one iteration. This is not the case for general nonlinear systems. Since nonlinear systems are approximated by a Taylor series expansion, there may be regions in the new DDP trajectory that are not valid about the nonlinear model. This may lead to divergence and have a larger cost function than the old trajectory. To fix this issue, a backtracking line-search parameter is introduced in the control update equation given by

$$\hat{\mathbf{u}}_k = \mathbf{u}_k + \alpha \mathbf{k}_k + K_k \left(\hat{\mathbf{x}}_k - \mathbf{x}_k \right), \tag{2.51}$$

where α is set to $\alpha = 1$ at the start of the forward pass. Then the expected cost reduction is considered using

$$\Delta J(\alpha) = \alpha \sum_{k=0}^{N-1} \mathbf{k}(k)^T Q_u(k) + \frac{\alpha^2}{2} \sum_{k=0}^{N-1} \mathbf{k}^T(k) Q_{uu}(k) \mathbf{k}(k). \tag{2.52}$$

A ratio z is determined using the actual and expected cost reduction given by

$$z = \frac{\left(J(\mathbf{x}_0, U) - J(\hat{\mathbf{x}}_0, \hat{U}) \right)}{\Delta J(\alpha)}, \tag{2.53}$$

where $J(\mathbf{x}_0, U)$ and $J(\hat{\mathbf{x}}_0, \hat{U})$ are the old and new cost respectively. The control update is accepted if the condition given by

$$0 < c_1 < z, \tag{2.54}$$

is met where c_1 is a parameter set by the user. The c_1 is usually set close to zero. If the condition is not met, the forward pass is restarted with a smaller α value which means that the new trajectory strayed farther than the system's region of validity. By using the α line search parameter, convergence can be achieved for nonlinear systems by iteratively deceasing α to obtain a cost reduction.

DDP Summary

An DDP iteration can be summarized in four steps. First, an initial rollout of the nonlinear dynamics given by Eq. (2.40) is integrated over time for a given control sequence U. If there is no good initialization of the control sequence, the control sequence can be set to $U = 0$. After the initial rollout, the derivatives of the cost function and nonlinear dynamics used in Eq. (2.43) are found. The derivatives are used in the third step which is to determine local control solutions using a backward pass. Using the terminal condition, $V(\mathbf{x}_N) = l_f(\mathbf{x}_N)$, local control solutions are found by iterating Eq. (2.43), (2.49), and (2.50) backwards at each time-step. When a non-positive definite $\tilde{Q}_{uu}$ is found, increase the regularization parameter μ and restart the backward pass. Once a local optimal policy is found, α is set to $\alpha = 1$, and Eqs. (2.47c) and (2.52) are propagated forward in time. If the integration diverged or cost reduction condition in Eq. (2.54) was not met, the forward pass is restarted with a smaller α.

2.4 Receding Horizon Control RFS Theory

An optimal solution, $\mathbf{u}$, can also be obtained by minimizing the objective, Eq. (2.25), using MPC or receding horizon control (Morgan et al., 2015). Conceptually, at a time k, the knowledge of the system model is used to derive a sequence $\mathbf{u}(k|k), \mathbf{u}(k+1|k), \mathbf{u}(k+2|k), \cdots, \mathbf{u}(k+T_p|k)$ where T_p is the finite prediction horizon from the current state $\mathbf{x}(k)$ (Findeisen and Allgöwer, 2002). With the input sequence, the state is moved forward in time by the control horizon, T_c; usually one time-step. Then the same strategy is repeated for time $k+1$. T_p can be chosen to be either small or large. As T_p increases, the degrees of freedom in the optimization increase which can slow down the algorithm considerably, even though more of the future reference trajectory would bring the output closer to the reference. With a smaller T_p, the computation time will be faster, but the optimization may be more suboptimal. Thus, the swarm may not converge to the desired configuration.

For the RFS control formulation, a $\mathbf{u}$ that controls the swarm intensities through their statistics (mean and covariance) is found by minimizing the objective as given by Eq. (2.26) and (2.27). This can be done by using MPC via the Quasi-Newton method or DDP. DDP is able to determine an optimal solution for nonlinear equations of motion and a nonquadratic objective function through an iterative process of finding the optimal solution involving second-order approximations of the dynamics and the objective function. The dynamical systems used in the results are linear, thus, DDP can be formed as its variant, ILQR. For the Quasi-Newton method, the optimal control input $\mathbf{u}$ is found using MATLAB's fminunc solver (Fletcher, 1980). Note that MPC via DDP or the Quasi-Newton method are both closed-loop control methods in terms of the statistics (mean and covariance) of the system. Then, the agents in the swarm are initialized to the closest Gaussian mixture using the Mahalanobis distance given by

$$D_M(\mathbf{x}, \mathbf{m}^i_{f,k}) = \sqrt{(\mathbf{x} - \mathbf{m}^i_{f,k})^T P^{-1} (\mathbf{x} - \mathbf{m}^i_{f,k})}, \tag{2.55}$$

which measures the distance of the agents to the means of the Gaussian mixtures. As the swarm evolves, this distance determines which agents belong to a given component. Agents are controlled according to their placement in each Gaussian mixture through an open loop method using the DDP or MPC control input obtained for each Gaussian mixture. Although an open loop method was used, feedback control can be used if the state estimates are determined from the PHD filter.

MPC and the Quasi-Newton algorithm can handle nonlinearities in the objective function, and it provides an initial basis in comparing the time-history responses for RFS control using different distributional distance-based costs. RFS control is extended to MPC with DDP which approximates (quadratizes) the objective function for value iteration to provide quick and reliable convergence to locally-optimal control solutions. The RFS control solution is demonstrated on spacecraft swarm relative motion simulation with and without perfect information combining the GM-PHD filter and DDP (formed as ILQR) in a closed-loop fashion given in Figure 1.1. In a single loop, the RFS control is determined from optimizing the objective containing the distributional distance based cost between the estimate and the desired intensity, the swarm dynamics are updated with new spawn, birth, or death of agents in the field, and measurements including clutter are incorporated into the overall system before a GM-PHD filter estimate is determined for control again. From this RFS-based architecture, the ability to determine an estimate of the cardinality and states of the swarm which is used directly for control using ILQR is realized.

2.5 Dynamical Models

To show viability of optimal swarm control via RFS, an acceleration model and a relative motion model, both linear systems, are used to describe rover and satellite dynamics, respectively. The dynamic equations of individual agents are used here to describe the dynamics of the Gaussian mixture components (means) given by the control objective Eqs. (2.26) and (2.27). Since linear dynamics are used, the DDP term can be expressed as ILQR.

2.5.1 Acceleration Model

On a 2D plane, the linear time-invariant (LTI) system of each agent can be described by the continuous state and control matrices

$$A_c = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad B_c = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, \tag{2.56}$$

and a state vector $\mathbf{x} = [x, y, \dot{x}, \dot{y}]^T$. Both x and y are defined to be the 2D positions of the agent respectively. The A_c and B_c matrices are discretized along a fixed time interval utilizing a zero-order hold assumption for the control (i.e. control is held constant over the time-interval). This results in discretized A and B matrices for the state-space equation,

$$\mathbf{x}_{k+1} = A\mathbf{x}_k + B\mathbf{u}_k. \tag{2.57}$$

2.5.2 Relative Motion using Clohessy-Wiltshire Equations

For a spacecraft in low Earth orbit, the relative dynamics of each spacecraft (agent), to a chief spacecraft in circular orbit, is given by the Clohessy-Wiltshire equations (Curtis, 2013)

$$\ddot{x} = 3n^2x + 2n\dot{y} + a_x, \tag{2.58a}$$

$$\ddot{y} = -2n\dot{x} + a_y, \tag{2.58b}$$

$$\ddot{z} = -n^2z + a_z, \tag{2.58c}$$

where x, y, and z are the relative positions in the orbital local-vertical local-horizontal (LVLH) frame and a_x, a_y, and a_z are the accelerations in each axis respectively. The variable n is defined as the orbital frequency given by

$$n = \sqrt{\frac{\mu}{a^3}}, \tag{2.59}$$

where μ is the standard gravitational parameter and a is the radius of the circular orbit. The continuous state-space representation is given by

$$A_c = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 3n^2 & 0 & 0 & 0 & 2n & 0 \\ 0 & 0 & 0 & -2n & 0 & 0 \\ 0 & 0 & -n^2 & 0 & 0 & 0 \end{bmatrix}, \quad B_c = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.60}$$

with a state vector $\mathbf{x} = [x, y, z, \dot{x}, \dot{y}, \dot{z}]$ and a control input $\mathbf{u} = [a_x, a_y, a_z]^T$. These equations are discretized similarly to the acceleration model discussed previously.

2.6 Results

Using the acceleration model, which is discretized from Eq. (2.56) to Eq. (2.57), a 4 Gaussian mixture swarm on a 2-D plane is initialized in a square grid where the mixtures 1, 2, 3, and 4 are defined counterclockwise starting on the first quadrant. With the 4 Gaussian mixture swarm, three different test cases are implemented to bring the intensity to the target trajectories and to test the control theory involved from the control formulation. The first test case compares the L_2^2 distance with varying initial conditions in a square grid with the L_2^2 plus quadratic distance with four desired Gaussian mixtures located at (±1,±1) using Quasi-Newton MPC. An L_2^2 plus quadratic distance comparison is also done using ILQR. The last two cases present the Quasi-Newton MPC and ILQR control using the L_2^2 distance with a quadratic term and varying desired Gaussian mixtures. For Case 2, three target destinations are located at (±1,1) and (-1,-1). Lastly, in Case 3, five target destinations are located at (±1,±1) and (0,0).

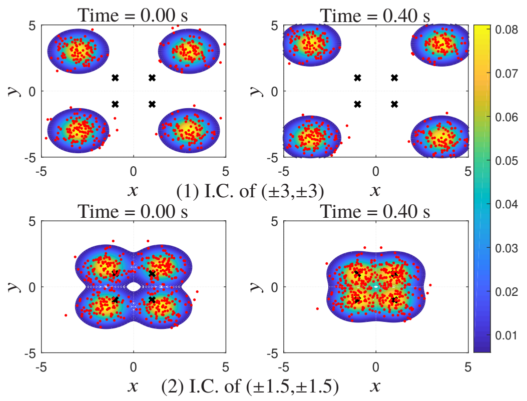
Using the results of the acceleration model, control using RFS is also expanded to satellite relative motion using the Clohessy-Wiltshire Equations. Specifically, the L_2^2 plus quadratic distance is used for spacecraft formation flight. A 77 Gaussian mixture swarm is initialized uniformly random between -1 and 1 on a 2D plane. Assuming that the spacecraft swarm is at lower Earth orbit, the goal for the spacecraft is to track a rotating star pattern moving counterclockwise at an orbital frequency of n.

2.6.1 Acceleration Model

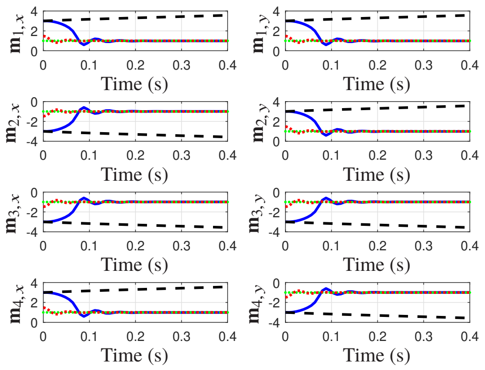
Case 1: L_2^2 vs. L_2^2+Quadratic Term, Four Desired Gaussian Mixtures

For Case 1, four swarm Gaussian mixtures are controlled to move towards the desired intensity at initial conditions farther away (square grid at (±3,±3)) and closer to (square grid at (±1.5,±1.5)) the desired intensity as shown by mean responses given by the black-dashed and red-dotted lines in Figure 2.2b respectively. From the trajectory snapshots given by Figure 2(a1), initial conditions that are far from the desired intensity do not have a converging control solution. From the surface visualization in Figure 2.1b, the general plane is flat in areas away from the desired and current states of the intensities. Therefore, optimization using Quasi-Newton MPC is more difficult in these flat areas and may not converge to a solution. If the current intensity is initialized much closer to the desired intensity as shown in Figure 2(a2), the flatness in the general plane is minimal, and the optimization step in Quasi-Newton MPC converges to a solution. By using the L_2^2 distance, converging control solutions can only be found for initial conditions and target destinations that are close. For the L_2^2 plus quadratic distance, four swarm Gaussian mixtures move towards the four desired Gaussian mixtures given by the blue-solid lines (mean responses) in Figure 2.2b. Figure 2.2c shows the trajectory snapshots and final states of each of the swarm Gaussian mixtures during the simulation. The target destinations are plotted as black x's. The red dots are the individual swarm agents that form the Gaussian mixture intensities. From the figure, all four mixtures converge to the desired mixtures in approximately 0.17 seconds and approximately 0.03 of steady-state error between the mixtures' position to the desired intensity. In comparison to only the L_2^2 distance, Figure 2.2b shows that for small distances between the initial state and the desired intensity, the L_2^2 distance is sufficient for state convergence, but as the distance increases, the L_2^2 distance diverges away. By adding the quadratic term to L_2^2, the optimization step can directly determine the minimum for the control solution shown in Figure 2.1c. Therefore, the desired intensity attracts the current swarm intensity at distances that fail for only L_2^2 distance given by Figure 2.2b.

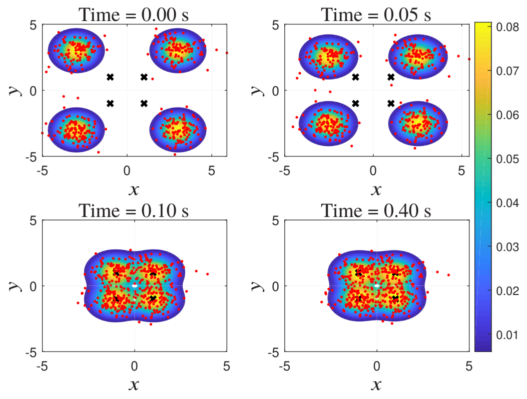
The L_2^2 plus quadratic distance is also extended to ILQR. Figure 2.3a shows the trajectory snapshots and final states of the simulation. All four Gaussian mixtures converge to the desired intensity in approximately 0.03 seconds and approximately 0.01 of steady-state error as shown in Figure 2.3b. In this figure, the x responses, y responses, and the desired intensity are given by blue, green, and red lines respectively. The entire simulation horizon is used to provide the prediction horizon for the ILQR trajectory. Even with a quadratic approximation of the objective function, ILQR is able to find control solutions that follow the L_2^2 plus quadratic characteristics that are presented using Quasi-Newton MPC.



(a) L_2^2 Distance Trajectory

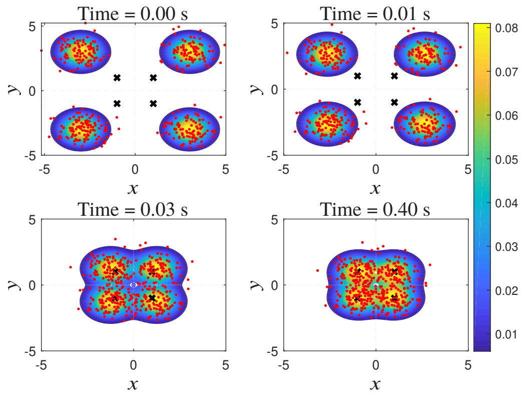


(b) L_2^2 and L_2^2+Quadratic Distance Position Time History

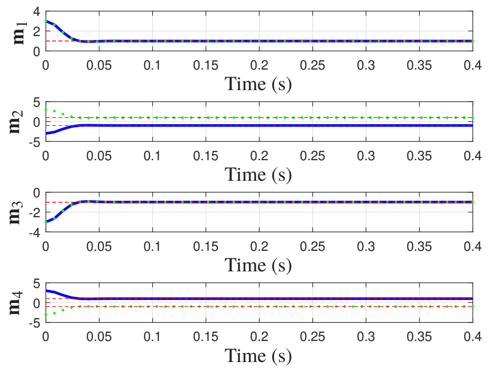


(c) L_2^2+Quadratic Distance Trajectory

Figure 2.2: Case 1: Figure 2.2a and Figure 2.2c show the controlled trajectories using the acceleration model and Quasi-Newton MPC. Figure 2.2b shows the intensity mean responses from the trajectories.



(a) L_2^2+ Quadratic Distance Trajectory

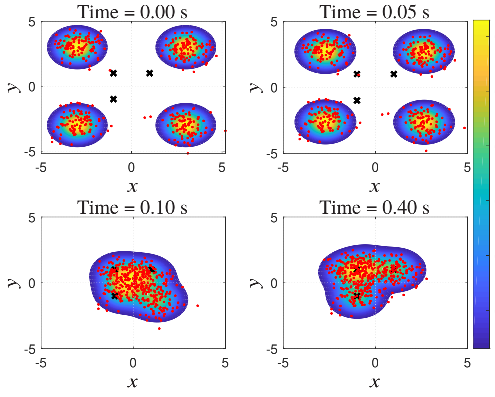


(b) L_2^2+Quadratic Distance Position Time History

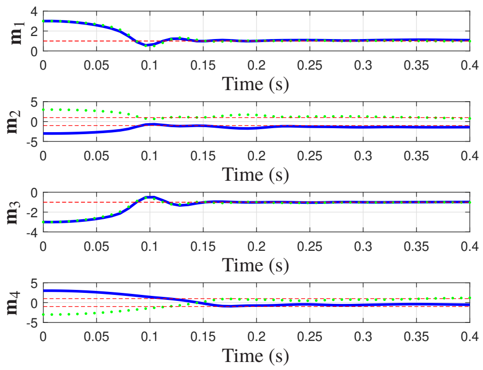
Figure 2.3: Case 1: 4 Gaussian mixture swarm controlled to four desired Gaussian mixtures via ILQR. Figure 2.3a shows the trajectories for the swarm and Figure 2.3b shows the position time history.

Case 2: Three Desired Gaussian Mixtures

Case 2 illustrates the effect of three desired Gaussian mixtures on the final trajectories of the four swarm Gaussian mixtures using Quasi-Newton MPC and ILQR. Using Quasi-Newton MPC, the current swarm intensity converges as given by the position time-history in Figure 2.4b. The trajectories for mixture 1 and mixture 3 reach their target, but mixtures 2 and 4 reach the third target with approximately 0.42 and 0.50 of steady-state error with 0.20 and 0.16 seconds of settling time respectively. From Figure 2.4a, it can be visually shown where the swarm intensity is located relative to the desired intensity at each time step. The results obtained follow directly from the RFS control theory using the L_2^2 plus quadratic distance term. By using this L_2^2 with a quadratic term in the objective function, the current intensity will attract towards the desired intensity while repulsing away from each other. This can be seen in the surface map shown in Figure 2.1c, where the hills are areas of repulsion and valleys, are areas of attraction. Thus, for Quasi-Newton MPC, mixtures 2 and 4 are



(a) L_2^2+ Quadratic Distance Trajectory



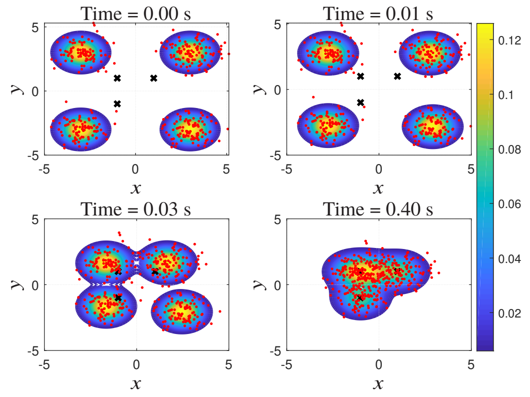
(b) L_2^2+Quadratic Distance Position Time History

Figure 2.4: Case 2: 4 Gaussian mixture swarm controlled to three desired Gaussian mixtures via Quasi-Newton MPC. Figure 2.4a shows the trajectories for the swarm and Figure 2.4b shows the position time history.

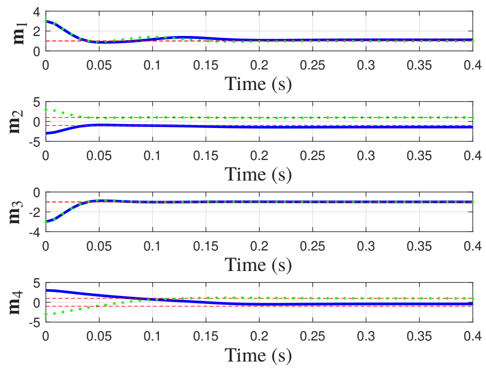
attracted to the same target, but they stay away from each other. This case is also extended to ILQR. Figures Figure 2.5a and Figure 2.5b show the trajectory snapshots and time-history of the same swarm using ILQR. As discussed previously, due to the approximation of the objective function, the mixtures 2 and 4 converged in 0.03 and 0.15 seconds with approximately 0.01 and 0.42 of steady-state error. By comparing the time-histories in Figure 2.5b and Figure 2.4b, the fourth intensity using ILQR follows very similarly to the MPC method. Therefore, there is a degree of accuracy in the approximation of the objective function to minimize for ILQR that allows the attraction of individual mixtures to the desired intensity while repulsing away from each other.

Case 3: Five Desired Gaussian Mixtures

Case 3 shows the effect of five desired Gaussian mixtures with the four swarm Gaussian mixtures using Quasi-Newton MPC and ILQR. Figure 2.6b shows the time his-

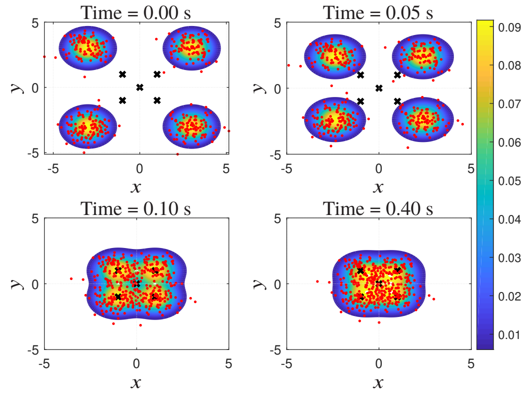


(a) L_2^2+ Quadratic Distance Trajectory

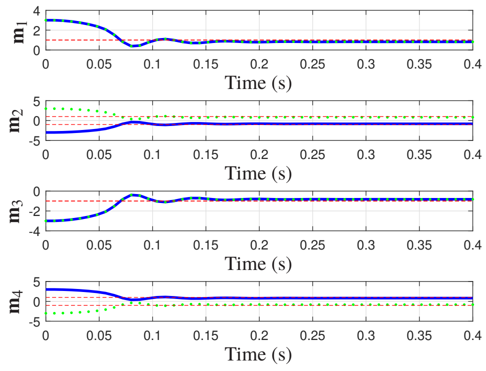


(b) L_2^2+Quadratic Distance Position Time History

Figure 2.5: Case 2: 4 Gausssian mixture swarm controlled to three desired Gaussian mixtures via ILQR. Figure 2.5a shows the trajectories for the swarm and Figure 2.5b shows the position time history.

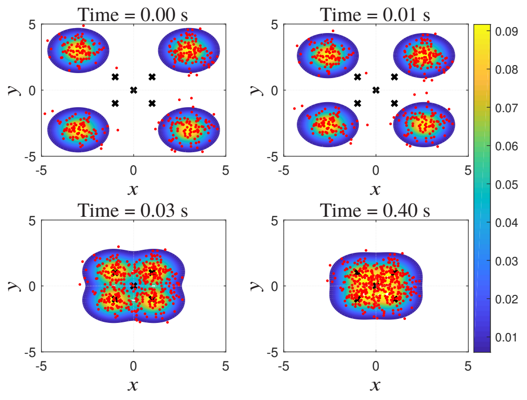


(a) L_2^2+ Quadratic Distance Trajectory

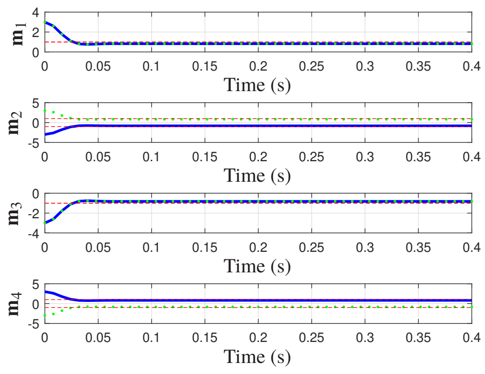


(b) L_2^2+Quadratic Distance Position Time History

Figure 2.6: Case 3: 4 Gaussian mixture swarm controlled to five desired Gaussian mixtures via Quasi-Newton MPC. Figure 2.6a shows the trajectories for the swarm and Figure 2.6b shows the position time history.



(a) L_2^2+ Quadratic Distance Trajectory



(b) L_2^2+Quadratic Distance Position Time History

Figure 2.7: Case 3: 4 Gaussian mixture swarm controlled to five desired Gaussian mixtures via ILQR. Figure 2.7a shows the trajectories for the swarm and Figure 2.7b shows the position time history.

tories for all the mixtures using Quasi-Newton MPC. The trajectory snapshots of the Gaussian mixtures are visually shown relative to the desired Gaussian mixtures in Figure 2.6a. From Figure 2.6b, the intensity converges in 0.19 seconds with a steady state error of approximately 0.17 which follow the theory as expected. Since the swarm Gaussian mixtures are far from each other, the effects of repulsion are minimal. Also, the mixtures are attracted to the four desired Gaussian mixtures that make up a square, but they are also attracted to the desired Gaussian mixture at the origin. This is due to the minimization of the objective function that has both an L_2^2 and a quadratic term where the individual mixtures will attract towards the desired intensity. Since there is an additional desired Gaussian mixture at the origin, all four swarm Gaussian mixtures are affected by the origin as they are moving towards the 4 square desired Gaussian mixtures. Thus, compared to Case 1 with only four desired Gaussian mixtures, the swarm intensity, in this case, will have a steady-state error due to the attraction to the additional desired Gaussian mixture. ILQR is also used to show how five desired Gaussian mixtures affect the quadratization of the L_2^2 plus quadratic objective function. Figures Figure 2.7a and Figure 2.7b show the trajectory snapshots and time-history respectively. The swarm converges in 0.03 seconds and 0.12 of steady-state error. This steady-state error shows the attraction of the desired Gaussian mixture at the origin which follows directly from results from the L_2^2 plus quadratic distance given by Figure 2.1c.

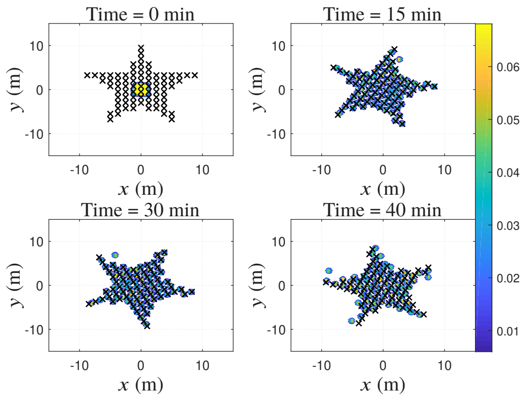
2.6.2 Clohessy-Wiltshire Relative Motion

Relative Motion with Perfect Information

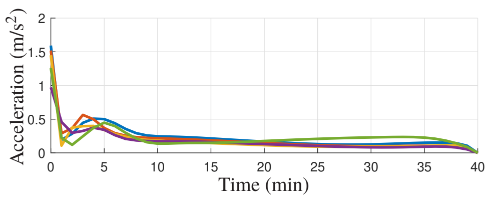
For the spacecraft relative motion, 77 Gaussian mixtures are birthed at the initial time from uniformly random initial conditions between -1 and 1 m from the chief satellite in a circular orbit. This is similar to the setup in (Eren et al., 2018). The goal is to control the spacecraft into a moving star-shaped pattern. Both the spacecraft and the rotating star pattern have an orbital frequency of $n = 0.00110678$ rad/s. It is assumed that the information received throughout the simulation is perfect. Figure 2.8a shows the trajectory snapshots of the spacecraft (contours) and the desire Gaussian mixtures (black x's) using ILQR and the L_2^2 plus quadratic distance. The Gaussian mixtures, represented by each contour, can be safely assumed to contain a single agent. As time progresses, the swarm intensities converge quickly into the formation and maintain the formation for the simulation time of 40 min. Figure 2.8b shows the acceleration for five spacecraft Gaussian mixtures to stay in the star formation. From these results, control using RFS can be expanded to physical spacecraft systems and can be used for moving targets.

Relative Motion with Imperfect Information

Next, the imperfect information (i.e. process, measurement, and clutter noise) is included in the simulation. In order to control with imperfect information, the GM-



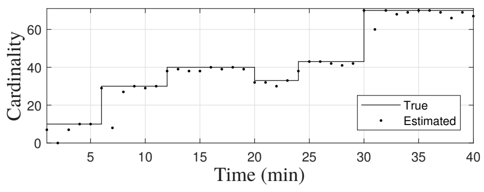
(a) Clohessy-Wiltshire Trajectory Snapshots



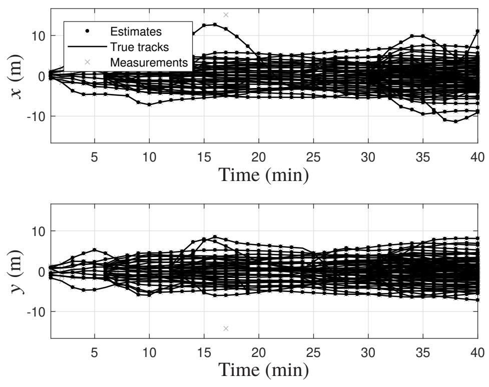
(b) Control Input

Figure 2.8: 77 Gaussian mixture spacecraft swarm controlled to a rotating star target via ILQR with perfect information. Figure 2.8a shows the trajectories and Figure 2.8b shows the acceleration for five spacecraft intensities.

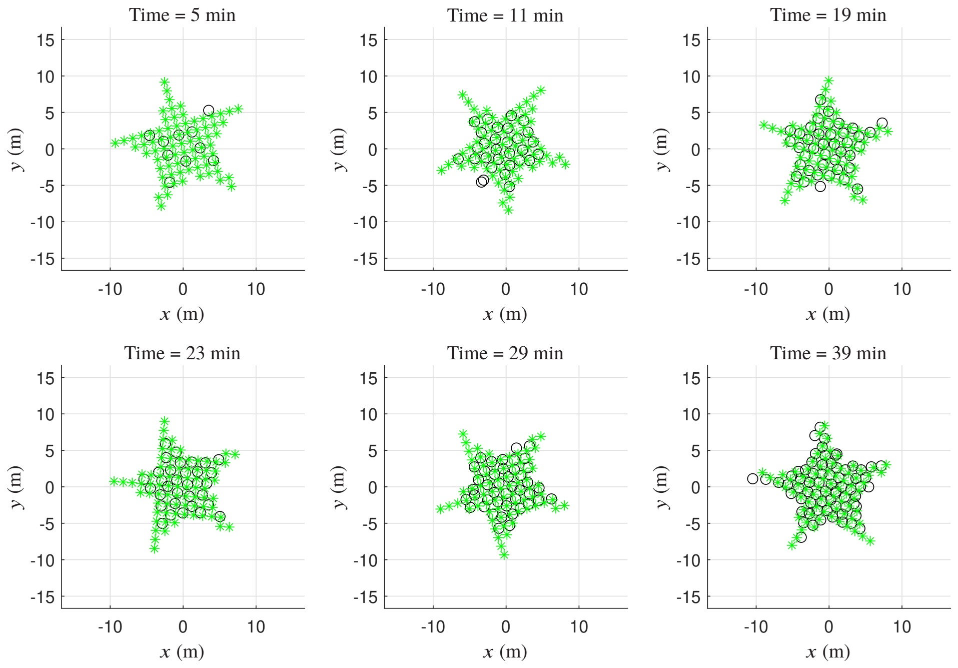
PHD filter is used in a feedback loop with the RFS control method. The GM-PHD filter determines the estimates of the intensities which is used for RFS control. The problem was altered to be more complicated by including differing birth and death times for the agents. With the addition of imperfect information and the added complication of changing number of agents, using the GM-PHD filter provides accurate estimates of the agents through time which allows for RFS control in the loop. Figure 2.9a shows the cardinality or number of agents in the swarm through time. The solid line is the true number of agents while the dotted line shows the estimate at each time-step. At each time-step, the agent estimates are fed through the RFS control using ILQR to obtain a control input for each agent. Then, the estimates are controlled and fed back to the GM-PHD filter at the next time-step. Figure 2.9c shows the snapshots of the controlled agents (black circles) and targets (green stars) at each time-step. Figure 2.9b shows the time history for the true agents (solid lines), estimated agents (black dots), and overall measurements (gray x's). From Figure 2.9a, as the true agents die or birth initially, estimates of the occurrence is accurate. As the number of agents increases, the estimates become less accurate. This is because the GM-PHD filter only uses the first-order statistical moment to propagate the cardinality information of agents (Mahler, 2007a). The cardinality distribution is unknown, and it is approximated as a Poisson distribution. For a Poisson distribution, the mean and variance are equal. Therefore, if there are a larger number of agents in the field, the corresponding variance of the cardinality distribution is also higher. Although the estimates are less accurate, the individual agents are controlled successfully into a star pattern in the presence of imperfect information. This is shown directly in Figure 2.9c and Figure 2.9b. As agents die or birth, the control input dies, or births with it, and due to the L_2^2 plus quadratic distance, agents are flexible to move into different parts of the formation.



(a) Cardinality



(b) Time History



(c) Clohessy-Wiltshire Trajectory Snapshots

Figure 2.9: 77-agent spacecraft swarm controlled to a rotating star target via ILQR with imperfect information. Figure 2.9a shows the true and estimated cardinality, Figure 2.9b shows the time history of the true tracks, the estimated tracks, and overall measurements, and Figure 2.9c shows the trajectories for the spacecraft agents and targets.

Chapter 3

Decentralized Control

The area of centralized swarm control may work well for small numbers of agents, but as the size of the swarm increases, various problems arise. Specifically, communication limitations, computational complexity due to the larger number of agents, and unknown environmental factors complicate the centralized control problem significantly (Bakule, 2008). Thus, it is necessary to break down the centralized control problem into smaller, more manageable subproblems which are weakly dependent or independent from each other. This becomes the area of decentralized or localized control. Decentralized control is able to control agents in a swarm by using different techniques on the swarm control (information) structure. Two different methods are of interest for decentralized control. The first area is the development of decentralized controllers under specific structural constraints (Fardad et al., 2009; Jovanovic, 2010; Fardad and Jovanović, 2011; Lin et al., 2011). An example of a structural constraint is sparsity requirements for an agent in the swarm which suggests that it only has access to the information structure from agents near it. The other area of interest is the development of decentralized control under communication constraints (delays). By adding delay and uncertainty into multi-agent systems, control can be degraded. Convex methods and optimal control have been tools used to develop decentralized systems that incorporate communication delays (Voulgaris et al., 2003; Bamieh and

Voulgaris, 2005).

In the original RFS control problem, the control (information) topology is assumed to be complete using all the state information obtained from the GM-PHD filter. This is centralized control in which the swarm computes the overall swarm control and manages the control execution for individual agents, allowing it to oversee the other agents' control processes.

For decentralized RFS control, the control topology is used in a localized or decentralized manner using sparse control matrices. The decentralized RFS control is realized using sparse LQR to sparsify the centralized RFS control gain matrix obtained using ILQR. This allows agents to use local information topology (information of agents near each other) or a fully decentralized topology (information of the agent's own information) to make a control decision. Sparse LQR allows for more stability and less performance degradation than truncating a centralized control matrix may provide. Sparsity and performance for decentralized RFS control are compared for different degrees of localization in the feedback control gains which show the viability for decentralized control for large collaborative swarms.

3.1 Decentralized Control Formulation

The framework for decentralizing RFS control for swarming agents is to design sparse control matrices using sparse LQR (Fardad et al., 2011; Lin et al., 2012, 2013). The following discussion on sparse LQR follows closely to Lin's work on sparse feedback gains (Lin et al., 2012).

3.1.1 Sparse LQR Problem

The continuous state-space representation of a linear time-invariant dynamical system with a structured control design is represented by

$$\dot{\mathbf{x}}(t) = A_c\mathbf{x}(t) + B_c\mathbf{u}(t) + B_{c2}\mathbf{d}(t), \tag{3.1a}$$

$$\mathbf{u}(t) = -F\mathbf{x}(t), \tag{3.1b}$$

where A_c is a continuous state transition matrix, B_c is a continuous control transition matrix, $\mathbf{d}(t)$ is a disturbance or external input for a time t, B_{c2} is the disturbance transition matrix, and F is a state feedback (control) gain dependent on the sparsity (structural) constraints $F \in \mathcal{S}$. A sparsity constraint subspace $\mathcal{S}$ is assumed to be non-empty for all sparsity patterns for controller gains that are stable. For an infinite horizon LQR, the total cost is quadratic in terms of the state and control given by

$$J(\mathbf{x}(t), \mathbf{u}(t)) = \int_0^\infty \mathbf{x}(t)^T Q\mathbf{x}(t) + \mathbf{u}(t)^T R\mathbf{u}(t), \tag{3.2}$$

where Q is a positive semi-definite state weight matrix and R is a positive definite control weight matrix. By plugging in Eq. (3.1) into Eq. (3.2) for control gain F (Levine and Athans, 1970), the optimal control problem with structural constraints becomes

$$\begin{aligned} &\min J(F) = \operatorname{trace}\left(B_{c2}^T \int_0^\infty \mathrm{e}^{(A-B_cF)^Tt}\left(Q + F^TRF\right)\mathrm{e}^{(A-B_cF)t}dtB_{c2}\right) \\ &\text{Subject to:} \quad F \in \mathcal{S}. \end{aligned} \tag{3.3}$$

The objective is to determine a control gain, $F \in \mathcal{S}$, that minimizes the LQR cost. Fortunately, the integral in Eq. (3.3) is bounded for stabilizing F, thus a control solution can be found using the Lyapunov equation given by

$$(A - B_cF)^TP + P(A - B_cF) = -(Q + F^TRF), \tag{3.4}$$

which reduces the $J(F)$ into

$$J(F) = \text{trace}\left(B_{c2}^TP(F)B_{c2}\right). \tag{3.5}$$

The control objective in Eq. (3.3) assumes the sparsity constraints are known before the optimization takes place, but these constraints may be unknown and appropriate sparsity patterns for decentralized control must be found. The optimization problem can be modified to provide a sparsity promoting optimal control solution which provides the performance and the topology for decentralized control. The sparsity-promoting optimal control problem is

$$\min J(F) + \gamma g_0(F), \tag{3.6a}$$

$$g_0(F) = \mathbf{nnz}(F), \tag{3.6b}$$

where $g_0(F)$ is the number of non-zeros ($\mathbf{nnz}(\cdot)$) for control gain F and $\gamma \geq 0$ is a scalar weight to penalize $g_0(F)$. By including the number of non-zeros in the control gain F into the control objective directly, sparsity in F is directly promoted in the optimization of the problem. More zeros (sparsity) in a control gain matrix corresponds to more localization in the information topology network. The weight γ follows similarly to how the Q and R matrices penalize $\mathbf{x}$ and $\mathbf{u}$, respectively, but γ penalizes the number of non-zeros in F. For example, when $\gamma >> 0$, the number of non-zeros in F is penalized heavily, thus, γ promotes more localized control. When $\gamma = 0$, no penalization of the control gain takes place, and a standard LQR solution with a centralized control gain matrix is found.

3.1.2 Sparsity-Promoting Optimal Control

The function $g_0(F)$ is a nonconvex argument in the optimization problem. As a result, finding the solution involves a brute-force search which becomes intractable. To circumvent this issue, the $g_0(F)$ function is substituted with the L_1 norm which is a nondifferentiable convex function given by

$$g_1(F) = ||F||_1 = \sum_{i,j} |F_{ij}|, \tag{3.7}$$

which gives higher costs to non-zeros elements in F with larger magnitudes (Boyd and Vandenberghe, 2004). This differs from $g_0(F)$ which gives the same cost to all non-zero elements. Therefore, the L_1 norm becomes a convex relaxation of the original problem, but the original $g_0(F)$ can be approximated better or recovered exactly by using a weighted L_1 norm given by

$$g_2(F) = \sum_{i,j} W_{ij}|F_{ij}|, \tag{3.8}$$

where W_{ij} are positive weights. The weights can be used to approximate the L_1 norm closer to $g_0(F)$, but if W_{ij} is chosen to be inversely proportional to $|F_{ij}|$ as given by

$$W_{ij} = \begin{cases} 1/|F_{ij}|, & \text{if } F_{ij} \neq 0, \\ \infty, & \text{if } F_{ij} = 0, \end{cases} \tag{3.9}$$

the weighted L_1 norm and $g_0(F)$ equate to

$$\sum_{i,j} W_{ij}|F_{ij}| = \mathbf{nnz}(F). \tag{3.10}$$

Although the weighted L_1 norm is viable to recover $g_0(F)$, the weights are dependent on the unknown feedback gain F. Therefore, an iterative algorithm, the alternating direction method of multipliers (ADMM), is used which trades off optimal performance, J, and sparsity, γ. First, initial centralized control gain, F, with $\gamma = 0$ is inputted into ADMM. Then, γ is increased and the ADMM iterative algorithm is used in conjunction with F and the previous γ to obtain a sparser F. Once the desired sparsity is found, the sparsity structure is fixed and the sparse control gain is found using the structured optimal control problem in Eq. (3.3). The method by which sparsity structures are identified using ADMM is explained in the next discussion.

3.1.3 Alternating Direction Method of Multipliers

The optimization problem in Eq. (3.6) can be rearranged into a constrained optimization problem

$$\begin{aligned}&\min J(F) + \gamma g(G),\\ &\text{Subject to:} \quad F - G = 0,\end{aligned} \tag{3.11}$$

where G decouples the sparsity cost separately from the performance cost. The equality constraint $F - G = 0$ makes Eq. (3.11) equivalent to Eq. (3.3). The associated augmented Lagrangian to the constrained optimization problem is

$$\mathcal{L}_\rho(F, G, \Lambda) = J(F) + \gamma g(G) + \text{trace}(\Lambda^T(F - G)) + \frac{\rho}{2}||F - G||_F^2, \tag{3.12}$$

where λ is the Langrange multiplier, $\rho > 0$ is scalar, and $|| \cdot ||_F$ is the Frobenius norm. By decoupling J and g, the structures for both J and g can be exploited using the ADMM algorithm optimization. The ADMM algorithm contains the F-minimization, G-minimization, and Lagrange multiplier steps in which F and G are minimized iteratively (Boyd et al., 2011). This is given by

$$F^{k+1} = \arg\min_F \mathcal{L}_\rho(F, G^k, \Lambda^k), \tag{3.13a}$$

$$G^{k+1} = \arg\min_G \mathcal{L}_\rho(F^{k+1}, G, \Lambda^k), \tag{3.13b}$$

$$\Lambda^{k+1} = \Lambda^k + \rho(F^{k+1} - G^{k+1}), \tag{3.13c}$$

and the convergence tolerance

$$||F^{k+1} - G^{k+1}||_F \leq \epsilon \quad \text{and} \quad ||G^{k+1} - G^k||_F \leq \epsilon. \tag{3.14}$$

The F-minimization and G-minimization alternate direction in terms of finding the optimal F and G, respectively, which gives ADMM its namesake. The Lagrange multiplier update steps with a size ρ which guarantees the feasibility of finding G^{k+1} and Λ^{k+1}.

For the sparsity-promoting optimization problem, ADMM provides benefits in the separability and differentiability of the sparsity cost and the performance cost. When calculating the performance cost using the control gain matrix, the matrix cannot be separated into individual elements to find optimal solutions. By separating optimization in the F-minimization and G-minimization steps, the G-minimization step can be decomposed into subproblems that involve individual elements (scalars) of the control gain matrix. Therefore, a optimal solution can be found analytically using either $g_0(F)$, $g_1(F)$, or $g_2(F)$. The other benefit to ADMM is differentiability. The performance cost is differentiable in terms of the control gain, but the sparsity cost is non-differentiable as discussed before. By separating the optimization problem in two steps, gradient descent algorithms can be used for the F-minimization step, and analytical solutions can be found for the G-minimization step. This is discussed next.

The F-Minimization Step Solution

The minimization of Eq. (3.13a) can use any descent method. Although gradient descent or Newton's methods can be used, the Anderson-Moore descent can converge faster than gradient descent and is simpler to implement than Newton's method (Makila and Toivonen, 1987). From the augmented Lagrangian in Eq. (3.12), an equivalent optimization problem can be obtained by completing the square given by

$$\begin{aligned} \min \phi(F) &= J(F) + (\rho/2)||F - U^k||_F^2, \\ U^k &= G^k - (1/\rho)\Lambda^k. \end{aligned} \tag{3.15}$$

Using methods developed in Levine and Athans (1970); Rautert and Sachs (1997), the necessary conditions for optimality are obtained as

$$(A - B_cF)L + L(A - B_cF)^T = -B_{c2}B_{c2}^T, \tag{3.16a}$$

$$(A - B_cF)^TP + P(A - B_cF) = -(Q + F^TRF), \tag{3.16b}$$

$$\nabla\phi(F) = 2RFL + \rho F - 2B_c^TPL - \rho U^k = 0, \tag{3.16c}$$

where Eqs. (3.16a) and (3.16b) are the controllability and observability grammians, respectively, and Eq. (3.16c) is the optimality condition for $\mathcal{L}_p$. Anderson-Moore iteratively solves for Eqs. (3.16a) and (3.16b) for L and P with a fixed F using the solution to the Lyapunov equations, and then solves F in Eq. (3.16c) with a fixed L and P using the solution to the Sylvester equation to obtain a new $\bar{F}$ (Makila and Toivonen, 1987; Lin et al., 2012). This consists of one iteration for the F-minimization step. To complete the F-minimization step, a descent direction, $\tilde{F} = \bar{F} - F$, is obtained to allow for convergence to a stationary point on ϕ. The stationary point ϕ is locally convex and provides a local minimum on ϕ. Note that step-size rules (i.e. determining s in $F + s\tilde{F}$ using the Armijo rule) can be used to guarantee convergence to the stationary point (Bertsekas, 2006).

The G-Minimization Step Solution

To find an analytical solution to the G-minimization in Eq. (3.13b), the first step is to complete the square of Eq. (3.12) with respect to G. This is given by

$$\begin{aligned} \min \phi(G) &= \gamma g(G) + (\rho/2)||G - V^k||_F^2, \\ V^k &= (1/\rho)\Lambda^k + F^{k+1}. \end{aligned} \tag{3.17}$$

This equation can be reduced into summation of element-wise components (scalars) by substituting the $g(\cdot)$ functions from Eqs. (3.6b), (3.7), or (3.8) and solving directly. The weighted L_1 , Eq. (3.8), is a general function for Eq. (3.7) when $W_{ij} = 1$ and Eq. (3.6b) when Eq. (3.9), so the objective can be reduced element-wise using a strictly convex Eq. (3.8) given by

$$\phi(G) = \sum_{i,j} \left(\gamma W_{ij} |G_{ij}| + (\rho/2)(G_{ij} - V_{ij}^k)^2 \right). \tag{3.18}$$

Thus, the minimization is

$$\min \phi_{ij}(G_{ij}) = \gamma W_{ij} |G_{ij}| + (\rho/2)(G_{ij} - V_{ij}^k)^2, \tag{3.19}$$

for each element in G. The unique solution to this problem is

$$G_{ij}^* = \begin{cases} V_{ij}^k - a, & V_{ij}^k \in (a, \infty), \\ 0, & V_{ij}^k \in (-a, a), \\ V_{ij}^k + a, & V_{ij}^k \in (-\infty, -a), \end{cases} \tag{3.20}$$

where $a = (\gamma/\rho)W_{ij}$ is a scalar. This equation is the shrinkage operator (Boyd et al., 2011), and it is the solution when Eq. (3.7) or (3.8) is substituted. The amount by which G^*_{ij} is minimized is the parameter a. If γ or W_{ij} is increased, the minimization becomes more forceful. This occurs similarly by reducing ρ. If Eq. (3.6b) is used, the G-minimization reduces to

$$\min \phi_{ij}(G_{ij}) = \gamma \mathbf{nnz}(G_{ij}) + (\rho/2)(G_{ij} - V^k_{ij})^2, \tag{3.21}$$

and has a unique solution given by

$$G^*_{ij} = \begin{cases} 0, & |V^k_{ij}| \le b, \\ V^k_{ij}, & |V^k_{ij}| > b, \end{cases} \tag{3.22}$$

where $b = \sqrt{2\gamma/\rho}$ is a scalar. This is the truncation operator (Lin et al., 2013). By using any of the $g(\cdot)$ functions, a unique solution for the optimization in Eq. (3.13b) can be found.

3.2 Application to RFS Control

The theory for using Sparse LQR for decentralized control is formulated in a continuous time representation given by Eqs. (3.1) and (3.2). Unfortunately, RFS control is formulated in discrete time with a zero-order hold on control. Therefore, a bridge between the two theories must be found. Previously, sparse feedback gains have been found in discrete time using non-convex sparsity-promoting penalty functions using sequential convex optimization (Fardad and Jovanović, 2014), but for this work, a less computationally intensive and theoretically extensive method is more useful. Work in discretizing the sparse LQR formulation has been made by high level discussion of using discrete Lyapunov and Sylvester equations, although no theory or algorithms have been presented (Verdoljak, 2016). Because discretization of sparse LQR has not been well documented, a bridge between the discrete RFS control and the continuous sparse LQR is developed instead.

The discrete state-space representation of a linear time-invariant system with a zero-order hold on control is represented by

$$\mathbf{x}_{k+1} = A_d\mathbf{x}_k + B_d\mathbf{u}_k, \tag{3.23a}$$

$$\mathbf{u}_k = -K\mathbf{x}_k, \tag{3.23b}$$

where A_d is the discrete state transition matrix, B_d is the discrete control transition matrix, and K is a discrete control gain. By substituting Eq. (3.23b) in to Eq. (3.23a) the discrete state-space equation can be reduced to

$$\begin{aligned}\mathbf{x}_{k+1} &= A_d\mathbf{x}_k - B_dK\mathbf{x}_k \\ &= (A_d - B_dK)\mathbf{x}_k \\ &= A_d^*\mathbf{x}_k,\end{aligned} \tag{3.24}$$

where $A_d^* = A_d - B_dK$. The conversion from a continuous-time to a discrete-time state-space is given by

$$A_d^* = \text{expm}(A_c^*dt), \tag{3.25}$$

where dt is the discrete time-step and $A_c^* = A_c - B_cF$ which follows a similar derivation from Eq. (3.1) where $B_{c2} = 0$ (DeCarlo, 1989). Converting from discrete-time to continuous-time is the inverse of Eq. (3.25) which uses the matrix logarithm given by

$$A_c^* = \text{logm}(A_d^*)/dt. \tag{3.26}$$

The discrete control gain K or the continuous control gain F can be obtained with

$$F = B_c^+(-A_c^* + A_c), \tag{3.27a}$$

$$K = B_d^+(-A_d^* + A_d), \tag{3.27b}$$

where $(\cdot)^+$ is the pseudoinverse. Therefore, the discrete control gain K can be converted to a continuous control gain F for sparse LQR and converted back to a decentralized control gain K. Note that the continuous control gain F is a continuous approximation of the discrete control gain K with a zero-order hold on control. That is, a discretization error results from approximating the continuous control gain F from K.

3.3 Results

Decentralized RFS control is implemented using the Clohessy-Wiltshire dynamics with different sparsity (γ) weights. Specifically, RFS control is implemented using the L_2^2 plus quadratic distance and ILQR. The dynamics model for agents within the swarm are decoupled from each other, but the distributional distance-based cost may have coupling between agents. Therefore, an RFS control gain that is found will be centralized due to coupling in the objective function. Then, the control gain matrix is decentralized by varying the γ parameter and using sparse LQR. Three cases with varying γ are implemented to show how changes in information topology affect performance of the agents in action.

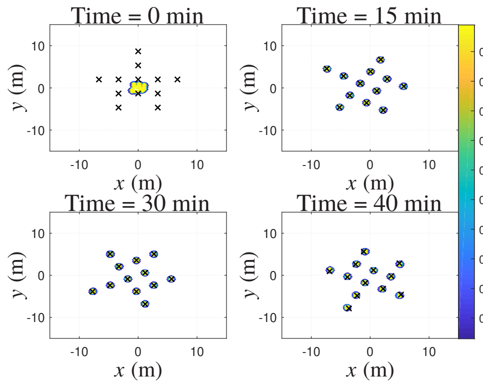
3.3.1 Case 1: Centralized Control

For Case 1, 12 swarm Gaussian mixtures are birthed at the initial time from uniformly random initial conditions between -1 and 1 m from a chief satellite in a circular orbit.

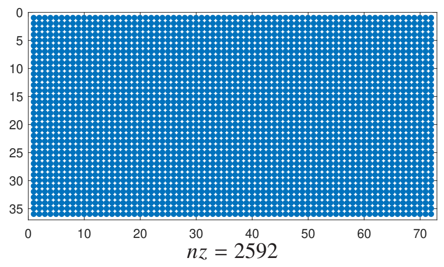
A $\gamma = 0$ is applied to the problem which provides no penalty in the sparsity-promoting objective. This setup is similar to the Relative Motion with Perfect Information in Section 2.6.2 but this example uses a 12 agent swarm instead. Again, each contour is safely assumed to be a single agent. Figure 3.1a shows the trajectory snapshots of the spacecraft (contours) and the desired Gaussian mixtures (black x's) using the aforementioned L_2^2 plus quadratic divergence and ILQR control. Through time, the swarm intensity converges quickly into the rotating star-shaped formation and maintains the formation for a duration of 40 min. Figure 3.1b shows the number of non-zeros in the control gain K. The control gain matrix of a single agent under Clohessy-Wiltshire dynamics is size 3×6. Therefore, the size for the control gain K of the entire 12 agent swarm is $(3 \cdot 12) \times (6 \cdot 12)$. With $\gamma = 0$, Figure 3.1b has no elements that are zero. Each sub-block that contains the 3×6 sized matrix is non-zero which totals to 2592 non-zero elements in K. Therefore, each agent in the swarm requires some control information from all the other agents in the field to take an action. Figure 3.4a shows the information graph between all the agents. Every agent in the field requires a signal to take an action, but the signals from agents further away from each other may provide a minimal control performance boost in terms of computational power needed. Thus, the control-gain is sparsified to reduce the complexity of the entire network to take an action.

3.3.2 Case 2: Localized Control

Case 2 illustrates the effect of promoting sparsity with a $\gamma = 10^{-19}$ for the same 12 agent problem. With $\gamma = 10^{-19}$, the number of non-zeros is penalized in the sparsity-promoting function in Eq. (3.6). Figure 3.2a shows the trajectory snapshots of the swarm using localized RFS control. Through time, the swarm intensity converges almost as quickly into the rotating star-shaped formation for the 40 min duration. Specifically from Table 3.1, there is only a reduction of 0.01% in performance in terms



(a) Trajectory Snapshots



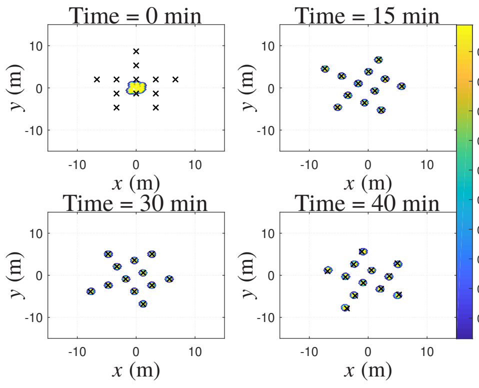
(b) Number of Non-Zeros in K

Figure 3.1: Trajectory and number of non-zeros of control gain K for the centralized RFS control case.

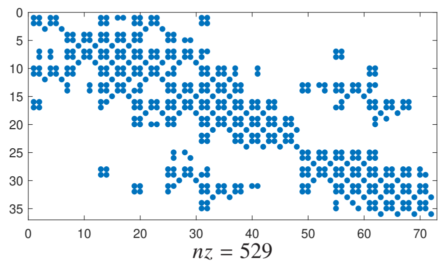
of the centralized performance, the J_c cost, due to localizing the control. Figure 3.2b shows the number of non-zeros in the control gain K. From the figure, the number of non-zeros is reduced to 529 which is 20.4% of the number of non-zeros from the centralized gain, K_c, case in Table 3.1. From Figure 3.2b and Figure 3.4b, the agents use the control information from agents local to it. As the control gain matrix becomes more localized, the number of non-zeros in K become increasingly diagonalized with a smaller spread. This is the inherent nature in decentralizing control using sparse LQR. The sparsity-promoting penalty function allows for reduction in the control information needed from individual agents to provide stable localized control with minimal effects on performance.

3.3.3 Case 3: Fully Decentralized Control

Case 3 shows the effect of promoting sparsity with a larger penalty, $\gamma = 1$, for the 12 agent problem. Figure 3.3a shows the trajectory snapshots of the swarm moving in a fully decentralized manner using RFS control. The swarm intensity converges into the rotating star-shaped formation for the 40 min duration. Performance-wise, there



(a) Trajectory Snapshots



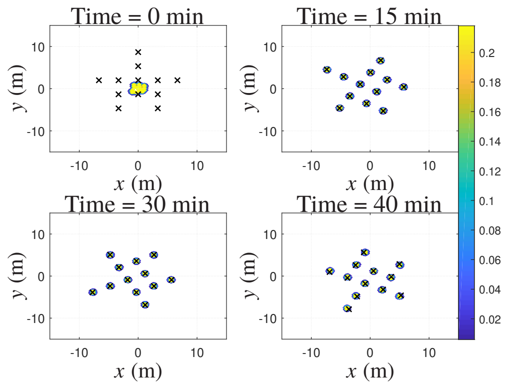
(b) Number of Non-Zeros in K

Figure 3.2: Trajectory and number of non-zeros of control gain K for the localized RFS control case.

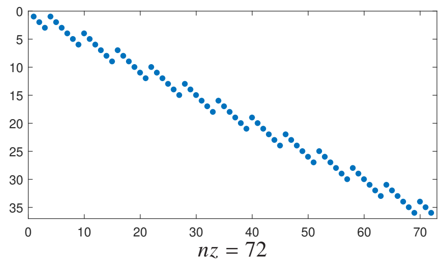
is only a 0.01% reduction in performance compared to $\gamma = 0$ example in Table 3.1. Figure 3.3b shows the number of non-zeros in the control gain K. The total number of non-zeros in K is 72 which is 2.8% of $\gamma = 0$ in Table 3.1. In this case, the 3×6 sub-matrices occur directly across the diagonal with no spread. No control information is collected from other agents in the swarm which is observed in Figure 3.4c. Increasing the γ weight penalizes the number of non-zeros in K which allows for more localized, and in this case, a fully decentralized RFS control.

Table 3.1: Sparsity vs. Performance for Swarm System

	Localized	Decentralized
$\mathbf{nnz}(K)/\mathbf{nnz}(K_c)$	20.4%	2.8%
$(J - J_c)/J_c$	0.01%	0.01%

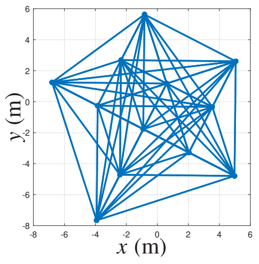


(a) Trajectory Snapshots

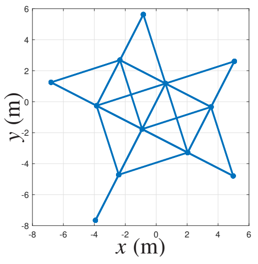


(b) Number of Non-Zeros in K

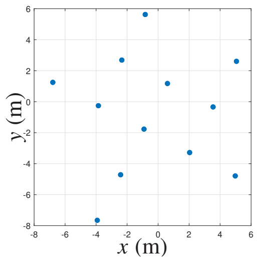
Figure 3.3: Trajectory and number of non-zeros of control gain K for the decentralized RFS control case.



(a) Centralized



(b) Localized



(c) Decentralized

Figure 3.4: Information graph of the 12 agent swarm for $\gamma = 0$, 10^{-19}, and 1 for Figure 3.4a, Figure 3.4b, and Figure 3.4c, respectively.

Chapter 4

Other Multi-Target Filters for RFS Control of Large Collaborative Swarms

As agents birth, spawn, or die in the swarm, the total number of agents may increase in the field. Although the GM-PHD filter can determine estimates from agents that birth, spawn, or die while avoiding explicit data associations between measurements and agents in a cluttered environment (Mahler, 2003; Goodman et al., 2013), the estimator performance decreases with an increase in cardinality in the swarm (Vo et al., 2006). The original PHD filter is an approximation of the intractable multi-agent Bayes filter (Mahler, 2003) in which the RFS intensity is propagated through time to avoid problems from data association. Although the PHD filter is still intractable to implement in a closed-form recursion (Pulford, 2005), methods that involve sequential Monte Carlo and Gaussian mixtures have been presented (Vo et al., 2005; Vo and Ma, 2006). Specifically, the GM-PHD filter has been implemented in control of large collaborative swarms (Doerr and Linares, 2018; Doerr et al., 2019), but the GM-PHD filter itself produces unreliable estimates when the cardinality becomes large.

In the GM-PHD filter, the RFS cardinality is assumed to be Poisson distributed. Thus, the mean and the variance are the same. The RFS cardinality mean is the total number of agents, so as the number of agents increase, the more varied the cardinality estimate becomes. To alleviate this problem, the Poisson distribution on the RFS cardinality is relaxed and the cardinality distribution itself is propagated (Mahler, 2006, 2007a). Then similarly to the GM-PHD filter, a closed-form recursion can be found using Gaussian mixtures. This Cardinalized Probability Hypothesis Density (CPHD) filter generalizes the GM-PHD filter by jointly propagating the cardinality distribution as well as the RFS intensity to produce reliable estimates at high cardinality.

Another problem with the GM-PHD filter is that it cannot label agents it estimates through time. The agents in the swarm are indistinguishable at each step in time. Thus, it makes it difficult to correspond an individual agent's trajectory with the estimated time-history data from the filter. The Generalized labeled Multi-Bernoulli (GLMB) filter alleviates this problem by incorporating labels into the RFS (Mahler, 2007b; Vo and Vo, 2013) on top of estimation that considers clutter, data association, births, and deaths. Thus, the GLMB is able to track individual trajectories through time. The GLMB filter also highlights the filter's exemption from the spooky effect, an event where the CPHD filter misses an agent trajectory and then shifts the weight of the undetected trajectory to other detected agents, leading to better performance of the GLMB filter over the CPHD filter (Franken et al., 2009).

For both the CPHD and the GLMB filter, the RFS estimate is propagated with the RFS control in feedback for the spacecraft relative motion problem. Specifically, the MPC-based ILQR is implemented to provide swarm control in a centralized manner. By using the CPHD and GLMB filters, the cardinality and state estimates become more accurate for RFS control for large collaborative swarms.

4.1 The Cardinalized Probability Hypothesis Density Filter for RFS Control

4.1.1 CPHD Filter Formulation

Instead of propagating the multi-agent posterior density through a multi-agent Bayes recursion, the CPHD filter propagates the intensity (or the first-order statistical moment of the RFS) and the probability distribution for the cardinality forward in time. The following discussion expands the CPHD filter estimation innovation (Vo et al., 2006) for RFS control.

The property for the intensity function, which is exactly the same as the GM-PHD filter, is given by

$$\mathbb{E}(|X \cap S|) = \int_S \nu(\mathbf{x}) d\mathbf{x}, \tag{4.1}$$

where the expected number of agents in the region S is the integral of the intensity function, $\nu(\mathbf{x})$. In other words, $\nu(\mathbf{x})$ is the probability of finding an agent per unit volume at $\mathbf{x}$. The integral gives the total mass or the expected number of agents of RFS X in a region S. The RFS is assumed to have agents that are independent and identically distributed and a general cardinality distribution. This is in contrast with the GM-PHD filter which assumes a cardinality distribution that is Poisson. For the CPHD filter, this is called a generalized Poisson RFS (Mahler, 2006, 2007a). To propagate the intensity, the following assumptions must hold. First, the birth and surviving RFS intensities are independent of one another. Secondly, the agents' motion and measurements are independent of each other. Lastly, the clutter intensity is a generalized Poisson RFS and independent from the measurement intensity. In this formulation, there is also no spawning. Therefore, between GM-PHD and the CPHD, the CPHD has no spawning and clutter is assumed to be generalized Poisson.

From these assumptions, a general CPHD recursion can be found.

The CPHD time update for the intensity, $v(\mathbf{x})$, and cardinality, $p(\mathbf{x})$, is

$$\bar{v}_k(\mathbf{x}) = b(\mathbf{x}) + \int p_s(\zeta) f(\mathbf{x}|\zeta) v(\zeta) d\zeta, \tag{4.2a}$$

$$\bar{p}_k(n) = \sum_{j=0}^{n} p_\Gamma(n-j)\Pi[v(\zeta), p(\zeta)](j), \tag{4.2b}$$

where $b(\mathbf{x})$ is the current birth intensity, $p_s(\zeta)$ is the probability of survival given the previous state ζ, $f(\mathbf{x}|\zeta)$ is the agent motion model given ζ, p_Γ is the cardinality distribution of births, $\bar{(\cdot)}$ denotes a value that has been time-updated, $n = N_{total(t)}$, and

$$\Pi[v,p](j) = \sum_{l=j}^{\infty} C_j^l \frac{\langle p_s, v\rangle^j \langle 1 - p_s, v\rangle^{l-j}}{\langle 1, v\rangle^l} p(l). \tag{4.3}$$

The function C_j^l is a binomial coefficient given by

$$C_j^l = \frac{l!}{j!(l-j)!}, \tag{4.4}$$

and $\langle \cdot, \cdot \rangle$ is the inner product between two real valued functions α and β given by

$$\langle \alpha, \beta \rangle = \int \alpha(\mathbf{x})\beta(\mathbf{x}) d\mathbf{x}. \tag{4.5}$$

The measurement update for the CPHD filter is

$$\begin{aligned} v_k(\mathbf{x}) = {} & (1 - p_d(\mathbf{x})) \frac{\Upsilon^1[\bar{v}_k(\mathbf{x}); Z_k]\bar{p}_k(n)}{\langle \Upsilon^0[\bar{v}_k(\mathbf{x}); Z_k], \bar{p}_k(n)\rangle} \bar{v}_k(\mathbf{x}) \\ & + \sum_{z \in Z_k} \psi(\mathbf{x})) \frac{\Upsilon^1[\bar{v}_k(\mathbf{x}); Z_k(z)]\bar{p}_k}{\langle \Upsilon^0[\bar{v}_k(\mathbf{x}); Z_k], \bar{p}_k\rangle} \bar{v}_k(\mathbf{x}), \end{aligned} \tag{4.6a}$$

$$p_k(n) = \frac{\Upsilon^0[\bar{v}_k(\mathbf{x}); Z_k](n)\bar{p}_k(n)}{\langle \Upsilon^0[\bar{v}_k(\mathbf{x}); Z_k], \bar{p}_k \rangle}, \tag{4.6b}$$

where $p_d(\mathbf{x})$ is the current agent detection probability, Z_k is the measurement RFS, and

$$\psi(\mathbf{x})) = \frac{\langle 1, \kappa_k \rangle}{\kappa_k(z)} g_k(z|\mathbf{x}) p_d(\mathbf{x}), \tag{4.7a}$$

$$\Upsilon^u_k[v; Z](n) = \sum_{j=0}^{\min(|Z|,n)} (|Z|-j)! p_\kappa(|Z|-j) P^n_{j+u} \frac{\langle 1-p_d, v \rangle^{n-(j+u)}}{\langle 1, v \rangle^n} e_j(\Xi_k(v, Z)). \tag{4.7b}$$

The κ_k and $g_k(z|\mathbf{x})$ in Eq. (4.7a) are the clutter intensity and the current measurement likelihood of a single agent, respectively. In Eq. (4.7b), p_κ is the clutter cardinality distribution, $\Xi_k(v, Z) = \{\langle v, \psi \rangle\} : z \in Z$, and P^n_{j+u} is the permutation coefficient in the form

$$P^n_{j+u} = \frac{n!}{(n-(j+u))!}. \tag{4.8}$$

The elementary symmetric function is defined about a finite set Z with the form

$$e_j(Z) = \sum_{S \subseteq Z, |S|=j} \prod_{\zeta \in S} \zeta. \tag{4.9}$$

Note by convention, $e_0(Z) = 1$ and Z is a finite set of real numbers. From both the time and measurement update equations, both the intensity and the cardinality distributions are propagated forward to obtain their posterior counterparts. This is in contrast with the original PHD filter which propagates the RFS through a single parameter. Therefore, the CPHD is still first order in terms of the multi-agent state, but it is higher order in the agent number (cardinality). Unfortunately, the CPHD filter is more complex than the PHD filter due to the cardinality propagation. Also, the CPHD filter follows the same problem with the PHD filter in which the recursion is intractable because the numerical integration suffers from higher computational time as the single agent state space X increases due to the increasing number of agents. But as with the PHD filter, a tractable closed-form solution can be found using a Gaussian mixture solution.

4.1.2 Gaussian Mixture CPHD Filter Closed Form Recursion

To obtain a closed-form solution to the CPHD filter recursion, the following assumptions are made. First, the multi-agent transition, $f(\mathbf{x}|\zeta)$, and likelihood, $g_k(z|\mathbf{x})$, density are Gaussian distributions given by

$$f(\mathbf{x}|\zeta) = \mathcal{N}(\mathbf{x}; A_k\zeta, Q_k), \tag{4.10a}$$

$$g_k(z|\mathbf{x}) = \mathcal{N}(z; H_k\mathbf{x}, R_k), \tag{4.10b}$$

where $\mathcal{N}(\cdot; \mathbf{m}, P)$ is a Gaussian distribution with m and P as the mean and covariance, repectively, A_k is the state transition matrix, Q_k is the process noise covariance, H_k is the observation matrix, and R_k is the measurement noise covariance. It is also assumed that the survival and detection probabilities are state independent (i.e. $p_s(\mathbf{x}) = p_s$ and $p_d(\mathbf{x}) = p_d$), and the birth intensity is a Gaussian mixture

$$\nu_{b,k}(\mathbf{x}) = \sum_{i=1}^{N_{b,k}} w_{b,k}^{(i)} \mathcal{N}(\mathbf{x}; \mathbf{m}_{b,k}^{(i)}, P_{b,k}^{(i)}), \tag{4.11}$$

where N_b is the number of birth multivariate Gaussian distributions and $w^{(i)}$ is the weight for the ith multivariate Gaussian distribution.

If the posterior intensity is assumed to be a Gaussian mixture at the previous time-step $k-1$ given by

$$\nu_{k-1}(\mathbf{x}) = \sum_{i=1}^{N_{f,k-1}} w_{k-1}^{(i)} \mathcal{N}(\mathbf{x}; \mathbf{m}_{k-1}^{(i)}, P_{k-1}^{(i)}), \tag{4.12}$$

the intensity is also a Gaussian mixture after the time-update which is

$$\bar{\nu}_k(\mathbf{x}) \triangleq \sum_{i=1}^{N_{f,k}} w_{f,k}^{(i)} \mathcal{N}\left(\mathbf{x}; \mathbf{m}_{f,k}^{(i)}, P_{f,k}^{(i)}\right) \triangleq \nu_{b,k}(\mathbf{x}) + \nu_{p_s,k}(\mathbf{x}). \tag{4.13a}$$

The birth Gaussian mixture is given by Eq. (4.11), and the surviving Gaussian mixture is given by

$$\nu_{p_s,k}(\mathbf{x}) = p_{s,k} \sum_{j=1}^{N_{s,k-1}} w_{k-1}^{(j)} \mathcal{N}(\mathbf{x}; \mathbf{m}_{p_s,k}^{(j)}, P_{p_s,k}^{(j)}), \tag{4.13b}$$

where $N_{s,k}$ is the number of surviving Gaussian distributions at a time-step k. The mean and covariance of the Gaussian mixtures are propagated through time simply using the dynamics for the swarm system given by

$$\mathbf{m}_{f,k+1}^{i} = A_k \mathbf{m}_{f,k}^{i} + B_k \mathbf{u}_{f,k}^{i}, \tag{4.13c}$$

$$P_{f,k+1}^{i} = A_k P_{f,k}^{i} A_k^T + Q_k. \tag{4.13d}$$

The predicted cardinality distribution is

$$\bar{p}_k(n) = \sum_{j=0}^{n} p_{\Gamma,k}(n-j) \sum_{l=j}^{\infty} C_j^l p_{k-1}(l) p_{s,k}^{(j)} (1-p_{s,k})^{l-j}. \tag{4.13e}$$

From the predicted intensity and cardinality in the time-update, the posterior intensity is determined using

$$\begin{aligned}\nu_k(\mathbf{x}) = f(\mathbf{x}) = (1-p_d(\mathbf{x}))\frac{\Upsilon^1[\mathbf{w}_{f,k};Z_k]\bar{p}_k}{\langle\Upsilon^0[\mathbf{w}_{f,k};Z_k],\bar{p}_k\rangle}\bar{\nu}_k(\mathbf{x}) \\ + \sum_{z\in Z_k}\sum_{j=1}^{N_{f,k}} w_k^{(j)}(z)\mathcal{N}(\mathbf{x};\mathbf{m}_k^{(j)}(z),P_k^{(j)}),\end{aligned} \tag{4.14a}$$

$$p_k(n) = \frac{\Upsilon^0[\mathbf{w}_{f,k};Z_k](n)\bar{p}_k(n)}{\langle\Upsilon^0[\mathbf{w}_{f,k};Z_k],\bar{p}_k\rangle}, \tag{4.14b}$$

where

$$\begin{aligned}\Psi_k^u[\mathbf{w},Z](n) = \sum_{j=0}^{\min(|Z|,n)}(|Z|-j)!p_{\kappa,k}(|Z|-j)P_{j+u}^n\frac{(1-p_{d,k})^{n-(j+u)}}{\langle 1,\mathbf{w}\rangle^{j+u}} \\ \times e_j(\Xi_k(\mathbf{w},Z)),\end{aligned} \tag{4.15a}$$

$$\Xi_k(\mathbf{w},Z) = \frac{\langle 1,\kappa_k\rangle}{\kappa_k(z)}p_{d,k}\mathbf{w}^T\mathbf{q}_k(z) : z\in Z, \tag{4.15b}$$

$$\mathbf{w}_{f,k} = \left[w_{f,k}^{(1)},\ldots,w_{f,k}^{(N_{f,k})}\right]^T, \tag{4.15c}$$

$$\mathbf{q}_k(z) = \left[q_k^{(1)}(z),\ldots,q_k^{(N_{f,k})}(z)\right]^T, \tag{4.15d}$$

$$q_k^{(j)}(z) = \mathcal{N}(z;H_k\mathbf{m}_{f,k}^{(j)},R_k+H_kP_{f,k}^{(j)}H_k^T), \tag{4.15e}$$

$$w_k^{(j)}(z) = p_{d,k}w_{f,k}^{(j)}q_k^{(j)}(z)\frac{\langle\Psi_k^1[\mathbf{w}_{f,k},Z_k(z)],\bar{p}_k\rangle}{\langle\Psi_k^0[\mathbf{w}_{f,k},Z_k],\bar{p}_k\rangle}, \tag{4.15f}$$

$$\mathbf{m}_k^{(j)}(z) = \mathbf{m}_{f,k}^{(j)} + K_k^{(j)}(z-H_k\mathbf{m}_{f,k}), \tag{4.15g}$$

$$P_k^{(j)} = [I-K_k^{(j)}H_k]P_{f,k}^{(j)}, \tag{4.15h}$$

$$K_k^{(j)} = P_{f,k}^{(j)}H_k^T(H_kP_{f,k}^{(j)}H_k^T+R_k)^{-1}. \tag{4.15i}$$

Therefore, Eqs. (4.13) and (4.15) provide a closed form solution to the CPHD filter recursion. It can be observed that these equations follow closely to the Kalman filter update equations (Kalman, 1960). Using the time and measurement update equations, the RFS control objective can be formed using Gaussian mixture intensity in which the means and covariances of the Gausssian mixture are propagated and controlled. Similarly to Doerr and Linares (2018); Doerr et al. (2019), the objective function is formed as

$$J(\mathbf{u}) = \sum_{k=1}^{T} \mathbf{u}_k^T R \mathbf{u}_k + D(\nu(\mathbf{x}, k), \nu_{des}(\mathbf{x}, k)), \tag{4.16}$$

where R is the positive definite control weight matrix, $\mathbf{u}_k$ is the control effort for the Gaussian mixture intensity shown in Eq. (4.13c), and $\nu_{des}(\mathbf{x}, k)$ is the desired formation given by

$$\nu_{des}(\mathbf{x}, k) \triangleq g(\mathbf{x}) \triangleq \sum_{i=1}^{N_g} w_g^{(i)} \mathcal{N}\left(\mathbf{x}; \mathbf{m}_g^i, P_g^i\right). \tag{4.17}$$

Both $\nu(\mathbf{x}, k)$ and $\nu_{des}(\mathbf{x}, k)$ are defined over the complete state space which include position and velocity parameters. $D(\cdot, \cdot)$ is the distance between Gaussian mixtures which has several closed-form solutions discussed in Doerr and Linares (2018); Doerr et al. (2019), and for this work, the same methodology is implemented to provide control using RFS theory. Specifically, an MPC based ILQR is implemented using the L_2^2 plus quadratic divergence to estimate and control the large collaborative swarm simultaneously.

4.2 The Generalized Labeled Multi-Bernoulli Filter for RFS Control

4.2.1 Notation

For this section, the following notation is used which is slightly different comparatively with the rest of the text. The state for a single agent is lowercase (i.e. x or $\mathbf{x}$), while RFSs are uppercase (i.e. X or $\mathbf{X}$). This is the same as with the rest of the text. The difference between this section and the rest of the text is that bold represents labeled states or RFSs (i.e. $\mathbf{x}$ or $\mathbf{X}$) instead of showing a vector. Spaces are represented as blackboard bold (i.e. $\mathbb{X}$ or $\mathbb{Y}$) and a finite subset of a space $\mathbb{X}$ is $\mathcal{X}$.

4.2.2 GLMB Filter Formulation

For the formulation of the GLMB filter, labeled RFSs are explored which follow closely to the discussion in Vo et al. (2014). As discussed in the previous chapter, a RFS is a set of random length that contains values that are random and un-ordered. Unfortunately, the identity of each value in the set cannot be determined as the RFS evolves through time. To mitigate this problem, labeled RFSs are introduced which incorporate the agent identity (label) into the RFS as it evolves through time. This label is chosen from a discrete countable space, $\mathbb{L} = \{\alpha_i : i \in \mathbb{N}\}$, where α_i is a distinct value in a $\mathbb{N}$ space of positive integers. A label $l \in \mathbb{L}$ is added to the state $x \in \mathbb{X}$ for each agent, therefore, the swarm is a finite set of $\mathbb{X} \times \mathbb{L}$. This provides each agent a label, but to track an agent through time, the labels must be distinct. Thus, a distinct label indicator, $\Delta(\mathbf{X})$, is introduced. If $\mathcal{L}(x, l) = l$ is the projection from $\mathcal{L} : \mathbb{X} \times \mathbb{L} \to \mathbb{L}$, the RFS $\mathbf{X}$ has distinct labels if and only if the cardinality for both the labels, $\mathcal{L}(\mathbf{X}) = \{\mathcal{L}(\mathbf{x}) : \mathbf{x} \in \mathbf{X}\}$, and $\mathbf{X}$ are the same. This is given by

$$\Delta(\mathbf{X}) \triangleq \delta_{|\mathbf{X}|}(|(\mathbf{X})|) = 1, \tag{4.18}$$

where $\delta_{|\mathbf{X}|}(|(\mathbf{X})|)$ is given by the form

$$\delta_Y(X) = \begin{cases} 1, & \text{if } X = Y, \\ 0, & \text{if } X \neq Y. \end{cases} \tag{4.19}$$

To obtain an unlabeled RFS, the labels are discarded. Thus, the cardinality distribution is the same for both the labeled and unlabeled variety of RFSs.

The labeled RFS is identified by the label $l = (k, i)$ where k is the birth of an agent at a discrete time and $i \in \mathbb{N}$ is an ascending index starting at one for agents birthing at the same time. This example is given by Figure 4.1. Initially, two agents birth at $k = 1$. Since multiple agents birth at the same time, an additional ascending index is included. Then another agent births at $k = 5$. From the three agents, the tracks can be determined due to their unique labels. Also to note is that the labels from $\mathbb{L}_{0:k-1}$ and $\mathbb{L}_{0:k}$ are disjoint. In the previous chapter, the PHD filter assumes that the RFS X can be propagated using the intensity function instead of using an intractable multi-agent Bayes recursion directly (Vo and Ma, 2006). With the PHD filter, the intensity is propagated with a Poisson distribution assumption on the cardinality. The cardinality distribution was generalized using the CPHD filter to obtain better estimates (Vo et al., 2006). For the GLMB filter, a solution can be found using the multi-agent Bayes recursion by using generalized labeled multi-Bernoulli distributions directly. The multi-agent Bayes filter is propagated using a

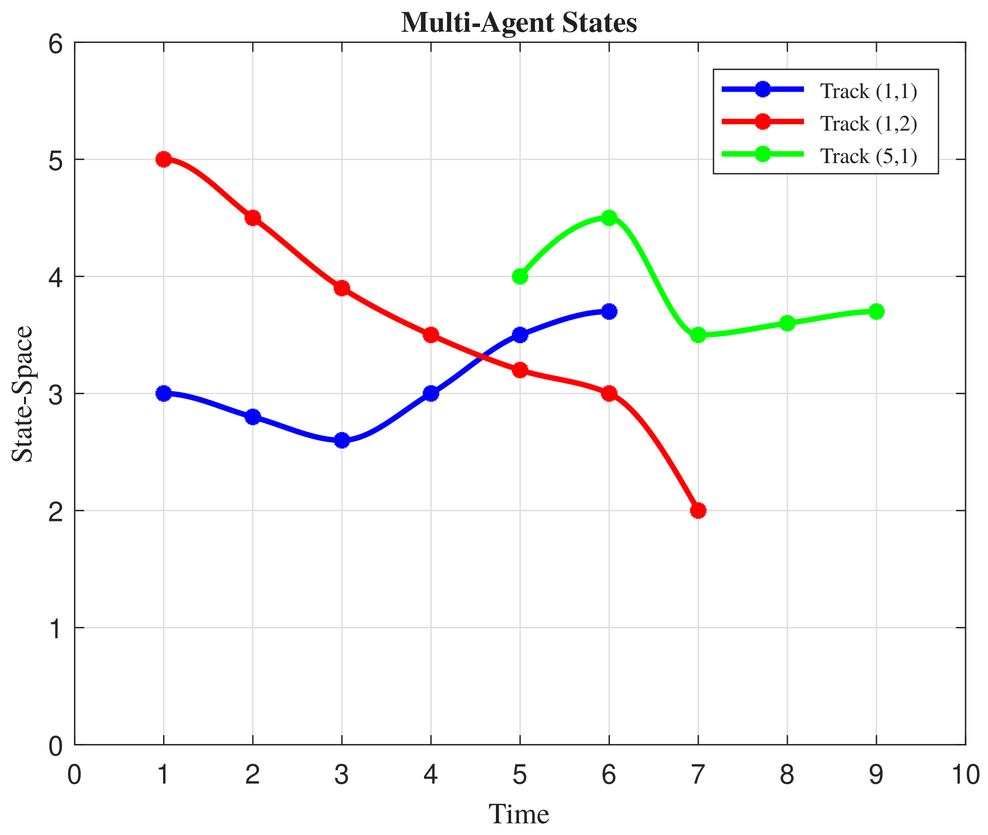


Figure 4.1: A time-history plot example that shows how individual agents are labeled. Two agents are birthed at $k = 1$ and are given unique labels. An additional agent births at $k = 5$.

time-update and measurement update given by

$$\pi_{k|k-1}(\mathbf{X}_k) = \int \mathbf{f}_{k|k-1}(\mathbf{X}_k|\mathbf{X}_{k-1})\pi_{k-1}(\mathbf{X}_{k-1}|Z_{k-1})\delta\mathbf{X}_{k-1}, \tag{4.20a}$$

$$\pi_k(\mathbf{X}_k|Z_k) = \frac{\pi_{k|k-1}(\mathbf{X}_k)g_k(Z_k|\mathbf{X}_k)}{\int \pi_{k|k-1}(\mathbf{X})g_k(Z_k|X)\delta\mathbf{X}}, \tag{4.20b}$$

in which the set integral has the form

$$\int \mathbf{f}(\mathbf{X})\delta\mathbf{X} = \sum_{i=0}^{\infty}\frac{1}{i!}\int \mathbf{f}(\mathbf{x}_1,\dots,\mathbf{x}_i)d(\mathbf{x}_1,\dots,\mathbf{x}_i). \tag{4.21}$$

A Bernoulli RFS X has the distribution

$$\pi(X) = \begin{cases} 1-r, & X = \emptyset, \\ rp(x), & X = \{x\}, \end{cases} \tag{4.22}$$

where a RFS exists in the state-space with a probability $r \in (0,1)$ and distributed with probability $p(x)$ over $\mathbb{X}$ or is empty with probability $1-r$. A multi-Bernoulli RFS is just the union of independent Bernoulli RFSs given by $X = \cup_{i=1}^{M} X^{(i)}$ where $i = 1,\dots,M$ is the number of Bernoulli RFSs. Again the parameters for each RFS, $X^{(i)}$, exists in the state-space with a probability $r^{(i)} \in (0,1)$ and a distribution $p(x)^{(i)}$. The probability density function for a multi-Bernoulli is

$$\pi(X = x_1,\dots,x_n) = \prod_{i=1}^{M}(1-r^{(i)}) \sum_{1\leq j_1 \neq \cdots \neq j_n \leq M} \prod_{i=1}^{n} \frac{r^{(j_i)}p^{(j_i)}(x_i)}{1-r^{(j_i)}}. \tag{4.23}$$

In order to simplify the notation of the GLMB recursion, for the rest of this chapter, the time-step k is omited (i.e. $\pi = \pi_k$, $\bar{\pi} = \pi_{k|k-1}$, $g = g_k$, and $\mathbf{f} = \mathbf{f}_{k|k-1}$).

First, the multi-agent motion model, $\mathbf{f}(\cdot)$, in Eq.(4.20) needs be found. From the previous time-step $k-1$ to the current time-step k, each agent has a survival probability of $p_s(x,l)$ and evolves through its dynamics $\mathbf{f}(x_k|x_{k-1},l_{k-1})\delta_{l_{k-1}}(l_k)$ or it dies with a probability $1-p_s(x,l)$. Also, birthed agents have the probability of

$$\mathbf{f}_b(\mathbf{Y}) = \Delta(\mathbf{Y})w_b(\mathcal{L}(\mathbf{Y}))[p_b]^{\mathbf{Y}}, \tag{4.24}$$

where the parameters w_b and p_b are provided parameters for the birth probability density $\mathbf{f}_b$ and $[p_b]^{\mathbf{Y}}$ has the form

$$[h]^X = \prod_{x\in X} h(x). \tag{4.25}$$

Note that h is a real-valued function where $[h]^{\emptyset} = 1$. For the birth probability density, note that $\mathbf{f}_b(\mathbf{Y}) = 0$ for any $\mathbf{y}$ with $\mathcal{L}(\mathbf{y}) \notin \mathbb{B}$. For propagation of the agents to the next time-step, it is assumed that surviving agents are independent from births and that agents evolve independently from each other. Therefore, the multi-agent model is

$$\mathbf{f}((\mathbf{X}_k|\mathbf{X}_{k-1})) = \mathbf{f}_{p_s}(\mathbf{X}_k \cap (\mathbb{X}\times\mathbb{L}))\mathbf{f}_b(\mathbf{X}_k - (\mathbb{X}\times\mathbb{L})), \tag{4.26}$$

where $\mathbb{L} = \mathbb{L}_{0:k-1}$ and

$$\mathbf{f}_{p_s}(\mathbf{W}|\mathbf{X}) = \Delta(\mathbf{W})\Delta(\mathbf{X})1_{\mathbf{L}(\mathbf{X})}(\mathcal{L}(\mathbf{W}))[\Phi(\mathbf{W};\cdot)]^{\mathbf{X}}, \tag{4.27a}$$

$$\Phi(\mathbf{W};x,l) = \begin{cases} p_s(x,l)f(x_k|x_{k-1},l_{k-1}), & \text{if } (x_k,l_{k-1}) \in \mathbf{W}, \\ 1-p_s(x,l), & \text{if } l \notin \mathcal{L}(\mathbf{W}). \end{cases} \tag{4.27b}$$

which is the superposition of the surviving and birthed agents. The inclusion function,

$1_{\mathbf{L}(\mathbf{X})}(\mathcal{L}(\mathbf{W}))$, has the form

$$1_Y(X) = \begin{cases} 1, & \text{if } X \subseteq Y, \\ 0, & \text{otherwise.} \end{cases} \tag{4.28}$$

From the multi-agent Bayes recursion in Eq. (4.20), the measurement likelihood function must also be found. Given an RFS $\mathbf{X}$ for a time-step k, each agent is either detected ($p_d(x,l)$) or not detected ($1-p_d(x,l)$). A measurement, z, is obtained with a likelihood ($g(z|x,l)$) if the agent is detected. Thus, a measurement RFS Z is obtained from all the detected agents and any clutter obtained defined by an intensity function κ. While conditioned on $\mathbf{X}$, by assuming that clutter and detections are independent from each other and detections are independent, the measurement likelihood is given by

$$g(Z|\mathbf{X}) = e^{-\langle \kappa, 1 \rangle} \kappa^Z \sum_{\theta \in \Theta(\mathcal{L}(\mathbf{X}))} [\psi_z(\cdot;\theta)]^{\mathbf{X}}, \tag{4.29}$$

where

$$\psi_Z(x,l;\theta) = \begin{cases} \frac{p_d(x,l) g(z_{\theta(l)}|z,l)}{\kappa(z_{\theta(l)})}, & \text{if } \theta(l) > 0, \\ 1 - p_d(x,l), & \text{if } \theta(l) = 0. \end{cases} \tag{4.30}$$

The inner product in Eq. (4.29) has the form

$$\langle f, g \rangle = \int f(x) g(x) dx. \tag{4.31}$$

The function $\theta(l)$ is an association map that represents how labeled tracks correspond to the measurements generated. For example, a track l corresponds to a measurement generated from $z_{\theta(l)} \in Z$.

Finally, the solution to the multi-agent Bayes filter can be found using the GLMB distribution given by

$$\pi(\mathbf{X}) = \Delta(\mathbf{X}) \sum_{\xi \in \Xi} w^{(\xi)}(\mathcal{L}(\mathbf{X}))[p^{(\xi)}]^{\mathbf{X}}, \tag{4.32}$$

where Ξ is a discrete index set, and the weights and probabilities follow

$$\sum_{L \subseteq \mathbb{L}} \sum_{\xi \in \mathbb{C}} w^{(\xi)}(\mathcal{L}(L)) = 1, \tag{4.33a}$$

$$\int p^{(\xi)}(x, l) dx = 1, \tag{4.33b}$$

with a discrete label space $\mathbb{L}$. The GLMB can be understood by a multi-agent exponential mixture. It includes a weight term that depends on the labels of the RFS state, $w^{(\xi)}(\mathcal{L}(\mathbf{X}))$, and it contains an RFS exponential which depends on the RFS state, $[p^{(\xi)}]^{\mathbf{X}}$. Note that an individual state and label pair (x, l) is not statistically independent in a GLMB. From the GLMB, the intensity function and cardinality distribution can be determined (Vo and Vo, 2013). The intensity function for the unlabeled GLMB is

$$v(x) = \sum_{\xi \in \Xi} \sum_{l \in \mathcal{L}} p^{(\xi)}(x, l) \sum_{L \subseteq \mathcal{L}} 1_L(l) w^{(\xi)}(L), \tag{4.34}$$

and the cardinality distribution is

$$p(n) = \sum_{L \in \mathcal{F}_n(\mathbb{L})} \sum_{\xi \in \Xi} w^{(\xi)}(L). \tag{4.35}$$

The Bayes filter recursion is closed under the GLMB (Vo and Vo, 2013), but the numerical implementation is unknown. But, a numerical implementation can be found using an alternate form called the $\delta-$GLMB. The probability density distribution using this alternate form is

$$\pi(\mathbf{X}) = \sum_{(I,\xi)\in\mathcal{F}(\mathbb{L})\times\Xi} w^{(I,\xi)}\delta_I(\mathcal{L}(\mathbf{X}))[p^{(\xi)}]^{\mathbf{X}}, \tag{4.36}$$

using the identity

$$w^{(\xi)}(J) = \sum_{I\in\mathcal{F}(\mathbb{L})} w^{(\xi)}(I)\delta_i(J), \tag{4.37}$$

since the components for summation are non-zero if and only if $I = J$. At a time-step, each $I \in \mathcal{F}(\mathbb{L})$ serves as a set of labeled tracks born, w^{ξ} is the weight of the hypothesis, ξ is the set of track labels, and $p(\cdot, l)$ is the probability density of a state in track $l \in I$. By allowing the $w^{(I,\xi)} = w^{(\xi)}(I)$, a numerical implementation can be found.

Assuming the density of $\delta-$GLMB at the previous time-step, $k-1$, is given by Eq. (4.36), the time-update for the multi-agent Bayes filter is

$$\bar{\pi}(\mathbf{X}_k) = \Delta(\mathbf{X}_k) \sum_{(I_k,\xi)\in\mathcal{F}(\mathbb{L}_k)\times\Xi} w_k^{(I_k,\xi)}\delta_{I_k}(\mathcal{L}(\mathbf{X}_k))[p_k^{(\xi)}]^{\mathbf{X}_k}, \tag{4.38a}$$

where

$$w_k^{(I_k,\xi)} = w_{p_s}^{(\xi)}(I_k \cap \mathbb{L})w_b(I_k \cap \mathbb{B}), \tag{4.38b}$$

$$w_{p_s}^{(\xi)}(L) = [\eta_{p_s}^{(\xi)}]^L \sum_{I\supseteq L}[1-\eta_{p_s}^{(\xi)}]^{I-L}w^{(I,\xi)}, \tag{4.38c}$$

$$\eta_{p_s}^{(\xi)}(l) = \langle p_s(\cdot,l), p^{(\xi)}(\cdot,l)\rangle, \tag{4.38d}$$

$$p_k^{(\xi)}(x,l) = 1_{\mathbb{L}}(l)p_s^{(\xi)}(x,l) + 1_{\mathbb{B}}(l)p_b(x,l), \tag{4.38e}$$

$$p_s^{(\xi)}(x,l) = \frac{\langle p_s(\cdot,l)f(x|\cdot,l), p^{(\xi)}(\cdot,l)\rangle}{\eta_{p_s}^{(\xi)}(l)}. \tag{4.38f}$$

The association history, ξ, is used for indexing while the label set I is used directly in the time-update. After the time-update, the posterior estimate can be determined by a measurement update using the multi-agent Bayes filter given by

$$\pi(\mathbf{X}_k|Z_k) = \Delta(\mathbf{X}_k) \sum_{(I,\xi)\in\mathcal{F}(\mathbb{L})\times\Xi} \sum_{\theta\in\Theta(I)} w^{(I,\xi,\theta)}(Z_k)\delta_I(\mathcal{L}(\mathbf{X}_k)))[p^{(\xi,\theta)}(\cdot|Z_k)]^{\mathbf{X}_k}, \tag{4.39a}$$

where the subset of association maps, $\Theta(I)$, has a domain on the labeled set I, and

$$w^{(I,\xi,\theta)}(Z) \propto w^{(I,\xi)}[\eta_Z^{(\xi,\theta)}]^I, \tag{4.39b}$$

$$\eta_Z^{(\xi,\theta)}(l) = \langle p^{(\xi)}(\cdot,l), \psi_Z(\cdot,l;\theta)\rangle, \tag{4.39c}$$

$$p^{(\xi,\theta)}(x,l|Z) = \frac{p^{(\xi)}(x,l)\psi_Z(x,l;\theta)}{\eta_Z^{(\xi,\theta)}(l)}. \tag{4.39d}$$

From both the time and measurement update, the $\delta-$GLMB recursion is parameterized by $\{w^{(I,\xi)}, p^{(\xi)} : (I,\xi \in \mathcal{F}(\mathbb{L}) \times \Xi\}$. Unfortunately, the number of hypotheses, $\{I^{(h)}, \xi^{(h)}, w^{(I^{(h)},\xi^{(h)})}, p^{(\xi^{(h)})}\}_{h=1}^{H}$ increases exponentially through time, so it is necessary to reduce the size of the parameter set by truncating (discarding) hypotheses that are unimportant (low weight) and keeping hypotheses that are high weight for propagation. Otherwise, if all the hypotheses are used, the GLMB recursion can become intractable. A solution to determining a tractable implementation of the GLMB filter is found by applying the K-shortest paths or ranked optimal assignment algorithms with a Gaussian mixture assumption.

4.2.3 Gaussian Mixture GLMB Filter Closed Form Recursion

Time Update using K-Shortest Paths

A tractable implementation of the $\delta-$GLMB recursion can be found by truncating unimportant hypotheses from the recursion at each time-step. The time-update in Eq. (4.38a) becomes more computationally intensive from summing all the supersets of L using Eq. (4.38c). From Vo and Vo (2013), Eq. (4.38a) can be expressed as

$$\begin{aligned}\bar{\pi}(\mathbf{X}_k) = \Delta(\mathbf{X}_k) &\sum_{(I,\xi)\in\mathcal{F}(\mathbb{L})\times\Xi} w^{(I,\xi)} \sum_{J\in\mathcal{F}(I)} [\eta_{p_s}^{(\xi)}]^J [1-\eta_{p_s}^{(\xi)}]^{I-J} \\ &\times \left(\sum_{L\in\mathcal{F}(\mathbb{B})} w_b(L)\delta_{J\cup L}(\mathcal{L}(\mathbf{X}_k))[p_k^{(\xi)}]^{\mathbf{X}_k} \right).\end{aligned} \tag{4.40}$$

From this equation, the time-updated hypothesis, $(J\cup L,\xi) : J \subseteq I, L \subseteq \mathbb{B}$, with weights $w_{p_s}^{(I,\xi)}(J)w_b(L)$, is generated from the previous hypothesis, (I,ξ), with weight $w^{(I,\xi)}$. The weight $w_{p_s}^{(I,\xi)}(J)$ has the form

$$w_{p_s}^{(I,\xi)}(J) = w^{(I,\xi)}[\eta_{p_s}^{(\xi)}]^J[1-\eta_{p_s}^{(\xi)}]^{I-J}. \tag{4.41}$$

The predicted $J\cup L$ is a label set that contains the union (in which the label sets are disjoint) between the surviving label set J and the birth label set L with weights $w_{p_s}^{(I,\xi)}(J)$ and $w_b(L)$, respectively. Therefore, the space for birth labels, $\mathbb{B}$, does not contain any surviving labels. The double sum over J and L can be truncated by separately truncating the sums individually since $J\cup L$ has a weight that is a product, $w_{p_s}^{(I,\xi)}(J)w_b(L)$.

The surviving label set, $J \subseteq I$ in Eq. (4.41) can be reworked as

$$w_{p_s}^{(I,\xi)}(J) = w^{(I,\xi)}[1 - \eta_{p_s}^{(\xi)}]^I \left[\frac{\eta_{p_s}^{(\xi)}}{1 - \eta_{p_s}^{(\xi)}}\right]^J, \tag{4.42}$$

for a given hypothesis I, ξ. The goal is to generate surviving labels in a non-increasing order of the term $\left[\frac{\eta_{p_s}^{(\xi)}}{1-\eta_{p_s}^{(\xi)}}\right]^J$, thus the largest weighted survival set are determined without computing all the hypothesis weights. By using the K-shortest path algorithm, the largest weights can be found and truncation can occur to reduce computations. A cost vector given by

$$C^{(I,\xi)} = [C^{(I,\xi)}(l_1), \ldots, C^{(I,\xi)}(l_{|L|})], \tag{4.43}$$

is defined where $|I|$ is the number of labels and the cost of the individual node $l_j \in I$ is

$$C^{(I,\xi)}(l_j) = -\ln\left[\frac{\eta_{p_s}^{(\xi)}(l_j)}{1 - \eta_{p_s}^{(\xi)}(l_j)}\right]. \tag{4.44}$$

The cost vector is ordered in decreasing order, and the distance between l_j and $l_{j'}$ is

$$d(l_j, l_{j'}) = \begin{cases} C^{(I,\xi)}(l_{j'}), & \text{if } j' > j, \\ \infty, & \text{else.} \end{cases} \tag{4.45}$$

The total path distance (total cost) between the start (PS) and the end (PE) that moves through the node set $J \subseteq I$ is

$$\begin{aligned}\sum_{l \in J} C^{(I,\xi)}(l) &= -\sum_{l \in J} \ln \left[\frac{\eta_{p_s}^{(\xi)}(l)}{1-\eta_{p_s}^{(\xi)}(l)} \right] \\ &= -\ln \left[\left(\frac{\eta_{p_s}^{(\xi)}(l)}{1-\eta_{p_s}^{(\xi)}(l)} \right)^J \right].\end{aligned} \tag{4.46}$$

The problem can also be defined similarly with agent births. By defining births as a labeled multi-Bernoulli model given by

$$w_b(L) = \prod_{l \in \mathbb{B}} (1 - r_b^{(l)}) \prod_{l \in L} \frac{1_{\mathbb{B}}(l) r_b^{(l)}}{1 - r_b^{(l)}}, \tag{4.47a}$$

$$p_b(x,l) = p_b^{(l)}(x), \tag{4.47b}$$

the K-shortest path cost vector is $C_b = [C_b(l_1), \ldots, C_b(l_{|\mathbb{B}|})]$ with a node cost about l_j is

$$C_b(l_j) = -\ln \left[\frac{r_b^{(l_j)}}{1 - r_b^{(l_j)}} \right], \tag{4.48}$$

where $|\mathbb{B}|$ is the number of births. This provides sets of $\mathbb{B}$ with the largest birth weights or lowest path costs. The shortest path between PS and PE is $J^* \subseteq I$ which gives the largest $\left(\frac{\eta_{p_s}^{(\xi)}(l)}{1-\eta_{p_s}^{(\xi)}(l)} \right)^{J^*}$ and the shortest distance $\sum_{l \in J^*} C^{(I,xi)}(l)$. Specifically, an enumeration of the shortest distance is found in non-decreasing order which corresponds to the enumeration of surviving label set $J \subseteq I$ with non-increasing weights. Note that when comparing the survival weights with the birth weights, the birth components are much smaller and may be discarded by the filter. To mitigate this problem, a larger number of birth components may be necessary to retain the birth hypotheses. The K-shortest path is a known solution to the combinatorial problem given the start and end of a weighted network (Eppstein, 1998). Specifically, the Bellman-Ford algorithm is used since the nodes are negative (Christofides et al., 1981).

The cost function for the K-shortest path for the time-update can be computed by assuming a Gaussian mixture for the multi-agent model. Thus, the survival probabilities are state independent (i.e. $p_s(x,l) = P_s$), and the multi-agent transition, $f(x_k|x,l) = \mathcal{N}(x_k; Ax, Q)$, is Gaussian distributed where A is the state transition matrix, Q is the process noise covariance, and the single agent state x transitions to x_k at the next time-step. The single agent density, $p^{(\xi)}(\cdot,l)$, is a Gaussian mixture given by

$$p^{(\xi)}(\cdot,l) = \sum_{i=1}^{J^{(\xi)}(l)} w_i^{(\xi)}(l)\mathcal{N}(x; m_i^{(\xi)}, P_i^{(\xi)}(l)), \tag{4.49}$$

where $J^{(\xi)}(l)$ is the number of tracks, and $m_i^{(\xi)}$ and $P_i^{(\xi)}(l)$ are the mean and covariance of the track. Also, the birth density, $p_b^{(l)}(x)$, is a Gaussian mixture with a similar form to Eq. (4.49). From this assumption, the time-update parameters from Eq. (4.38d) and (4.38e) become

$$\eta_{p_s}^{(\xi)}(l) = p_s, \tag{4.50a}$$

$$p_k^{(\xi)}(x,l) = 1_{\mathbb{L}}(l) \sum_{i=1}^{J^{(\xi)}(l)} w_i^{(\xi)}(l)\mathcal{N}(x; m_{p_s,i}^{(\xi)}(l), P_{p_s,i}^{(\xi)}(l)) + 1_{\mathbb{B}}(l)p_b^{(l)}(x), \tag{4.50b}$$

where

$$m_{p_s,i}^{(\xi)}(l) = Am_i^{(\xi)}(l) + Bu_i^{(\xi)}(l), \tag{4.51a}$$

$$P_{p_s,i}^{(\xi)}(l) = Q + FP_i^{(\xi)}(l)F^T. \tag{4.51b}$$

The term B is the control transition matrix and $u_i^{(\xi)}(l)$ is the control input for the track. Note if the time-update parameters are dependent on label l, the survival probability is $p_s = p_s(l)$, the state transition matrix is $F = F(l)$, and the process noise matrix is $Q = Q(l)$. From these equations, truncation using K-shortest paths is solved by evaluating Eq. (4.44) and the time-update is determined with a Gaussian mixture assumption.

Measurement Update using Rank Optimal Assignment

In the measurement weight update given by Eq. (4.39b), every hypothesis, from the time-update with weight $w^{(I,\xi)}$, is updated to form a set of hypotheses with weight $w^{(I,\xi,\theta)}(Z)$. If a set of association maps, $\theta \in \Theta(I)$, is produced in descending order of $[\eta_Z^{(\xi,\theta)}]^I$, for a given (I,ξ), the largest set of weights is chosen without solving for every single new hypothesis. Through an optimal assignment problem, the unimportant hypotheses can be truncated.

An assignment matrix S, size $|I| \times |Z|$, can be represented by the number of measurements, $|Z|$, and the number of labels, $|I|$, which consists of binary entries that add up to 1 for both rows and columns. For example, by letting $i \in \{1, \ldots, |I|\}$ and $j \in \{1, \ldots, |Z|\}$, $S_{i,j} = 1$ if the jth measurement and l_i track correspond, that is, $\theta(l_i) = j$. If row i is all zero, the track with l_i is not detected. If the column j is all zero, the measurement z_j is false. A cost matrix can be formed for the S matrix given by

$$C_Z^{(I,\xi)} = \begin{bmatrix} C_{1,1} & \cdots & C_{1,|Z|} \\ \vdots & \ddots & \vdots \\ C_{|I|,1} & \cdots & C_{|I|,|Z|} \end{bmatrix}, \tag{4.52}$$

where the individual cost of the assignment between z_j and l_i is

$$C_{i,j} = -\ln\left(\frac{\langle p^{(\xi)}(\cdot, l_i), p_d(\cdot, l_i) g(z_j|\cdot, l_i)\rangle}{\langle p^{(\xi)}(\cdot, l_i), 1 - p_d(\cdot, l_i)\rangle \kappa(z_j)}\right). \tag{4.53}$$

The total cost of the assignment is given by the Frobenius inner product given by

$$\text{tr}(S^T C_Z^{(I,\xi)}) = \sum_{i=1}^{|I|}\sum_{j=1}^{|Z|} C_{i,j} S_{i,j}. \tag{4.54}$$

For the updated measurement weight given by Eq. (4.39b), the cost of the association map, θ, can be included in $[\eta_Z^{(\xi,\theta)}]^I$ by

$$[\eta_Z^{(\xi,\theta)}]^I = \exp\left(-\text{tr}(S^T C_Z^{(I,\xi)})\right), \tag{4.55}$$

with the substitution of Eq. (4.30) and (4.54). The goal is to determine the optimal assignment matrix S^* which minimizes $\text{tr}(S^T C_Z^{(I,\xi)})$. Specifically, an enumeration of the lowest cost assignment matrices are found in non-decreasing order which corresponds to the enumeration of the association map, θ in an order with non-increasing $[\eta_Z^{(\xi,\theta)}]^I$ (Murthy, 1968).

The cost function for the optimal assignment problem for the measurement update can be computed directly using a Gaussian mixture assumption for the model. For the measurement update, the Gaussian mixture detection probabilities are state independent (i.e. $p_d(x,l) = p_d$), and the measurement likelihood function is Gaussian distributed (i.e. $g(z|x,l) = \mathcal{N}(z; Hx, R)$) where the Gaussian $\mathcal{N}(\cdot; m, P)$ has a mean m, and a covariance P. The quantities H and R are the observation matrix and the measurement noise covariance, respectively. Each agent has a Gaussian mixture density given by Eq. (4.49). From this assumption, the cost function is

$$C_{i,j} = -\ln\left(\frac{p_d \sum_{k=1}^{J^{(\xi)}(l_i)} w_k^{(\xi)}(l_i) q_k^{(\xi)}(z_j; l_i)}{(1-p_d)\kappa(z_j)}\right), \tag{4.56}$$

and Eqs. (4.39c) and (4.39d) become

$$\eta_Z^{(\xi,\theta)}(l) = \sum_{i=1}^{J^{(\xi)}(l)} w_{Z,i}^{(\xi,\theta)}(l), \tag{4.57a}$$

$$p^{(\xi,\theta)}(x,l|Z) = \sum_{i=1}^{J^{(\xi)}(l)} \frac{w_{Z,i}^{(\xi,\theta)}(l)}{\eta_Z^{(\xi,\theta)}(l)} \mathcal{N}(x; m_{Z,i}^{(\xi,\theta)}, P_i^{(\xi,\theta)}(l)), \tag{4.57b}$$

where

$$w_{Z,i}^{(\xi,\theta)}(l) = w_i^{(\xi)}(l) \begin{cases} \frac{p_d q_i^{(\xi)}(z_{\theta(l)};l)}{\kappa(z_{\theta(l)})}, & \text{if } \theta(l) > 0, \\ (1-p_d), & \text{if } \theta(l) = 0, \end{cases} \tag{4.57c}$$

$$q_i^{(\xi)}(Z;l) = \mathcal{N}(z; Hm_i^{(\xi)}(l), HP_i^{(\xi)}(l)H^T + R), \tag{4.57d}$$

$$m_{Z,i}^{(\xi,\theta)}(l) = \begin{cases} m_i^{\xi}(l) + K_i^{(\xi,\theta)}(l)(z_{\theta(l)} - Hm_i^{(\xi)}(l)), & \text{if } \theta(l) > 0, \\ m_i^{\xi}(l), & \text{if } \theta(l) = 0, \end{cases} \tag{4.57e}$$

$$P_i^{(\xi,\theta)}(l) = [I - K_i^{(\xi,\theta)}(l)H]P_i^{(\xi)}(l), \tag{4.57f}$$

$$K_i^{(\xi,\theta)}(l) = \begin{cases} P_i^{(\xi)}(l)H^T[HP_i^{(\xi)}(l)H^T + R]^{-1}, & \text{if } \theta(l) > 0, \\ 0, & \text{if } \theta(l) = 0. \end{cases} \tag{4.57g}$$

Note if the measurement update parameters are dependent on the label, l, the probability of detection is $p_d = p_d(l)$, the observation matrix is $H = H(l)$, and the measurement noise covariance is $R = R(l)$. From these equations, the truncation using rank optimal assignment is solved and the measurement update is determined using the Gaussian mixture assumption.

Truncation of Hypotheses

Truncating for both the time and measurement update is very similar although two different optimization methods (K-shortest paths and ranked optimal assignment) are used. For hypotheses, H, the time update is

$$\bar{\pi}^{(h)}(\mathbf{X}_k) = \sum_{h=1}^{H} \bar{\pi}^{(h)}(\mathbf{X}_k), \tag{4.58a}$$

where

$$\bar{\pi}^{(h)}(\mathbf{X}_k) = \sum_{J \subseteq I^{(h)}} \sum_{L \subseteq \mathbb{B}} w_{p_s}^{(I^{(h)},\xi^{(\xi)})}(J) w_b(L) \delta_{J \cup L}(\mathcal{L}(\mathbf{X}_k)) \left[p_k^{(\xi^{(h)})}\right]^{\mathbf{X}_k}, \tag{4.58b}$$

and the measurement update is

$$\pi(\mathbf{X}_k|Z_k) = \sum_{h=1}^{H} \pi^{(h)}(\mathbf{X}_k|Z_k), \tag{4.59a}$$

where

$$\pi^{(h)}(\mathbf{X}_k|Z_k) = \Delta(\mathbf{X}_k) \sum_{j=1}^{|\Theta(I^{(h)})|} w^{(h,j)} \delta_{I^{(h)}}(\mathcal{L}(\mathbf{X}_k)) \left[p^{(h,j)}\right]^{\mathbf{X}_k} \tag{4.59b}$$

$$w^{(h,j)} = w^{(I^{(h)},\xi^{(h)},\theta^{(h,j)})}(Z_k), \tag{4.59c}$$

$$p^{(h,j)} = p^{(\xi^{(h)},\theta^{(h,j)})}(\cdot|Z_k), \tag{4.59d}$$

where the optimization is solved for $h = 1, \dots, H$ to obtain truncated hypotheses with the largest weights. For the time-update, K-shortest path solution is implemented on the survival and birth costs, $C^{(I^{(h)},\xi^{(h)})}$ and C_b, to obtain truncated hypotheses

$J^{(h,j)} : j = 1, \ldots, K^h$ and $L^{(d)} : d = 1, \ldots, K^b$, respectively. Both $K^{(h)}$ and K_b are subsets of the largest survival and birth weights which are user chosen and dependent on the problem itself. By increasing $H + K_b$, there is an increase in computational cost to running the filter. For the measurement update, the ranked optimal assignment problem is solved on the cost, $C_Z^{(I^{(h)},\xi^{(h)})}$, for each hypothesis to obtain $\theta^{(h,j)} : j = 1, \ldots, T^{(h)}$. The $T^{(h)}$ is a subset of the largest weights for the association map which again are user defined and dependent on the problem itself. Similarly to the time-update, by increasing T, there is a T-fold increase in the computational complexity of the problem. The truncated versions for the time and measurement update are given by

$$\hat{\bar{\pi}}_k^{(h)}(\mathbf{X}_k) = \Delta(\mathbf{X}_k) \sum_{j=1}^{K^{(h)}} \sum_{d=1}^{K_b} w_{sb}^{(h,j,d)} \delta_{J^{(h,j)} \cup L^{(d)}}(\mathcal{L}(\mathbf{X}_k)) \left[p_k^{(h)}\right]^{\mathbf{X}_k}, \tag{4.60a}$$

where

$$w_{sb}^{(h,j,d)} = w_{p_s}^{(I^{(h)},\xi^{(h)})}(J^{(h,j)}) w_b(L^{(d)}), \tag{4.60b}$$

$$p_k^{(h)} = p_k^{(\xi^{(h)})}, \tag{4.60c}$$

and

$$\hat{\pi}^{(h)}(\mathbf{X}_k|Z_k) = \Delta(\mathbf{X}_k) \sum_{j=1}^{T^{(h)}} w^{(h,j)} \delta_{I^{(h)}}(\mathcal{L}(\mathbf{X}_k)) \left[p^{(h,j)}\right]^{\mathbf{X}_k}, \tag{4.61}$$

respectively. The $(\hat{\cdot})$ denotes that the time and measurement update have been truncated.

Control Objective using the GLMB Filter

By using a Gaussian mixture assumption and by truncating unimportant hypotheses, a closed form solution to the GLMB filter recursion is found with Eqs. (4.60) and (4.61). Using the time and measurement update equations, the RFS control objective is formed with a Gaussian mixture assumption on the single agent densities given by Eq. (4.49). This is substituted into the unlabeled intensity function given by Eq. (4.34) to provide a distributional distance-based objective between the current intensity, $\nu(\mathbf{x}, k) \triangleq f(\mathbf{x})$, and the desired intensity, $\nu_{des}(\mathbf{x}, k) \triangleq g(\mathbf{x}) \triangleq \sum_{i=1}^{N_g} w_g^{(i)} \mathcal{N}\left(\mathbf{x}; \mathbf{m}_g^i, P_g^i\right)$, given by

$$J(\mathbf{u}) = \sum_{k=1}^{T} \mathbf{u}_k^T R \mathbf{u}_k + D(\nu(\mathbf{x}, k), \nu_{des}(\mathbf{x}, k)), \tag{4.62}$$

where R is the positive definite control weight matrix and $\mathbf{u}_k$ is the control effort for the Gaussian mixture shown in Eq. (4.51a). Note that the objective function includes the time-step parameter k. The filter equations provide a recursion from one time-step k to the next. To make the filter and control objective consistent, all the equations have the time-step, k (e.g. Eq. (4.34) is supplemented with k to form $v(x, k)$). For this work, an MPC based ILQR is implemented using the L_2^2 plus quadratic divergence to estimate and control the large collaborative swarm simultaneously.

4.3 Results

RFS control is implemented using the Clohessy-Wiltshire dynamics in feedback with either the CPHD or GLMB filter. Note that imperfect information (i.e. process, measurement, and clutter noise) is included in the simulation. To complicate the problem even more, differing birth and death times for the agents are also simulated. Additionally, RFS control is implemented using the L_2^2 plus quadratic distance and

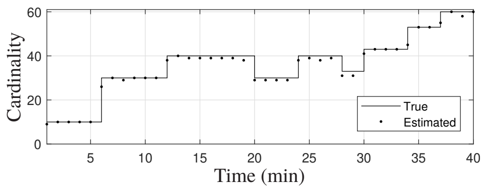
MPC-based ILQR. The first case involves control of a 77-agent swarm using the CPHD filter. The second case involves control of a 16-agent swarm involving the GLMB filter. The goal for each case is to track a rotating star pattern moving in a counterclockwise motion with a orbital frequency of $n = 0.00110678$ rad/s. With the addition of imperfect information and the added complication of changing number of agents, using the CPHD and GLMB filters provide accurate estimates of the agents through time which allows for RFS control in-the-loop.

4.3.1 Case 1: RFS Control using the CPHD Filter

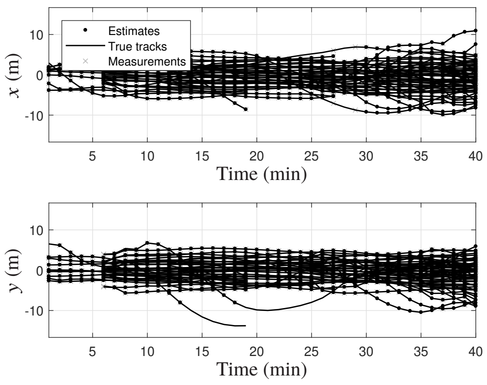
Case 1 illustrates the use of the CPHD filter in conjunction with RFS control for a 77-agent swarm. In this case, the agents are birthed at different time intervals from a uniformly random initial condition between -1 and 1 m from a chief satellite in a circular orbit. Figure 4.2a shows the cardinality of the swarm as agents die or birth through time. The estimated cardinality (dotted line) follows the true cardinality (solid line) as agents die or birth very accurately. Compared to the cardinality from the GM-PHD filter, the CPHD filter has a more accurate cardinality estimate as the number of agents are increased in the swarm. This is because the CPHD filter propagates a general cardinality distribution through time instead of assuming a cardinality distribution that is Poisson distributed. Although this increases the computational complexity of the CPHD filter in relation to the GM-PHD filter, the state and cardinality estimates become more accurate for RFS control, especially for large collaborative swarms. Figure 4.2c shows the snapshots of the controlled agents (black circles) and targets (green stars) at each time-step. As agents birth or die at each time-step, the agents move and maintain the rotating star formation. Note that at 39 min, agents are still converging into the star formation because new agents have birthed between -1 and 1 m from the origin at 38 min. Thus, all agents that have settled before time = 38 min may have to move to allow the birthed agents to move into the formation. Figure 4.2b shows the time history for the true agents (solid lines), estimated agents (black dots), and overall measurements (gray x's). The CPHD filter is able to detect the agents in motion and successfully provide estimates for RFS control.

4.3.2 Case 2: RFS Control using the GLMB Filter

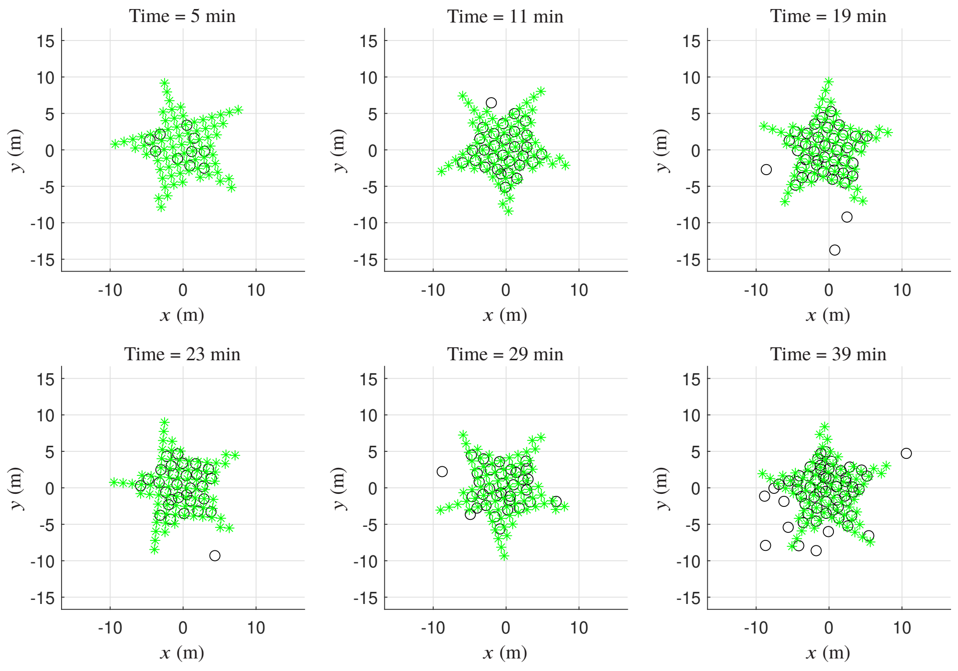
For case 2, 16 agents are birthed at different time intervals from a uniformly random initial condition between -1 and 1 m from a chief satellite in a circular orbit. Figure 4.3a shows the cardinality in the swarm through time. The solid and dotted lines show the true and estimated cardinality, respectively. Throughout the simulation, it can be observed that six agents die which is considered in the cardinality and state estimate of the swarm. These estimates are fed through the RFS control using ILQR to obtain a control input for each surviving agent. Figure 4.3c shows the snapshots of the controlled agents (black circles) and targets (green stars) at each time-step. Figure 4.3b shows the time history for the the true agents (solid lines), estimated agents that are labelled (colored dots), and overall measurements (gray x's). This figure shows the benefits of the GLMB filter. With the GLMB filter, an individual agent's track (time history) can be determined separately from other agents in the swarm. This identification is useful in providing specific control to an agent instead of relying on the Mahalanobis distance to compare an agent to the closest Gaussian mixture (estimate). Although obtaining track information is valuable for control of swarms, the GLMB filter becomes more computationally intensive as the number of agents increase in the swarm. The multi-agent Bayes filter recursion using the GLMB distribution is closed form but the number of hypotheses increases exponentially through time. The fix for this is to truncate the hypotheses that are unimportant by setting user-defined limits in the birth (K_b), survival ($K^{(h)}$), and measurement update hypotheses ($T^{(h)}$), but as the number of agents increase, these limits also have to



(a) Cardinality



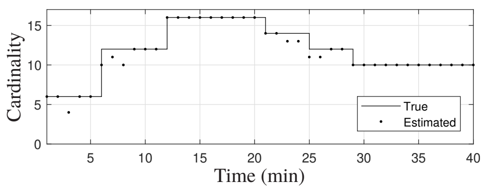
(b) Time History



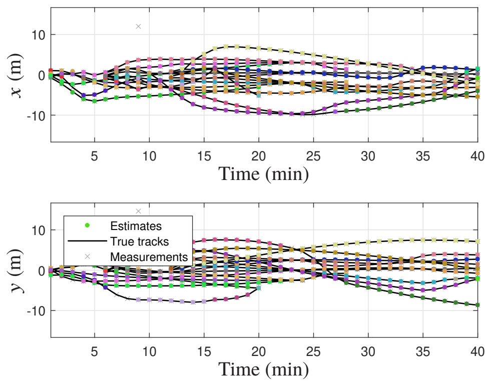
(c) Clohessy-Wiltshire Trajectory Snapshots

Figure 4.2: 77-agent spacecraft swarm controlled to a rotating star target via ILQR with imperfect information. The CPHD filter is used. Figure 4.3a shows the true and estimated cardinality, Figure 4.3b shows the time history of the true tracks, the estimated labelled tracks, and overall measurements, and Figure 4.3c shows the trajectories for the spacecraft agents and targets.

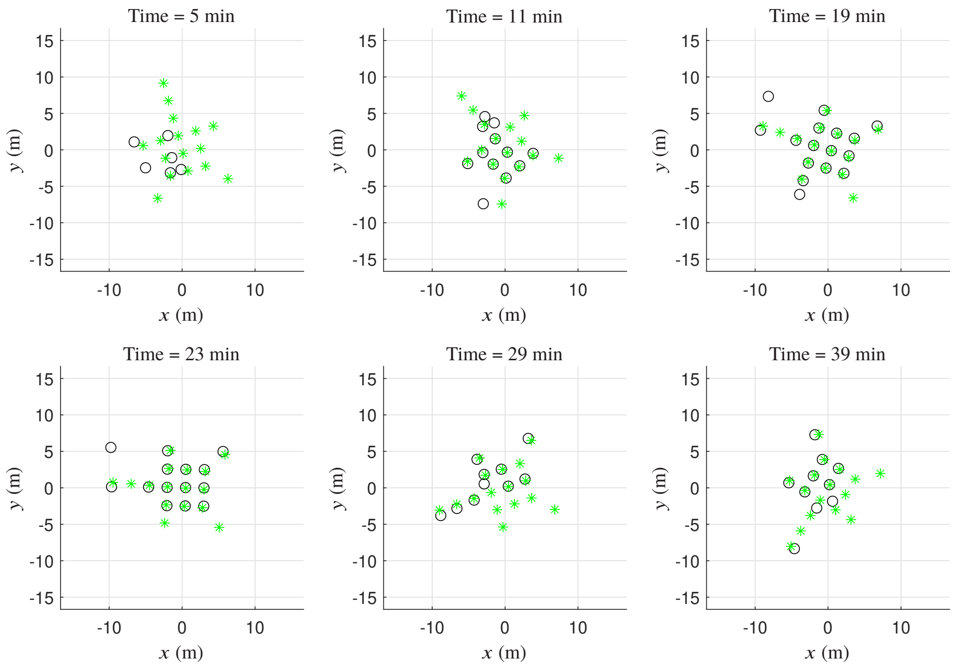
be increased to obtain an accurate estimate. Thus, less hypotheses are truncated which leads to higher computational complexity. For this reason, a 16-agent swarm is simulated since a 77-agent swarm becomes computationally intractable for finding accurate estimates. Even with this shortcoming for large swarms, the GLMB filter works very well in determining state, cardinality, and track estimates for swarm sizes that do not become computationally intractable.



(a) Cardinality



(b) Time History



(c) Clohessy-Wiltshire Trajectory Snapshots

Figure 4.3: 16-agent spacecraft swarm controlled to a rotating star target via ILQR with imperfect information. The GLMB filter is used. Figure 4.3a shows the true and estimated cardinality, Figure 4.3b shows the time history of the true tracks, the estimated labelled tracks, and overall measurements, and Figure 4.3c shows the trajectories for the spacecraft agents and targets.

Chapter 5

Conclusion

The objective of this work is to formulate the multi-target estimation and control background for large collaborative swarms. Multi-target estimation using the GM-PHD, CPHD, and GLMB filters and optimal control (i.e. ILQR or a Quasi-Newton optimization) are combined using RFS theory. RFS control is also decentralized by considering sparse control gain matrices obtained from sparse LQR.

By setting up the problem using information divergence to define the distance between the swarm RFS and the desired distribution, an optimal control problem is found that tracks a linear system with a nonquadratic objective function through the use of Quasi-Newton MPC and ILQR. Specifically, minimizing the L_2^2 plus quadratic distance provides control solutions which converges to a desired intensity. It also provides collision-reducing trajectory solutions. In consideration of agents that birth or spawn through time, the GM-PHD filter is used to determine accurate estimates of the multi-agent swarm problem. RFS control and the GM-PHD filter are combined for control of a large number of Gaussian mixtures and rotating targets. This allows for a converging RFS control solution of variable swarm size.

To provide better estimates for cardinality, the CPHD filter is implemented in conjunction with RFS control. By propagating a general cardinality distribution rather than a Poisson distribution assumption, cardinality and state estimates become more accurate as the number of agents in the swarm increase. The GLMB filter and RFS control are also combined which accounts for the labels of each agent. With each agent labelled, individual trajectories of each agent can be determined. Thus, by using RFS control and the GLMB filter, an individual agent's track can be determined separately from others in the swarm and be provided specific control instructions. In other words, specific agents can be controlled using the labelling information of the GLMB filter.

It is assumed through these results that the control topology is complete and it is used in a centralized manner. To provide a control topology that is localized or decentralized, sparse control gain matrices are obtained by sparsifying the RFS control gain matrix using sparse LQR. This allows agents to use local information topology or fully decentralized topology to drive an agent to a target.

In conclusion, by using a RFS-based architecture, the ability to estimate the cardinality and states of the swarm for control is realized. Thus, applications in scientific exploration, communication relaying, self-assembly, and surveillance become tangible by using RFS theory and optimal control for a large number of collaborative swarms.

5.0.1 Future Work

The results presented using RFS control show an implicit proof of the method's stability and optimality for control of collaborative swarms. Future work will entail theoretical proofs into the stability and optimality of the RFS control formulation to show these properties without the results directly. Also, this work provides RFS control using both centralized and decentralized control methods. These methods are planned to be expanded to hardware systems (i.e. UAVs and robots) to show the viability of the RFS control theory to swarms of such systems.

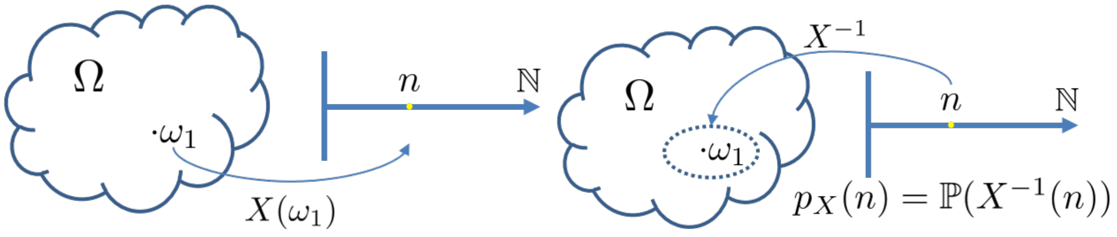
Lastly, future work will include exploration into the tensor train decomposition.

For large collaborative swarms, the state-space becomes larger as the cardinality increases. This becomes a problem for optimal control methods due to the curse of dimensionality. For example, ILQR uses both gradient and Hessian state information to approximate a quadratic cost for optimization, but this information becomes exponentially larger as the state increases. Tensor train decomposition allows for the reduction of these vectors to provide computationally easier and faster problems to solve. Thus, RFS control using tensor train decomposition is expected to be computationally faster than its normal matrix and vector counterpart.

Appendix A

Modelling and Derivation of the PHD Filter

In the development of multi-target estimation using RFSs, the PHD filter can be modeled and derived more intensively by using general terminology through point processes. From Mahler (2006), the PHD filter consists of random finite sets, multi-target densities, and set integrals. These are specific quantities that pertain to general terminology of point processes that consist of sequences of points, probability measures, probability densities, and measure-theoretical integrals. The RFS is equivalent to a simple point process and higher-order moments can be more easily described by measure-theoretical integrals as opposed to set integrals. Thus, the PHD filter in Eq. (2.10) can be derived by the use of point processes. Appendix A.1 provides an introduction to estimating the cardinality for a swarm using integer-valued random variables and probability generating functions (PGFs). Appendix A.2 extends this theory to point processes and probability generating functionals (PGFLs) to derive the PHD filter. For more discussion in point processes, see the extensive review presented by Daley and Vere-Jones (2003, 2007). The following derivation of the PHD filter using point processes follows closely to that of Houssineau et al. (2013) and lecture notes based off of Clark et al. (2016).



(a) (b)

Figure A.1: Mapping from probability space $(\Omega, \mathcal{F}, \mathbb{P})$ to space $\mathcal{X}$ for random variable X.

A.1 Integer-valued Random Variables

The number of agents in a swarm are represented by an integer-valued random variable X in which each agent is counted by an integer and the cardinality is unknown. The random variable maps between a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ and a set of non-negative integers $\mathbb{N}$ given by Figure A.1a. Ω is the outcome space which contains all the possible outcomes regarding the random experiment, $\mathcal{F}$ is a subset of space Ω and is considered the $\sigma-$algebra on Ω, and $\mathbb{P}$ is the probability measure. Individual outcomes are denoted as w_i and individual realizations of $\mathbb{N}$ are n. The quantity $X^{-1}(n)$ represents the collection of all possible w_i that leads to the realization n given in Figure A.1b. The probability measure $\mathbb{P}$ measures the size of $X^{-1}(n)$, thus n becomes more likely to be chosen when sampling from X. The equation

$$p_X(n) = \mathbb{P}(X^{-1}(n)), \tag{A.1}$$

gives the probability (likelihood) of choosing n when X is sampled. This is defined by the event $X = n$. This probability has the property of

$$\sum_{n\geq 0} p_X(n) = \sum_{n\geq 0} \mathbb{P}(X^{-1}(n)) = 1, \tag{A.2}$$

which ensures that the probability of all possible outcomes is equal to 1. $p_X(n)$ in this case is a cardinality probability which describes the swarm with n agents.

The representation or behaviorial description of X is found by defining the random variable's moments. The kth order non-factorial, $\mu_X^{(k)}$, and factorial, $\alpha_X^{(k)}$, moments of X about integer $k \geq 0$ are

$$\mu_X^{(k)} = \mathbb{E}[X^{(k)}] = \sum_{n\geq 0} p_X(n)n^{(k)}, \tag{A.3a}$$

$$\alpha_X^{(k)} = \mathbb{E}[X(X-1)\dots(X-k+1)] = \sum_{n\geq k} p_X(n)n(n-1)\dots(n-k+1). \tag{A.3b}$$

The non-factorial moments are used to determine the central moments. For a random variable X, the variance can formed from the central moment given by

$$\mathrm{var}_X = \mu_X^{(2)} - \left(\mu_X^{(1)}\right)^2. \tag{A.4}$$

This describes the distance or spread of realizations about the average value of X, $\mu_X^{(1)}$. Similarly, the joint moment between random variables X and Y is formed as

$$\mathrm{cov}_{X,Y} = \mu_{XY}^{(1)} - \mu_X^{(1)}\mu_Y^{(1)}. \tag{A.5}$$

For the factorial moment, the first order is used to provide the mean value of X given by

$$\mu_X = \sum_{n\geq 0} p_X(n)n = \mu_X^{(1)} = \alpha_X^{(1)}. \tag{A.6}$$

The cardinality distribution is a set of cardinality probabilities $\{p_x(n)\}_{n\geq 0}$ which characterizes the entirety of X, but may be unavailable or intractable to estimate through time. For a multi-target estimation problem, the variables are updated with new measurements. Thus, random variables that describe the cardinality are also updated. This update is difficult to describe, so consequently, another representation must be found using probability generating functions.

A.1.1 Probability Generating Functions

The generating function G is a function which is generated by a sequence of real numbers. For a sequence of real numbers, $(u_n)_{n\geq 0}$, the generating function G is

$$G(s) = \sum_{n\geq 0} u_n S^n, \tag{A.7}$$

where $s \in \mathbb{R}^+$ and the sum is finite. The generating function relates a set of non-negative real numbers $\mathbb{R}^+ \to \mathbb{R}$. The random variable X with cardinality distribution $\{p_x(n)\}_{n\geq 0}$ can be substituted into the equation to produce the PGF G_X given by the expectation

$$\begin{aligned} G_X(s) &= \mathbb{E}[s^X] \\ &= \sum_{n\geq 0} p_X(n) s^n. \end{aligned} \tag{A.8}$$

By using probabilities in the generating function, the test variable s is constrained from $0 \leq s \leq 1$. By setting s to 0 and 1, two properties of PGFs are given by

$$G_X(0) = \sum_{n\geq 0} p_X(n) 0^n = p_X(0), \tag{A.9a}$$

$$G_X(1) = \sum_{n\geq 0} p_X(n) 1^n = \sum_{n\geq 0} p_X(n) = 1. \tag{A.9b}$$

For $s = 0$, the cardinality probability $p_X(0)$ is extracted from the PGF. For $s = 1$, the PGF sums to 1.

For multi-target estimation, joint random variables must also be studied. For example, a random variable that consists of measurements can be determined given by the number of agents in the field. Thus, a joint PGF $\mathcal{G}_{Z,X}$ is given by the expectation

$$\begin{aligned}\mathcal{G}_{Z,X}(t,s) &= \sum_{m,n\geq 0} p_Z(m)p_X(n)t^m s^n \\ &= \left(\sum_{m\geq 0} p_Z(m)t^m\right)\left(\sum_{n\geq 0} p_X(n)s^n\right) \\ &= G_Z(t)G_X(s),\end{aligned} \tag{A.10}$$

where t is the test variable for random variable Z. Several properties coincide to manipulate PGFs into other forms.

The derivative can be obtained from a PGF since PGFs are real-valued functions. For a function f with $\mathbb{R} \to \mathbb{R}$, the derivative is given by

$$f'(x) = \lim_{\epsilon\to 0} \frac{f(x+\epsilon) - f(x)}{\epsilon}, \tag{A.11}$$

evaluated at $x \in \mathbb{R}$ for small $\epsilon \in \mathbb{R}$. The properties for a function f and g are

$$\text{sum: } (f+g)'(x) = f'(x) + g'(x), \tag{A.12a}$$

$$\text{product: } (f \cdot g)'(x) = f'(x)g(x) + f(x)g'(x), \tag{A.12b}$$

$$\text{power: } (f^m)'(x) = mf^{m-1}(x)f'(x), \tag{A.12c}$$

$$\text{chain: } (f \circ g)'(x) = g'(x)f'(g(x)). \tag{A.12d}$$

It is also useful to know the properties of using an exponential function with a PGF as it can provide tractable and implementable solutions to the cardinality estimation problem. The properties include

$$\text{ordinary differentiation: } \exp'(x) = \exp(x), \tag{A.13a}$$

$$\text{chain: } (\exp\circ f)'(x) = f'(x)(\exp\circ f)(x), \tag{A.13b}$$

$$\text{Taylor expansion: } \exp(x) = \sum_{n\geq 0}\frac{\exp^{(n)}(0)}{n!}x^{(0)} = \sum_{n\geq 0}\frac{x^{(n)}}{n!}. \tag{A.13c}$$

With these properties for a function f, differentiation can be applied directly on the PGF. For a random variable X and a PGF G_X, the goal is to determine the cardinality distribution $\{p_x(n)\}_{n\geq 0}$. The derivatives for G_X are given by

$$G_X(s) = \sum_{n\geq 0} p_X(n)s^n, \tag{A.14a}$$

$$G'_X(s) = \sum_{n\geq 0} p_X(n)(s^n)' = \sum_{n\geq 1} p_X(n)ns^{n-1}, \tag{A.14b}$$

$$G^{(2)}_X(s) = \sum_{n\geq 1} p_X(n)n(s^{n-1})' = \sum_{n\geq 2} p_X(n)n(n-1)s^{n-2}, \tag{A.14c}$$

$$\begin{aligned} G^{(k)}_X(s) &= \sum_{n\geq k} p_X(n)n(n-1)\dots(n-k+1)s^{n-k} \\ &= \sum_{n\geq k} p_X(n)\frac{n(n-1)\dots(n-k+1)(n-k)\dots 1}{(n-k)\dots 1}s^{n-k} \\ &= \sum_{n\geq k} p_X(n)\frac{n!}{(n-k)!}s^{n-k}. \end{aligned} \tag{A.14d}$$

By substituting $s = 0$ or $s = 1$ into Eq. (A.14d), the kth derivative simplifies to

$$G^{(k)}_X(0) = \sum_{n\geq k} p_X(n)\frac{n!}{(n-k)!}0^{n-k} = p_X(k)\frac{k!}{(k-k)!} = k!p_X(k), \tag{A.15a}$$

$$G^{(k)}_X(1) = \sum_{n\geq k} p_X(n)n(n-1)\dots(n-k+1) = \alpha^{(k)}_X. \tag{A.15b}$$

Thus, the cardinality probability and the factorial moment can be directly extracted from the derivatives of the PGF for $s = 0$ or $s = 1$. By knowing the PGF G_X, the full characterization of the random variable is known by computing the cardinality distribution directly. These equations are given by

$$p_X(k) = \frac{G_X^{(k)}(0)}{k!}, \tag{A.16a}$$

$$\mu_x = \alpha_X^{(1)} = G'_X(1). \tag{A.16b}$$

Similarly, a derivation using joint random variables, X and Z, can also occur using joint PGFs assuming $Z = m$. This PGF, $\mathcal{G}_{Z=m,X}(s)$ is determined directly from a general PGF $\mathcal{G}_{Z,X}$ given by

$$\begin{aligned} \mathcal{G}_{Z=m,X}(s) &= \sum_{n\geq 0} p_{Z,X}(m,n)s^n \\ &= \frac{1}{m!}\frac{d^m}{dt^m}\mathcal{G}_{Z,X}(t,s)|_{t=0}. \end{aligned} \tag{A.17}$$

Then, it can be differentiated with respect to s to produce the cardinality probability $\{p_{Z,X}(m,n)\}_{m,n\geq 0}$ using Eqs. (A.14d) and (A.15a).

Similar to random variables, operations can also occur with PGFs. The first operation of discussion is marginalization. For the joint behavior of random variables Z and X, the behavior of one random variable can be determined directly by marginalization. For example, to find the behavior of Z from the joint behavior of Z and X, the joint behavior is marginalized (integrated or summed) over all the possible realizations of X given by

$$\forall m \in \mathbb{N}, \quad p_Z(m) = \sum_{n\geq 0} p_{Z,X}(m,n). \tag{A.18}$$

Marginalization works similarly with PGFs. For the joint PGF in Eq. (A.10), the

PGF for Z is determined by marginializing over X. The test variable is set to $s = 1$ and the joint PGF can be simplified given by

$$\begin{aligned}\mathcal{G}_{Z,X}(t,1) &= \sum_{m,n\geq 0} p_{Z,X}(m,n)t^m 1^n \\ &= \sum_{m\geq 0}\left(\sum_{n\geq 0} p_{Z,X}(m,n)\right) t^m \\ &= \sum_{m\geq 0} p_Z(m)t^m \\ &= G_Z(t).\end{aligned} \tag{A.19}$$

The operation using sums of two realizations of random variables $X + Y = Z$ is also of interest for PGFs for scenarios where X and Y are independent. The PGF of Z in terms of X and Y is given by

$$\begin{aligned}G_Z(s) &= \mathbb{E}[s^Z] \\ &= \mathbb{E}[s^{X+Y}] \\ &= \mathbb{E}[s^X s^Y] \\ &= \mathbb{E}[s^X]\mathbb{E}[s^Y] \\ &= G_X(s)G_Y(s),\end{aligned} \tag{A.20}$$

if X and Y are independent. If they are not independent, Eq. (A.20) results in $G_Z(s) = \mathbb{E}[s^X s^Y]$.

One last operation of interest is known as branching, or a special kind of dependence between X and Y. For a realization m of parent random variable Y, the daughter random variable X will have a superposition of m identical but independent random variables T. This is directly related to the spawning of agents. The parent random variable spawns a number of agents in the daughter random variable following some transition described by T. Suppose that the PGFs for the parent and transitional random variable Y and T are known by G_Y and G_T, respectively. The goal is to describe X in terms of these variables. The PGF for the joint behavior between X and Y is

$$\begin{aligned}\mathcal{G}_{Y,X}(t,s) &= \sum_{m,n\geq 0} p_{Y,X}(m,n)t^m S^n \\ &= \sum_{m,n\geq 0} p_Y(m)p_{X|Y}(n|m)t^m s^n \\ &= \sum_{m\geq 0} p_Y(m)\left(\sum_{n\geq 0} p_{X|Y}(n|m)s^n\right)t^m \\ &= \sum_{m\geq 0} p_Y(m)\mathcal{G}_{X|Y}(s|m)t^m,\end{aligned} \tag{A.21}$$

where $\mathcal{G}_{X|Y}(s|m)$ is the conditional PGF of X given the realization of Y. If $Y=m$, $X|Y$ is the superposition of m copies that are independent of the transitional random variable T. This results in

$$\mathcal{G}_{X|Y}(s|m) = (G_T(s))^m. \tag{A.22}$$

By substituting Eq. (A.22) into Eq. (A.21), the joint PGF becomes

$$\begin{aligned}\mathcal{G}_{Y,X}(t,s) &= \sum_{m\geq 0} p_Y(m)(G_T(s))^m t^m \\ &= \sum_{m\geq 0} p_Y(m)(tG_T(s))^m \\ &= G_Y(tG_T(s)).\end{aligned} \tag{A.23}$$

This describes the joint behavior between dependencies of X and Y. To obtain the description of X, marginalization is applied about Y which is given by

$$\begin{aligned} G_X(s) &= \mathcal{G}_{Y,X}(1, s) \\ &= G_Y(G_T(s)). \end{aligned} \tag{A.24}$$

Examples of Random Variables and PGFs

With the basics of random variables and PGFs, families of these quantities are discussed which is used directly in multi-target filtering.

The first of these random variables is a Bernoulli random variable X which has a parameter $0 \leq p \leq 1$ defined by

$$X = \begin{cases} 0, & \text{if } 1 - p, \\ 1, & \text{if } p. \end{cases} \tag{A.25}$$

The parameter p is the probability that an event 1 will occur. This is used in single random experiments that asks a yes/no question or a coin flip. The PGF G_X is constructed using Eq. (A.8) given by

$$\begin{aligned} G_X(s) &= \sum_{n \geq 0} p_X(n) s^n \\ &= p_X(0) + p_X(1)s = \sum_{n \geq 2} p_X(n) s^n \\ &= 1 - p + ps. \end{aligned} \tag{A.26}$$

For multi-target tracking, the Bernoulli random variable depicts the agent survival or agent detection.

The other random variable of interest is the Poisson random variable X with a parameter $\lambda_X \geq 0$ defined by

$$\forall n \geq 0, \quad X = n, \quad p_X(n) = \exp(-\lambda_X)\frac{\lambda_X^n}{n!}. \tag{A.27}$$

The parameter λ_X is the average number of events per interval, and it is considered a rate parameter. It is used for modelling the number of times an event occurs for some particular space or time interval. From Eq. (A.8), the PGF for a Poisson distribution is

$$\begin{aligned} G_X(s) &= \sum_{n\geq 0} p_X(n)s^n \\ &= \sum_{n\geq 0} \exp(-\lambda_X)\frac{\lambda_X^n}{n!}s^n \\ &= \exp(-\lambda_X)\sum_{n\geq 0}\frac{(\lambda_X s)^n}{n!}, \end{aligned} \tag{A.28}$$

and by using a Taylor expansion, Eq. (A.13c), the equation simplifies to

$$\begin{aligned} G_X(s) &= \exp(-\lambda_X)\exp(\lambda_X s) \\ &= \exp(\lambda_X(s-1)). \end{aligned} \tag{A.29}$$

The mean can be determined directly by Eq. (A.16b) which results in

$$\begin{aligned} \mu_X &= G_X'(s)|_{s=1} \\ &= (\exp(\lambda_X(s-1)))'|_{s=1} \\ &= \lambda_X \exp(\lambda_X(s-1))|_{s=1} \\ &= \lambda_X \exp(\lambda_X(1-1)) \\ &= \lambda_X. \end{aligned} \tag{A.30}$$

The mean of a Poisson distribution is just λ_X. Also, the var_X is λ_X using a similar derivation for variance. For multi-agent filtering, the Poisson distribution is attractive since it allows for tractability in filtering recursions.

A.1.2 Cardinality Estimation

With the introduction to random variables and PGFS, these tools can be used to derive a cardinality estimator from a multi-target Bayesian estimation perspective which consists of time-update and measurement update steps. The goal is to estimate and propagate the mean number of agents through time by observing measurements. Note that this derivation estimates only the number of agents and not the state of each agent. The construction of the problem consists of relating X, the agents after the time-update, with Y, the agents before the time-update, and relating $X|Z$ from X where Z contains the current measurements and $X|Z$ contains the measurement updated agent estimate. The other part of this problem consists of extracting information (i.e. μ_X, μ_Y, and $\mu_{X|Z}$) from differentiation of the PGFs to determine the cardinality estimates. Both of these steps take in assumptions on the physical environment that the agents are set in. Thus, a proper set-up is required to determine the best estimates from the filter.

For the time-update (prediction step), it is assumed that the agents are independent of each other, each agent survives with probability p_s or dies with probability $1-p_s$, and agents birth independently from the surviving agents on the field. For the time-update, the survival random variable X_s is a Bernoulli random variable with a parameter p_s given by

$$\mathcal{G}_s(s) = 1 - p_s + p_s s. \tag{A.31}$$

The number of surviving agents is denoted by X_{sur} which occurs by branching the parent random variable Y with the transition random variable X_s given by

$$\mathcal{G}_{sur}(s) = G_Y(\mathcal{G}_s(s)). \tag{A.32}$$

The time-updated number of agents is described by X which is the union (sum) of the surviving agents and birthed agents given by

$$G_X(s) = \mathcal{G}_{sur}(s)\mathcal{G}_b(s). \tag{A.33}$$

Therefore, the predicted PGF for the cardinality is

$$G_X(s) = G_Y(1 - p_s + p_s s)\mathcal{G}_b(s). \tag{A.34}$$

This PGF provides the entire description of agents as they birth or survive to the next time-step without computing each agent's individual cardinality probability.

For the measurement update, it is assumed that measurements are independently measured from each other, an agent is detected with a probability p_d from a single measurement or it is not detected with a probability $1 - p_d$, and any clutter measurements are obtained independently from target measurements. For the measurement update, the observation random variable Z_{obs} is a Bernoulli random variable with a parameter p_d given by

$$\mathcal{G}_{obs}(t) = 1 - p_d + p_d t. \tag{A.35}$$

The number of agent measurements is denoted by Z_{tar} which occurs by branching the parent random variable X with the transition random variable Z_{obs} given by

$$\mathcal{G}_{Z_{tar},X}(t, s) = G_X(s\mathcal{G}_{obs}(t)). \tag{A.36}$$

The joint PGF of measured agents is the sum of the measurements and clutter given by

$$\mathcal{G}_{Z,X}(t,s) = \mathcal{G}_{Z_{tar},X}(t,s)\mathcal{G}_c(t). \tag{A.37}$$

Therefore, the measurement updated PGF for the cardinality is

$$\mathcal{G}_{Z,X}(t,s) = G_X(s(1-p_d+p_d t))\mathcal{G}_c(t). \tag{A.38}$$

With the measurement update PGF, the goal is to estimate the number of agents conditioned on $Z = m$. By using Bayes' rule given by

$$p_{X|Z}(n|m) = \frac{p_{Z,X}(m,n)}{p_Z(m)}, \tag{A.39}$$

a posterior probability of $X = n$ agents on the field given $Z = m$ measurements can be found using the joint probability of $X = n$ agents and $Z = m$ measurements and the probability of $Z = m$ measurements. Bayes' rule can be manipulated by multiplying s^n on both sides and summing up all realizations of X given by

$$\sum_{n\geq 0} p_{X|Z}(n|m)s^n = \frac{\sum_{n\geq 0} p_{Z,X}(m,n)s^n}{p_Z(m)}. \tag{A.40}$$

By substituting Eqs. (A.17), (A.8), and (A.16a) into Eq. (A.40), the equation reduces to

$$\begin{aligned} \mathcal{G}_{X|Z}(s|m) &= \frac{\mathcal{G}_{Z=m,X}(s)}{p_Z(m)} \\ &= \frac{\frac{1}{m!}\frac{d^m}{dt^m}\mathcal{G}_{Z,X}(t,s)|_{t=0}}{\frac{1}{m!}G_Z^{(m)}(0)}. \end{aligned} \tag{A.41}$$

The denominator can be expanded out to all possible agents using Eq. (A.19) to

$$\mathcal{G}_{X|Z}(s|m) = \frac{\frac{d^m}{dt^m}\mathcal{G}_{Z,X}(t,s)|_{t=0}}{\frac{d^m}{dt^m}\mathcal{G}_{Z,X}(t,1)|_{t=0}} \tag{A.42}$$

where $\mathcal{G}_{Z,X}(t,s) = G_X(s(1-p_d+p_dt))\mathcal{G}_c(t)$. These PGFs can be exploited using differentiation to obtain the cardinality estimates directly.

For the time-update, μ_X is determined by Eq. (A.16b) about G_X given by

$$\mu_X = G'_X(s)|_{s=1}. \tag{A.43}$$

By substituting Eq. (A.34), the equation simplifies to

$$\mu_X = (G_Y(1-p_s+p_ss)\mathcal{G}_b(s))'|_{s=1}. \tag{A.44}$$

The derivative is solved using the product rule given by Eq. (A.12b) which reduces to

$$\mu_X = (G_Y(1-p_s+p_ss))'|_{s=1}\mathcal{G}_b(s)|_{s=1} + G_Y(1-p_s+p_ss)|_{s=1}\mathcal{G}'_b(s)|_{s=1}. \tag{A.45}$$

By using chain rule, Eq. (A.12d), the equation becomes

$$\begin{aligned}\mu_X &= (1-p_s+p_ss)'|_{s=1}G'_Y(1-p_s+p_ss)|_{s=1}\mathcal{G}_b(1) + G_Y(1)G'_b(1)\\ &= p_sG'_Y(1)\mathcal{G}_b(1) + G_Y(1)G'_b(1).\end{aligned} \tag{A.46}$$

By using the property when the test variable $s=1$ given by Eq. (A.9b) and the PGF derivative property given by Eq. (A.16b), the mean reduces to

$$\begin{aligned}\mu_X &= p_sG'_Y(1) + G'_b(1)\\ &= p_s\mu_Y + \mu_b.\end{aligned} \tag{A.47}$$

This provides a direct time update without assuming anything about the prior cardinality or birth cardinality.

For the measurement update, the derivative of $\mathcal{G}_{X|Z}(\cdot|m)$ is

$$\mu_{X|Z=m} = G'_{X|Z}(s|m)|_{s=1}. \tag{A.48}$$

By substituting Eq. (A.42), the equation becomes

$$\mu_{X|Z=m} = \frac{\frac{d^{m+1}}{dsdt^m}\mathcal{G}_{Z,X}(t,s)|_{t=0,s=1}}{\frac{d^m}{dt^m}\mathcal{G}_{Z,X}(t,1)|_{t=0}} \tag{A.49}$$

The equation from Eq. (A.49) becomes intractable without any assumptions on the predicted cardinality X or the clutter. But, if the predicted cardinality, X, and clutter, Z_c, are Poisson distributed, tractable solutions can be determined using Eq. (A.29). This becomes

$$\begin{aligned}\mathcal{G}_{Z,X}(t,s) &= G_X(s(1-p_d+p_dt))\mathcal{G}_c(t) \\ &= \exp(\mu_X(s(1-p_d+p_dt)-1))\exp(\mu_c(t-1)) \\ &= \exp(\mu_X(s(1-p_d+p_dt)-1)+\mu_c(t-1)).\end{aligned} \tag{A.50}$$

With the exponential form, the equation can be simplified using Eq. (A.13b) yielding

$$\begin{aligned}\frac{d}{dt}\mathcal{G}_{Z,X}(t,s) &= \frac{d}{dt}\exp(\mu_X(s(1-p_d+p_dt)-1)+\mu_c(t-1)) \\ &= \frac{d}{dt}(\mu_X(s(1-p_d+p_dt)-1)+\mu_c(t-1))\exp(\mu_X(s(1-p_d+p_dt)-1) \\ &\quad +\mu_c(t-1)) \\ &= (\mu_X sp_d+\mu_c)\exp(\mu_X(s(1-p_d+p_dt)-1)+\mu_c(t-1)).\end{aligned} \tag{A.51}$$

The term in front of the exponential is independent of t, therefore the mth derivative of the joint PGF with respect to t is

$$\frac{d^m}{dt^m}\mathcal{G}_{Z,X}(t,s) = (\mu_X s p_d + \mu_c)^m \exp(\mu_X(s(1-p_d+p_d t)-1)+\mu_c(t-1)). \tag{A.52}$$

The numerator in Eq. (A.49) requires the differentiation of Eq. (A.52) with respect to s using Eq. (A.12b). This becomes

$$\begin{aligned}\frac{d^{m+1}}{dsdt^m}\mathcal{G}_{Z,X}(t,s) &= \frac{d}{ds}((\mu_X s p_d + \mu_c)^m)\exp(\mu_X(s(1-p_d+p_d t)-1)+\mu_c(t-1)) \\ &\quad + (\mu_X s p_d+\mu_c)^m \frac{d}{ds}\exp(\mu_X(s(1-p_d+p_d t)-1)+\mu_c(t-1)).\end{aligned} \tag{A.53}$$

The first term and second term can be simplified with Eqs. (A.12c) and (A.13b), repectively. This is given by

$$\begin{aligned}\frac{d^{m+1}}{dsdt^m}\mathcal{G}_{Z,X}(t,s) &= m(\mu_X s p_d+\mu_c)^{m-1}\mu_X p_d \exp(\mu_X(s(1-p_d+p_d t)-1)+\mu_c(t-1)) \\ &\quad + (\mu_X s p_d+\mu_c)^m \mu_X(1-p_d+p_d t)\exp\left(\mu_X(s(1-p_d+p_d t)-1)\right. \\ &\quad \left.+\mu_c(t-1)\right).\end{aligned} \tag{A.54}$$

Then the quantities in Eqs. (A.54) and (A.52) are substituted into Eq. (A.49) with $s=1$ and $t=0$ to obtain the estimated cardinality result

$$\begin{aligned}\mu_{X|Z=m} &= m\frac{\mu_X p_d}{\mu_X s p_d+\mu_c} + \mu_X(1-p_d+p_d t) \\ &= m\frac{\mu_X p_d}{\mu_X p_d+\mu_c} + \mu_X(1-p_d).\end{aligned} \tag{A.55}$$

Thus, the estimated cardinality is obtained using

$$\mu_X = p_s \mu_Y + \mu_b, \tag{A.56a}$$

$$\mu_{X|Z=m} = m \frac{\mu_X p_d}{\mu_X p_d + \mu_c} + \mu_X (1 - p_d). \tag{A.56b}$$

A.2 PHD Filter using Point Processes

The methodology is extended to a full PHD filter formulation which estimates both the cardinality and state estimates of agents in the swarm. Specifically, point processes are introduced in conjunction with probability generating functionals (PGFLs) to derive the PHD filter equations directly.

A.2.1 Introduction to Point Processes

For a number of agents in an environment, the cardinality and the individual states (i.e. position and velocity) for each agent are unknown. The description of the swarm can be described as a random finite set or more generally as a point process Φ. A point process is a random variable where the size of the sequence and elements within are both random realizations. The agent state space is $\mathbf{X} \subseteq \mathbb{R}^{d_x}$ where d_x is the agent state vector size. The measurements, z, produced from the system have a state space $\mathbf{Z} \subseteq \mathbb{R}^{d_z}$ where d_z is the measurement vector size.

The point process, Φ, maps between a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ and the space $\mathcal{X} = \cup_{k \geq 0} \mathbf{X}^k$. This relation is given by Figure A.2a. The outcomes w_i can be associated to realizations ϕ. $\Phi^{-1}(d\phi)$ represents the collection of all possible w_i that achieves a realization in the neighborhood of ϕ with components $d\phi$ given in Figure A.2b. The probability measure $\mathbb{P}$ measures the size of $\Phi^{-1}(d\phi)$, thus, $d\phi$ becomes more likely to be chosen when sampling from Φ. The point process probability is

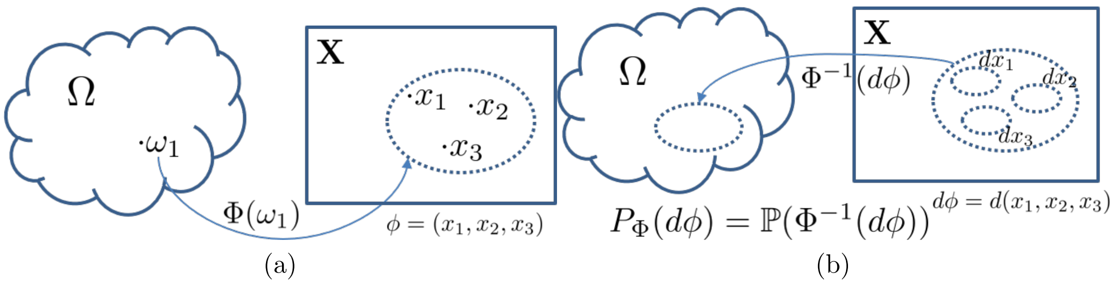


Figure A.2: Mapping from probability space $(\Omega, \mathcal{F}, \mathbb{P})$ to space $\mathcal{X}$ for point process Φ.

given by

$$P_\Phi(d\phi) = \mathbb{P}(\Phi^{-1}(d\phi)), \tag{A.57}$$

which gives the likelihood of choosing $d\phi$ when Φ is sampled. Specifically, $P_\Phi(d\phi)$ is the probability that the swarm, described by Φ, has n agents and the ith agent in $d\phi = d(x_1, \ldots, x_n)$ is localized in the neighborhood dx_i. When the probability distribution is integrated about the sequence of points in $\mathbf{X}$, the result becomes 1 which ensures that P_Φ is a probability measure. This is given by

$$\int_{\mathcal{X}} P_\Phi(d\phi) = \int_{\mathcal{X}} \mathbb{P}(\Phi^{-1}(d\phi)) = 1. \tag{A.58}$$

By comparing point processes to integer-valued random variables, both have very similar attributes. A property of point processes of importance is that the probability distributions are defined by symmetric functions. That is, any arrangements of a realization will occur with equal probability and is exemplified by the example of $P_\phi(d(x_1, x_2, x_3)) = P_\phi(d(x_3, x_2, x_1))$. Also point process realizations may have sequences of points that are point-wise distinct. This is a simple point process, and this property is assumed throughout the literature.

The projection measure $P_\Phi^{(n)}$ characterizes the probability distribution P_Φ for $n \geq$

0, and the nth-order projection is defined on $\mathbf{X}^n$ for $n \geq 1$. The projection measure provides both the point process probability of n points and the probability distribution of the n points, but the projection measures are not probability measures. The probability of an empty point process is given by $P_\Phi^{(0)}$. The cardinality distribution of a point process is given by

$$\rho_\Phi(n) = \int_{\mathbf{X}^n} P_\Phi^{(n)}(d(x_1, \ldots, x_n)), \tag{A.59}$$

for $n \geq 0$. The probability $\rho_\Phi(n)$ is the probability that a realization ϕ about Φ has a sequence with n points. Due to the symmetry of P_Φ, $P_\Phi^{(n)}$ is also symmetric. Thus, Janossy measures are introduced with group projection measure information over all possible arrangement of points. The nth order Janossy measure about Φ, $J_\Phi^{(n)}$, is defined by

$$\begin{aligned} J_\Phi^{(n)}(B_1 \times \cdots \times B_n) &= \sum_{\sigma(n)} P_\Phi^{(n)}(B_{\sigma_1} \times \cdots \times B_{\sigma_n}) \\ &= n! P_\Phi^{(n)}((B_1 \times \cdots \times B_n)), \end{aligned} \tag{A.60}$$

for $n \geq 0$. The term B_i is a region of $\mathbf{X}$ with $1 \leq i \leq n$ and $\sigma(n)$ is a set of all arrangements (permutations) of $1, \ldots, n$. The rate of change of P_Φ over a unit volume of state space is given by density $p_\Phi(x_1, \ldots, x_n)$. This can be roughly defined as the probability that $\Phi = (x_1, \ldots, x_n)$. Similarly, both $P_\Phi^{(n)}$ and $J_\Phi^{(n)}$ have the densities $p_\Phi^{(n)}$ and $j_\Phi^{(n)}$, respectively.

For the probability P_Φ, a set integral formulation can be obtained to describe a point process. By assuming that f is a function on $\mathcal{X}$, the set integral of f and P_Φ is

$$\begin{aligned}
P_\Phi(f) &= \int_{\mathcal{X}} f(\phi) P_\Phi(d\phi) \\
&= \int_{\mathcal{X}} f(\phi) p_\Phi(\phi) d\phi \\
&= \sum_{n\geq 0} \int_{\mathbf{X}^n} f(x_1,\ldots,x_n) P_\Phi^{(n)}(d(x_1,\ldots,x_n)) \\
&= \sum_{n\geq 0} \int_{\mathbf{X}^n} f(x_1,\ldots,x_n) p_\Phi^{(n)}(x_1,\ldots,x_n) dx_1 \ldots dx_n \\
&= \sum_{n\geq 0} \frac{1}{n!} \int_{\mathbf{X}^n} f(x_1,\ldots,x_n) J_\Phi^{(n)}(d(x_1,\ldots,x_n)) \\
&= \sum_{n\geq 0} \frac{1}{n!} \int_{\mathbf{X}^n} f(x_1,\ldots,x_n) j_\Phi^{(n)}(x_1,\ldots,x_n) dx_1 \ldots dx_n.
\end{aligned} \tag{A.61}$$

Note that although a set integral formulation using densities is used, the properties of point processes can be obtained with a measure-theoretic formulation as well. Specifically, measure-theoretic integrals can easily describe higher-order moments, but to simplify the derivation, the use of probabilities and set integrals are used instead. Set integrals also have a compact expression for multi-target filtering, but they do not have the properties of measure-theoretic integrals.

Just like with random variables, the full knowledge is usually not available for multi-target estimation. Thus, a limited description of point processes has to be provided by its moment measures or densities. Both the factorial and non-factorial moments are defined for point processes, but they take more work to derive. For the PHD filter, only the first-order moment density or intensity, μ_Φ, is necessary for derivation. The intensity, $\mu_\Phi(x)$, is defined as the density of the average number of agents with state x per unit volume. This is roughly the density of the average number of agents with state x. The intensity function is given by

$$\begin{aligned}\mu_\Phi(x) &= \int_{\mathcal{X}} \left(\sum_{x_i \in \phi} \delta_x(x_i) \right) p_\Phi(\phi) d\phi \\ &= \sum_{n \geq 1} \sum_{\mathbf{X}^n} \left(\sum_{i=1}^{n} \delta_x(x_i) \right) p_\Phi^{(n)}(x_1, \ldots, x_n) dx_1 \ldots dx_n,\end{aligned} \tag{A.62}$$

which considers all realizations about Φ for each agent with state x. $\delta_x(\cdot)$ is the Dirac delta function, and the equation can be simplified to

$$\begin{aligned}\mu_\Phi(x) &= \sum_{k \geq 1} \int_{\mathbf{X}^{k-1}} \left(\sum_{i=1}^{n} p\Phi^{(n)}(x_1, \ldots, x_i, \ldots, x_{n-1}) \right) dx_1 \ldots dx_{n-1} \\ &= \sum_{n \geq 1} \sum_{\mathbf{X}^{n-1}} n p_\Phi^{(n)}(x, x_1, \ldots, x_{n-1}) dx_1 \ldots dx_{n-1} \\ &= \sum_{n \geq 0} (n+1) \int_{\mathbf{X}^n} p_\Phi^{(n+1)}(x, x_1, \ldots, x_n) dx_1 \ldots dx_n\end{aligned} \tag{A.63}$$

which shows that the intensity function contains all realizations of Φ that consider x which is marginalized over all feasible cardinalities and the rest of the feasible states. Just as with random variables, probability densities of point processes are unavailable or become intractable to propagate, so another representation must be found using probability generating functionals.

A.2.2 Probability Generating Functionals

A generating functional is defined as a mapping of G about a function h on $\mathbf{X}$ with $\mathbf{X} \to \mathbb{R}^+$ to $\mathbb{R}$. In other words, a functional is a function of a sequence of functions where $(u_n)_{n \geq 0} : \mathbf{X}^n \to \mathbb{R}^+$. The functional G is given by

$$G(h) = \sum_{n \geq 0} \int_{\mathbf{X}^n} \left(\prod_{i=1}^{n} h(x_i) \right) u_k(x_1, \ldots, x_n) dx_1 \ldots dx_n, \tag{A.64}$$

about any $h : \mathbf{X} \to \mathbb{R}^+$ and sequence $(u_n)_{n\geq 0}$ that makes the resultant finite. The point process, Φ, with a density p_Φ can be substituted into the functional to produce the PGFL, $\mathcal{G}_\Phi(h)$, given by

$$\begin{aligned}\mathcal{G}_\Phi(h) &= \mathbb{E}\left[\prod_{x\in\Phi} h(x)\right] \\ &= \int_{\mathcal{X}} \left(\prod_{x\in\Phi} h(x)\right) p_\Phi(\phi)d\phi \\ &= \sum_{n\geq 0}\int_{\mathbf{X}^n}\left(\prod_{i=1}^{n} h(x_i)\right) p_\Phi^{(n)}(x_1,\ldots,x_n)dx_1\ldots dx_n.\end{aligned} \tag{A.65}$$

The generating sequence is a probability density which restricts the value for the test function h to $h : \mathbf{X} \to [0 \quad 1]$. This is very similar to the generating function for a random variable in which the test variable s is a real number constrained from $0 \leq s \leq 1$ while the test function h maps the space $\mathbf{X}$ to $[0 \quad 1]$. This mapping includes the sum over all possible cardinalities and adds the component which integrates over all of the individual agents' states. Similar to PGFs, PGFLs have the properties

$$\mathcal{G}_\Phi(0) = \sum_{n\geq 0}\int_{\mathbf{X}^n}\left(\prod_{i=1}^{n} 0\right) p_\Phi^{(n)}(x_1,\ldots,x_n)dx_1\ldots dx_n = \rho_\Phi(0), \tag{A.66a}$$

$$\mathcal{G}_\Phi(1) = \sum_{n\geq 0}\int_{\mathbf{X}^n}\left(\prod_{i=1}^{n} 1\right) p_\Phi^{(n)}(x_1,\ldots,x_n)dx_1\ldots dx_n = 1. \tag{A.66b}$$

For mapping $h : x \in \mathbf{X} \to 0$, the cardinality probability $\rho_\Phi(0)$ is extracted which corresponds to the the probability of no agents in the field. For mapping $h : x \in \mathbf{X} \to 0$, the PGFL becomes 1.

Just like with random variables for multi-target estimation, the joint behavior of point processes must also be studied. For example, a point process that consists of a measurement configuration can be found given the configuration of agents on the field. Thus, a joint PGFL $\mathcal{G}_{\Xi,\Phi}$ for two point processes is given by

$$\begin{aligned}\mathcal{G}_{\Xi,\Phi}(g,h) &= \mathbb{E}\left[\left(\prod_{z\in\Xi} g(z)\right)\left(\prod_{x\in\Phi} h(x)\right)\right] \\ &= \int_{\mathcal{Z}}\int_{\mathcal{X}} \left(\prod_{z\in\Xi} g(z)\right)\left(\prod_{x\in\Phi} h(x)\right) p_{\Xi,\Phi}(\xi,\phi)d\xi d\phi,\end{aligned} \tag{A.67}$$

where g is the test function for Ξ and $p_{\Xi,\Phi}(\xi,\phi)$ is the joint probability density per unit volume at ξ and ϕ. This is roughly the joint probability of $\Xi = \xi$ and $\Phi = \phi$. If the point processes Ξ and Φ are independent, the joint PGFL simplifies to

$$\begin{aligned}\mathcal{G}_{\Xi,\Phi}(g,h) &= \int_{\mathcal{Z}}\int_{\mathcal{X}} \left(\prod_{z\in\Xi} g(z)\right)\left(\prod_{x\in\Phi} h(x)\right) p_{\Xi}(\xi)p_{\Phi}(\phi)d\xi d\phi \\ &= \left(\int_{\mathcal{Z}} \left(\prod_{z\in\Xi} g(z)\right) p_{\Xi}(\xi)d\xi\right)\left(\int_{\mathcal{X}} \left(\prod_{x\in\Phi} h(x)\right) p_{\Phi}(\phi)d\phi\right) \\ &= \mathcal{G}_{\Xi}(g)\mathcal{G}_{\Phi}(h),\end{aligned} \tag{A.68}$$

where $p_{\Xi,\Phi}(\xi,\phi) = p_{\Xi}(\xi)p_{\Phi}(\phi)$. With PGFLs, several properties coincide to manipulate the PGFLs into other forms.

The derivative for a functional F at h is given by

$$F'(h) = \lim_{\eta\to 0} \frac{F(h+\eta) - F(h)}{\eta}, \tag{A.69}$$

where $\eta : \mathbb{X} \to [0 \quad 1]$ has the same form as h. Unfortunately, the convergence of η to 0 and division by η are not well defined. However, an alternative method exists to obtain the functional derivative. The chain derivative of F is given by

$$\delta F(h;\eta) = \lim_{n\to\infty} \frac{F(h+\epsilon_n\eta_n) - f(h)}{\epsilon_n} \tag{A.70}$$

which is evaluated at h and in the direction of η where $h, \eta : \mathbf{X} \to \mathbb{R}^+$. The sequence of $\{\eta_n\}_{n\geq 0}$ converges to η while the positive real number sequence $\{\epsilon_n\}_{n\geq 0}$ converges to 0. Note that a functional derivative contains the dependencies h and η, i.e. $\delta F(h;\eta)$, while an ordinary derivative is notated as $f'(x)$. Other than the generating functional given by Eq. (A.64), other functionals are of importance for the multi-target estimation problem. One example is

$$F(h) = \int h(x)dx, \tag{A.71}$$

which integrates the test function h over the state space $\mathbf{X}$. These have values that are $\mathbb{R}$. Another example are transformations of a function into another form in the space of real-valued functions given by

$$F(h)(\cdot) = [h(\cdot)]^2. \tag{A.72}$$

This transformation squares the real-valued function, and $(\cdot)$ denotes a real-valued function. The functionals in Eqs. (A.71) and (A.72) produce derivatives that are a real number or real-valued function, repectively. Note that when a functional derivative of Eq. (A.72) is taken, $\delta F(h;\eta)$ is a function evaluated at $x \in \mathbf{X}$ and becomes $\delta F(h;\eta)(x)$. To make the notation clearer, a simplified notation is given by

$$\delta F(h(x);\eta) = \delta F(h;\eta)(x). \tag{A.73}$$

The functional derivative and ordinary derivative are related by substituting a real-valued function f at $a \in \mathbb{R}$ and direction $b \in \mathbb{R}$ into Eq. (A.70) given by

$$\begin{aligned}\delta f(a;b) &= \lim_{n\to\infty} \frac{f(a+\epsilon_n b_n) - f(a)}{\epsilon_n} \\ &= \lim_{n\to\infty} b_n \frac{f(a+\epsilon_n b_n) - f(a)}{\epsilon_n b_n} \\ &= b \lim_{\epsilon\to 0} \frac{f(a+\epsilon) - f(a)}{\epsilon} \\ &= b f'(a).\end{aligned} \tag{A.74}$$

If $b = 1$, the equation reduces to

$$\delta f(a;1) = f'(a), \tag{A.75}$$

which relates the ordinary derivative to the functional derivative. The functional derivatives contains properties that are useful in deriving other quantities. For functionals, F and G, these properties are given by

$$\text{sum: } \delta(F+G)(h;\eta) = \delta F(h;\eta) + \delta G(h;\eta), \tag{A.76a}$$

$$\text{product: } \delta(F\cdot G)(h;\eta) = \delta F(h;\eta)G(h) + F(h)\delta G(h;\eta), \tag{A.76b}$$

$$\text{chain: } \delta(F\circ G)(h;\eta) = \delta F(G(h);\delta G(h;\eta)). \tag{A.76c}$$

It is also necessary to obtain higher-order derivations of chain differentiation. Originally, Di Bruno (1857) established a formula for higher-order chain differentials which can be used directly with functionals (Clark and Houssineau, 2013; Clark et al., 2015). Specifically, the 2nd order chain differential is used which is given by

$$\begin{aligned}\delta^2(F\circ G)(h;\eta_1,\eta_2) = \delta F(G(h);\delta^2 G(h;\eta_1,\eta_2)) \\ + \delta^2 F(G(h);\delta G(h;\eta_1),\delta G(h;\eta_2)).\end{aligned} \tag{A.77}$$

It is also useful to know the differentiation of several other PGFLs to derive the multi-target estimation problem. For a functional F_x such as $F_x(h) = h(x)$, where $x \in \mathbf{X}$ is a fixed point, the goal is to differentiate the functional that describes a single agent in the state x with a probability of 1. By substituting into Eq. (A.70), the functional derivative is given by

$$\begin{aligned}
\delta F_x(h;\eta) &= \lim_{n\to\infty} \frac{F_x(h+\epsilon_n\eta_n) - F_x(h)}{\epsilon_n} \\
&= \lim_{n\to\infty} \frac{h(x) + \epsilon_n\eta_n(x) - h(x)}{\epsilon_n} \\
&= \lim_{n\to\infty} \eta_n(x) \\
&= \eta(x).
\end{aligned} \tag{A.78}$$

So the functional derivative reduces to

$$\delta(h(x);\eta) = \eta(x). \tag{A.79}$$

Note that this is a shorthand notation of a more rigorous notation given by

$$\delta(\cdot \to \cdot(x))(h;\eta) = \eta(x). \tag{A.80}$$

The resultant functional derivative is a real number. It is also informative to obtain the same resultant using a different real test function. Specifically, an identity function can be used to obtain the same result. This is given by

$$\begin{aligned}\delta F_{id}(h;\eta) &= \lim_{n\to\infty}\frac{F_{id}(h+\epsilon_n\eta_n)-F_{id}(h)}{\epsilon_n}\\ &= \lim_{n\to\infty}\frac{h+\epsilon_n\eta_n-h}{\epsilon_n}\\ &= \lim_{n\to\infty}\eta_n\\ &= \eta.\end{aligned} \tag{A.81}$$

So for any $x \in \mathbf{X}$,

$$\begin{aligned}\delta F_{id}(h;\eta)(x) &= \eta(x)\\ &= \delta(h(x);\eta).\end{aligned} \tag{A.82}$$

The functional derivative result in Eq. (A.79) is generalized by expanding the expression to

$$\begin{aligned}\delta((h(x))^k;\eta) &= k\eta(x)(h(x))^{k-1}\\ &= \delta(\cdot \to (\cdot(x))^k)(h;\eta).\end{aligned} \tag{A.83}$$

Another functional of interest is $F(h) = \int h(x)f(x)dx$ about space $\mathbf{X}$. If it is assumed that $\int f(x)dx = 1$, the functional F can be a PGFL that describes a single agent with a state of probability 1 and distributed about f. By substituting into Eq.

(A.70), the functional derivative is

$$\begin{aligned}
\delta F(h;\eta) &= \lim_{n\to\infty} \frac{F(h+\epsilon_n\eta_n - F(h)}{\epsilon_n} \\
&= \lim_{n\to\infty} \frac{\int (h(x)+\epsilon_n\eta_n(x))f(x)dx - \int h(x)f(x)dx}{\epsilon_n} \\
&= \lim_{n\to\infty} \frac{\int h(x)f(x)dx + \epsilon_n \int \eta_n(x)f(x)dx - \int h(x)f(x)dx}{\epsilon_n} \\
&= \lim_{n\to\infty} \int \eta_n(x)f(x)dx \\
&= \int \eta(x)f(x)dx,
\end{aligned} \tag{A.84}$$

thus,

$$\begin{aligned}
\delta\left(\int h(x)f(x)dx;\eta\right) &= \int \eta(x)f(x)dx \\
&= \delta\left(\cdot \to \int \cdot(x)f(x)dx\right)(h;\eta).
\end{aligned} \tag{A.85}$$

It can be observed from the functional derivative that the functional $F : h \to \int h(x)f(x)dx$ shows up directly.

An additional functional to consider is $F(h) = \int G(h|x)f(x)dx$ about space $\mathbf{X}$ and functional G. If it is assumed that $\int f(x)dx = 1$, the functional F can be a PGFL that describes a point process that is marginalized about all possible x's for an agent and G can be a PGFL of a point process that depends on an agent state.

By substituting into Eq. (A.70), the functional derivative is

$$\begin{aligned}\delta F(h;\eta) &= \lim_{n\to\infty} \frac{F(h+\epsilon_n\eta_n - F(h)}{\epsilon_n} \\ &= \lim_{n\to\infty} \frac{\int G(h+\epsilon_n\eta_n|x)f(x)dx - \int G(h|x)f(x)dx}{\epsilon_n} \\ &= \int \lim_{n\to\infty} \frac{G(h+\epsilon_n\eta_n|x) - G(h|x)}{\epsilon_n} f(x)dx \\ &= \int \delta G(h|x;\eta)f(x)dx \\ &= F(\delta G(h|\cdot;\eta)),\end{aligned} \tag{A.86}$$

thus,

$$\begin{aligned}\delta\left(\int G(h|x)f(x)dx;\eta\right) &= \int \delta G(h|x;\eta)f(x)dx \\ &= \delta\left(\cdot \to \int G(\cdot|x)f(x)dx\right)(h;\eta).\end{aligned} \tag{A.87}$$

The functional derivative in Eq. (A.87) has an integral which does not depend on the test function h. Thus, the derivative δ and the integral can be rearranged to form the functional derivative

$$\begin{aligned}\delta\left(\int_{\mathcal{X}} G(h|\phi)f(\phi)d\phi;\eta\right) &= \int_{\mathcal{X}} \delta G(h|\phi;\eta)f(\phi)d\phi \\ &= \delta\left(\cdot \to \int_{\mathcal{X}} G(\cdot|\phi)f(\phi)d\phi\right)(h;\eta).\end{aligned} \tag{A.88}$$

One last functional of interest is the exponential. The exponential, just like with the derivation of the multi-target cardinality estimator, is used considerably due to the simple nature of taking the derivative. By comparing the relation between the functional and ordinary derivations in Eq. (A.74) and the exponential derivative in

Eq. (A.13a), the functional derivative for an exponential is given by

$$\delta \exp(a; b) = b \exp(a). \tag{A.89}$$

The exponential functional can also be substituted into the chain rule for functionals, Eq. (A.76c), with a corresponding general functional F given by

$$\delta(\exp \circ F)(h; \eta) = \delta F(h; \eta)(\exp \circ F)(h). \tag{A.90}$$

In comparison to the ordinary chain rule using the exponential, Eq. (A.13b), the exponential functional chain rule is very similar in form. Another useful property can be obtained by assuming that F is an identity functional, $F_{id}(h) = h$. By substituting into Eq. (A.90) and using the property obtained in Eq. (A.82), the expression simplifies to

$$\begin{aligned} \delta \exp(h(x); \eta) &= \delta \exp(h; \eta)(x) \\ &= \delta(\exp \circ F_{id})(h; \eta)(x) \\ &= \delta F_{id}(h; \eta)(x)(\exp \circ F_{id})(h)(x) \\ &= \eta(x) \exp(h(x)) \end{aligned} \tag{A.91}$$

With these properties for a functional, differentiation can be applied directly on the PGFL. For a point process Φ and a PGFL $\mathcal{G}_\Phi$, the goal is to determine the information needed, i.e. probability density $p_\Phi(x)$, for multi-target estimation. Similar to determining the mean and variance from PGFs, a similar derivation can be found for PGFLs. First, the goal is to differentiate G_Φ about δ_x given by

$$\delta \mathcal{G}_\Phi(h; \delta_x) = \delta \left(\sum_{n \geq 0} \int_{\mathbf{X}^n} \left(\prod_{i=1}^{n} h(x_i) \right) p_\Phi^{(n)}(x_1, \ldots, x_n) dx_1 \ldots dx_n; \delta_x \right). \tag{A.92}$$

The rearrangement of the derivative and integral can be applied directly to the expression from Eq. (A.88) which becomes

$$\delta \mathcal{G}_\Phi(h;\delta_x) = \sum_{n\geq 1} \int_{\mathbf{X}^n} \delta\left(\left(\prod_{i=1}^{n} h(x_i)\right); \delta_x\right) p_\Phi^{(n)}(x_1,\dots,x_n) dx_1 \dots dx_n, \tag{A.93}$$

which can be simplified further by using the product rule, Eq. (A.76b), to obtain

$$\delta \mathcal{G}_\Phi(h;\delta_x) = \sum_{n\geq 1} \int_{\mathbf{X}^n} \left(\sum_{i=1}^{n} \delta(h(x_i);\delta_x) \left[\prod_{j\neq i} h(x_j)\right]\right) p_\Phi^{(n)}(x_1,\dots,x_n) dx_1 \dots dx_n. \tag{A.94}$$

By using Eq. (A.79), the expression is simplified to the general form

$$\begin{aligned}
\delta \mathcal{G}_\Phi(h;\delta_x) &= \sum_{n\geq 1} \int_{\mathbf{X}^n} \left(\sum_{i=1}^{n} \delta_x(x_i) \left[\prod_{j\neq i} h(x_j)\right]\right) p_\Phi^{(n)}(x_1,\dots,x_n) dx_1 \dots dx_n \\
&= \sum_{n\geq 1} \int_{\mathbf{X}^{n-1}} \sum_{i=1}^{n} \left(\prod_{j\neq i} h(x_j)\right) \\
&\times p_\Phi^{(n)}(x_1,\dots,x,\dots,x_n) dx_1 \dots dx_{i-1} dx_{i+1} \dots dx_n \\
&= \sum_{n\geq 1} \int_{\mathbf{X}^{n-1}} n \left(\prod_{i=1}^{n-1} h(x_i)\right) p_\Phi^{(n)}(x, x_1,\dots,x_{n-1}) dx_1 \dots dx_{n-1} \\
&= \sum_{n\geq 1} n \int_{\mathbf{X}^{n-1}} \left(\prod_{i=1}^{n-1} h(x_i)\right) p_\Phi^{(n)}(x, x_1,\dots,x_{n-1}) dx_1 \dots dx_{n-1} \\
&= \sum_{n\geq 0} (n+1) \int_{\mathbf{X}^{n}} \left(\prod_{i=1}^{n} h(x_i)\right) p_\Phi^{(n+1)}(x, x_1,\dots,x_{n}) dx_1 \dots dx_{n}.
\end{aligned} \tag{A.95}$$

Similar to PGFs, the functional derivative of the PGFL can be evaluated at $h = 0$

and $h = 1$ given by

$$\begin{aligned}\delta\mathcal{G}_\Phi(h;\delta_x)|_{h=0} &= \sum_{n\geq 0}(n+1)\int_{\mathbf{X}^n}\left(\prod_{i=1}^{n} 0\right) p_\Phi^{(n+1)}(x,x_1,\ldots,x_n)dx_1\ldots dx_n \\ &= p_\Phi^{(1)}(x),\end{aligned} \tag{A.96a}$$

$$\begin{aligned}\delta\mathcal{G}_\Phi(h;\delta_x)|_{h=1} &= \sum_{n\geq 0}(n+1)\int_{\mathbf{X}^n}\left(\prod_{i=1}^{n} 1\right) p_\Phi^{(n+1)}(x,x_1,\ldots,x_n)dx_1\ldots dx_n \\ &= \sum_{n\geq 0}(n+1)\int_{\mathbf{X}^n} p_\Phi^{(n+1)}(x,x_1,\ldots,x_n)dx_1\ldots dx_n \\ &= \mu_\Phi(x).\end{aligned} \tag{A.96b}$$

Higher order differentiation of Eq. (A.95) evaluated at $h = 0$ will yield a probability density calculated at a set of a desired size. Higher order differentiation of Eq. (A.95) evaluated at $h = 1$ yields factorial moment densities which is not used in this derivation. Thus, the PGFL $\mathcal{G}_\Phi$ provides a full characterization of the point process Φ given by

$$p_\Phi^{(k)}(x_1,\ldots,x_k) = \frac{1}{k!}\delta^k\mathcal{G}_\Phi(h;\delta_{x_1},\ldots,\delta_{x_k})|_{h=0} = \frac{j_\Phi^{(k)}(x_1,\ldots,x_n)}{k!}, \tag{A.97a}$$

$$\mu_\Phi(x) = \delta\mathcal{G}_\Phi(h;\delta_x)|_{h=1}, \tag{A.97b}$$

which follows Eq. (A.16) very similarly.

Another necessary tool to derive the multi-target PHD filter using point processes is Campbell's theorem. The goal in this theorem is to evaluate a real-valued function f on space $\mathbf{X}$ for each point $x \in \phi$. That is, the goal is to find the expected value of

f about Φ. This equation is given by

$$\begin{aligned}\mathbb{E}\left[\sum_{x\in\phi} f(x)\right] &= \sum_{n\geq 1}\int_{\mathbf{X}^n}\left(\sum_{i=1}^{n} f(x_i)\right) p_{\Phi}^{(n)}(x_1,\ldots,x_n)dx_1\ldots dx_n \\ &= \sum_{n\geq 1}\left(\sum_{i=1}^{n}\int_{\mathbf{X}^n} f(x_i)p_{\Phi}^{(n)}(x_1,\ldots,x_n)dx_1\ldots dx_n\right),\end{aligned} \tag{A.98}$$

and since p_Φ is symmetrical

$$\begin{aligned}\mathbb{E}\left[\sum_{x\in\phi} f(x)\right] &= \sum_{n\geq 1} n\int_{\mathbf{X}^n} f(x_1)p_{\Phi}^{(n)}(x_1,\ldots,x_n)dx_1\ldots dx_n \\ &= \int f(x)\left(\sum_{n\geq 1} n\int_{\mathbf{X}^{n-1}} p_{\Phi}^{(n)}(x,x_1,\ldots,x_{n-1})dx_1\ldots dx_{n-1}\right)dx \\ &= \int f(x)\left(\sum_{n\geq 0}(n+1)\int_{\mathbf{X}^n} p_{\Phi}^{(n+1)}(x,x_1,\ldots,x_n)dx_1\ldots dx_n\right)dx.\end{aligned} \tag{A.99}$$

Then, Eq. (A.96b) can be substituted into the expression to obtain

$$\mathbb{E}\left[\sum_{x\in\phi} f(x)\right] = \int f(x)\mu_\Phi(x)dx = \int_{\mathcal{X}}\left(\sum_{x\in\phi} f(x)\right)p_\Phi(\phi)d\phi. \tag{A.100}$$

This expressions shows that evaluating f at each $x \in \mathbf{X}$ times the average number of agents at x is identical to evaluating f at each agent $x \in \phi$ over all realizations of Φ. This reduces the sequence of points of $\mathcal{X}$ to a smaller space $\mathbf{X}$.

Differentiation of joint PGFLs can also occur. The PGFL that describes the joint behavior of Ξ and Φ, where Ξ has the realization $\xi = (z_1,\ldots,z_m)$, is

$$\begin{aligned}\mathcal{G}_{\Xi=\xi,\Phi}(h) &= \int_{\mathcal{X}}\left(\prod_{x\in\phi} h(x)\right)p_{\Xi,\Phi}(\xi,\phi)d\phi \\ &= \frac{1}{m!}\delta^m\mathcal{G}_{\Xi,\Phi}(g,h;\delta_{z_1},\ldots,\delta_{z_m})|_{g=0},\end{aligned} \tag{A.101}$$

which is derived from Eqs. (A.65) and (A.96). Note that points z relate to g over an observation space $\mathbf{Z}$ while points x, y relate to h over a state space $\mathbf{X}$.

Just like with PGFs, operations can also occur with PGFLs. The first operation of discussion is marginalization between two joint point processes Ξ and Φ. The goal is to obtain the behavior of Ξ by marginalizing or integrating the joint behavior over Φ. This is given by

$$\forall \xi \in \mathbf{Z}, \quad p_\Xi(\xi) = \int_\mathcal{X} p_{\Xi,\Phi}(\xi, \phi) d\phi. \tag{A.102}$$

Thus, for a known joint PGFL, $\mathcal{G}_{\Xi,\Phi}$, the PGFL $\mathcal{G}_\Xi(g)$ is found by a marginalization given by

$$\begin{aligned}
\mathcal{G}_{\Xi,\Phi}(g, 1) &= \int_\mathcal{Z} \int_\mathcal{X} \left(\prod_{z \in \xi} g(z) \right) \left(\prod_{x \in \phi} 1 \right) p_{\Xi,\Phi}(\xi, \phi) d\xi d\phi \\
&= \int_\mathcal{Z} \left(\prod_{z \in \xi} g(z) \right) \left(\int_\mathcal{X} p_{\Xi,\Phi}(\xi, \phi) d\phi \right) d\xi \\
&= \int_\mathcal{Z} \left(\prod_{z \in \xi} g(z) \right) p_\Xi(\xi) d\xi \\
&= \mathcal{G}_\Xi(g).
\end{aligned} \tag{A.103}$$

The operation that involves the superposition of point processes is also of interest for PGFLs for scenarios where Φ_1 and Φ_2 are independent. If Ξ is the union of point processes, Φ_1 and Φ_2, the superposition of the two point process can be formed by

$$\begin{aligned}
\mathcal{G}_{\Xi}(h) &= \mathbb{E}\left[\prod_{x\in\Xi} h(x)\right] \\
&= \mathbb{E}\left[\prod_{x\in\Phi_1\cup\Phi_2} h(x)\right] \\
&= \mathbb{E}\left[\prod_{x_1\in\Phi_1} h(x) \prod_{x_2\in\Phi_2} h(x)\right] \\
&= \mathbb{E}\left[\prod_{x_1\in\Phi_1} h(x)\right] \mathbb{E}\left[\prod_{x_2\in\Phi_2} h(x)\right] \\
&= \mathcal{G}_{\Phi_1}(h)\mathcal{G}_{\Phi_2}(h)
\end{aligned} \tag{A.104}$$

using Eq. (A.65). To go from the third to the fourth step, Φ_1 and Φ_2 are assumed to be independent of each other.

One last operation of interest is branching which is a special kind of dependence between Ξ and Φ. For a realization $\xi = (z_1, \ldots, z_m)$ of parent process Ξ, the daughter process Φ will have a superposition m but with an independent point process $\Upsilon|z_i$. This operation is very similar to the spawning description for PGFs. Suppose that the PGFLs for the parent, Ξ and transitional process, $\Upsilon|\cdot$, are known, and the goal is to describe the daughter process Φ. The PGFL that describes the joint behavior is

$$\begin{aligned}
\mathcal{G}_{\Xi,\Phi}(g,h) &= \int_{\mathcal{Z}}\int_{\mathcal{X}} \left(\prod_{z\in\xi} g(z)\right)\left(\prod_{x\in\phi} h(x)\right) p_{\Xi,\Phi}(\xi,\phi)d\xi d\phi \\
&= \int_{\mathcal{Z}}\int_{\mathcal{X}} \left(\prod_{z\in\xi} g(z)\right)\left(\prod_{x\in\phi} h(x)\right) p_{\Xi}(\xi)p_{\Phi|\Xi}(\phi|\xi)d\xi d\phi \\
&= \int_{\mathcal{Z}} \left(\prod_{z\in\xi} g(z)\right)\left(\int_{\mathcal{X}}\left(\prod_{x\in\phi} h(x)\right) p_{\Phi|\Xi}(\phi|\xi)d\phi\right) p_{\Xi}(\xi)d\xi \\
&= \int_{\mathcal{Z}} \left(\prod_{z\in\xi} g(z)\right) \mathcal{G}_{\Phi|\Xi}(h|\xi)p_{\Xi}(\xi)d\xi,
\end{aligned} \tag{A.105}$$

where $\mathcal{G}_{\Phi|\Xi}(h|\xi)$ is the conditional PGFL of Φ given the realization $\Xi = \xi$. With the realization $\Xi = \xi$, $\Phi|\Xi$ is a superposition of $|\xi|$ independent $\Upsilon|z_i : z_i \in \xi$. So,

$$\mathcal{G}_{\Phi|\Xi}(h|\xi) = \prod_{z\in\xi} \mathcal{G}_{\Upsilon}(h|z). \tag{A.106}$$

The superposition relation is substituted into Eq. (A.105) to obtain

$$\begin{aligned}\mathcal{G}_{\Xi,\Phi}(g,h) &= \int_{\mathcal{Z}} \left(\prod_{z\in\xi} g(z)\right)\left(\prod_{z\in\xi}\mathcal{G}_{\Upsilon}(h|z)\right) p_{\Xi}(\xi)d\xi \\ &= \int_{\mathcal{Z}} \left(\prod_{z\in\xi} g(z)\mathcal{G}_{\Upsilon}(h|z)\right) p_{\Xi}(\xi)d\xi \\ &= \mathcal{G}_{\Xi}(g\mathcal{G}_{\Upsilon}(h|\cdot)).\end{aligned} \tag{A.107}$$

This reduces the joint behavior of two point processes in terms of conditional PGFLs and transitional point processes. Branching is used to derive the measurement update for the PHD filter, and it is in a similar form to the PGF branching behavior given in Eq. (A.23). To obtain the description of Φ, marginalization is used about Ξ which is given by

$$\begin{aligned}\mathcal{G}_{\Phi}(h) &= \mathcal{G}_{\Xi,\Phi}(1,h) \\ &= \mathcal{G}_{\Xi}(\mathcal{G}_{\Upsilon}(h|\cdot)).\end{aligned} \tag{A.108}$$

A.2.3 Examples of Point Process and PGFLs

With the basics of point processes and PGFLs, families of these quantities are discussed which is used directly in multi-target filtering.

The first of these point processes is a Bernoulli point process Φ which has a parameter $0 \le p \le 1$ and a spatial distribution $s : \int s(x)dx = 1$. The Bernoulli point process is given by

$$\Phi = \begin{cases} \emptyset, & \text{if } 1-p, \\ x, & \text{if } ps(x), \end{cases} \tag{A.109}$$

and it describes a case in which there is an agent in the environment with a state distributed about s or a case in which there is no agent in the environment. Thus, it can describe the behavior of an individual agent or a measurement. The associated PGFL for a Bernoulli point process is given by

$$\begin{aligned} \mathcal{G}_\Phi(h) &= \sum_{n\geq 0} \int_{\mathbf{X}^n} \left(\prod_{i=1}^{n} h(x_i) \right) p_\Phi^{(n)}(x_1, \ldots, x_n) dx_1 \ldots dx_n \\ &= p_\Phi^{(0)}(\emptyset) + \int h(x) p_\Phi^{(1)}(x) dx \\ &\quad + \sum_{n\geq 2} \int_{\mathbf{X}^n} \left(\prod_{i=1}^{n} h(x_i) \right) p_\Phi^{(n)}(x_1, \ldots, x_n) dx_1 \ldots dx_n \\ &= 1 - p + p \int h(x) s(x) dx, \end{aligned} \tag{A.110}$$

where $p_\Phi^{(0)}(\emptyset) = 1-p$, $p_\Phi^{(1)}(x) = ps(x)$, and $p_\Phi^{(n)}(x_1, \ldots, x_n) = 0$ in the second step.

The Poisson point process is also used in the derivation of the PHD filter. The Poisson point process, Φ, is defined by a parameter $\lambda \geq 0$ and spatial distribution s given by

$$\forall n \geq 0, \quad |\Phi| = n, \quad p_\Phi = \exp(-\lambda) \frac{\lambda^n}{n!}, \tag{A.111}$$

where the agent states are independently and identically distributed about s. It is used for modeling a population in which the number of elements is Poisson distributed. The point patterns produce spatial randomness by using a spatial distribution that is uniform over the state space. The PGFL for a point process is given by

$$\begin{aligned}\mathcal{G}_\Phi(h) &= \int_{\mathcal{X}} \left(\prod_{x\in\phi} h(x)\right) p_\Phi(\phi)d\phi \\ &= \sum_{n\geq 0} \int_{\mathbf{X}^n} \left(\prod_{i=1}^{n} h(x_i)\right) \exp(-\lambda)\frac{\lambda^n}{n!} \left(\prod_{i=1}^{n} s(x_i)\right) dx_1 \dots dx_n \\ &= \exp(-\lambda)\sum_{n\geq 0} \frac{\lambda^n}{n!} \int_{\mathbf{X}^n} \left(\prod_{i=1}^{n} h(x_i)s(x_i)\right) \\ &= \exp(-\lambda)\sum_{n\geq 0} \frac{\lambda^n}{n!} \left(\int h(x)s(x)dx\right)^n \\ &= \exp(-\lambda)\sum_{n\geq 0} \frac{\left(\lambda \int h(x)s(x)dx\right)^n}{n!}.\end{aligned} \tag{A.112}$$

Then using the exponential Taylor expansion in Eq. (A.13c), the expression simplifies to

$$\begin{aligned}\mathcal{G}_\Phi(h) &= \exp\left(-\lambda\right)\exp\left(\lambda \int h(x)s(x)dx\right) \\ &= \exp\left(\lambda(\int h(x)s(x)dx - 1)\right).\end{aligned} \tag{A.113}$$

As discussed with Poisson random variables, the PGFL of a Poisson point process is an exponential, thus, it can be differentiated easily and produce tractable filtering recursions. Specifically, the first moment density can be obtained for some point $y \in \mathcal{X}$ by differentiating the PGFL with Eq. (A.97b) given by

$$\begin{aligned}\mu_\Phi(y) &= \delta\mathcal{G}_\Phi(h;\delta_y)|_{h=1} \\ &= \delta\left(\exp\left(\lambda\left(\int h(x)s(x)dx - 1\right)\right);\delta_y\right)\Bigg|_{h=1}.\end{aligned} \tag{A.114}$$

The chain rule in Eq. (A.90) is used which reduces the equation to

$$\begin{aligned}\mu_\Phi(y) &= \delta\left(\lambda\left(\int h(x)s(x)dx - 1\right);\delta_y\right)\exp\left(\lambda\left(\int h(x)s(x)dx-1\right)\right)\Big|_{h=1}\\ &= \lambda\delta\left(\int h(x)s(x)dx;\delta_y\right)\Big|_{h=1}\exp\left(\lambda\left(\int s(x)dx-1\right)\right)\\ &= \lambda\delta\left(\int h(x)s(x)dx;\delta_y\right)|_{h=1}.\end{aligned} \tag{A.115}$$

This expression can be simplified further using Eq. (A.85) which becomes

$$\begin{aligned}\mu_\Phi(y) &= \lambda\int \delta_y(x)s(x)dx\\ &= \lambda s(y).\end{aligned} \tag{A.116}$$

Thus, the Poisson point process intensity (first moment density) is the spatial distribution times the Poisson rate which fully characterizes the Poisson point process Φ. If the intensity μ_Φ is known, the parameters λ and $s(\cdot)$ can be found by

$$\lambda = \int \mu_\Phi(x)dx, \tag{A.117a}$$

$$s(\cdot) = \lambda^{-1}\mu_\Phi(\cdot). \tag{A.117b}$$

Also the PGFL can be expressed by the intensity using Eqs. (A.113) and (A.117) given by

$$\mathcal{G}_\Phi(h) = \exp\left(\int (h(x)-1)\mu_\Phi(x)dx\right), \tag{A.118}$$

in which μ_Φ is propagated by the PHD filter and is useful for deriving the PHD filter equations.

A.2.4 PHD Filter Derivation

With the introduction to point processes and PGFLs, these tools can be used to derive the PHD filter to estimate the number of agents in the swarm and their states. The goal is to estimate and propagate the mean number of agents and their states through time by observing measurements. The construction of the problem consists of relating Φ, the agents after the time-update, from Ψ, the agents before the time-update, and relating $\Phi|\Xi$ from X where Ξ contains the current measurements and $X|Z$ contains the measurement updated agent estimate. The other part of this problem consists of extracting information (i.e. μ_Φ, μ_Ψ, and $\mu_{\Phi|\Xi}$) from differentiation of the PGFLs to determine the PHD estimates. Both of these steps take in assumptions about the physical environment that the agents interact with. Thus, a proper set-up is required to obtain the best estimates from the filter.

For the time-update, it is assumed that the agents are independent of each other, each agent with state $x \in \mathbf{X}$ survives and evolves with probability $p_s(x)$ to a new state $y \in \mathbf{X}$ according to distribution $m(y|x)$ or dies with probability, $1 - p_s(x)$, and agents birth independently from the surviving agents with point process Φ_b and $\mathbf{G}_b$. The survival point process Φ_s is a Bernoulli point process with a survival parameter $p_s(x)$ and spatial distribution $m(\cdot|x)$ given by

$$\mathbf{G}_s(h|\cdot) = 1 - p_s(\cdot) + p_s(s) \int h(y)m(y|\cdot)dy. \tag{A.119}$$

Comparing the PGF in Eq. (A.31) to the PGFL in Eq. (A.119), the PGFL considers the spatial distribution (the states) of the surviving agents in the formulation. Specifically, the spatial distribution is dependent on the previous state x, thus the point process $\mathcal{G}_s$ is dependent on the prior states. The surviving agents are denoted by a point process Φ_{sur} which occurs by branching the parent point process Ψ with the transition point process Φ_s given by

$$\mathcal{G}_{sur}(h) = \mathcal{G}_\Psi(\mathcal{G}_s(h|\cdot)). \tag{A.120}$$

The time-updated agents are described by Ψ which is the superposition of the surviving and birthed agents given by

$$\mathcal{G}_\Phi(h) = \mathcal{G}_{sur}(h)\mathcal{G}_b(h). \tag{A.121}$$

Therefore, the predicted PGFL for the PHD filter is

$$\mathcal{G}_\Phi(h) = \mathcal{G}_\Psi\left(1 - p_s(\cdot) + p_s(\cdot)\int h(y)m(y|\cdot)dy\right)\mathcal{G}_b(h) \tag{A.122}$$

which provides the entire description of agents as they birth or survive to the next time-step without computing each agent's individual probability $p_\Phi(\phi) : \phi \in \mathcal{X}$.

For the measurement update, it is assumed that measurements are independently measured from each other, an agent with state $x \in \mathbf{X}$ is detected with a probability $p_d(x)$ from a single measurement $z \in Z$ according to distribution $l(z|x)$ or it is not detected with a probability $1 - p_d(x)$, and any clutter measurements (with point process Ξ_c and PGFL $\mathcal{G}_c$) are obtained independently from agent measurements. The observation point process $\Xi_{obs}|x$ is a Bernoulli point process with a detection parameter $p_d(x)$ and a spatial distribution $l(\cdot|x)$ given by

$$\mathcal{G}_{obs}(g|\cdot) = 1 - p_d(\cdot) + p_d(\cdot)\int g(z)l(z|\cdot)dz. \tag{A.123}$$

The agent measurement point process is denoted by Ξ_{tar} which occurs by branching the parent point process Φ with the transition point process Ξ_{obs} given by

$$\mathcal{G}_{\Xi_{tar},\Phi}(g,h) = \mathcal{G}_{\Phi}(h\mathcal{G}_{obs}(g|\cdot)). \tag{A.124}$$

The joint PGFL of measured agents is the superposition of the measurements and clutter given by

$$\mathcal{G}_{\Xi,\Phi}(g,h) = \mathcal{G}_{\Xi_{tar},\Phi}(g,h)\mathcal{G}_{c}(g), \tag{A.125}$$

therefore, by substituting Eqs. (A.123) and (A.124) into the expression, the joint PGFL becomes

$$\mathcal{G}_{\Xi,\Phi}(g,h) = \mathcal{G}_{\Phi}\left(h(1-p_d(\cdot)+p_d(\cdot)\int g(z)l(z|\cdot)dz)\right)\mathcal{G}_{c}(g). \tag{A.126}$$

With the measurement update PGFL, the goal is to estimate the agent state and cardinality conditioned on $\Xi = Z$ where $Z = (z_1, \ldots, z_m)$ is the measurement set. By using Bayes' rule given by

$$p_{\Phi|\Xi}(\phi|Z) = \frac{p_{\Xi,\Phi}(Z,\phi)}{p_{\Xi}(Z)}, \tag{A.127}$$

a posterior probability of $\Phi = \phi$ agents given $\Xi = Z$ measurements can be found using the joint probability of $\Phi = \phi$ agents and $\Xi = Z$ measurement and the probability of $\Xi = Z$ measurements. Bayes' rule can be manipulated by multiplying $\prod_{x\in\phi} h(x)$ on both sides and integrating up all realizations of Φ given by

$$\int_{\mathcal{X}}\left(\prod_{x\in\phi} h(x)\right)p_{\Phi|\Xi}(\phi)d\phi = \frac{\int_{\mathcal{X}}\left(\prod_{x\in\phi} h(x)\right)p_{\Xi,\Phi}(Z,\Phi)d\phi}{p_{\Xi}(Z)}. \tag{A.128}$$

By substituting Eqs. (A.65), and (A.101) into Eq. (A.128), the expression is reduced to

$$\mathcal{G}_{\Phi|\Xi}(h|Z) = \frac{\mathcal{G}_{\Xi=Z,\Phi}(h)}{p_{\Xi}(Z)}, \tag{A.129}$$

which can be simplified further using Eqs. (A.101) and (A.97a) to become

$$\mathcal{G}_{\Phi|\Xi}(h|Z) = \frac{\frac{1}{m!}\delta^m \mathcal{G}_{\Xi,\Phi}(g,h;\delta_{z_1},\ldots,\delta_{z_m})|_{g=0}}{\frac{1}{m!}\delta^m \mathcal{G}_{\Xi}(g;\delta_{z_1},\ldots\delta_{z_m})|_{g=0}}, \tag{A.130}$$

where $\mathcal{G}_{\Xi,\Phi}(g,h) = \mathcal{G}_{\Phi}\left(h(1-p_d(\cdot)+p_d(\cdot)\int g(z)l(z|\cdot)dz)\right)\mathcal{G}_c(g)$. This is the measurement update for the PGFL, and these PGFLS can be exploited using differentiation to obtain the PHD filter estimates directly.

For the time-update, μ_{Φ} is determined by Eq. (A.97b) about $\mathcal{G}_{\Phi}$ given by

$$\mu_{\Phi}(x) = \delta\mathcal{G}_{\Phi}(h;\delta_x)|_{h=1}. \tag{A.131}$$

By substituting Eq. (A.122) and using the product rule in Eq. (A.76b), the equation simplifies to

$$\begin{aligned}\mu_{\Phi}(x) &= \delta(\mathcal{G}_{\Psi}(\mathcal{G}_s(h|\cdot))\mathcal{G}_b(h);\delta_x)|_{h=1}\\ &= \delta(\mathcal{G}_{\Psi}(\mathcal{G}_s(h|\cdot));\delta_x)|_{h=1}\mathcal{G}_b(h)|_{h=1} + \mathcal{G}_{\Psi}(\mathcal{G}_s(h|\cdot))|_{h=1}\delta\mathcal{G}_b(h;\delta_x)|_{h=1}\\ &= \delta(\mathcal{G}_{\Psi}(\mathcal{G}_s(h|\cdot));\delta_x)|_{h=1} + \delta\mathcal{G}_b(h;\delta_x)|_{h=1},\end{aligned} \tag{A.132}$$

where $\mathcal{G}_b(h)|_{h=1} = \mathcal{G}_b(1) = 1$ and $\mathcal{G}_{\Psi}(\mathcal{G}_s(h|\cdot))|_{h=1} = \mathcal{G}_{\Psi}(\mathcal{G}_s(1|\cdot)) = \mathcal{G}_{\Psi}(1) = 1$. By using the relationship between the intensity and the first functional derivative in Eq. (A.97b), the equation becomes

$$\mu_{\Phi}(x) = \delta(\mathcal{G}_{\Psi}(\mathcal{G}_s(h|\cdot));\delta_x)|_{h=1} + \mu_b(x). \tag{A.133}$$

To obtain a form for $\delta(\mathcal{G}_{\Psi}(\mathcal{G}_s(h|\cdot));\delta_x)|_{h=1}$, Campbell's theorem in Eq. (A.100) can be used. Note although Campbell's theorem is used for a general form of Ψ, if Ψ were assumed Poisson, Eq. (A.90) which is a special case for chain rule using exponentials can be used.

The definition for the PGFL for $\delta(\mathcal{G}_\Psi(\mathcal{G}_s(h|\cdot)); \delta_x)|_{h=1}$ is given by

$$\delta(\mathcal{G}_\Psi(\mathcal{G}_s(h|\cdot)); \delta_x)|_{h=1} = \delta\left(\int_{\mathcal{X}}\left(\prod_{\bar{x}\in\phi}\mathcal{G}_s(h|\bar{x})\right)p_\Phi(\phi)d\phi; \delta_x\right)\Bigg|_{h=1}, \tag{A.134}$$

using Eq. (A.65), and the derivative and integral can be rearranged using Eq. (A.88) to

$$\delta(\mathcal{G}_\Psi(\mathcal{G}_s(h|\cdot)); \delta_x)|_{h=1} = \int_{\mathcal{X}}\delta\left(\prod_{\bar{x}\in\phi}\mathcal{G}_s(h|\bar{x}); \delta_x\right)\Bigg|_{h=1} p_\Phi(\phi)d\phi. \tag{A.135}$$

Next, the product rule is used to expand the expression given by

$$\begin{aligned}\delta(\mathcal{G}_\Psi(\mathcal{G}_s(h|\cdot)); \delta_x)|_{h=1} &= \int_{\mathcal{X}}\sum_{\bar{x}\in\phi}\left(\delta\mathcal{G}_s(h|\bar{x}; \delta_x)|_{h=1}\prod_{\bar{x}\in\phi}\mathcal{G}_s(h|\bar{x})|_{h=1}\right)p_\Phi(\phi)d\phi \\ &= \int_{\mathcal{X}}\left(\sum_{\bar{x}\in\phi}\delta\mathcal{G}_s(h|\bar{x}; \delta_x)|_{h=1}\right)p_\Phi(\phi)d\phi,\end{aligned} \tag{A.136}$$

where $\mathcal{G}_s(h|\bar{x})|_{h=1} = \mathcal{G}_s(1|\bar{x}) = 1$. Since the inner functional $\mathcal{G}_s$ is a simple Bernoulli process, the inner functional can be reduced to

$$\begin{aligned}\delta\mathcal{G}_s(h|\bar{x}; \delta_x)|_{h=1} &= \delta\left(1 - p_s(\bar{x}) + p_s(\bar{x})\int h(y)m(y|\bar{x})dy; \delta_x\right)\Bigg|_{h=1} \\ &= p_s(\bar{x})\delta\left(\int h(y)m(y|\bar{x})dy; \delta_x\right)\Bigg|_{h=1},\end{aligned} \tag{A.137}$$

and simplified further using Eq. (A.85) to

$$\begin{aligned}\delta\mathcal{G}_s(h|\bar{x};\delta_x)|_{h=1} &= p_s(\bar{x})\int \delta_x(y)m(y|\bar{x})dy \\ &= p_s(\bar{x})m(x|\bar{x}).\end{aligned} \tag{A.138}$$

By substituting Eq. (A.138) back into Eq. (A.136), the equation yields a Campbell's theorem form given by

$$\delta(\mathcal{G}_\Psi(\mathcal{G}_s(h|\cdot));\delta_x)|_{h=1} = \int_{\mathcal{X}}\left(\sum_{\bar{x}\in\phi} p_s(\bar{x})m(x|\bar{x})\right)p_\Psi(\phi)d\phi, \tag{A.139}$$

which can be reduced using Campbell's theorem, Eq. (A.100), to obtain

$$\delta(\mathcal{G}_\Psi(\mathcal{G}_s(h|\cdot));\delta_x)|_{h=1} = \int p_s(\bar{x})m(x|\bar{x})\mu_\Psi(\bar{x})d\bar{x}. \tag{A.140}$$

Thus, the time-update intensity recursion for the PHD filter can be extracted by substituting Eq. (A.140) into Eq. (A.133) to yield

$$\mu_\Phi(x) = \int p_s(\bar{x})m(x|\bar{x})\mu_\Psi(\bar{x})d\bar{x} + \mu_b(x). \tag{A.141}$$

This provides a direct time-update without assuming anything about the model of Ψ or Φ_b.

For the measurement update, the goal is to derive the intensity of the posterior agents given by

$$\mu_{\Phi|\Xi}(x|Z) = \delta\mathcal{G}_{\Phi|\Xi}(h|Z)|_{h=1}. \tag{A.142}$$

This follows a similar derivation to finding the posterior using PGFs. By substituting

Eq. (A.130) into the expression, the equation becomes

$$\mu_{\Phi|\Xi}(x|Z) = \frac{\delta^{m+1}\mathcal{G}_{\Xi,\Phi}(g,h;\delta_{z_1},\ldots,\delta_{z_m},\delta_x)|_{g=0,h=1}}{\delta^m\mathcal{G}_{\Xi,\Phi}(g,1;\delta_{z_1,\ldots,\delta_{z_m}})|_{g=0}}. \tag{A.143}$$

The equation from Eq. (A.143) becomes intractable without any assumptions on the predicted Φ or the clutter. But, if the predicted agents, Φ, and clutter, Z_c, are Poisson processes, tractable solutions can be determined using Eq. (A.118). This becomes

$$\begin{aligned}\mathcal{G}_{\Xi,\Phi} &= \mathcal{G}_\Phi\left(h(1-p_d(\cdot)+p_d(\cdot)\int g(z)l(z|\cdot)dz\right)\mathcal{G}_c(g)\\ &= \exp\left(\int(h(y)(1-p_d(y)+p_d(y)\int g(z)l(z|y)dz)-1)\mu_\Phi(y)dy\right)\\ &\times\exp\left(\int(g(z)-1)\mu_c(z)dz\right)\\ &= \exp\left(\int(h(y)(1-p_d(y)+p_d(y)\int g(z)l(z|y)dz)-1)\mu_\Phi(y)dy\right.\\ &\qquad\left.+\int(g(z)-1)\mu_c(z)dz\right)\\ &= \exp\left(F(g,h)\right),\end{aligned} \tag{A.144}$$

with an inner functional given by

$$\begin{aligned}F(g,h) = &\int\left[h(y)(1-p_d(y)+p_d(y)\int g(z)l(z|y)dz)-1\right]\mu_\Phi(y)dy\\ &+\int[g(z)-1]\mu_c(z)dz.\end{aligned} \tag{A.145}$$

With the exponential form, the equation can be simplified using Eq. (A.90) to

$$\begin{aligned}\delta\mathcal{G}_{\Xi,\Phi}(g,h;\delta_{z_1}) &= \delta\left(\exp\left(F(g,h)\right);\delta_{z_1}\right)\\ &= \delta F(g,h;\delta_{z_1})\exp\left(F(g,h)\right).\end{aligned} \tag{A.146}$$

For $\delta F(g,h;\delta_{z_1})$, the integral and the differential can be rearranged and differentiated using Eqs. (A.87) and (A.85), respectively, to obtain

$$\begin{aligned}\delta F(g,h;\delta_{z_1}) &= \int h(y)p_d(y)\delta\left(\int g(z)l(z|y)dz;\delta_{z_1}\right)\mu_\Phi(y)dy \\ &\quad + \delta\left(\int g(z)\mu_c(z)dz;\delta_{z_1}\right) \\ &= \int h(y)p_d(y)\left(\int \delta_{z_1}(z)l(z|y)dz\right)\mu_\Phi(y)dy + \int \delta_{z_1}(z)\mu_c(z)dz \\ &= \int h(y)p_d(y)l(z_1|y)\mu_\Phi(y)dy + \mu_c(z_1).\end{aligned} \tag{A.147}$$

Therefore, the first-order functional derivative is

$$\delta\mathcal{G}_{\Xi,\Phi}(g,h;\delta_{z_1}) = \left[\int h(y)p_d(y)l(z_1|y)\mu_\Phi(y)dy + \mu_c(z_1)\right]\exp(F(g,h)). \tag{A.148}$$

The term in front of the exponential is independent of g, so the mth derivative of the joint PGFL with respect to g is

$$\delta^m\mathcal{G}_{\Xi,\Phi}(g,h;\delta_{z_1},\ldots,\delta_{z_m}) = \prod_{i=1}^{m}\left[\int h(y)p_d(y)l(z_i|y)\mu_\Phi(y)dy + \mu_c(z_i)\right]\exp(F(g,h)). \tag{A.149}$$

The next step is to differentiate the expression with respect to h to obtain the numerator in Eq. (A.130). First, the product rule, Eq. (A.76b), is used to get

$$
\begin{aligned}
&\delta^{m+1}\mathcal{G}_{\Xi,\Phi}(g,h;\delta_{z_1},\ldots,\delta_{z_m},\delta_x) \\
&= \sum_{i=1}^{m}\Bigg[\delta\left(\int h(y)p_d(y)l(z_i|y)\mu_\Phi(y)dy+\mu_c(z_i);\delta_x\right) \\
&\qquad \prod_{\substack{j=1\\ j\neq i}}^{m}\left[\int h(y)p_d(y)l(z_j|y)\mu_\Phi(y)dy+\mu_c(z_j)\right]\Bigg]\exp(F(g,h)) \\
&+\prod_{i=1}^{m}\left[\int h(y)p_d(y)l(z_i|y)\mu_\Phi(y)dy+\mu_c(z_i)\right]\delta(\exp(F(g,h)),\delta_x),
\end{aligned}
\tag{A.150}
$$

and then Eq. (A.85) is used for the first term and Eq. (A.90) is used for the second term to obtain

$$
\begin{aligned}
&\delta^{m+1}\mathcal{G}_{\Xi,\Phi}(g,h;\delta_{z_1},\ldots,\delta_{z_m},\delta_x) \\
&= \sum_{i=1}^{m}\Bigg[\int \delta_x(y)p_d(y)l(z_i|y)\mu_\Phi(y)dy \\
&\qquad \prod_{\substack{j=1\\ j\neq i}}^{m}\left[\int h(y)p_d(y)l(z_j|y)\mu_\Phi(y)dy+\mu_c(z_j)\right]\Bigg]\exp(F(g,h)) \\
&+\prod_{i=1}^{m}\left[\int h(y)p_d(y)l(z_i|y)\mu_\Phi(y)dy+\mu_c(z_i)\right]\delta F(g,h;\delta_x)\exp(F(g,h)) \\
&= \sum_{i=1}^{m}\left[p_d(x)l(z_i|x)\mu_\Phi(x)\prod_{\substack{j=1\\ j\neq i}}^{m}\left[\int h(y)p_d(y)l(z_j|y)\mu_\Phi(y)dy+\mu_c(z_j)\right]\right]\exp(F(g,h)) \\
&+\prod_{i=1}^{m}\left[\int h(y)p_d(y)l(z_i|y)\mu_\Phi(y)dy+\mu_c(z_i)\right]\delta F(g,h;\delta_x)\exp(F(g,h)).
\end{aligned}
\tag{A.151}
$$

The $\delta F(g, h; \delta_x)$ term can be simplified using Eq. (A.85) to

$$\begin{aligned}\delta F(g, h; \delta_x) &= \int \delta_x(y) \left(1 - p_d(y) + p_d(y) \int g(z) l(z|y) dz\right) \mu_\Phi(y) dy \\ &= \left(1 - p_d(x) + p_d(x) \int g(z) l(z|x) dz\right) \mu_\Phi(x).\end{aligned} \tag{A.152}$$

Then, substituting the expression back into Eq. (A.151) is

$$\begin{aligned}&\delta^{m+1} \mathcal{G}_{\Xi,\Phi}(g, h; \delta_{z_1}, \ldots, \delta_{z_m}, \delta_x) \\ &= \sum_{i=1}^{m} \left[p_d(x) l(z_i|x) \mu_\Phi(x) \prod_{\substack{j=1 \\ j \neq i}}^{m} \left[\int h(y) p_d(y) l(z_j|y) \mu_\Phi(y) dy + \mu_c(z_j) \right] \right] \exp(F(g, h)) \\ &+ \prod_{i=1}^{m} \left[\int h(y) p_d(y) l(z_i|y) \mu_\Phi(y) dy + \mu_c(z_i) \right] \\ &\times \left(1 - p_d(x) + p_d(x) \int g(z) l(z|x) dz\right) \mu_\Phi(x) \exp(F(g, h)).\end{aligned} \tag{A.153}$$

The last step is to divide the numerator, Eq. (A.153), by the denominator, Eq. (A.149) given by

$$\begin{aligned}\frac{\delta^{m+1} \mathcal{G}_{\Xi,\Phi}(g, h; \delta_{z_1}, \ldots, \delta_{z_m}, \delta_x)}{\delta^{m} \mathcal{G}_{\Xi,\Phi}(g, h; \delta_{z_1}, \ldots, \delta_{z_m})} &= \sum_{i=1}^{n} \frac{p_d(x) l(z_i|x) \mu_\Phi(x)}{\int h(y) p_d(y) l(z_i|y) \mu_\Phi(y) dy + \mu_c(z_i)} \\ &+ \left(1 - p_d(x) + p_d(x) \int g(z) l(z|x) dz\right) \mu_\Phi(x),\end{aligned} \tag{A.154}$$

and set $h = 1$ and $g = 0$ to form the general measurement update equation for the PHD filter given by

$$\mu_{\Phi|\Xi}(x|Z) = \sum_{z \in Z} \frac{p_d(x) l(z|x) \mu_\Phi(x)}{\int p_d(y) l(z|y) \mu_\Phi(y) dy + \mu_c(z)} + (1 - p_d(x)) \mu_\Phi(x). \tag{A.155}$$

Thus, the PHD filter recursion for the time and measurement updates are given by

$$\mu_\Phi(x) = \mu_b(x) + \int p_s(\bar{x}) m(x|\bar{x}) \mu_\Psi(\bar{x}) d\bar{x}, \tag{A.156a}$$

$$\mu_{\Phi|\Xi}(x|Z) = (1 - p_d(x))\mu_\Phi(x) + \sum_{z \in Z} \frac{p_d(x) l(z|x) \mu_\Phi(x)}{\int p_d(y) l(z|y) \mu_\Phi(y) dy + \mu_c(z)}, \tag{A.156b}$$

which follow the structure of the cardinality estimator derived from random variables, Eq. (A.56), and follows directly to Eq. (2.10) in the main text. This derivation does not include spawning, but it can also be included as well.

References

Açikmeşe, B. and Bayard, D. S. (2012). A markov chain approach to probabilistic swarm guidance. In *2012 American Control Conference (ACC)*, pages 6300–6307. IEEE.

Açıkmeşe, B. and Bayard, D. S. (2014). Probabilistic swarm guidance for collaborative autonomous agents. In *2014 American control conference*, pages 477–482. IEEE.

Bakule, L. (2008). Decentralized control: An overview. *Annual reviews in control*, 32(1):87–98.

Bamieh, B. and Voulgaris, P. G. (2005). A convex characterization of distributed control problems in spatially invariant systems with communication constraints. *Systems & control letters*, 54(6):575–583.

Bandyopadhyay, S., Chung, S.-J., and Hadaegh, F. Y. (2013). Inhomogeneous markov chain approach to probabilistic swarm guidance algorithm. In *5th Int. Conf. Spacecraft Formation Flying Missions and Technologies.*

Bandyopadhyay, S., Chung, S.-J., and Hadaegh, F. Y. (2014). Probabilistic swarm guidance using optimal transport. In *2014 IEEE Conference on Control Applications (CCA)*, pages 498–505. IEEE.

Banerjee, A., Merugu, S., Dhillon, I. S., and Ghosh, J. (2005). Clustering with bregman divergences. *Journal of machine learning research*, 6(Oct):1705–1749.

Bellman, R. et al. (1954). The theory of dynamic programming. *Bulletin of the American Mathematical Society*, 60(6):503–515.

Belta, C. and Kumar, V. (2004). Abstraction and control for groups of robots. *IEEE Transactions on robotics*, 20(5):865–875.

Bertsekas, D. (2006). Nonlinear programming, athena scientific, 1999. *REFER ENCIAS BIBLIOGR AFICAS*, 89.

Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J., et al. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122.

Boyd, S. and Vandenberghe, L. (2004). *Convex optimization.* Cambridge university press.

Camacho, E. F., Bordons, C., and Johnson, M. (1999). Model predictive control. advanced textbooks in control and signal processing.

Chattopadhyay, I. and Ray, A. (2009). Supervised self-organization of homogeneous swarms using ergodic projections of markov chains. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(6):1505–1515.

Christofides, N., Mingozzi, A., and Toth, P. (1981). Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. *Mathematical programming*, 20(1):255–282.

Clark, D., Delande, E., and Houssineau, J. (2016). Basic concepts for multi-object estimation lecture notes. Heriot-Watt University.

Clark, D. E. and Houssineau, J. (2013). Faa di bruno's formula for chain differentials. *arXiv preprint arXiv:1310.2833.*

Clark, D. E., Houssineau, J., and Delande, E. D. (2015). A few calculus rules for chain differentials. *arXiv preprint arXiv:1506.08626*.

Curtis, H. D. (2013). *Orbital mechanics for engineering students*. Butterworth-Heinemann.

Daley, D. J. and Vere-Jones, D. (2003). An introduction to the theory of point processes. vol. i. probability and its applications.

Daley, D. J. and Vere-Jones, D. (2007). *An introduction to the theory of point processes: volume II: general theory and structure*. Springer Science & Business Media.

DeCarlo, R. A. (1989). *Linear systems: A state variable approach with numerical implementation*. Prentice-Hall, Inc.

Demir, N., Eren, U., and Açıkmeşe, B. (2015). Decentralized probabilistic density control of autonomous swarms with safety constraints. *Autonomous Robots*, 39(4):537–554.

Di Bruno, F. F. (1857). Note sur une nouvelle formule de calcul différentiel. *Quarterly J. Pure Appl. Math*, 1(359-360):12.

Di Cairano, S., Park, H., and Kolmanovsky, I. (2012). Model predictive control approach for guidance of spacecraft rendezvous and proximity maneuvering. *International Journal of Robust and Nonlinear Control*, 22(12):1398–1427.

Doerr, B. and Linares, R. (2018). Control of large swarms via random finite set theory. In *2018 Annual American Control Conference (ACC)*, pages 2904–2909. IEEE.

Doerr, B., Linares, R., Zhu, P., and Ferrari, S. (2019). Random finite set theory and optimal control of large spacecraft swarms. In *2019 Space Flight Mechanics Meeting*, pages 3729–3732.

Eppstein, D. (1998). Finding the k shortest paths. *SIAM Journal on computing*, 28(2):652–673.

Eren, U. and Açıkmeşe, B. (2017). Velocity field generation for density control of swarms using heat equation and smoothing kernels. *IFAC-PapersOnLine*, 50(1):9405–9411.

Eren, U., Demirer, N., and Açıkmeşe, B. (2018). Density-based feedback control for earth orbiting swarms via velocity fields. *IFAC-PapersOnLine*, 51(12):44–49.

Fardad, M. and Jovanović, M. R. (2011). Design of optimal controllers for spatially invariant systems with finite communication speed. *Automatica*, 47(5):880–889.

Fardad, M. and Jovanović, M. R. (2014). On the design of optimal structured and sparse feedback gains via sequential convex programming. In *2014 American Control Conference*, pages 2426–2431. IEEE.

Fardad, M., Lin, F., and Jovanović, M. R. (2009). On the optimal design of structured feedback gains for interconnected systems. In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 978–983. IEEE.

Fardad, M., Lin, F., and Jovanović, M. R. (2011). Sparsity-promoting optimal control for a class of distributed systems. In *Proceedings of the 2011 American Control Conference*, pages 2050–2055. IEEE.

Ferrari, S., Foderaro, G., Zhu, P., and Wettergren, T. A. (2016). Distributed optimal control of multiscale dynamical systems: a tutorial. *IEEE Control Systems Magazine*, 36(2):102–116.

Findeisen, R. and Allgöwer, F. (2002). An introduction to nonlinear model predictive control. In *21st Benelux meeting on systems and control*, volume 11, pages 119–141. Technische Universiteit Eindhoven Veldhoven Eindhoven, The Netherlands.

Fletcher, R. (1980). *Practical Methods Of Optimization: Vol. 1 Unconstrained Optimization*. John Wiley & Sons.

Foderaro, G., Ferrari, S., and Wettergren, T. A. (2014). Distributed optimal control for multi-agent trajectory optimization. *Automatica*, 50(1):149–154.

Foderaro, G., Zhu, P., Wei, H., Wettergren, T. A., and Ferrari, S. (2018). Distributed optimal control of sensor networks for dynamic target tracking. *IEEE Transactions on Control of Network Systems*, 5(1):142–153.

Franken, D., Schmidt, M., and Ulmke, M. (2009). "spooky action at a distance" in the cardinalized probability hypothesis density filter. *IEEE Transactions on Aerospace and Electronic Systems*, 45(4):1657–1664.

Garcia, C. E., Prett, D. M., and Morari, M. (1989). Model predictive control: theory and practicea survey. *Automatica*, 25(3):335–348.

Goodman, I. R., Mahler, R. P., and Nguyen, H. T. (2013). *Mathematics of data fusion*, volume 37. Springer Science & Business Media.

Hadaegh, F. Y., Chung, S.-J., and Manohara, H. M. (2016). On development of 100-gram-class spacecraft for swarm applications. *IEEE Systems Journal*, 10(2):673–684.

Hoang, H. G., Vo, B.-N., Vo, B.-T., and Mahler, R. (2015). The cauchy–schwarz divergence for poisson point processes. *IEEE Transactions on Information Theory*, 61(8):4475–4485.

Houssineau, J., Delande, E., and Clark, D. (2013). Notes of the summer school on finite set statistics. *arXiv preprint arXiv:1308.2586*.

Huang, M., Malhamé, R. P., Caines, P. E., et al. (2006). Large population stochastic dynamic games: closed-loop mckean-vlasov systems and the nash certainty equivalence principle. *Communications in Information & Systems*, 6(3):221–252.

Inaba, M. and Corke, P. (2016). *Robotics Research: The 16th International Symposium ISRR*, volume 114. Springer.

Jovanovic, M. R. (2010). On the optimality of localised distributed controllers. *International Journal of Systems, Control and Communications*, 2(1-3):82–99.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45.

Kim, D. H., Wang, H., and Shin, S. (2006). Decentralized control of autonomous swarm systems using artificial potential functions: Analytical design guidelines. *Journal of Intelligent and Robotic Systems*, 45(4):369–394.

Kube, C. R. and Zhang, H. (1993). Collective robotics: From social insects to robots. *Adaptive behavior*, 2(2):189–218.

Levine, W. and Athans, M. (1970). On the determination of the optimal constant output feedback gains for linear multivariable systems. *IEEE Transactions on Automatic control*, 15(1):44–48.

Liao, L.-Z. and Shoemaker, C. A. (1991). Convergence in unconstrained discrete-time differential dynamic programming. *IEEE Transactions on Automatic Control*, 36(6):692–706.

Liao, L.-z. and Shoemaker, C. A. (1992). Advantages of differential dynamic programming over newton's method for discrete-time optimal control problems. Technical report, Cornell University.

Lin, F., Fardad, M., and Jovanovic, M. R. (2011). Augmented lagrangian approach to design of structured optimal state feedback gains. *IEEE Transactions on Automatic Control*, 56(12):2923–2929.

Lin, F., Fardad, M., and Jovanović, M. R. (2012). Sparse feedback synthesis via the alternating direction method of multipliers. In *2012 American Control Conference (ACC)*, pages 4765–4770. IEEE.

Lin, F., Fardad, M., and Jovanović, M. R. (2013). Design of optimal sparse feedback gains via the alternating direction method of multipliers. *IEEE Transactions on Automatic Control*, 58(9):2426–2431.

Lunze, J. (1992). *Feedback control of large-scale systems*. Prentice Hall New York.

Ma, W.-K., Vo, B.-N., Singh, S. S., and Baddeley, A. (2006). Tracking an unknown time-varying number of speakers using tdoa measurements: A random finite set approach. *IEEE Transactions on Signal Processing*, 54(9):3291–3304.

Mahler, R. (2006). A theory of phd filters of higher order in target number. In *Signal Processing, Sensor Fusion, and Target Recognition XV*, volume 6235, page 62350K. International Society for Optics and Photonics.

Mahler, R. (2007a). Phd filters of higher order in target number. *IEEE Transactions on Aerospace and Electronic systems*, 43(4):1523–1543.

Mahler, R. P. (2003). Multitarget bayes filtering via first-order multitarget moments. *IEEE Transactions on Aerospace and Electronic systems*, 39(4):1152–1178.

Mahler, R. P. (2007b). *Statistical multisource-multitarget information fusion*, volume 685. Artech House Norwood, MA.

Makila, P. and Toivonen, H. (1987). Computational methods for parametric lq problems–a survey. *IEEE Transactions on Automatic Control*, 32(8):658–671.

Manikonda, V., Arambel, P., Gopinathan, M., Mehra, R., and Hadaegh, F. (1999). A model predictive control-based approach for spacecraft formation keeping and attitude control. In *Proceedings of the 1999 American Control Conference (Cat. No. 99CH36251)*, volume 6, pages 4258–4262. IEEE.

Mayne, D. Q., Rawlings, J. B., Rao, C. V., and Scokaert, P. O. (2000). Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814.

Mondada, F., Gambardella, L. M., Floreano, D., Nolfi, S., Deneuborg, J.-L., and Dorigo, M. (2005). The cooperation of swarm-bots: Physical interactions in collective robotics. *IEEE Robotics & Automation Magazine*, 12(2):21–28.

Morgan, D., Chung, S.-J., and Hadaegh, F. (2012). Spacecraft swarm guidance using a sequence of decentralized convex optimizations. In *AIAA/AAS Astrodynamics Specialist Conference*, page 4583.

Morgan, D., Chung, S.-J., and Hadaegh, F. Y. (2013). Decentralized model predictive control of swarms of spacecraft using sequential convex programming. *Advances in the Astronautical Sciences*, (148):1–20.

Morgan, D., Chung, S.-J., and Hadaegh, F. Y. (2015). Swarm assignment and trajectory optimization using variable-swarm, distributed auction assignment and model predictive control. In *AIAA guidance, navigation, and control conference*, page 0599.

Murthy, K. G. (1968). An algorithm for ranking all the assignments in order of increasing costs. *Operations research*, 16(3):682–687.

Pace, M., Birattari, M., and Dorigo, M. (2013). The swarm/potential model: Modeling robotics swarms with measure-valued recursions associated to random finite sets. *IEEE Transactions on Robotics, page submitted.*

Pulford, G. (2005). Taxonomy of multiple target tracking methods. *IEE Proceedings-Radar, Sonar and Navigation*, 152(5):291–304.

Rautert, T. and Sachs, E. W. (1997). Computational design of optimal output feedback controllers. *SIAM Journal on Optimization*, 7(3):837–852.

Reif, J. H. and Wang, H. (1999). Social potential fields: A distributed behavioral control for autonomous robots. *Robotics and Autonomous Systems*, 27(3):171–194.

Rubenstein, M., Cornejo, A., and Nagpal, R. (2014). Programmable self-assembly in a thousand-robot swarm. *Science*, 345(6198):795–799.

Rudd, K., Foderaro, G., and Ferrari, S. (2013). A generalized reduced gradient method for the optimal control of multiscale dynamical systems. In *52nd IEEE Conference on Decision and Control*, pages 3857–3863. IEEE.

Ryan, A., Zennaro, M., Howell, A., Sengupta, R., and Hedrick, J. K. (2004). An overview of emerging results in cooperative uav control. In *2004 43rd IEEE Conference on Decision and Control (CDC)(IEEE Cat. No. 04CH37601)*, volume 1, pages 602–607. IEEE.

Sommerville, I. (2016). *Software Engineering GE.* Pearson Australia Pty Limited.

Sun, J. and Chen, H. (2018). A decentralized and autonomous control architecture for large-scale spacecraft swarm using artificial potential field and bifurcation dynamics. In *2018 AIAA Guidance, Navigation, and Control Conference*, page 1860.

Tassa, Y., Erez, T., and Todorov, E. (2012). Synthesis and stabilization of complex behaviors through online trajectory optimization. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4906–4913. IEEE.

Tassa, Y., Mansard, N., and Todorov, E. (2014). Control-limited differential dynamic programming. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1168–1175. IEEE.

Todorov, E. and Li, W. (2005). A generalized iterative lqg method for locally-optimal feedback control of constrained nonlinear stochastic systems. In *Proceedings of the 2005, American Control Conference, 2005.*, pages 300–306. IEEE.

Vassev, E., Hinchey, M., and Paquet, J. (2008). Towards an assl specification model for nasa swarm-based exploration missions. In *Proceedings of the 2008 ACM symposium on Applied computing*, pages 1652–1657. ACM.

Verdoljak, R. (2016). Application of sparse feedback control strategies to civil structures.

Vo, B.-N. and Ma, W.-K. (2006). The gaussian mixture probability hypothesis density filter. *IEEE Transactions on signal processing*, 54(11):4091–4104.

Vo, B.-N., Singh, S., and Doucet, A. (2005). Sequential monte carlo methods for multitarget filtering with random finite sets. *IEEE Transactions on Aerospace and electronic systems*, 41(4):1224–1245.

Vo, B.-N., Vo, B.-T., and Phung, D. (2014). Labeled random finite sets and the bayes multi-target tracking filter. *IEEE Transactions on Signal Processing*, 62(24):6554–6567.

Vo, B.-T. and Vo, B.-N. (2013). Labeled random finite sets and multi-object conjugate priors. *IEEE Transactions on Signal Processing*, 61(13):3460–3475.

Vo, B.-T., Vo, B.-N., and Cantoni, A. (2006). The cardinalized probability hypothesis density filter for linear gaussian multi-target models. In *2006 40th Annual Conference on Information Sciences and Systems*, pages 681–686. IEEE.

Voulgaris, P. G., Bianchini, G., and Bamieh, B. (2003). Optimal h2 controllers for spatially invariant systems with delayed communication requirements. *Systems & control letters*, 50(5):347–361.