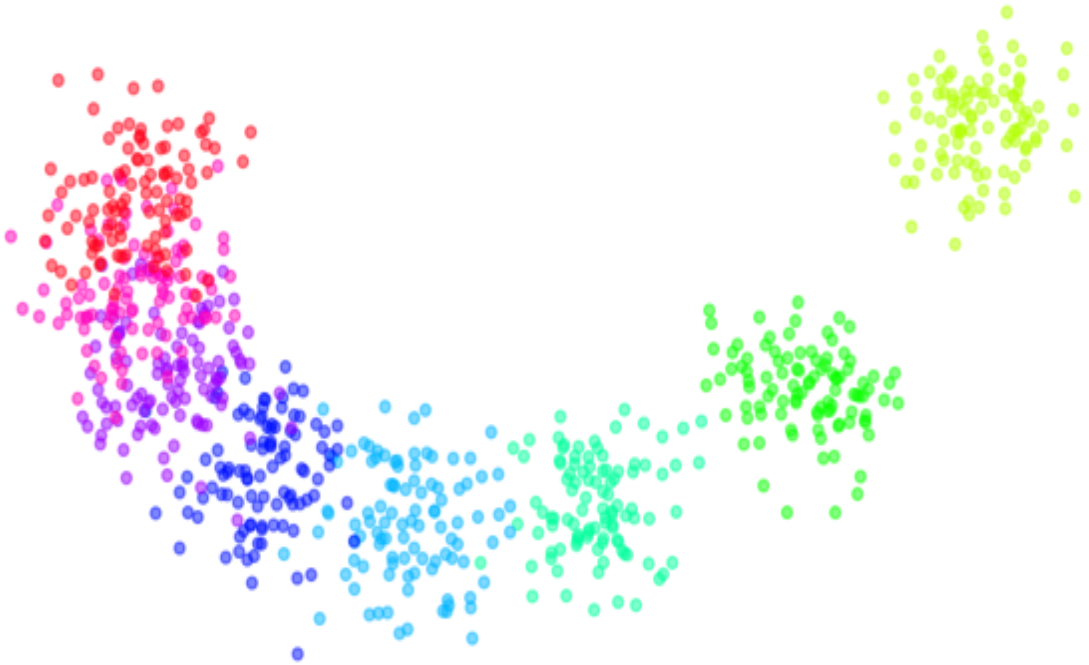
Quantization of Gaussian Mixture Models

With applications in both control theory and machine learning

Elize Alwash

Master of Science Thesis



TU Delft Delft University of Technology

Delft Center for Systems and Control

Quantization of Gaussian Mixture Models

With applications in both control theory and machine learning

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

Elize Alwash

July 16, 2025

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Delft University of Technology
Department of
Delft Center for Systems and Control (dcsc)

The undersigned hereby certify that they have read and recommend to the Faculty of Mechanical, Maritime and Materials Engineering (3mE) for acceptance a thesis entitled

Quantization of Gaussian Mixture Models

by

Elize Alwash

in partial fulfillment of the requirements for the degree of

Master of Science Systems and Control

Dated: July 16, 2025

Supervisor(s):

Dr. Luca Laurenti

Steven Adams PhD

Reader(s):

Dr. Mohammad Khosravi

Abstract

Gaussian Mixture Models (GMMs) are powerful tools for representing arbitrary distributions or data sets, especially in complex non-linear systems. They are often used as approximators due to their flexibility. However, in many cases, such as for dynamical systems, these must be propagated through non-linear functions. How we can do this is still an open problem.

In this work, we propose a scalable method for quantizing GMMs with formal bounds on the approximation error using the Wasserstein distance. This ensures the method is suitable for safety-critical applications such as autonomous vehicles and UAVs, where formal guarantees are essential. Our approach, called the multi-grids method, constructs local grids at locations of high density, which are identified using clustering techniques. Around each cluster, a hyper-rectangular grid is formed, and the locations are placed ensuring minimal error in terms of the Wasserstein distance. This design allows the method to scale efficiently to GMMs of large sizes in terms of both the dimension and the number of components, a known limitation of existing methods.

We validate our method against the state of the art, across various settings, demonstrating significantly lower approximation errors in terms of the Wasserstein distance. Even at higher dimensions of 60 it can still find efficient quantizations with little computational costs. Additionally, we apply the multi-grids method to the uncertainty propagation problem in dynamical systems, including the benchmark Dubins Car, highlighting its practical effectiveness in real-time systems.

Contents

Preface & Acknowledgements

This thesis marks the end of my Master's degree in Systems and Control. My academic journey began with a Bachelors in Mechanical Engineering, where I was exposed to many interesting disciplines. Among them, Systems and Control stood out as the most fascinating field.

Next to studying I took up many student jobs, I especially loved working as a student assistant for the mechanics seminars, which was very rewarding. Later I joined the Micro-Ariel-Vehicle (MAV) lab at Aerospace Engineering, where I was exposed to a more practical aspect of engineering.

During this journey, I experienced my share of challenges. However I am deeply grateful for all the learning opportunities that shaped both my academic and personal development.

To quickly introduce the project, this thesis presents the research and development of new quantization operator for Gaussian Mixture Models (GMMs). Due to their powerful properties, GMMs have been applied in many different fields. However their complexity can be problematic when deploying them in real-time. Therefore this thesis project proposes an efficient and computationally favourable approximation scheme for GMMs.

I would like to sincerely thank my supervisors Dr. Luca Laurenti and Steven Adams PhD for their guidance and support during the thesis process. Their availability, constructive feedback, and encouragement made this a truly enriching experience. I can definitely say I could not have asked for better supervisors, thank you! It was very inspiring to work alongside them and I am especially grateful for their influence in encouraging me to pursue a PhD, an opportunity I now look forward to with enthusiasm.

One a more personal note, I'm so grateful I got to struggle through the Master's together with Veronika and Valerio, you made all the difference. Rachel and Anneke, meeting you on the first day of the Bachelor's shaped the years that followed in the best way. Amaryllis, who I could always go to for anything, and always had my back, thank you. The athletics trainings would've been a lot duller without you. Lennart, who supported me through everything, both my studies and personal struggles, thank you for always listening and helping me get through the tough times. And lastly, thank you to my family. Even though we all experienced some difficult and strange times over the last few years, I am grateful you were always there to

support me in all the possible ways you could.

Delft, University of Technology Elize Alwash
July 16, 2025

Chapter 1

Introduction

The increasing deployment of autonomous systems, such as self driving cars [17] and Unmanned Aerial Vehicles (UAVs) [9], demand control strategies capable of handling complex, non-linear dynamics under uncertainty. To design realistic models, the disturbances or uncertainty these systems are subjected to must be accounted for in system modelling. This uncertainty may arise from noise in the dynamics, sensor measurements, or incomplete knowledge of the system. For autonomous driving, it may be the interaction with other cars or pedestrians, while UAVs could suffer from unexpected weather conditions. Therefore, in practice, new control strategies are being used to account for such uncertainty, for example stochastic model predictive control (MPC) [20] or deep aware reinforcement learning [18].

A system is described by a set of states, which are variables that fully capture the systems behaviour at a given time, given its inputs. The system model is typically a function that maps current states and inputs to future states, allowing us to simulate system behaviour over time. In deterministic settings, these states are represented as fixed values. However, in stochastic or uncertain environments, such as for UAVs, it is more appropriate to represent the states as probability distributions rather than point values, to incorporate uncertainty. A common assumption is that the states are Gaussian, which is an analytically convenient way to capture this uncertainty.

Modelling states as distributions allows us to quantify uncertainty explicitly. As the system evolves over time, it becomes important to understand how this uncertainty changes, a process known as uncertainty propagation. For non-linear dynamics there is no general solution for uncertainty propagation [12]. Additionally, propagating distributions such as a Gaussian through non-linear dynamics, results in an output that is no longer Gaussian and often intractable to propagate further.

With the use of Gaussian Mixture Models (GMMs) we can approximate these distributions, using methods like the Expectation-Maximization algorithm [25]. Uncertainty propagation with the use of mixture models has already been studied in various literature, [13], [33], [30]. However, in most cases, there are no closed-form solutions for propagating the entire GMM through the system dynamics [16]. Therefore additional approximation schemes are

required to make the computations tractable. That is why it is important to develop scalable approximation schemes for GMMs, as we'll do in this paper.

Approximating these models will introduce additional uncertainty. This added error from the approximation step must be accounted for. It is critical in applications like autonomous driving, as unaccounted error can compromise system safety and performance. Therefore, beyond designing a scalable approximation scheme, it is essential to establish formal bounds on the approximation error.

The work of [2], introduces the 'signature operator', which directly solves the problem presented above. It finds a discrete approximation of a GMM with formal guarantees. However, for similar performance, the method suffers from scalability limitations. It will become computationally expensive when the dimension or the number of mixture components increases. This paper presents a new method that ensures scalability while maintaining accuracy.

Our approach, which we call the 'multi-grids method', constructs a discrete approximation of a GMM with formal guarantees. This method combines clustering techniques with the Wasserstein distance to efficiently place the discrete locations. The foundation of the method is to identify high-density regions within the GMM, so the majority of locations are placed there, capturing the most significant information.

The method is as follows, we first apply a density-based clustering algorithm on a sampled version of the GMM. These clusters serve as a foundation for constructing a discrete approximation using a grid-based representation. Around each cluster a hyper-rectangular box is built, which represents the boundary of the grid, within these grids the discrete locations are placed.

For the quantization of a Gaussian, we can actually find the most optimal locations in terms of the Wasserstein distance using the work of [2]. As we have found clusters which represent the dense areas of a GMM, we can approximate these clusters by Gaussians. In doing so, we can apply the work of [2] to place the locations within these hyper-rectangular boxes. Any remain mass outside of the boxes is assigned to an "outer location", denoted by z. This ensures the complete mass of the original GMM is conserved within the discrete approximation.

We use a grid-based representation such that we can leverage fast computations of the Wasserstein distance. The Wasserstein distance is particularly suitable for this quantization task due to its favourable properties: it is a true metric and is well-defined between distributions with different supports [26]. This overall strategy balances computational efficiency with formal guarantees. In summary, our main contributions are:

- the development of a new quantization operator that scales efficiently to high dimensional settings and large scale GMMs,
- benchmarking our method against the previous state-of-the-art operator of [2], comparing both approximation error, in terms of the Wasserstein distance, and runtime.
- an application of the proposed operator to dynamic systems, including high dimensional examples and the Dubins Car benchmark [4].

The paper is structured as follows, Chapter 2 introduces the necessary preliminaries and theoretical foundations, Chapter 3 formally defines the problem setting, Chapter 4 presents our approach, and, finally, Chapter 5 showcases the performance of our new operator through empirical experiments.

Chapter 2

Preliminaries

To approximate GMMs, the theory of quantization is used, with the quantization error formally bounded by the Wasserstein distance. To provide context for how this problem is addressed, we begin by discussing GMMs and their properties. This is followed by an explanation of quantization theory and the role of the Wasserstein distance in measuring this approximation error. Finally, we review related work, including relevant clustering techniques.

2-1 Notation

- A probability distribution gives the probability of a certain outcome. It is a mathematical representation of a random phenomenon, defined by its sample space and the probabilities assigned to events. In this report we will be using Gaussian distributions modeled by the function $\mathcal{N}(\mu, \Sigma)$ where μ is the mean and Σ the covariance.
- The expectation operator $\mathbb{E}$ is defined as $\mathbb{E}[X] = \int_{-\infty}^{\infty} x f(x)\, dx$ for a random variable X with a probability density functions $f(x)$. If X has an infinite countable set, then it is defined as $\mathbb{E}[X] = \sum_{i=1}^{\infty} x_i\, p_i$, with x_i possible outcomes and p_i probabilities.
- The push-forward operator $P_{\mathcal{X}}$ maps the probability distribution γ to its marginal distribution α in the set $\mathcal{X}$, defined by $P_{\mathcal{X}} \# \gamma = \alpha$.
- $\mathcal{P}(\mathbb{R}^d)$ is the set of probability distributions on $\mathbb{R}^d$.
- $\mathcal{P}_p(\mathbb{R}^d)$ is the Wasserstein space, which is the space of probability distributions of finite moment order of p, defined by $\mathcal{P}_p(X) := \{\mu \in \mathcal{P}(X) \mid \int_X d(x_0, x)^p\, \mu(dx) < +\infty\}$ [32].
- $\mathcal{P}_p(\mathbb{R}^d \times \mathbb{R}^d)$ is the product space for all joint probability distributions with marginals in the space of $\mathcal{P}_p(\mathbb{R}^d)$.
- The Dirac Delta function is described by δ_{c_i} at locations c_i.
- $D_N(\mathbb{R}^d)$ is the set of discrete probability distributions on $\mathbb{R}^d$ with N locations, in the set of $\mathcal{P}(\mathbb{R}^d)$.

- $sup(\mathbb{P})$ describes the support of distribution $\mathbb{P}$.
- The Euclidean norm for a vector $x \in \mathbb{R}^d$ is given by $\|x\|$ and defined on $\mathbb{R}^d$.
- The weights $\bar{\pi}^{(i)}$ are used as weights for the Gaussian components in the GMMs. They are defined by $\bar{\pi}^{(i)} \in \Pi$ where $\Pi = \{\bar{\pi}^{(i)} \in \mathbb{R}^N_{\geq 0} : \bar{\pi}^{(i)} = 1\}$.
- Discrete distributions are defined by $d \in D_N(\mathbb{R}^d) = \{\sum_{i=1}^N \pi^{(i)} \delta_{c_i} | \quad \forall i \in 1, \ldots, N\}$. Where the set of points $C = \{c_i\}_{i=1}^N \subset \mathbb{R}^d$ are the locations of probability mass for the discrete distribution with index i. The Dirac Delta's are weighted by their respective probability mass $\pi^{(i)}$.
- The indicator function is defined as $\mathbb{1}_{\mathcal{X}} := \begin{cases} 1, & \text{if } x \in \mathcal{X} \\ 0, & \text{otherwise} \end{cases}$ for vector $x \in \mathbb{R}^d$ and region $\mathcal{X} \subset \mathbb{R}^d$

2-2 Gaussian Mixture Models

GMMs are a combination of multiple weighted Gaussian distributions. Often Gaussians are used as an approximation of a distribution, such as the noise or uncertainty in a system. Using this assumption can save a lot of computations due to the model's tractability. However, for a more accurate representation of complex or multimodal distributions, GMMs are better suited. Yet, this improved accuracy comes at the cost of increased complexity.

Definition 1. *GMMs are defined by a sum of weighted Gaussian distributions:*

$$GMM_d(N) = \sum_{i=1}^{\bar{M}} \bar{\pi}^{(i)} \mathcal{N}\left(\mu_i, \Sigma_i\right)^d,$$

where $\bar{\pi}_{(i)}$ are the weights for each component and constraint by: $\bar{\pi}^{(i)} \in \Pi$. The GMM is of dimension d with $\bar{M}$ components.

2-2-1 GMMs as Approximators

GMMs are so powerful as they are able to model any probability distribution of finite moments arbitrary well with a set error of ε [8]. It is a linear combination of $\bar{M}$ Gaussian components, parametric in the mean, variance and weight of the separate Gaussian components. This property is highly useful when it comes to data clustering and unsupervised learning tasks [7]. GMMs are widely used in different fields to model big and complex data due to their computationally favourable representations. Areas such as marketing, medicine or astronomy use GMMs as models for statistical analysis of complex data sets [19].

In the use case of clustering, each cluster within the data is represented by a different Gaussian component in the mixture. The components can be weighted differently, representing their dominance in the data set. This way of representing data is interpretable and gives us a good indication of how the data is exactly distributed. Figure 2-1 shows a visual representation of such a situation. More examples of the wide use of GMMs include weight sharing in a neural network, [23], solving inverse image problems [37], classification problems [39] and the expectation maximization algorithm [19].

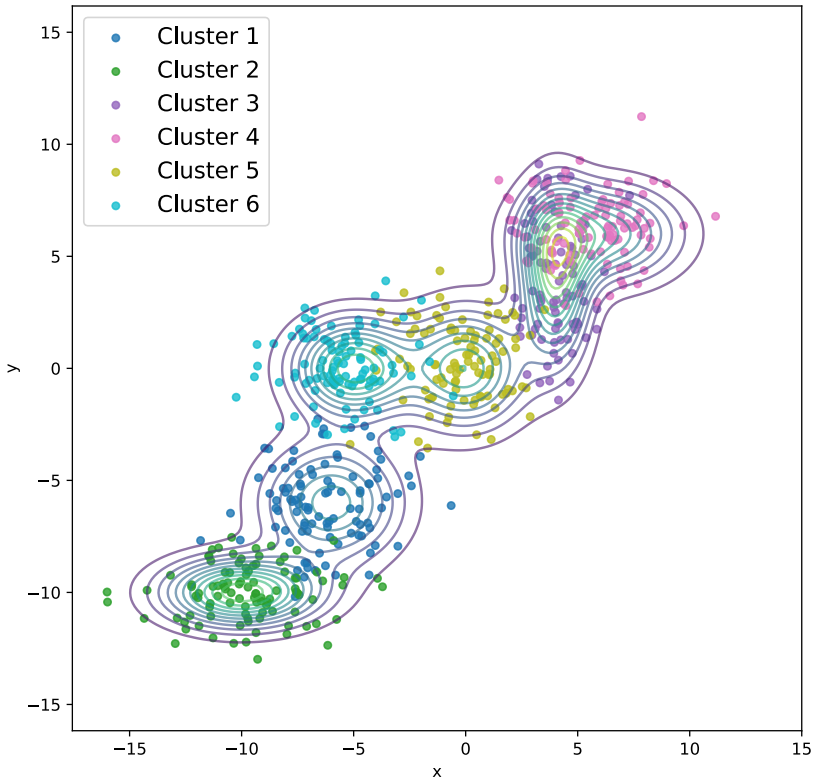


Figure 2-1: The clusters represent the data, each cluster represents a Gaussian component. Approximating the full distribution using these clusters leads to a GMM, which is represented by the level lines in the plot, approximating the data set.

2-2-2 GMMs in Quantization

To approximate a GMM by a discrete distribution, it is most advantageous to place the discrete locations at regions of maximum density. For a GMM this means concentrating the points around the modes. A GMM is a multi-modal function, where each mode corresponds to a local maximum of the probability density function [38]. However there is no general closed-form solution for identifying these modes [6]. Multiple algorithms for example, the Expectation-Maximisation (EM) algorithm [22]. The EM algorithm is an iterative method for finding maximum likelihood estimates of the parameters within the GMM based on the training data.

Another way is to use clustering methods. EM is naturally adapted to represent clusters of data as Gaussian components [25], however our focus is on finding modes within an already constructed GMM. Clustering methods such as DBSCAN use density constraints to find high density regions.

After dense regions are found within the GMM, an additional step to consider is calculating

the mass of the discrete distribution, $\pi^{(i)}$. To do so we need the marginals of the GMM over regions of mass [12].

2-3 Quantization

Quantization originates from the theory of signal processing [15]. However, for probability distributions it comes down to approximating a d-dimensional distribution by at most n locations. The works of [3], [2], [12], extend quantization further by computing formal bounds on the Wasserstein distance between the original distribution and its approximation.

2-3-1 The Quantization Operator

The quantization operator is defined by a partitioning $\mathcal{R} := \{\mathcal{R}_i\}_{i=1}^N$ and a set of locations $\mathcal{C} := \{c_i\}_{i=1}^N \subset \mathbb{R}^d$. The operator is defined as

$$\Delta_{\mathcal{R},\mathcal{C}}(x) := \sum_{i=1}^{N} c_i \mathbb{1}_{\mathcal{R}_i}(x), \tag{2-1}$$

where any point inside region $\mathcal{R}_i$ is brought to location c_i [12]. For a probability distribution $\mathbb{P} \in \mathcal{P}(\mathbb{R}^d)$, the push-forward operator is defined as,

$$\Delta_{\mathcal{R},\mathcal{C}} \# \mathbb{P} = \sum_{i=1}^{N} \mathbb{P}(\mathcal{R}_i)\delta_{c_i} \in D_N(\mathbb{R}^d). \tag{2-2}$$

To calculate $\mathbb{P}(\mathcal{R}_i)$ we take the product of the marginals of each dimension, resulting in the total mass over the partitioning $\mathcal{R}_i$,

$$\mathbb{P}(\mathcal{R}) = \prod_{j}^{d} \mathbb{P}(\mathcal{R}^{(j)})$$

To apply this rule, the regions must be axis-aligned with the distributions and hyper-rectangular. Axis-aligned means that the partition boundaries must align with the coordinate axes of the distribution. Additionally, when computing the product of marginal distributions, any dependency between dimensions introduces cross terms, which breaks the separability assumption. Therefore, a second requirement is that the distribution's dimensions must be independent, ensuring that the joint distribution can be described by the product of its univariate marginals.

When $\mathbb{P}$ is a GMM, the problem in Equation 2-2 becomes intractable, as calculating the probability mass $\mathbb{P}(\mathcal{R}_i)$ is infeasible. However, Gaussian distributions do exhibit independent dimensions, therefore the quantization operator can be applied per Gaussian component and the probability mass can be summed due to the additivity property of GMMs,

$$\mathbb{P}(\mathcal{R}) = \sum_{i}^{\bar{M}} \bar{\pi}^{(i)} \prod_{j}^{d} \mathbb{P}_i(\mathcal{R}^{(j)}),$$

where $\mathbb{P}_i$ represents each Gaussian component with respective weight $\bar{\pi}^{(i)}$.

2-3-2 Approximations

As solving the quantization of a GMM for any arbitrary partitioning $\mathcal{R}$ and set of locations $\mathcal{C}$ may be intractable, approximations of the quantization operator are made, by constraining both $\mathcal{R}$ and $\mathcal{C}$.

In the work of [2] the locations $\mathcal{C}$ are optimized with respect to the minimal Wasserstein error of the quantization of a Gaussian distribution. These locations are found per dimension and the Cartesian product is taken resulting in a grid of locations. The partitioning $\mathcal{R}$ is set as the Voronoi partitioning of the locations. As the dimensions of a Gaussian are independent, the grid will result in hyper-rectangular Voronoi partitioning, ensuring we can calculate the mass of $\mathbb{P}(\mathcal{R}_i)$. The Voronoi partitioning is be defined by,

$$\mathcal{R}_i = \left\{ z \in \mathbb{R}^d : \|z - c_i\| \leq \|z - c_j\|, \forall j \in \{1, \ldots, N\}, j \neq i \right\}. \tag{2-3}$$

To approximate the quantization of a GMM, the operator is applied on each Gaussian component individually, then the union is taken of the resulting quantizations $d = \sum_{i=1}^{N} \mathbb{P}(\mathcal{R}_i)\delta_{c_i}$. This operator will be named as the 'per-component' method from now on.

Another approach to approximating the quantization operator is to apply a shared partitioning and set of locations across all components, rather than computing a separate quantization for each component individually. The work of [12] presents this method. However, to ensure computations are tractable, as in the per-component method, the partitioning $\mathcal{R}$ must be a set of axis-aligned hyper-rectangles. We refer to this strategy as the 'one-grid' approach.

2-4 Wasserstein Distance

The Wasserstein-p distance, is as a metric that represents the solution to the optimal transport problem, typically using the l_2-norm as the cost function. It is the minimum cost to transfer one distributions' mass to another, defined as a metric on the space $\mathcal{P}_p(\mathbb{R}^d)$ [26].

Definition 2. *For $p \geq 1$, the Wasserstein (p-squared) distance between arbitrary distributions is defined by* (2-4)*, where the cost function is the p-squared Euclidean norm between values x and y sampled from the distributions α and β respectively. The distributions α and β live in the space $\mathcal{P}_p(\mathbb{R}^d)$.*

$$W_p(\alpha, \beta) := \left(\inf_{\gamma \in \Gamma(\alpha,\beta)} \mathbb{E}_{(x,y)\sim\gamma} \|x - y\|^p \right)^{1/p} = \left(\inf_{\gamma \in \Gamma(\alpha,\beta)} \int_{\mathbb{R}^d \times \mathbb{R}^d} [\|x - y\|^p] \gamma(dx, dy) \right)^{1/p} \tag{2-4}$$

$\Gamma(\alpha, \beta)$ is defined as the set of joint probability distributions γ in the space $\mathcal{P}_p(\mathbb{R}^d \times \mathbb{R}^d)$, with marginal distributions α and β.

2-4-1 Properties of the Wasserstein distance

A very advantageous property of the Wasserstein-p distance is that there is an explicit formulation when α and β are Gaussians and p is set to 2, defined by:

$$W_2^2(\alpha, \beta) = \|\mu_k - \mu_l\|_2^2 + \operatorname{tr}\left(\Sigma_k + \Sigma_l - 2 \left(\Sigma_l^{1/2} \Sigma_k \Sigma_l^{1/2} \right)^{1/2} \right), \tag{2-5}$$

where the Gaussians are given by $\mathcal{N}(\mu_k, \Sigma_k)$ and $\mathcal{N}(\mu_l, \Sigma_l)$ respectively [8]. We often write the solution in terms of the Wasserstein-squared distance instead of the Wasserstein distance, since it is simpler to express. Given this property, all subsequent references to the Wasserstein distance will specifically refer to the Wasserstein-2 distance from here onwards.

The Wasserstein distance is a true metric, meaning that it satisfies symmetry, the triangle inequality and non-degeneracy [26]. The Wasserstein distance is defined between distributions of different supports, which is very important for quantization, while metrics such as TV cannot. Another unique and powerful feature of the Wasserstein distance is that closeness in the Wasserstein distance implies closeness in moments. This is formalized in Lemma 2 of [2].

There are only a few well known cases where there are closed form solutions available to compute the Wasserstein distance. This includes problems with one-dimensional distributions, which have a tractable cumulative distribution function (CDF), and for Gaussian distributions of all dimensions, as shown in Equation 2-5. However, more importantly, the problem at hand is computing the true Wasserstein distance between GMMs, which is intractable. Closed form solutions of the Wasserstein distance, for example between Gaussians, are often leveraged in approximations. An example of this is the $MW2$ distance from [8]. It splits up the GMM in its respective Gaussian components and approximates the true Wasserstein distance by the pairwise Wasserstein distances between the Gaussian components.

2-4-2 Semi-Discrete Optimal Transport

The quantization of a distribution can be interpreted as the semi-discrete optimal transport problem defined between a distribution $\mathbb{P}$ and its approximate discrete distribution d. The Wasserstein distance between a distribution and its quantization becomes the the measure of error for the quantization step, defined by

$$W_2^2(d, \mathbb{P}) = \inf_{\zeta(z'|z) \in \mathcal{Z}[d,\mathbb{P}} \sum_{j=1}^{N} \beta_j \mathbb{E}_{z' \sim \zeta(z'|z)} \left[c(z_j, z') \right], \tag{2-6}$$

where the locations of mass in $\mathbb{P}$ are defined by z' and the locations in d are defined by z^j [3]. The cost function is defined by $c(z_j, z')$, which in our case is the l_2-norm. The set $\mathcal{Z}[q_N, p]$ is the set of conditional distributions that fulfil the marginalization constraint:

$$\mathcal{Z}[q_N, p] = \left\{ \zeta(z'|z) \;\middle|\; \sum_{j=1}^{N} \beta_j \zeta(z'|z_j) = p(z') \right\}. \tag{2-7}$$

This marginalization constraint signifies that the weights β_j must be assigned to the mass of the distribution $\mathbb{P}$ at location z', by the transport plan $\zeta(z'|z)$ [3].

The quantization of a Gaussian for the minimal Wasserstein error has already been solved by [2]. However no optimal quantization has been found for a GMM yet. Current methods include the per-component method, where the Wasserstein distance is expressed as

$$W_2^2 \left(\sum_{i=1}^{\bar{M}} \bar{\pi}_i \mathcal{N}(m_i, \Sigma_i), \sum_{i=1}^{\bar{M}} \bar{\pi}_i \Delta_{\mathcal{C}_i} \# \mathcal{N}(m_i, \Sigma_i) \right) \leq \sum_{i=1}^{\bar{M}} \bar{\pi}_i W_2^2 \left(\mathcal{N}(m_i, \Sigma_i), \Delta_{\mathcal{C}_i} \# \mathcal{N}(m_i, \Sigma_i) \right). \tag{2-8}$$

Which is upper-bounded by the weighted sum of the Wasserstein errors per Gaussian quantization separately [2].

Instead of taking separate locations and partitioning per-component, you can also apply the same partitioning $\mathcal{R}$ and locations $\mathcal{C}$ to each component in the GMM. In this case, the Wasserstein distance can be expressed as,

$$W_2^2(\mathbb{P}, \Delta_{\mathcal{R},\mathcal{C}}\#\mathbb{P}) \leq \sum_{k=1}^{N} \int_{\mathcal{R}_k} \|x - c_k\|^2 d\mathbb{P}(x). \tag{2-9}$$

The work of [12] presents an analytically tractable solution for the constrained 2nd-moment in Equation 2-9, for a specific group of distributions, if $\mathcal{R}$ is a set of axis-aligned hyper-rectangles.

If $\mathcal{R}$ is chosen to be the Voronoi partitioning w.r.t $\mathcal{C}$, then Equation 2-9 holds with equality [12]. Additionally, it will ensure the lowest transport cost [2]. To maintain a hyper-rectangular Voronoi partition aligned with the distribution's dimensions, the locations c_i are constrained to a grid formed by placing points along each axis and taking their Cartesian product.

Definition 3. *Let the vectors $\{V^i(N) \mid i = 1, \cdots n\}$ represent the axes of the dimensions of the distribution $\mathbb{P} \in \mathbb{R}^d$, with an arbitrary number of points N per axes. The (axes-aligned) grid's locations are defined by*

$$C_{grid} = \{V^1(N) \times V^2(N) \times \cdots \times V^n(N)\}. \tag{2-10}$$

which contains all viable locations. The total grid is defined by its locations $\mathcal{C}$, and the partitioning $\mathcal{R}$, which is set to Voronoi partitioning for the lowest transport cost.

The eigen-basis of a covariance matrix defines the axes of the dimension of its corresponding Gaussian. For a single Gaussian component, the dimensions are independent, so Equation 2-9 has a closed-form solution. As mentioned earlier, for a GMM there will be dependencies across the dimensions. To get around this, the quantization is performed individually for each Gaussian component wihin the GMM. To ensure that the quantization grid aligns consistently across all components, it is easiest to constrain all components within the GMM to have the same eigen-basis.

If we apply the same grid to all components, the Wasserstein distance of this quantization method of the GMM can be calculated using Equation 2-11,

$$W_2^2(\Delta_{\mathcal{R},\mathcal{C}}\#\mathbb{P}, \mathbb{P}) = \sum_{i=1}^{\bar{M}} \bar{\pi}_i W_2^2(\Delta_{\mathcal{R},\mathcal{C}}\#\mathbb{P}_i, \mathbb{P}_i), \tag{2-11}$$

where $W_2^2(\Delta_{\mathcal{R},\mathcal{C}}\#\mathbb{P}_i, \mathbb{P}_i)$, is calculated by the constrained 2nd-moment in Equation 2-9. The proof is as follows: for a mixture of probability distribution $\mathbb{P} = \sum_{i=1}^{\bar{M}} \pi_i \mathbb{P}_i \in \mathbb{R}^d$, and a finite set of points $\mathcal{C} \subset \mathbb{R}^d$, and $\mathcal{R} \subset \mathbb{R}^d$ the Voronoi partition of $\mathbb{R}^d$ w.r.t. $\mathcal{C}$, according to [12], we have that,

$$\mathbb{W}_2^2\left(\mathbb{P}, \Delta_{\mathcal{C}}\#\mathbb{P}\right) = \sum_{k=1}^{|\mathcal{C}|} \int_{R_k} \|x - c_k\|^2 d\mathbb{P}(x) \tag{2-12}$$

$$= \sum_{k=1}^{|\mathcal{C}|} \int_{R_k} \sum_{i=1}^{\bar{M}} \bar{\pi}_i \|x - c_k\|^2 d\mathbb{P}_i(x) \tag{2-13}$$

$$= \sum_{i=1}^{\bar{M}} \bar{\pi}_i \sum_{k=1}^{|\mathcal{C}|} \int_{R_k} \|x - c_k\|^2 d\mathbb{P}_i(x) \tag{2-14}$$

$$= \sum_{i=1}^{\bar{M}} \bar{\pi}_i \mathbb{W}_2^2(\mathbb{P}_i, \Delta_{\mathcal{C}}\#\mathbb{P}_i) \tag{2-15}$$

The constraints we will use for the quantization of a GMM are summarized by Lemma 1 below.

Lemma 1. *Assuming that all components in the GMM share a common eigen-basis that is aligned with the partitioning $\mathcal{R}$ and that the locations $\mathcal{C}$ are defined on a grid aligned with the distribution's axes (as per Definition 3), we can apply the same grid to each component. This allows us to compute the Wasserstein distance using Equation 2-11. As it is solved per-component, the integral in Equation 2-9 can be solved using closed form expressions for Gaussians defined in the work of [12].*

2-5 Clustering Algorithms

As mentioned in the properties of GMMs, finding the modes is an intractable problem, however methods such as the EM algorithm or clustering methods can be used to find them. The focus will lie on clustering methods to find these modes.

There have several notable works, reviewing all known clustering algorithms. The survey [35] discusses clustering algorithms specific for statistics, computer science and machine learning, and applied them on a few benchmark datasets. The survey [34] focuses on the basic operations clustering algorithms use and compares traditional clustering methods with modern ones. The more recent work [11] discusses an up-to-date comparison of clustering techniques. Finally, [36] gives a rapid overview, using all these surveys mentioned above, with the goal of helping users select the best clustering algorithm for their application. Table 2-1 summarizes the works mentioned and presents all clustering options suited for our problem.

Table 2-1: Chosen clustering algorithms based off [35], [34], [11], [36].

Category	Algorithm	Complexity
Partition based	K-means	$O(NKT)$
Hierarchy based	BIRCH	$O(N)$
Density based	DBSCAN	$O(N \log N)$
Fuzzy based	Fuzzy c-means	$O(NKT)$

Partition based methods such as K-means and Fuzzy c-means require the number of clusters to be given as an input. Since the modes of the GMM are not known in advance, it is difficult to determine the appropriate number of clusters beforehand. Hierarchy methods such as BIRCH cluster the data based off distances. Density based methods find clusters based off distances and density requirements, therefore most suitable for our use case. Additionally, DBSCAN scales well and can be extended to large data sets and/or larger dimensions [10].

Chapter 3

Problem definition

The goal is to find a computationally efficient quantization of a GMM with formal guarantees. The problem is defined as follows.

Problem 1. *For a given threshold error $\epsilon \geq 0$ and support size M,*

$$\begin{aligned} &\textit{find} \quad d \in \mathcal{D}_M(\mathbb{R}^d) \\ &\textit{s.t.} \quad W_2(d, \mathbb{P}) \leq \epsilon \end{aligned} \tag{3-1}$$

To solve Problem 1, we are required to compute the Wasserstein distance between GMMs. This is known to be an intractable problem as there is no closed-form solution, and empirical methods are computationally expensive and scale poorly [26]. Moreover, to ensure mass conservation in the approximation, the discrete representation must preserve the total probability mass of the original GMM. This requires computing masses over regions, which is infeasible in closed form for a GMM due to the mixture's structure as mentioned in Section 2-3-1.

3-1 Approach

This work focuses on the quantization of GMMs, as they offer a more expressive representation of complex distributions compared to single Gaussians. While the quantization problem in Equation 3-1 can be solved analytically for Gaussians by the method introduced by [2], no such optimal solution exists for GMMs. Prior efforts to address this gap include the methods proposed in [2] and [12], but these suffer from scalability limitations.

In contrast to other works, our approach leverages a sample-based method. This is such that we can identify high-density areas within a GMM. Around each cluster, a hyper-rectangular box is built and the locations are strategically placed within. This strategy allows our method to scale effectively while maintaining strong approximation performance.

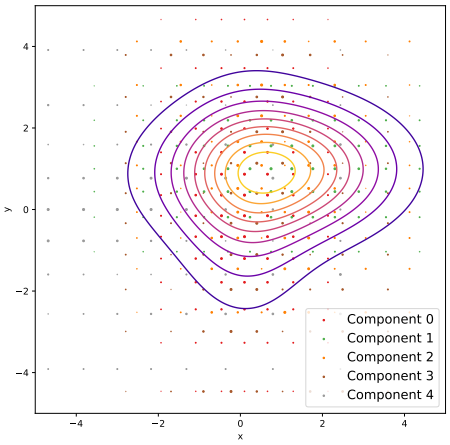
To calculate the Wasserstein distance for the quantization of a GMM we need to solve the constrained 2nd-moment in Equation 2-9. As described in the Preliminaries chapter, GMMs

have dependent dimensions, therefore we cannot calculate the product of marginals or moments over a region. In the case of Gaussian distributions, a closed-form solution exists, making the computation especially efficient.

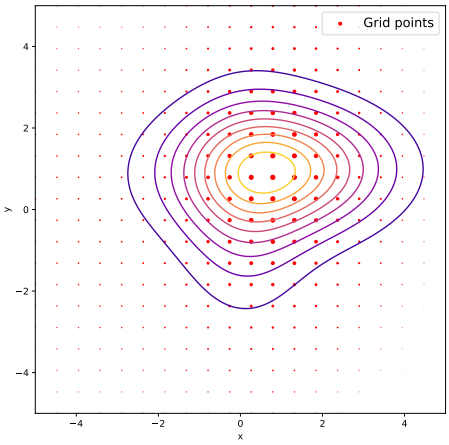
To extend this to GMMs, the quantization operator is applied per Gaussian component. Additionally, we use a grid-based representation for the quantization, where the locations are constrained to a grid structure as according to Definition 3. This will ensure we have regions that are axes-aligned with the dimensions of the distributions such that we can leverage fast, closed-form evaluations of the Wasserstein distance.

3-2 Related Works

The previous state-of-the-art is the quantization operator introduced by [2], which we refer to as the per-component method. To the best of our knowledge, no other dedicated quantization operators for GMMs have been proposed in the literature. As described in the preliminaries, this operator is applied to each Gaussian component separately, finding quantizations per component and taking the union as the approximate quantization of the GMM. This leads to sub-optimal results as it does not take into account the geometry of the whole distribution but just its consecutive parts. Figure 3-1a visualizes the use of the per-component method on a 2D GMM. This way we can also visualize the sub-optimality, as for overlapping components within a GMM, a lot of redundant discrete locations are placed. Other works, such as [12],



(a) Per-component method with 100 locations per component, total of 400 locations.



(b) One-grid method using an uniform grid with 400 locations over space (-5,5) in each dimension.

Figure 3-1: Discretization of an arbitrary 2D GMM with 5 components using methods from previous works, [2] and [12].

apply the same grid to each component, resulting in one quantization over all components. An example is shown in Figure 3-1b. However as this uses a single grid over the entire space, it will automatically place unnecessary locations in areas of empty space.

Chapter 4

The Multi-Grid Method

The aim is to find a discrete approximation of a GMM with minimal locations with formal guarantees using the Wasserstein distance. We can rewrite the problem statement from Chapter 3 using the quantization operator from Equation 2-1:

$$\text{find} \quad \{\mathcal{R}, \mathcal{C}\} \quad \text{for} \quad \Delta_{\mathcal{R},\mathcal{C}} \# \mathbb{P} \tag{4-1}$$

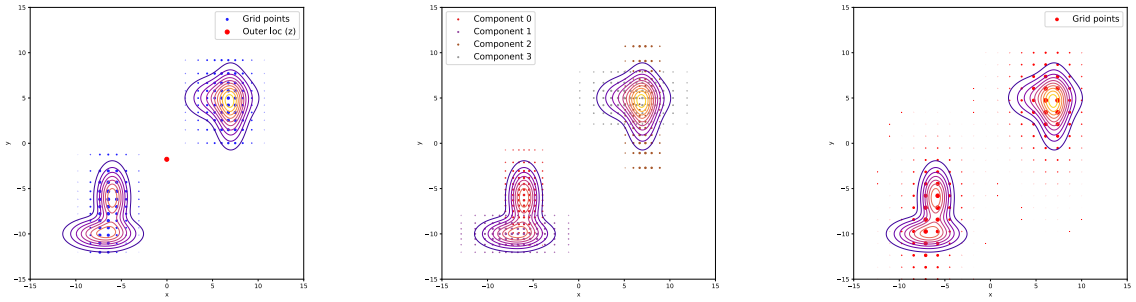
$$\text{s.t.} \quad W_2(\Delta_{\mathcal{R},\mathcal{C}} \# \mathbb{P}, \mathbb{P}) \leq \epsilon \tag{4-2}$$

where the objective is to find a grid, defined by the partitioning $\mathcal{R}$, and locations $\mathcal{C}$, such that the Wasserstein distance of the quantization is bounded by ϵ. The problem still resembles the problem of the previous chapter, because, by defining the quantization using $\mathcal{R}$ and $\mathcal{C}$, the resulting signature can still represent any discrete distribution d. As mentioned in Chapter 2, the operator will be applied component-wise on the GMM. Additionally, as the regions $\mathcal{R}$ are required to align with the GMM, we use the following constraint, defined in Remark 1.

Remark 1. *To define a common grid aligned with all Gaussian components in the GMM, we assume that all components share the same eigen-basis for their covariance matrices. This is not a restrictive assumption, as any distribution can still be approximated by a GMM with axis-aligned covariance matrices, as illustrated in Equation 4-5.*

Lastly, to ensure minimal transport cost over the regions, the Voronoi partitioning is chosen. With this setup we can benefit from closed form solutions for the Wasserstein distance, as shown in Lemma 1 in the preliminaries.

The motivation for the proposed multi-grid method arises from limitations in previous approaches, such as the per-component method. As illustrated in Figure 4-1, in per-component method, each component is treated independently, which can lead to an excessive number of signature locations, many of which may be redundant in densely packed GMMs. Moreover, these methods often fail to distribute mass effectively across signature locations, since they ignore the global structure. Conversely, applying a single uniform grid across the entire space can also be suboptimal, particularly when components are far apart, as it may waste locations in low-density or empty regions, see Figure 4-1. The multi-grid method addresses both



(a) Multi-grid **(b)** Per-component **(c)** One-grid

Figure 4-1: Quantizations of a 2D GMM using the toolbox from [8]. The multi-grid method uses 201 locations in total, over 2 grids, whereas the per-component method uses 400 locations, 100 locations per grid and the one-grid method uses a uniform grid with 400 locations over the space of (-15,10) in each dimension.

issues by adapting to the GMMs geometry: it places grids strategically at the modes of the distribution, ensuring better coverage and efficiency.

The idea is to apply local grids at the modes of the GMM. Each grid will be defined on a bounded space, instead of on the whole $\mathbb{R}^d$, such that it captures just the most important part(s) of the GMM. To find these modes, we benefit from density-based clustering methods that can identify these regions while keeping a low computational cost. Assuming that most mass is within these grids, the mass outside of the grid(s) will be placed at one arbitrary location, named by the outer location z. Once the high density regions and the outer location have been identified, the local grids can be built, defining both $\mathcal{R}$ and $\mathcal{C}$ for the quantization operator, $\Delta_{\mathcal{R},\mathcal{C}}$. The complete multi-grid method is defined by Algorithm 1.

Algorithm 1 Multi-Grids method

Input: $\mathbb{P} = \sum_{i=1}^{\bar{M}} \bar{\pi}_i \mathbb{P}_i$, where $\mathbb{P}_i = \mathcal{N}(\mu_i, \Sigma_i)$
Output: $d = \sum_{k=1}^{N} \mathbb{P}(\mathcal{R}_k)\delta_{c_k}$ and $W_2^2(\mathbb{P}, d)$
1: Find partitioning $\mathcal{R}$ and locations $\mathcal{C}$ by Algorithm 2.
2: Apply quantization operator $\Delta_{\mathcal{R},\mathcal{C}}$ on $\mathbb{P}$ and calculate $W_2^2(\mathbb{P}, d)$ by Algorithm 4.
3: **return** $d = \sum_{k=1}^{N} \mathbb{P}(\mathcal{R}_k)\delta_{c_k}$ and $W_2^2(\mathbb{P}, d)$

The first step of Algorithm 1 is to find the grids, defined by the partitioning $\mathcal{R}$ and locations $\mathcal{C}$. This will be done by Algorithm 2, described in the next section below. Thereafter, the quantization is found, along with the Wasserstein distance, which is presented in Section 4-2 by Algorithm 4.

4-1 Finding Local Grids

The following section outlines the method for selecting appropriate grids. Figure 4-2 represents the problem for an arbitrary 2D GMM, where the quantization locations are constrained by the grid structure defined in Definition 3, resulting in a hyper-rectangular partitioning of the space.

Definition 4. *The boundary of the grid is defined as the 'shell' of a grid.*

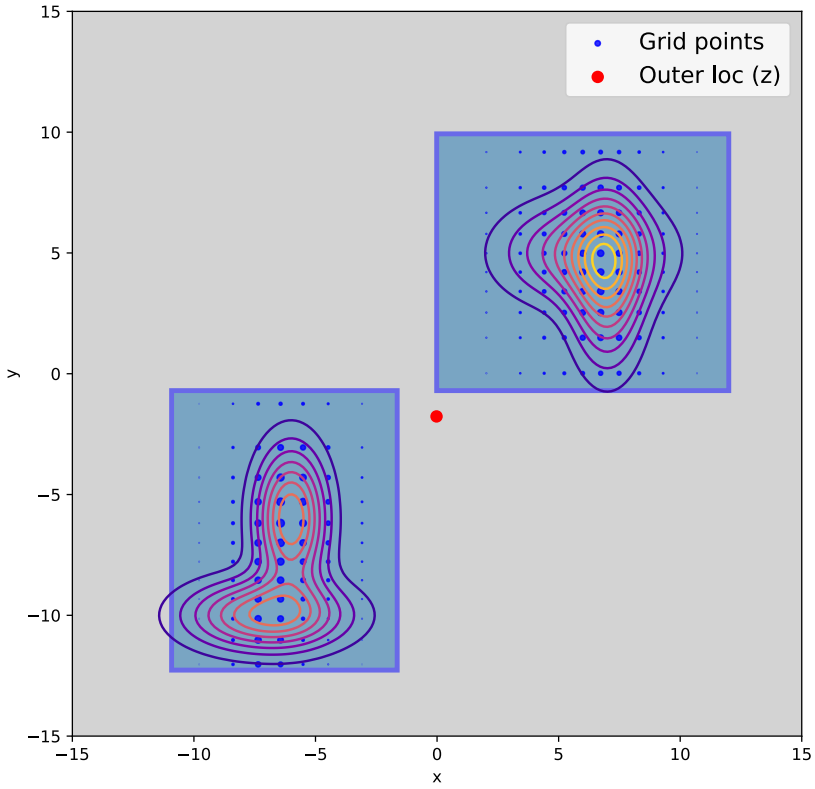


Figure 4-2: Visualization of the 'shell' problem.

There is no closed-form solution to find the optimal shell(s). Therefore an alternative heuristic method has been developed to choose the number of shells, their size, and location, for any arbitrary distribution. The requirement for the shells is that they must be hyper-rectangular in shape and disjoint.

The power of this method is that no prior information about the distribution is required. Instead, to get information about the GMMs structure, a sample-based approach is applied. The clustering method that is most suitable for our use is Density-Based Spatial Clustering of Applications with Noise (DBSCAN), with an application to large spatial databases [10]. It

does not require a set amount of clusters to be found, and no optimization of the centroids is done as in K-means [35]. It will find an arbitrary amount of clusters based on its hyper-parameters, which define the density constraints for a cluster to be formed [10]. The focus lies on finding dense areas within a database, therefore making it the most suitable clustering algorithm for our method. Algorithm 2 outlines the method.

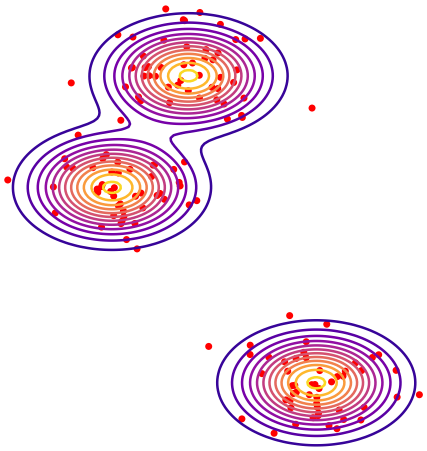
Algorithm 2 Grid Creation via clustering

Input: $\mathbb{P} = \sum_{i=1}^{\bar{M}} \bar{\pi}_i \mathbb{P}_i$, where $\mathbb{P}_i = \mathcal{N}(\mu_i, \Sigma_i)$
Output: $\{(\mathcal{R}_k, \mathcal{C}_k)\}_{k=1}^{K}$

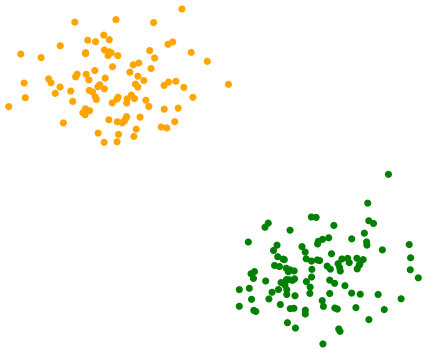
Draw N samples from the distribution: $\{x_j\}_{j=1}^{N} \sim \mathbb{P}$
Cluster data set using the DBSCAN algorithm
for $c = 1$ to C clusters **do**
    Find best Gaussian fit of cluster $\sim \mathcal{N}(\mu_c, \Sigma_c)$.
    Project Σ_c onto basis of shared eigen-basis of grid.
    Set shells as tightest hyper-rectangle around each cluster.
    Expand shells' size by maximum gap between shells in each dimension.
end for
Check overlap of shells, if they do, merge shells together.
for $k = 1$ to K remaining shells **do**
    Generate W2-optimal locations $\mathcal{C}_k$ for $\mathcal{N}(c_k, \Sigma_k)$ using [2] for a set number of locations.
    Define the partitioning $\mathcal{R}_k$ as the Voronoi partitioning w.r.t $\mathcal{C}_k$ within the shell k.
    Generate bounded spatial grid $(\mathcal{R}_k, \mathcal{C}_k)$ using shell and locations $\mathcal{C}_k$
end for
Compute outer location as $z = \sum_{k=1}^{K} \mu_k$.
Add z to the set of locations $\mathcal{C}_k$.
return $\{(\mathcal{R}_k, \mathcal{C}_k)\}_{k=1}^{K}$

The first step is to sample from the GMM. Once the samples have been clustered, the next step is to build shells around each separate cluster. First the tightest shell is built around each cluster. The distances between these shells are then assessed by measuring the gaps along each dimension. To maximize coverage without causing overlap, each shell is subsequently expanded by the largest available gap in its respective dimensions.

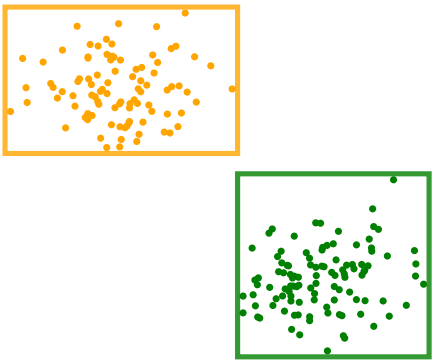
Next, the clusters are approximated by Gaussians based on their centres and spreads. Using [2] we can find the optimal locations, $\mathcal{C}$, for each cluster-approximated-Gaussian. Using the locations $\mathcal{C}$, we define the partitioning $\mathcal{R}$ as the Voronoi partitioning w.r.t $\mathcal{C}$ inside the shell. Using $\mathcal{R}$ and $\mathcal{C}$ we can create the local grids. The algorithm is visualized by Figure 4-3.



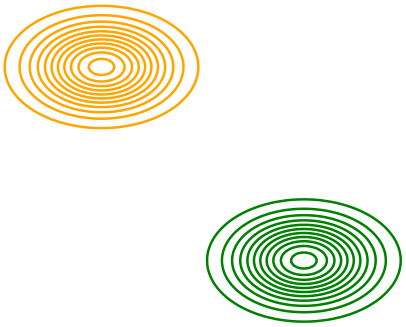
(a) Step 1: Sample from GMM



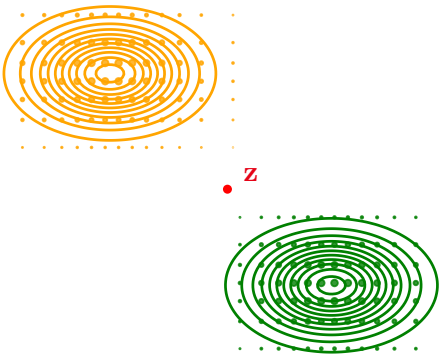
(b) Step 2: Clustering using samples



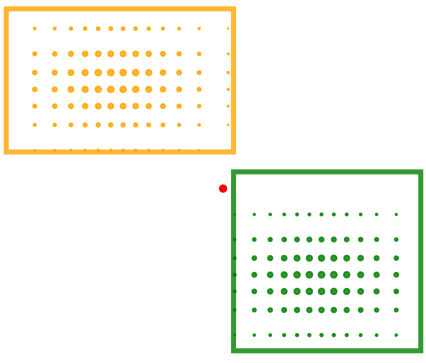
(c) Step 3: Construct shells



(d) Step 4: Approximate clusters by Gaussians



(e) Step 5: Generate locations using [2], place z, and set partitioning.



(f) Step 6: Build local grids using locations $\mathcal{C}$ and partitioning $\mathcal{R}$.

Figure 4-3: Visualization of Algorithm 2.

4-1-1 Complexity & Memory analysis

For the quantization step, we leverage fast closed-form computations of the Wasserstein distance and probability masses. Yet, the addition of Algorithm 2 does add additional complexity. The complexity and memory of Algorithm 2 are analyzed in Table 4-1. The complexity is primarily dependent on the number of samples N, therefore it should be limited. In the Appendix, Figure A-1 shows the effect of the number of samples used in clustering on the overall Wasserstein distance. The samples are kept quite relatively low, at $100 \times \bar{M}$ for $\bar{M}$ components in the GMM, still ensuring performance. Therefore the algorithm is easily scalable to big GMMs, in terms of both dimension and mixture size.

Step	**Time Complexity**	**Memory**
Sampling from GMM	$\mathcal{O}(Nd)$	$\mathcal{O}(Nd)$
Clustering using DBSCAN	$\mathcal{O}(N \log N)$ *	$\mathcal{O}(N)$ to $\mathcal{O}(N^2)$
Find closest Gaussian	$\mathcal{O}(Nd)$	$\mathcal{O}(Kd)$
Find tightest shell per cluster	$\mathcal{O}(K^2 d)$	$\mathcal{O}(Kd)$
Overlap detection and merging	$\mathcal{O}(K^2 d)$	$\mathcal{O}(Kd)$
Compute outer loc z	$\mathcal{O}(Kd)$	$\mathcal{O}(d)$
Total	$\boldsymbol{\mathcal{O}(N \log N + K^2 d)}$	$\boldsymbol{\mathcal{O}(Nd)}$

Table 4-1: Computational Complexity and Memory Usage of Algorithm 2 for N samples, d dimensions, $\bar{M}$ components and K clusters.
*Worst-case DBSCAN complexity is $\mathcal{O}(N^2)$, eg in higher dimensions (d $>$ 5) when KD-tree is not suitable [5]

4-1-2 Hyper-parameter tuning

Within DBSCAN you have two hyper-parameters; ε and *min_samples*. The parameter ε sets the maximum distance of points to be apart from each other to be considered to be in the same cluster. The parameter *min_samples* defines the minimum number of points to be in ε distance from a point to be defined as a core point. The first step in DBSCAN is identifying core points. The clusters are generated around core points. Points that are not reachable from any core point are labelled as noise. The definition of core points and noise can be visualized by Figure 4-4. The next question is how to set these hyper-parameters correctly for our use case.

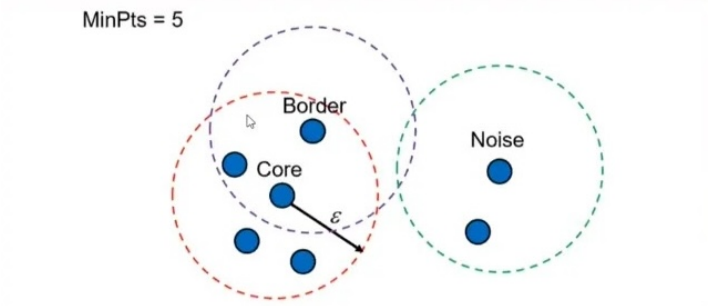


Figure 4-4: Visualization of core, border and noise points in DBSCAN [28].

DBSCAN density parameters

The search for the optimal ε has been extensively researched in literature [21], [1], [27]. However it comes down to the same method that was first introduced in 1996 alongside the DBSCAN algorithm itself by [10].

The distances to each points k nearest neighbors (k-NNs) are computed and sorted to generate a k-distance plot, as shown in Figure 4-5. The parameter k is set equal to $min\_samples$. The "elbow" or "knee" of the curve corresponds to the point where the slope begins to flatten significantly, indicating a transition from dense to sparse regions. Setting ε at this point ensures that only samples in dense regions are grouped into clusters, while outliers and noise are excluded. A practical way to estimate the knee is by taking the 95th percentile of the k-distances. This approach ignores the top 5% of the most sparsely located points (left end of the curve) and emphasizes the denser regions (right end of curve). It also scales with dimension, as in higher dimensions the Euclidean distance grows, leading to higher epsilon values as well. The addition of this step adds a complexity of $\mathcal{O}(NlogN)$ due to the k-NNs search, however DBSCAN already has a minimum complexity of $\mathcal{O}(NlogN)$ so it won't greatly affect the total complexity. The method is summarized in Algorithm 3.

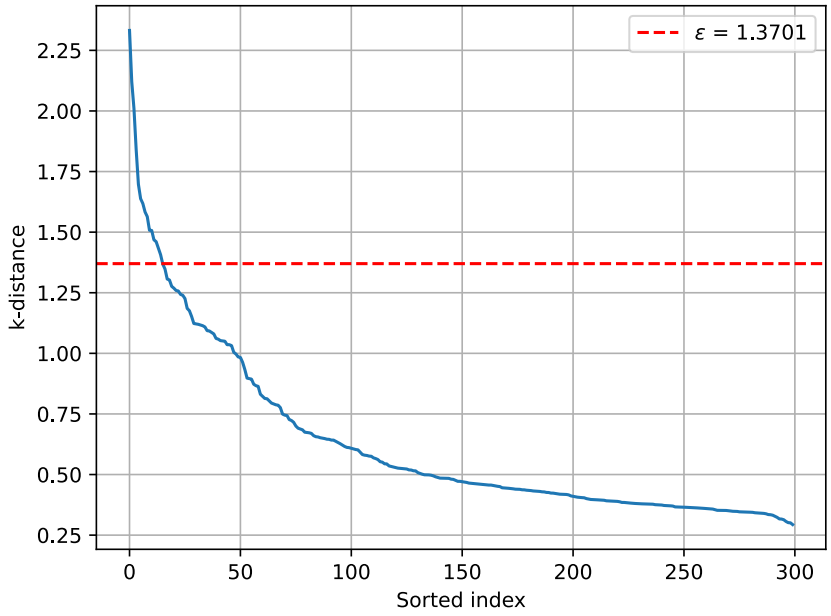


Figure 4-5: k-NN distances in ascending order for samples of an arbitrary 2D GMM

Algorithm 3 ε Selection

Input: Samples $\{x_j\}_{j=1}^{N} \sim \sum_{i=1}^{M} \bar{\pi}_i \mathcal{N}(\mu_i, \Sigma_i)$, $min\_samples = 0.1N$

1: Compute k-NN distances for samples x using $k = min\_samples$
2: Sort distances in ascending order
3: Use 95th percentile or locate Knee in curve using KneeLocator to find ε

Output: ε

Unlike the search for the optimal ε, the parameter *min_samples* has been overlooked [21]. Generally the rule of thumb is to use twice the amount of features in the data, most often being the dimension $(2 \times D)$ [10]. The value of *min_samples* has been tuned manually, and set at 10% of the total number of samples.

GMM samples N
Setting the number of samples for the clustering algorithm is a trade-off between computation time and accuracy. With less samples the shells are a less accurate representation of the distribution leading to higher Wasserstein distances of the resulting quantization. As a quick assessment on the affect of the samples on the Wasserstein distance, 10 arbitrary GMMs are generated and quantized for different sample sizes, see Figure A-1 in the Appendix. The number of samples is set at 100 per component to ensure performance but limit computational complexity.

4-1-3 Validation of Algorithm 2

To verify the performance of the heuristic method to find the optimal shells, we compare the sizes of the shells from Algorithm 2 with an optimization based method.

The optimization chosen is a 1D optimization method, called the Golden-section method [24]. It is an iterative optimization algorithm where at each step a search direction is chosen and then a line search is done to find the optimum. The optimization variable is the shell size, defined by a radius ω, see Figure 4-6. The optimization is done to find the lowest Wasserstein distance of the quantization step using this shell size for the grids.

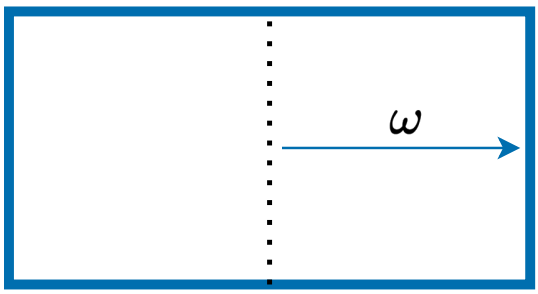


Figure 4-6: Shell defined by ω

The comparison of methods is based on both the probability mass contained within the shells and the corresponding Wasserstein distances. As the primary evaluation metric, the normalized Wasserstein distance is used in place of the Wasserstein distance, its definition and justification are provided in Chapter 5.

From the results in Table 4-2 we can take away that the heuristic can find close to optimal values for the shells, resulting in similar Wasserstein distances and coverage of the distribution. The full results are recorded in the Appendix in Table A-2.

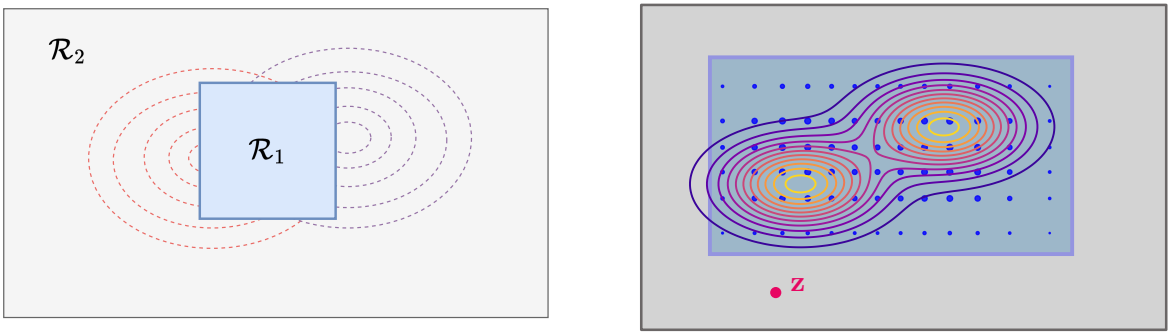
Method	$\bar{W}_2$-error	$\mathbb{P}(\mathcal{R})$
Grid search	0.234 ± 0.089	0.998 ± 0.007
Algorithm 2	0.239 ± 0.091	1.00 ± 0.000

Table 4-2: Comparison of the normalized Wasserstein distance and shell probability mass for Algorithm 2 versus optimized shell sizes using the Golden-section method [24]. The experiment is run for 100 GMMs with different sizes $\bar{M}$ and dimensions d ranging from $[2, 100]$ and $[2, 60]$ respectively. The average values are recorded with their standard deviation. The full results are in Table A-2.

4-2 Quantization

Assuming the grids are pre-determined (generated by Algorithm 2), we can make an approximation of the quantization of a GMM using the method we will now call the 'multi-grid' method.

When using these local grids, the space can be split up into two main regions, the region inside the grid $\mathcal{R}_1$ and the region outside the grid $\mathcal{R}_2$, as seen in Figure 4-7. The goal is to use the Voronoi partitioning inside these grids, as previously mentioned, for minimal Wasserstein distance. For the outside region, all mass will be placed at location z.



(a) Partitioning **(b)** Locations

Figure 4-7: Local grids visualization. (a) Partitioning of regions $\mathcal{R}$. (b) Corresponding locations $\mathcal{C}$ in partitioned regions.

4-2-1 Wasserstein Bound Computation

By using the rule of integration over the union of disjoint sets, the Wasserstein distance can be calculated per region of space. Using the grid structure shown in Figure 4-7, we define the grids by,

$$\{\mathcal{R}, \mathcal{C}\} := \{\{\mathcal{R}_1, \mathcal{R}_2\}, \{\mathcal{C}_1, \mathcal{C}_2\}\},$$

where $\mathcal{R}_1$ is the inner region and $\mathcal{R}_2 = \mathbb{R}^d / \mathcal{R}_1$ is the outer region. The locations inside the grid, $\mathcal{C}_1$ are defined as the locations found by step 11 in Algorithm 2, whereas location $\mathcal{C}_2$ is defined as the outer location, z.

Using the partitions $\mathcal{R} = \{\mathcal{R}_1, \mathcal{R}_2\}$ we can split the 2nd-moment in Equation 2-9 into two parts,

$$W_2^2(\mathbb{P}, \Delta_{\mathcal{R},\mathcal{C}}\#\mathbb{P}) \leq \sum_{k=1}^{N} \int_{\mathcal{R}_1} \|x - c_k\|^2 d\mathbb{P}(x) + \sum_{k=1}^{N} \int_{\mathcal{R}_2} \|x - c_k\|^2 d\mathbb{P}(x).$$

For $\mathcal{R}_1$, the internal region of the grid is split into $\mathcal{R}_k$-Voronoi partitions using locations $\mathcal{C}_1$. For region $\mathcal{R}_2$, the respective locations $\mathcal{C}_2$ are defined as a single point $c_{k=1} = z$,

$$\sum_{k=1}^{N} \int_{\mathcal{R}_2} \|x - c_k\|^2 d\mathbb{P}(x) := \int_{\mathcal{R}_2} \|x - z\|^2 d\mathbb{P}(x).$$

Since the region $\mathcal{R}_2$ is not hyper-rectangular, it does not satisfy Definition 3, and the closed-form solution for the second moment in Equation 2-9 from [12] cannot be applied.

To address this, we apply the rule of integration over the union of disjoint sets, allowing the formulation to be rewritten [29]. For any measurable sets A, B with $B \subset A$, we can write

$$\int_A f(x)\, d\mathbb{P}(x) = \int_B f(x)\, d\mathbb{P}(x) + \int_{A \setminus B} f(x)\, d\mathbb{P}(x).$$

We can split the integral over $\mathcal{R}_2$ (B) over the space of $\mathcal{R}_1 = \mathbb{R}^d/\mathcal{R}_2$ and $\mathbb{R}^d$ (A). Re-arranging results in the Wasserstein distance defined by;

$$\begin{aligned} W_2^2(\mathbb{P}, \Delta_{\mathcal{R},\mathcal{C}}\#\mathbb{P}) &\leq \sum_{k=1}^{N} \left(\int_{\mathcal{R}_1} \|x - c_k\|^2 d\mathbb{P}(x) + \int_{\mathcal{R}_2} \|x - c_k\|^2 d\mathbb{P}(x) \right) \\ &= \int_{\mathbb{R}^d} \|x - z\|^2 \, d\mathbb{P}(x) + \left(\sum_{k=1}^{N} \int_{\mathcal{R}_1} \|x - c_k\|^2 d\mathbb{P}(x) - \int_{\mathcal{R}_1} \|x - z\|^2 \, d\mathbb{P}(x) \right). \end{aligned} \tag{4-3}$$

Where the first and third integral account for the mass of moving region $\mathcal{R}_2$ to location z, by taking the difference between the cost of moving all the mass in $\mathbb{R}^d$ to z and of moving all the mass inside region $\mathcal{R}_1$ to z. The second accounts for the distribution of mass within the grid $\mathcal{R}_1$ with locations $\mathcal{C}_1$. By using the equality in Equation 2-11 we can apply this to the whole GMM.

For multiple disjoint grids, we define the partitions and locations of the grids as:

$$\{\mathcal{R}, \mathcal{C}\} := \{\{\mathcal{R}_{\text{grid}}, \mathcal{R}_{\text{outer}}\}, \{\mathcal{C}, z\}\}$$

where each inner grid is defined by:

$$\{\mathcal{R}_{\text{grid}}, \mathcal{C}\} = \{\mathcal{R}_j, \mathcal{C}_j\}_{j=1}^{G}.$$

Let $\mathcal{R}_{\text{grid}} = \{\mathcal{R}_j \mid j = 1, \ldots, G\}$ be a collection of **disjoint regions**, and let $\mathcal{C} = \{\mathcal{C}_j \mid j = 1, \ldots, G\}$ denote the associated sets of locations within each region. For the outer region we have the space defined by $\mathcal{R}_{\text{outer}}$ with location z.

This construction results in a *multi-grid*, rather than a complete grid as defined previously in Definition 3. The integrals are then combined through additive and subtractive terms as follows.

Theorem 1. *The Wasserstein distance of the multi-grids quantization operator is defined by:*

$$W_2^2(\mathbb{P}, \Delta_{\mathcal{R},\mathcal{C}}\#\mathbb{P}) \leq W_2^2(\mathbb{P}, \Delta_{\mathbb{R}^d,z}\#\mathbb{P}) + \sum_{j=1}^{G} W_2^2(\mathbb{P}, \Delta_{\mathcal{R}_j,C_j}\#\mathbb{P}) - W_2^2(\mathbb{P}, \Delta_{\mathcal{R}_j,z}\#\mathbb{P}) \tag{4-4}$$

$$= \int_{\mathbb{R}^d} \|x - z\|^2 \, d\mathbb{P}(x) + \sum_{j=1}^{G} \left(\sum_{k=1}^{N} \int_{\mathcal{R}_{j,k}} \|x - c_k\|^2 \, d\mathbb{P}(x) - \int_{\mathcal{R}_j} \|x - z\|^2 \, d\mathbb{P}(x) \right)$$

Remark 2. *An important factor to account for is that if the regions were to overlap, it would result in an underestimate of the bound. Therefore, the regions* ***must be disjoint****, otherwise there is no guarantee that Equation 4-4 will upper bound the true Wasserstein distance.*

Additionally, in the case where the GMM does not have components with an equal eigen-basis for their covariance matrix, we can apply the approximation in Remark 3.

Remark 3. *If a GMM has a component $\mathcal{G}$ with a non-diagonal covariance matrix, we can approximate this component by a GMM, $\mathbb{P}_{\mathcal{G}} = \sum_i \bar{\pi}_i \mathcal{N}(\mu_i, \Sigma_i)$, that uses only diagonal covariances Σ_i:*

$$\mu_{\mathcal{G}} = \sum_i \bar{\pi}_i \mu_i$$

$$\Sigma_{\mathcal{G}} = \sum_i \left(\Sigma_i + (\mu_i - \mu_{\mathcal{G}})(\mu_i - \mu_{\mathcal{G}})^\top \right) \tag{4-5}$$

$$\tag{4-6}$$

The additional Wasserstein distance incurred by this approximation can be bounded by the 'discrete method' from [8]:

$$W_2(\mathcal{G}, \mathbb{P}_{\mathcal{G}}) \leq M W_2(\mathcal{G}, \mathbb{P}_{\mathcal{G}}). \tag{4-7}$$

4-2-2 Quantization Step

As the calculation is linearly dependent on the number of grids, number of partitions per grid, and quadratic in the dimensions (due to product of marginals), $\mathcal{O}(NGd^2)$, the computations are relatively fast. The main complexity lies with the computation of the grids, as discussed in the previous section. The complete quantization of the GMM and the calculation of the Wasserstein distance is presented in Algorithm 4.

Algorithm 4 Multi-grid Quantization of GMMs

Input: $\mathbb{P} = \sum_{i=1}^{\bar{M}} \bar{\pi}_i \mathbb{P}_i$, where $\mathbb{P}_i = \mathcal{N}(\mu_i, \Sigma_i)$, $\{\mathcal{R}, \mathcal{C}\}$
Output: $d = \sum_{k=1}^{N} \mathbb{P}(\mathcal{R}_k)\delta_{c_k}$ and $W_2^2(\mathbb{P}, \Delta_{\mathcal{R},\mathcal{C}}\#\mathbb{P})$
1: Define grid for space $\mathbb{R}^d$ and with outer location z
2: Compute $W_2^2(\mathbb{P}, \Delta_{\mathbb{R}^d,z}\#\mathbb{P})$ using Equation 2-11.
3: **for** $j \in G$ **do**
4: Compute $W_2^2(\mathbb{P}, \Delta_{\mathcal{R}_j,C_j}\#\mathbb{P})$ using Equation 2-11.
5: Evaluate $\mathbb{P}(\mathcal{R}_j) = \sum_{i=1}^{M} \bar{\pi}_i \mathbb{P}_i(\mathcal{R}_j)$
6: Define grid for shell of $\mathcal{R}_j$ with outer location z.
7: Compute $W_2^2(\mathbb{P}, \Delta_{\mathcal{R}_j,z}\#\mathbb{P})$ using Equation 2-11.
8: **end for**
9: Compute residual mass and location of z:
10: $z_{\text{mass}} \leftarrow 1 - \sum_{j=1}^{G} \mathbb{P}(\mathcal{R}_j)$
11: Compute total Wasserstein distance using Equation 4-4
12: Stack grid partitions and locations:
13: $\{R, C\} \leftarrow \{\{\mathbb{P}(\mathcal{R}_j), z_{\text{mass}}\}, \{C_j, z\} \mid j \in G\}$
14: **return** $d = \sum_{k=1}^{N} \mathbb{P}(\mathcal{R}_k)\delta_{c_k}$ and $W_2^2(\mathbb{P}, \Delta_{\mathcal{R},\mathcal{C}}\#\mathbb{P})$

The first step of Algorithm 4 involves defining an initial grid over the full space $\mathcal{R} = \mathbb{R}^d$ with locations $\mathcal{C} = z$. The corresponding Wasserstein distance is then computed.

Next, inner grids are defined for each shell, using the partitioning $\mathcal{R} = \mathcal{R}_j$ and locations $\mathcal{C} = \mathcal{C}_j$. In parallel, a grid is defined over each $\mathcal{R}_j$ using the outer location z as the locations $\mathcal{C}$. Wasserstein distances are again computed for these configurations.

The probability masses associated with the inner grids are then accumulated, allowing us to compute the mass at the outer location z. Finally, the total Wasserstein distance is calculated, and the quantization operator from Equation 2-1 is applied to the distribution $\mathbb{P}$.

Chapter 5

Experiments

In the following chapter, we present a series of experiments comparing the performance of our proposed multi-grids method to the current state of the art, referred to as the per-component method. Finally, we demonstrate the application of the multi-grids method within a dynamic system setting. All code is available at github.

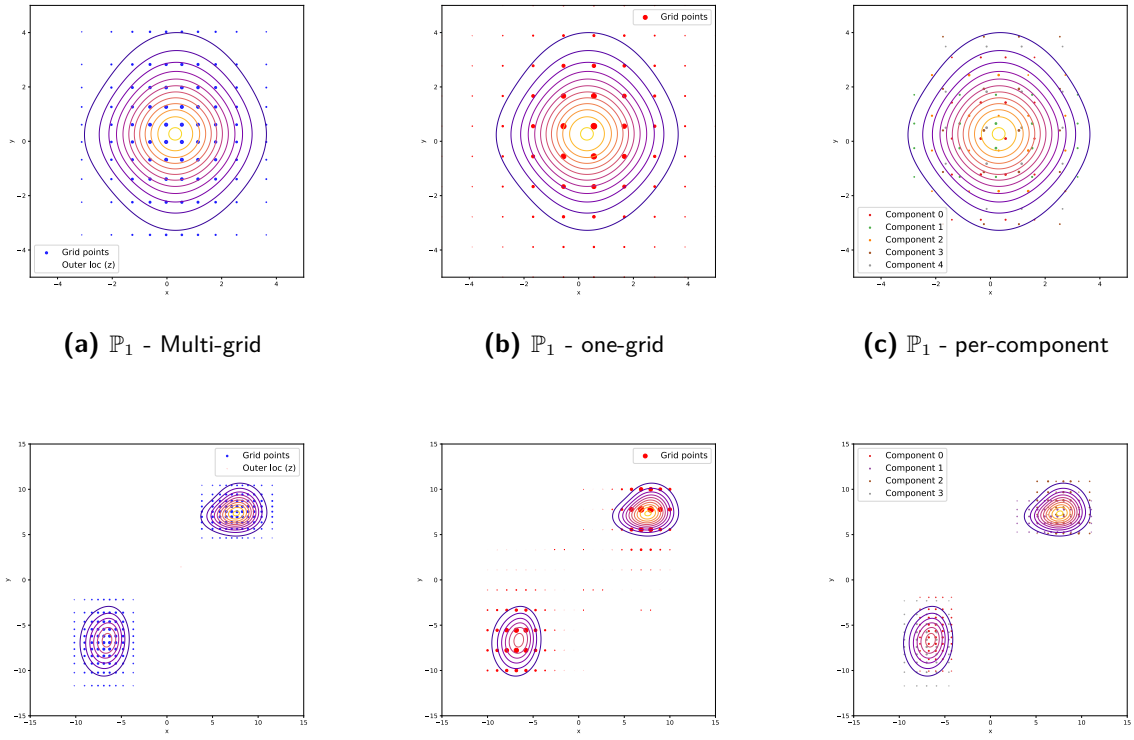
5-1 Baseline Comparison on Low-Dimensional GMMs

We begin by comparing three quantization methods on two representative 2D GMMs: a tightly clustered configuration $\mathbb{P}_1$ and a widely spaced configuration $\mathbb{P}_2$, visualized in the plots below in Figure 5-1.

The one-grid method will be using an uniform grid over the space of the GMM. The per-component method refers to the work of [2] as previously discussed. As the per-component method generates a separate grid per-component, the grids' locations are coloured differently in Figure 5-1 to signify that it is not applied to the entire distribution. The numerical results are presented in Table 5-1, comparing the Wasserstein distances for the equivalent size of the quantization.

GMM	**Method**	W_2-error	Nr of locations
$\mathbb{P}_1$	Multi-grid	0.3549	101
	Per-component	0.6672	100
	One-grid	0.4579	100
$\mathbb{P}_2$	Multi-grid	0.3849	199
	Per-component	0.4392	199
	One-grid	0.7482	200

Table 5-1: Comparison of Wasserstein distance and support size for three quantization methods across two GMM configurations: $\mathbb{P}_1$ (tightly packed) and $\mathbb{P}_2$ (widely spaced).



(a) $\mathbb{P}_1$ - Multi-grid **(b)** $\mathbb{P}_1$ - one-grid **(c)** $\mathbb{P}_1$ - per-component

(d) $\mathbb{P}_2$ - Multi-grid **(e)** $\mathbb{P}_2$ - one-grid **(f)** $\mathbb{P}_2$ - per-component

Figure 5-1: Visual representations of the quantizations of GMMs $\mathbb{P}_1$ (tightly packed) and $\mathbb{P}_2$ (widely spaced). Each row compares three methods: multi-grid, one-grid, and the per-component quantization method. For the one-grid approach, in setting $\mathbb{P}_1$, the grid is defined over (-5,5) in each dimension for 100 locations. In setting $\mathbb{P}_2$ the grid is defined over (-10,10) in each dimension for 200 locations.

The results below clearly demonstrate the advantage of the multi-grid quantization method over the per-component approach. Given the same number of locations, the multi-grid achieves a lower Wasserstein distance by placing locations based on the overall shape of the GMM, rather than treating each component independently. For the case of components spread out throughout space, the comparison is made between multi-grid and one-grid. Instead of using a grid to cover the entire space, multi-grid can place separate local grids, ensuring much lower Wasserstein distances for the same amount of locations.

5-2 Benchmark Tests

In this section we evaluate the performance of our proposed method in comparison to the current state of the art, known as the per-component method from [2].

Setup

To evaluate performance in more general settings, we consider GMMs with densely located components, an important case that motivates our method. The core idea is that when many

components are tightly clustered, it is more efficient to discretize the distribution using a shared grid rather than allocating a separate grid to each component and taking the union of these grids. This strategy allows the representation to better capture the overall shape of the distribution.

We generate synthetic GMMs with dimensions up to $d = 60$ and number of components $\bar{M} \in [2, 100]$. The means are initialized so that any two components are separated by at most $0.5/\sqrt{d}$ units, simulating densely packed components within the GMM. To ensure that all dimensions contribute meaningfully to the distribution, the component variances are also bounded below by a minimum value of 0.1. Without this lower bound, very low variance in some dimensions could lead to an unrepresentative or degenerate GMM structure, especially in high dimensions. We define our GMMs as:

$$\mathbb{P} = \sum_{i=1}^{\bar{M}} \bar{\pi}_i \mathcal{N}(\mu_i, \Sigma_i)^d,$$

where the component means are sampled as

$$\mu_i \sim U(-0.5, 0.5)^d \cdot \frac{1}{\sqrt{d}},$$

and the covariance matrices Σ_i are diagonal, defined as

$$\Sigma_i = \text{diag}(\sigma_{i1}^2, \ldots, \sigma_{id}^2),$$

with

$$\sigma_{ij}^2 = \max(0.1, \tilde{\sigma}_{ij}^2), \quad \tilde{\sigma}_{ij}^2 \sim U(0, 1).$$

To ensure a fair comparison, both methods are restricted to a total of 100 quantization locations, $M \leq M_{max} = 100$. In the per-component method, this is divided evenly across components, i.e., each component is allocated approximately $100/M$ locations. While the multi-grid method uses the full 100 locations for its local grid(s). It is important to note that the multi-grid method will always have $M + 1$ total locations due to the placement of the outer location, z, outside of the grids.

Metrics

The metric used to compare quantizations is $\bar{W}_2$, taken from the paper [2], which is relative to the 2nd-moment of the distribution,

$$\bar{W}_2(\mathbb{P}, \Delta_{\mathcal{R},\mathcal{C}}\mathbb{P}) = \frac{W_2(\mathbb{P}, \Delta_{\mathcal{R},\mathcal{C}}\mathbb{P})}{\|\mu\|^2 + \text{trace}(\Sigma)},$$

where μ is the mean of the GMM and Σ its covariance matrix. This way we can compare the Wasserstein distances across different scenarios.

5-2-1 GMM Size

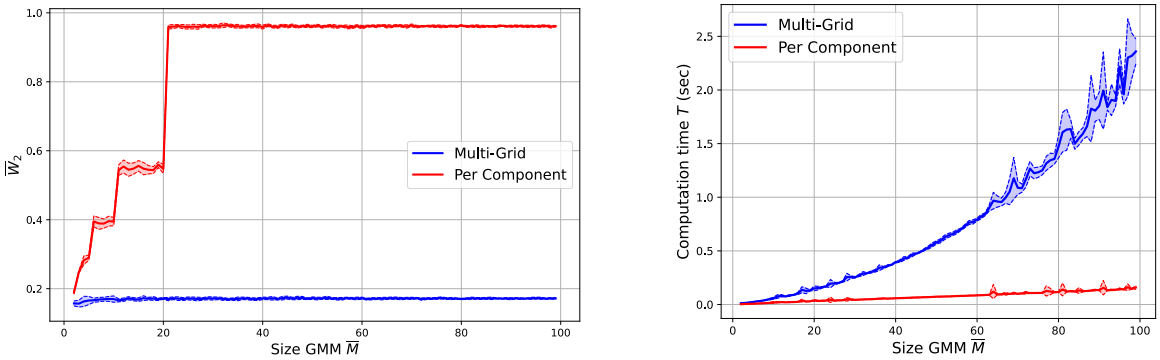
To compare the methods based on their ability to distribute the quantization locations well, we compare 2D GMMs with a varying number of components. For each GMM size the experiment is run 10 times and the average values are recorded.

$$\mathbb{P} = \sum_{i=1}^{\bar{M}} \bar{\pi}_i \mathcal{N}(\mu_i, \Sigma_i)^{d=2},$$

Table 5-2 shows a subset of the results for a varying number of components within the GMM. The full results are available in the Appendix, Tables A-4 and A-5.

$\bar{M}$	$\bar{W}_2$-multi	$\bar{W}_2$-pc	M-multi	M-pc
2	0.157 ± 0.007	0.189 ± 0.005	99.900 ± 0.539	98.300 ± 1.187
4	0.163 ± 0.015	0.283 ± 0.013	100.100 ± 0.539	79.000 ± 1.000
6	0.166 ± 0.006	0.395 ± 0.016	100.600 ± 0.490	57.700 ± 1.005
12	0.168 ± 0.005	0.554 ± 0.019	100.600 ± 0.490	53.700 ± 1.847
22	0.172 ± 0.004	0.960 ± 0.006	100.700 ± 0.458	22.600 ± 1.200
42	0.172 ± 0.005	0.960 ± 0.004	100.800 ± 0.400	42.000 ± 0.000
62	0.171 ± 0.003	0.962 ± 0.003	101.000 ± 0.000	62.000 ± 0.000
72	0.170 ± 0.003	0.961 ± 0.002	101.000 ± 0.000	72.000 ± 0.000
82	0.172 ± 0.003	0.961 ± 0.003	101.000 ± 0.000	82.000 ± 0.000
99	0.172 ± 0.003	0.961 ± 0.002	101.000 ± 0.000	99.000 ± 0.000

Table 5-2: Subset of results from the quantization of 2D GMMs for selected GMM sizes $\bar{M}$ using the multi-gird (multi) and per-component (pc) methods. The methods are both constraint to 100 total quantization locations for their grids. For each chosen GMM size the experiment is run 10 times for different GMMs and the average values are recorded along with their standard-deviation. The full results are recorded in Table A-4.



(a) Average $\bar{W}_2$ **(b)** Runtime T (in seconds)

Figure 5-2: Results of quantization of 2D GMMs with varying number of components $\bar{M}$. $\bar{W}_2$ is the normalized Wasserstein-2 distance and M is the number of quantization locations for each method. The methods are multi-grid (multi) and per-component (pc). The total number of quantization locations is constraint to 100 for each method. For each chosen GMM size the experiment is run 10 times for different GMMs and the average values are recorded along with their standard-deviation. Full numerical results are presented in Table A-4

As the per-component method splits the number of locations over the $\bar{M}$ components, especially for higher $\bar{M}$, the Wasserstein distances increase significantly. Figure 5-2a visualizes the relationship. It is clear to see that the multi-grid method can spread its locations over the space of the entire GMM, leading to lower Wasserstein distances, while the per-component method is more restricted in its placement. For example, in the case of 82 or 99 mixture

components, the multi-grid method can still spread its locations to favour the GMM's shape, while the per-component method is limited to placing a single location per component in the GMM, leading to significantly higher Wasserstein distances. Matching performance with the multi-grid method would demand more quantization points, which comes at the cost of higher complexity when applying it in real-time settings.

It is clear from Table 4-1 that increasing the sample size raises the computational complexity for the multi-grid method, whereas the per-component method is less affected. This trend is further illustrated in Figure 5-2b.

5-2-2 GMM Dimension

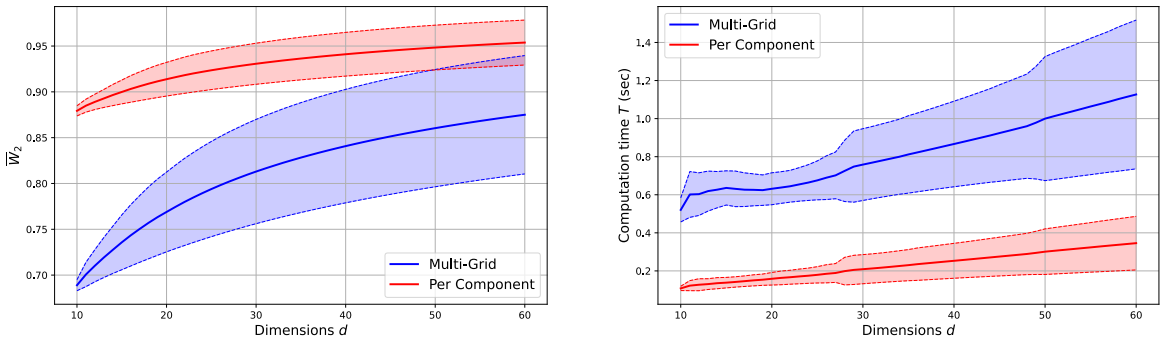
Next, the methods are tested for higher dimensional GMMs ranging from dimensions 10 to 60. The size of the GMM is set at 20 components. For each dimension, 10 different GMMs are taken and their averaged $\bar{W}_2$ and times are taken. The full results are available in the Appendix, Table A-7.

$$\mathbb{P} = \sum_{i=1}^{\bar{M}=20} \bar{\pi}_i \mathcal{N}(\mu_i, \Sigma_i)^d,$$

In higher dimensions, the probability mass of a Gaussian distribution does not concentrate near the mean, but rather forms a region of mass at a distance that scales with $\sqrt{d}$ from the centre [31]. The implication of this is that the components in a GMM become even more dispersed. Implying that for the same amount of quantization locations, a less accurate representation is made, and thus leading to an increase in Wasserstein distance.

For the per-component method the implication is even bigger as it must spread its locations over all components over a bigger space. While the multi-grid method can adapt more easily to the full geometry of the GMM. It is clear to see how well multi-grid can adapt to higher dimensions in Figure 5-3a.

For higher dimensions, the computational complexity of Algorithm 2 increases to order $\mathcal{O}(N^2)$ (see Table 4-1) leading to slightly greater computational times in the multi-grid method. However, the computation time remains on the order of seconds, as in Figure 5-2b. To further reduce runtime, one could decrease the number of samples N. Nonetheless, this comes at the cost of increased Wasserstein distances, as illustrated in Figure A-1.



(a) Average $\bar{W}_2$ **(b)** Runtime T (in seconds)

Figure 5-3: Quantization results of densely packed GMMs in increasing dimensions ($d = 10$ to 60), comparing the multi-grid (multi) and per-component (pc) methods in terms of Wasserstein distance and runtime. For each dimension case, 10 arbitrary GMMs are taken and the average values are recorded with their standard deviations for both $\bar{W}_2$ and time T. Each method is limited to 100 total quantization locations.

5-2-3 GMM's Variance

The last experiment will focus on the affect of the variance of a GMM on the quantization methods. We take an arbitrary GMM, with set means and variance, however we scale the variance by $\sigma_{ij}^2 = 0.1 + \texttt{Iterations} * 0.1$ at every iteration.

$$\mathbb{P} = \sum_{i=1}^{\bar{M}=15} \bar{\pi}_i \mathcal{N}(\mu_i, \Sigma_i)^{d=10},$$

Figure 5-4 highlights the advantage of the multi-grid method over the per-component approach. The full results are in Table A-9 in the Appendix. The per-component method must place its locations equally across all components, which limits its ability to adapt to the spread of the overall distribution. While the multi-grid method can place locations across one or more shared grids, enabling it to more effectively capture the global structure and shape of the distribution, even as the variance increases.

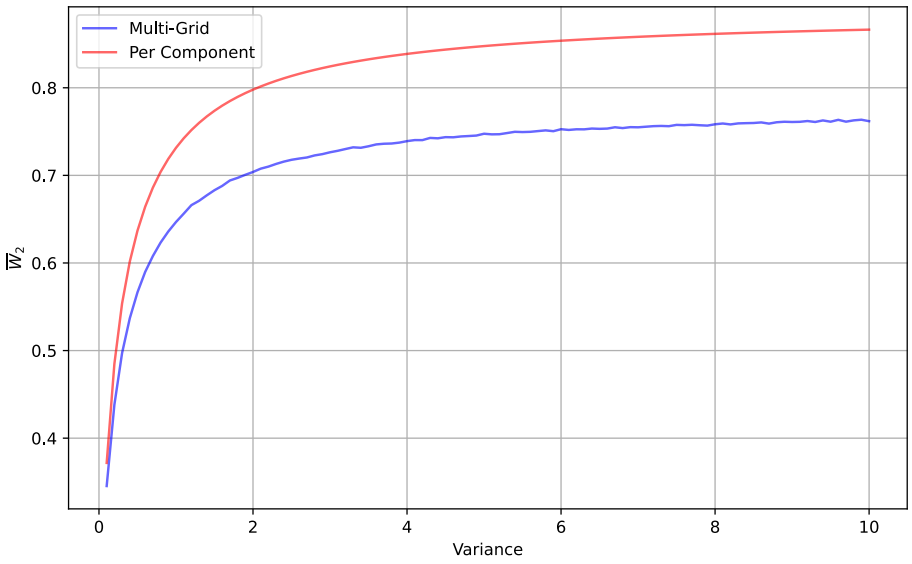


Figure 5-4: Results of quantization of a 15D-GMM with 10 components, where the variance is scaled for 100 runs, such that it increases by $0.1 + \texttt{Iterations} * 0.1$ in each dimension. The $\bar{W}_2$ metric is plotted for the multi-grid and per-component method over the variance. The total number of quantization locations is constraint to 100 for each method.

5-3 Uncertainty Propagation

As mentioned in the motivation, in model-based decision making, distributions can be used to model parts of the dynamics. When these distributions are propagated through model functions, approximations must be made. In the next section we will apply the multi-grid quantization method during uncertainty propagation of two different models.

5-3-1 System Evaulation

Say we have a dynamical system with additive noise;

$$x_{k+1} = f(x_k, u_k) + \epsilon, \tag{5-1}$$

$$u_k = g(x_k) \tag{5-2}$$

where $\epsilon \sim \mathcal{N}(\mu_\epsilon, \Sigma_\epsilon)$ and $x_0 \sim \mathcal{N}(\mu_0, \Sigma_0)$. The states are described by distributions,

$$\mathbb{P}(x_{k+1}) = f\#\mathbb{P}(x_k) * \mathbb{P}(\epsilon), \tag{5-3}$$

where $\mathbb{P}(x_0) = \mathcal{N}(\mu_0, \Sigma_0)$. However as it is intractable to solve Equation 5-3, the distributions are approximated such that we can propagate the dynamics,

$$\tilde{\mathbb{P}}(x_{k+1}) = f\#(\Delta_{\mathcal{R},\mathcal{C}}\#\tilde{\mathbb{P}}(x_k)) * \mathbb{P}(\epsilon), \tag{5-4}$$

where $\tilde{\mathbb{P}}(x_0) = \mathbb{P}(x_0)$. As we take the convolution of the quantization of $\tilde{\mathbb{P}}(x_k)$ with the noise, described by a normal distribution, the output results in a GMM, $\sum_i^N \pi^i \mathcal{N}(c_i + \mu_\epsilon, \Sigma_\epsilon)$, where the mean is the addition of the previous locations of the quantization with the mean of the noise, while the covariance is just dependent on the noise, see Remark 4.

Remark 4. *Taking the locations c_i from $\Delta_{\mathcal{R},\mathcal{C}}\#\tilde{\mathbb{P}}(x_k) = \sum_i^N \delta_{c_i}\pi^i$, and applying the push forward from the dynamics on the quantization, it results in $\sum_i^N \delta_{c_i}\pi^i$. Then taking the convolution with the noise distribution $\mathbb{P}(\epsilon)$, it results in;*

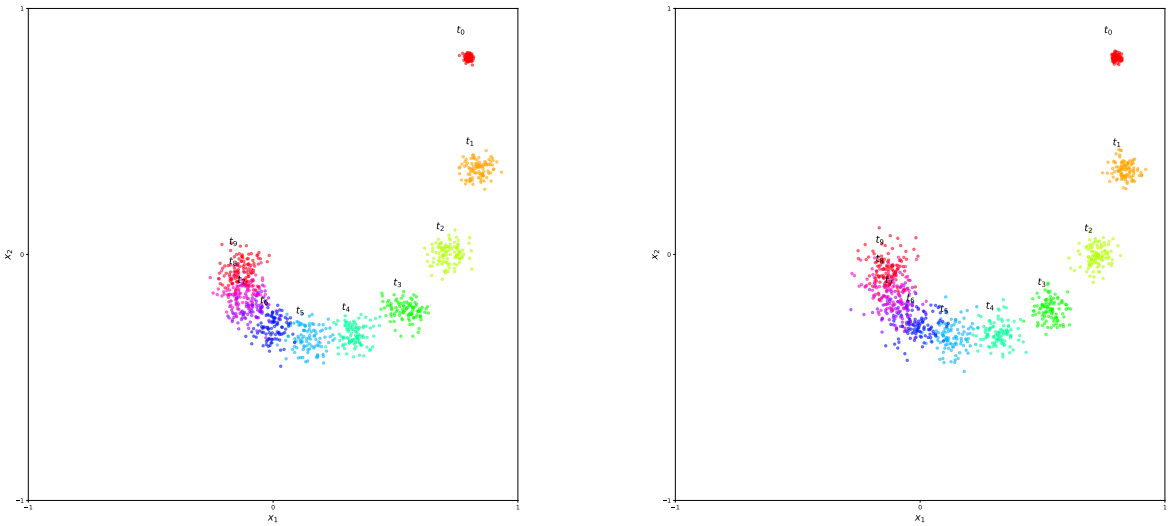
$$(\sum_i^N \delta_{c_i}\pi^i) * \mathcal{N}(\mu_\epsilon, \Sigma_\epsilon) := \sum_i^N \pi^i(\delta_{c_i} * \mathcal{N}(\mu_\epsilon, \Sigma_\epsilon)) := \sum_i^N \pi^i \mathcal{N}(c_i + \mu_\epsilon, \Sigma_\epsilon).$$

As the covariance matrix of all components in the GMM is the same, we have an equal eigenbasis, the perfect setting to apply our new quantization method.

5-3-2 2D Linear System

We will propagate the distributions through the model functions of a simple 2D linear system. The system is given by $f(x) = Ax$, with initial state $x_0 \sim \mathcal{N}\left(\begin{bmatrix}0.8\\0.8\end{bmatrix}, \begin{bmatrix}0.0001 & 0\\0 & 0.0001\end{bmatrix}\right)$, $A = \rho\begin{bmatrix}\cos(\theta) & -\sin(\theta)\\ \sin(\theta) & \cos(\theta)\end{bmatrix} + \delta$, with $\theta = -\frac{\pi}{8}$, $\rho = 0.8$, and $\delta = 0$. The additive noise is given by $\epsilon \sim \mathcal{N}\left(\begin{bmatrix}0\\0\end{bmatrix}, 0.001 \cdot \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\right)$

Figure 5-5 shows the evolution of the distribution $\mathbb{P}(x_k)$ over 10 time steps under linear dynamics $f(x) = Ax$ with additive Gaussian noise. We compare the propagated distribution using the multi-grid quantization operator (left), represented by $\tilde{\mathbb{P}}(x_k)$, with the empirical estimate of the true distribution (right), $\mathbb{P}(x_k)$.



(a) Multi-grid quantization **(b)** Empirical estimate

Figure 5-5: Propagation of $\mathbb{P}(x_k)$ using linear dynamics with additive noise $\sim \mathcal{N}(0, 10^{-3}I)$ over 10 time steps. For both the quantization and empirical approximation 100 locations/samples are used.

Time Step t	$W_2(\tilde{\mathbb{P}}(x_k), \Delta_{\mathcal{R},\mathcal{C}}\#\tilde{\mathbb{P}}(x_k))$	$W_2(\mathbb{P}(x_{k+1}), \tilde{\mathbb{P}}(x_{k+1}))$
0	0.0021	0.0165
1	0.0070	0.0207
2	0.0087	0.0200
3	0.0096	0.0287
4	0.0101	0.0217
5	0.0104	0.0301
6	0.0106	0.0248
7	0.0108	0.0299
8	0.0108	0.0235
9	0.0109	0.0252

Table 5-3: W_2 distances per time step for linear dynamics. The first column is the quantization error and the second is the Wasserstein distance between the quantization and the true distribution for the next state. This value indicates how close our multi-grid method is to the 'true' distribution. For both the quantization and empirical approximation 100 locations/samples are used.

Table 5-3 provides the Wasserstein distances per time step for the quantization step, $W_2(\tilde{\mathbb{P}}(x_k), \Delta_{\mathcal{R},\mathcal{C}}\#\tilde{\mathbb{P}}(x_k))$. The second Wasserstein distance is between the quantization and the true distribution, using an empirical estimate, $W_2(\mathbb{P}(x_{k+1}), \tilde{\mathbb{P}}(x_{k+1}))$. To calculate $W_2(\mathbb{P}(x_{k+1}), \tilde{\mathbb{P}}(x_{k+1}))$ we use discrete optimal transport, which equates to a linear programming problem [14]. The

multi-grid method consistently maintains a low Wasserstein distance for both its quantization step as the comparison with the empirical distribution.

The quantization error stabilizes after just a few time steps (after step 4), suggesting that the discrete locations and grid structure remain well-aligned with the evolving distribution. Additionally, from these results we can also conclude that 100 discrete locations is enough to capture all fine-scale features of the distributions.

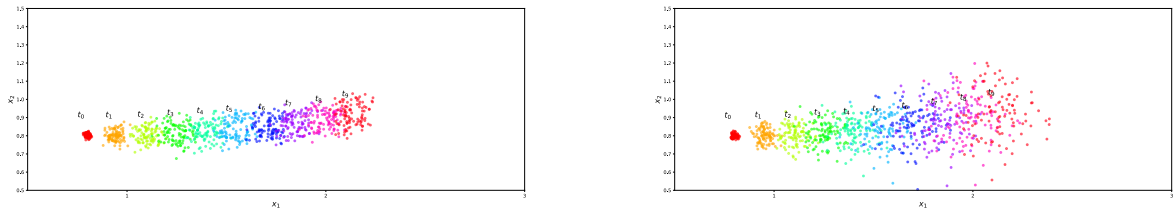
The Wasserstein distance between the quantization and the empirical distribution also stays quite low and consistent over time, which confirms that our method provides reliable approximations.

5-3-3 3D Dubins Car

The next model is the 3D Dubins car model from [4]. We set the velocity at a constant value of $v = 1.5$, with constant input $\theta = 0.2$, and initial state $x_0 \sim \mathcal{N}\left(\begin{bmatrix}0.8\\0.8\\0.0\end{bmatrix}, 0.0001\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}\right)$.

The additive noise is defined by $\epsilon \sim \mathcal{N}\left(\begin{bmatrix}0\\0\\0\end{bmatrix}, 0.001 \cdot \begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}\right)$.

Figure 5-6 compares the multi-grid propagated distribution with the empirical result across 10 time steps. Again the left plot is the propagated distribution using the multi-grid quantization operator, represented by $\tilde{\mathbb{P}}(x_k)$ (left), and the right plot is the empirical estimate of the true distribution, $\mathbb{P}(x_k)$.



(a) Multi-grid quantization **(b)** Empirical estimate

Figure 5-6: Propagation of $\mathbb{P}(x_k)$ under Dubins Car dynamics with control input $\theta = 0.2$, $\Delta t = 0.1$, and noise $\sim \mathcal{N}(0, 10^{-3}I)$ for 10 time steps. For both the quantization and empirical approximation 100 locations/samples are used.

Table 5-4 presents the Wasserstein distances for this system. The Wasserstein distances are slightly higher in comparison to the linear dynamics. Unlike the linear case, the state distribution deforms in more complex ways, making accurate approximation more challenging. This is especially clear in the last few time steps in Figure 5-6.

The quantization error, grows gradually but remains bounded. However, the Wasserstein distance between the quantized and empirical distributions increases more sharply over time. This suggests that a fixed number of quantization locations may not be sufficient to fully capture the evolving complexity of the distribution, and that additional locations may be necessary for improved accuracy.

Time Step t	$W_2(\tilde{\mathbb{P}}(x_k), \Delta_{\mathcal{R},\mathcal{C}}\#\tilde{\mathbb{P}}(x_k))$	$W_2(\mathbb{P}(x_{k+1}), \tilde{\mathbb{P}}(x_{k+1}))$
0	0.0053	0.0264
1	0.0174	0.0368
2	0.0229	0.0429
3	0.0268	0.0528
4	0.0300	0.0541
5	0.0327	0.0604
6	0.0353	0.0565
7	0.0386	0.0627
8	0.0397	0.0720
9	0.0413	0.0751

Table 5-4: W_2 distances per time step for Dubins Car dynamics. The first column is the quantization error and the second is the Wasserstein distance between the quantization and the true distribution for the next state. This value indicates how close our multi-grid method is to the 'true' distribution. For both the quantization and empirical approximation 100 locations/samples are used.

Chapter 6

Conclusion

We introduced a new quantization operator for GMMs, named the multi-grid method, with formal error bounds defined by the Wasserstein distance. Our approach leverages clustering algorithms to identify the modes of a GMM. The method is compared against the previous state-of-the-art - the per-component method from [2] - in various different settings. In terms of the Wasserstein distance, in all experiments, the error bound for the multi-grid method is significantly lower than of the previous method. The key advantage of our approach lies in its scalability to higher-dimensional spaces and larger sized GMMs, as demonstrated empirically.

Matching the performance of the multi-grid method would require the per-component approach to use a substantially larger number of quantization locations, which becomes impractical in real-time applications. This further highlights the efficiency and scalability of the multi-grid method.

Finally the operator is applied during uncertainty propagation of two different models, a simple 2D linear system and Dubins car [4], further illustrating its practical effectiveness.

One potential direction for future research involves relaxing the assumption that all components in the GMM share a common eigen-basis for their covariance matrices. This assumption is currently made to ensure we can use a shared, axis-aligned grid across all components. Investigating methods that allow for varying eigen-bases, while still ensuring efficient quantizations, could broaden the applicability of our method. Another potential direction is investigating other ways of finding and placing the shells. Currently we assume we do not know any information about the input distribution, which does make our approach very general. However, including information from the distribution, being a GMM, could help with the placement of even more optimal shells, possibly leading to even lower Wasserstein distances. In addition to improving the search algorithm for the optimal shells, another related direction is the development of a more computationally efficient algorithm. Although the current method maintains runtimes in the order of seconds, this may become a limitation in time-critical applications such as autonomous driving systems [17].

Appendix A

Appendix

A-1 Tables

The following tables include the full numerical results from the experiments run in Chapter 4 and 5.

A-1-1 Validation Algorithm 2

Run ID	$\bar{W}_2$ Grid Search	$\bar{W}_2$ DBSCAN	$\mathbb{P}(\mathcal{R}_{\text{grid search}})$	$\mathbb{P}(\mathcal{R}_{\text{DBSCAN}})$
1	0.199552	0.199300	0.9999	1.0000
2	0.339783	0.340206	0.9704	1.0000
3	0.293735	0.295457	1.0000	1.0000
4	0.145545	0.145853	1.0000	1.0000
5	0.232156	0.232255	1.0000	1.0000
6	0.316742	0.316877	0.9991	1.0000
7	0.246115	0.245851	1.0000	1.0000
8	0.320529	0.322475	1.0000	1.0000
9	0.277583	0.346630	0.9966	1.0000
10	0.070830	0.070775	1.0000	1.0000
11	0.329692	0.328687	0.9999	1.0000
12	0.070202	0.070394	1.0000	1.0000
13	0.261273	0.261817	1.0000	1.0000
14	0.183417	0.183232	1.0000	1.0000
15	0.284980	0.287673	1.0000	1.0000
16	0.194808	0.193281	1.0000	1.0000
17	0.153758	0.138773	0.9937	1.0000
18	0.344619	0.343804	1.0000	1.0000
19	0.251889	0.251624	0.9993	1.0000

Run ID	$\bar{W}_2$ Grid Search	$\bar{W}_2$ DBSCAN	$\mathbb{P}(\mathcal{R}_{\text{grid search}})$	$\mathbb{P}(\mathcal{R}_{\text{DBSCAN}})$
20	0.290009	0.288885	0.9974	1.0000
21	0.296745	0.296989	0.9999	1.0000
22	0.264472	0.264485	1.0000	1.0000
23	0.222611	0.224915	1.0000	1.0000
24	0.337109	0.336891	0.9990	1.0000
25	0.133929	0.134288	1.0000	1.0000
26	0.176090	0.196157	1.0000	1.0000
27	0.312282	0.311320	1.0000	1.0000
28	0.143817	0.143756	1.0000	1.0000
29	0.295662	0.300516	0.9997	1.0000
30	0.230004	0.231449	0.9975	1.0000
31	0.058955	0.058202	1.0000	1.0000
32	0.204173	0.349511	0.9810	1.0000
33	0.070253	0.070261	1.0000	1.0000
34	0.141051	0.141246	1.0000	1.0000
35	0.320181	0.329097	1.0000	1.0000
36	0.069243	0.069391	1.0000	1.0000
37	0.223744	0.223531	0.9940	1.0000
38	0.079768	0.079604	1.0000	1.0000
39	0.179273	0.179866	1.0000	1.0000
40	0.264519	0.264836	0.9975	1.0000
41	0.334993	0.332780	0.9997	1.0000
42	0.352297	0.350945	1.0000	1.0000
43	0.223195	0.241696	0.9907	1.0000
44	0.347873	0.347286	0.9985	1.0000
45	0.243716	0.244052	1.0000	1.0000
46	0.252307	0.254596	0.9990	1.0000
47	0.171904	0.171957	0.9905	1.0000
48	0.261009	0.261179	1.0000	1.0000
49	0.142159	0.141374	1.0000	1.0000
50	0.229909	0.230237	1.0000	1.0000
51	0.119021	0.117810	0.9983	1.0000
52	0.077816	0.078523	1.0000	1.0000
53	0.144878	0.145461	1.0000	1.0000
54	0.299195	0.302096	0.9992	1.0000
55	0.192090	0.191846	1.0000	1.0000
56	0.177599	0.178036	1.0000	1.0000
57	0.227608	0.282403	0.9420	1.0000
58	0.258100	0.258542	1.0000	1.0000
59	0.342675	0.343738	0.9980	1.0000
60	0.348659	0.349576	0.9999	1.0000
61	0.246315	0.247820	1.0000	1.0000
62	0.312095	0.313944	0.9977	1.0000
63	0.281783	0.283641	0.9850	1.0000
64	0.306550	0.307195	1.0000	1.0000

Run ID	$\bar{W}_2$ Grid Search	$\bar{W}_2$ DBSCAN	$\mathbb{P}(\mathcal{R}_{\text{grid search}})$	$\mathbb{P}(\mathcal{R}_{\text{DBSCAN}})$
65	0.217746	0.217694	1.0000	1.0000
66	0.302441	0.304350	0.9999	1.0000
67	0.322972	0.323384	0.9909	1.0000
68	0.259070	0.256677	1.0000	1.0000
69	0.336525	0.334711	0.9992	1.0000
70	0.334013	0.336275	1.0000	1.0000
71	0.220377	0.220116	1.0000	1.0000
72	0.333583	0.337188	0.9979	1.0000
73	0.077685	0.077846	1.0000	1.0000
74	0.183632	0.183245	1.0000	1.0000
75	0.363674	0.363215	1.0000	1.0000
76	0.360642	0.361447	0.9993	1.0000
77	0.137597	0.137551	1.0000	1.0000
78	0.230468	0.230903	0.9999	1.0000
79	0.310566	0.310231	1.0000	1.0000
80	0.288759	0.288588	0.9999	1.0000
81	0.290437	0.290190	1.0000	1.0000
82	0.135662	0.138965	1.0000	1.0000
83	0.336289	0.336758	1.0000	1.0000
84	0.224124	0.222854	1.0000	1.0000
85	0.322948	0.322746	1.0000	1.0000
86	0.351149	0.349838	0.9979	1.0000
87	0.186491	0.186823	0.9999	1.0000
88	0.185410	0.185532	1.0000	1.0000
89	0.137854	0.137811	1.0000	1.0000
90	0.065469	0.065424	1.0000	1.0000
91	0.201980	0.243525	1.0000	1.0000
92	0.352942	0.353719	0.9990	1.0000
93	0.315168	0.326893	1.0000	1.0000
94	0.071637	0.071718	1.0000	1.0000
95	0.232275	0.231823	1.0000	1.0000
96	0.068365	0.068327	1.0000	1.0000
97	0.265056	0.265189	1.0000	1.0000
98	0.361121	0.361958	1.0000	1.0000
99	0.073641	0.073588	1.0000	1.0000
100	0.252329	0.335007	0.9967	1.0000

Table A-2: Comparison of the normalized Wasserstein-2 errors $\bar{W}_2$ and probability mass within the grids of Algorithm 2 versus optimized shell sizes for minimal Wasserstein error using the Golden-section method. Experiment is run for 100 GMMs with different sizes and dimensions.

A-1-2 GMM Size Tests

$\bar{M}$	$\bar{W}_2$-multi	$\bar{W}_2$-pc	M-multi	M-pc
2	0.157 ± 0.007	0.189 ± 0.005	99.900 ± 0.539	98.300 ± 1.187
3	0.157 ± 0.010	0.249 ± 0.004	100.700 ± 0.640	88.600 ± 0.917
4	0.163 ± 0.015	0.283 ± 0.013	100.100 ± 0.539	79.000 ± 1.000
5	0.166 ± 0.012	0.290 ± 0.007	100.400 ± 0.663	98.800 ± 1.600
6	0.166 ± 0.006	0.395 ± 0.016	100.600 ± 0.490	57.700 ± 1.005
7	0.168 ± 0.006	0.389 ± 0.017	100.400 ± 0.663	67.800 ± 1.249
8	0.169 ± 0.006	0.389 ± 0.014	100.500 ± 0.500	77.400 ± 0.917
9	0.170 ± 0.007	0.396 ± 0.014	100.700 ± 0.458	86.400 ± 1.281
10	0.170 ± 0.009	0.395 ± 0.012	100.900 ± 0.300	97.000 ± 1.095
11	0.166 ± 0.004	0.544 ± 0.015	100.100 ± 0.539	49.500 ± 2.156
12	0.168 ± 0.005	0.554 ± 0.019	100.600 ± 0.490	53.700 ± 1.847
13	0.172 ± 0.004	0.545 ± 0.019	100.700 ± 0.458	57.900 ± 1.300
14	0.169 ± 0.005	0.548 ± 0.016	100.600 ± 0.490	61.800 ± 1.778
15	0.168 ± 0.005	0.556 ± 0.015	100.800 ± 0.400	65.400 ± 1.908
16	0.173 ± 0.007	0.548 ± 0.018	101.000 ± 0.000	70.900 ± 2.343
17	0.171 ± 0.004	0.545 ± 0.012	100.800 ± 0.400	74.500 ± 1.628
18	0.172 ± 0.005	0.544 ± 0.010	101.000 ± 0.000	80.100 ± 1.700
19	0.169 ± 0.003	0.560 ± 0.008	101.000 ± 0.000	82.800 ± 1.778
20	0.172 ± 0.002	0.547 ± 0.013	100.800 ± 0.400	90.000 ± 2.191
21	0.170 ± 0.005	0.959 ± 0.006	100.800 ± 0.400	21.600 ± 1.200
22	0.172 ± 0.004	0.960 ± 0.006	100.700 ± 0.458	22.600 ± 1.200
23	0.171 ± 0.005	0.962 ± 0.004	100.800 ± 0.400	23.000 ± 0.000
24	0.170 ± 0.004	0.960 ± 0.006	100.800 ± 0.400	25.800 ± 1.470
25	0.172 ± 0.004	0.959 ± 0.006	100.600 ± 0.490	25.300 ± 0.900
26	0.171 ± 0.005	0.960 ± 0.007	100.600 ± 0.490	26.000 ± 0.000
27	0.170 ± 0.005	0.959 ± 0.007	100.700 ± 0.458	27.000 ± 0.000
28	0.171 ± 0.004	0.963 ± 0.004	100.900 ± 0.300	28.000 ± 0.000
29	0.172 ± 0.004	0.961 ± 0.004	100.900 ± 0.300	29.000 ± 0.000
30	0.172 ± 0.005	0.961 ± 0.005	100.900 ± 0.300	30.000 ± 0.000
31	0.172 ± 0.004	0.961 ± 0.005	100.900 ± 0.300	31.000 ± 0.000
32	0.170 ± 0.004	0.964 ± 0.007	100.900 ± 0.300	32.000 ± 0.000
33	0.169 ± 0.003	0.964 ± 0.004	100.900 ± 0.300	33.000 ± 0.000
34	0.170 ± 0.003	0.962 ± 0.005	101.000 ± 0.000	34.000 ± 0.000
35	0.170 ± 0.005	0.961 ± 0.005	101.000 ± 0.000	35.000 ± 0.000
36	0.172 ± 0.004	0.960 ± 0.004	100.900 ± 0.300	36.000 ± 0.000
37	0.171 ± 0.004	0.961 ± 0.005	101.000 ± 0.000	37.000 ± 0.000
38	0.170 ± 0.004	0.961 ± 0.005	100.900 ± 0.300	38.000 ± 0.000
39	0.170 ± 0.004	0.962 ± 0.006	101.000 ± 0.000	39.000 ± 0.000
40	0.172 ± 0.004	0.959 ± 0.006	101.000 ± 0.000	40.000 ± 0.000
41	0.172 ± 0.003	0.964 ± 0.004	101.000 ± 0.000	41.000 ± 0.000
42	0.172 ± 0.005	0.960 ± 0.004	100.800 ± 0.400	42.000 ± 0.000
43	0.170 ± 0.003	0.959 ± 0.004	101.000 ± 0.000	43.000 ± 0.000
44	0.173 ± 0.005	0.963 ± 0.004	101.000 ± 0.000	44.000 ± 0.000
45	0.174 ± 0.004	0.961 ± 0.005	101.000 ± 0.000	45.000 ± 0.000
46	0.171 ± 0.003	0.963 ± 0.004	101.000 ± 0.000	46.000 ± 0.000

$\bar{M}$	$\bar{W}_2$-multi	$\bar{W}_2$-pc	M-multi	M-pc
47	0.173 ± 0.005	0.960 ± 0.004	101.000 ± 0.000	47.000 ± 0.000
48	0.173 ± 0.004	0.961 ± 0.003	100.900 ± 0.300	48.000 ± 0.000
49	0.171 ± 0.002	0.962 ± 0.005	101.000 ± 0.000	49.000 ± 0.000
50	0.173 ± 0.003	0.960 ± 0.003	101.000 ± 0.000	50.000 ± 0.000
51	0.171 ± 0.004	0.963 ± 0.004	101.000 ± 0.000	51.000 ± 0.000
52	0.172 ± 0.003	0.959 ± 0.004	100.900 ± 0.300	52.000 ± 0.000
53	0.173 ± 0.005	0.962 ± 0.003	101.000 ± 0.000	53.000 ± 0.000
54	0.171 ± 0.002	0.962 ± 0.005	101.000 ± 0.000	54.000 ± 0.000
55	0.171 ± 0.004	0.959 ± 0.004	101.000 ± 0.000	55.000 ± 0.000
56	0.171 ± 0.004	0.961 ± 0.002	100.900 ± 0.300	56.000 ± 0.000
57	0.173 ± 0.002	0.959 ± 0.005	101.000 ± 0.000	57.000 ± 0.000
58	0.172 ± 0.003	0.958 ± 0.004	101.000 ± 0.000	58.000 ± 0.000
59	0.172 ± 0.004	0.962 ± 0.004	101.000 ± 0.000	59.000 ± 0.000
60	0.172 ± 0.003	0.958 ± 0.005	101.000 ± 0.000	60.000 ± 0.000
61	0.170 ± 0.002	0.961 ± 0.003	100.900 ± 0.300	61.000 ± 0.000
62	0.171 ± 0.003	0.962 ± 0.003	101.000 ± 0.000	62.000 ± 0.000
63	0.171 ± 0.002	0.961 ± 0.004	100.900 ± 0.300	63.000 ± 0.000
64	0.171 ± 0.004	0.963 ± 0.002	101.000 ± 0.000	64.000 ± 0.000
65	0.172 ± 0.003	0.961 ± 0.005	101.000 ± 0.000	65.000 ± 0.000
66	0.172 ± 0.003	0.961 ± 0.003	101.000 ± 0.000	66.000 ± 0.000
67	0.172 ± 0.003	0.963 ± 0.003	101.000 ± 0.000	67.000 ± 0.000
68	0.171 ± 0.003	0.962 ± 0.003	101.000 ± 0.000	68.000 ± 0.000
69	0.172 ± 0.003	0.960 ± 0.005	101.000 ± 0.000	69.000 ± 0.000
70	0.172 ± 0.003	0.964 ± 0.004	101.000 ± 0.000	70.000 ± 0.000
71	0.172 ± 0.002	0.960 ± 0.004	101.000 ± 0.000	71.000 ± 0.000
72	0.170 ± 0.003	0.961 ± 0.002	101.000 ± 0.000	72.000 ± 0.000
73	0.171 ± 0.003	0.962 ± 0.002	101.000 ± 0.000	73.000 ± 0.000
74	0.171 ± 0.002	0.961 ± 0.003	101.000 ± 0.000	74.000 ± 0.000
75	0.172 ± 0.003	0.961 ± 0.003	101.000 ± 0.000	75.000 ± 0.000
76	0.173 ± 0.003	0.961 ± 0.004	101.000 ± 0.000	76.000 ± 0.000
77	0.171 ± 0.002	0.960 ± 0.003	101.000 ± 0.000	77.000 ± 0.000
78	0.171 ± 0.003	0.962 ± 0.003	101.000 ± 0.000	78.000 ± 0.000
79	0.171 ± 0.002	0.960 ± 0.003	101.000 ± 0.000	79.000 ± 0.000
80	0.170 ± 0.002	0.963 ± 0.002	101.000 ± 0.000	80.000 ± 0.000
81	0.171 ± 0.002	0.961 ± 0.003	101.000 ± 0.000	81.000 ± 0.000
82	0.172 ± 0.003	0.961 ± 0.003	101.000 ± 0.000	82.000 ± 0.000
83	0.172 ± 0.003	0.961 ± 0.004	101.000 ± 0.000	83.000 ± 0.000
84	0.171 ± 0.003	0.960 ± 0.003	100.900 ± 0.300	84.000 ± 0.000
85	0.172 ± 0.003	0.961 ± 0.002	101.000 ± 0.000	85.000 ± 0.000
86	0.172 ± 0.004	0.961 ± 0.002	101.000 ± 0.000	86.000 ± 0.000
87	0.172 ± 0.002	0.960 ± 0.003	101.000 ± 0.000	87.000 ± 0.000
88	0.171 ± 0.001	0.963 ± 0.002	101.000 ± 0.000	88.000 ± 0.000
89	0.172 ± 0.002	0.961 ± 0.002	101.000 ± 0.000	89.000 ± 0.000
90	0.171 ± 0.003	0.961 ± 0.003	101.000 ± 0.000	90.000 ± 0.000
91	0.172 ± 0.002	0.962 ± 0.003	101.000 ± 0.000	91.000 ± 0.000

$\bar{M}$	$\bar{W}_2$-multi	$\bar{W}_2$-pc	M-multi	M-pc
92	0.171 ± 0.002	0.962 ± 0.003	101.000 ± 0.000	92.000 ± 0.000
93	0.171 ± 0.001	0.961 ± 0.003	101.000 ± 0.000	93.000 ± 0.000
94	0.172 ± 0.003	0.961 ± 0.003	101.000 ± 0.000	94.000 ± 0.000
95	0.173 ± 0.003	0.961 ± 0.002	101.000 ± 0.000	95.000 ± 0.000
96	0.172 ± 0.003	0.961 ± 0.004	101.000 ± 0.000	96.000 ± 0.000
97	0.171 ± 0.001	0.962 ± 0.002	101.000 ± 0.000	97.000 ± 0.000
98	0.172 ± 0.002	0.960 ± 0.004	101.000 ± 0.000	98.000 ± 0.000
99	0.172 ± 0.003	0.961 ± 0.002	101.000 ± 0.000	99.000 ± 0.000

Table A-4: Results of quantization of 2D GMMs with varying number of components $\bar{M}$. $\bar{W}_2$ is the normalized Wasserstein-2 distance and M is the number of support locations for each method. The methods are multi-grid (multi) and per component (pc). The total number of quantization locations is constraint to 100 for each method. For each chosen GMM size the experiment is run 10 times for different GMMs and the average values are recorded along with their standard-deviation.

$\bar{M}$	T-multi (s)	T-pc (s)
2	0.0122 ± 0.0079	0.0031 ± 0.0000
3	0.0148 ± 0.0009	0.0050 ± 0.0004
4	0.0189 ± 0.0007	0.0063 ± 0.0002
5	0.0236 ± 0.0006	0.0076 ± 0.0001
6	0.0297 ± 0.0028	0.0095 ± 0.0005
7	0.0342 ± 0.0006	0.0107 ± 0.0002
8	0.0406 ± 0.0022	0.0125 ± 0.0006
9	0.0498 ± 0.0059	0.0152 ± 0.0019
10	0.0578 ± 0.0095	0.0190 ± 0.0088
11	0.0751 ± 0.0164	0.0209 ± 0.0038
12	0.0816 ± 0.0104	0.0221 ± 0.0028
13	0.0756 ± 0.0018	0.0204 ± 0.0007
14	0.0848 ± 0.0090	0.0217 ± 0.0007
15	0.0903 ± 0.0014	0.0232 ± 0.0008
16	0.1033 ± 0.0111	0.0249 ± 0.0012
17	0.1377 ± 0.0281	0.0373 ± 0.0131
18	0.1265 ± 0.0257	0.0279 ± 0.0012
19	0.1321 ± 0.0045	0.0300 ± 0.0017
20	0.1503 ± 0.0218	0.0377 ± 0.0089
21	0.1530 ± 0.0167	0.0332 ± 0.0045
22	0.1605 ± 0.0056	0.0351 ± 0.0033
23	0.1706 ± 0.0069	0.0371 ± 0.0025
24	0.1951 ± 0.0377	0.0446 ± 0.0249
25	0.1956 ± 0.0101	0.0369 ± 0.0020
26	0.2012 ± 0.0051	0.0388 ± 0.0023
27	0.2145 ± 0.0056	0.0399 ± 0.0024

$\bar{M}$	T-multi (s)	T-pc (s)
28	0.2557 ± 0.0354	0.0484 ± 0.0164
29	0.2558 ± 0.0121	0.0474 ± 0.0076
30	0.2535 ± 0.0064	0.0436 ± 0.0007
31	0.2706 ± 0.0099	0.0459 ± 0.0007
32	0.2883 ± 0.0187	0.0469 ± 0.0009
33	0.2918 ± 0.0034	0.0487 ± 0.0009
34	0.3039 ± 0.0102	0.0492 ± 0.0006
35	0.3144 ± 0.0115	0.0517 ± 0.0031
36	0.3438 ± 0.0265	0.0542 ± 0.0041
37	0.3444 ± 0.0093	0.0565 ± 0.0040
38	0.3606 ± 0.0091	0.0554 ± 0.0011
39	0.3705 ± 0.0051	0.0563 ± 0.0010
40	0.3945 ± 0.0058	0.0582 ± 0.0013
41	0.4046 ± 0.0114	0.0595 ± 0.0016
42	0.4282 ± 0.0174	0.0609 ± 0.0027
43	0.4364 ± 0.0089	0.0623 ± 0.0014
44	0.4627 ± 0.0178	0.0640 ± 0.0025
45	0.4737 ± 0.0053	0.0657 ± 0.0018
46	0.4886 ± 0.0051	0.0661 ± 0.0004
47	0.5103 ± 0.0098	0.0678 ± 0.0012
48	0.5305 ± 0.0137	0.0687 ± 0.0004
49	0.5559 ± 0.0191	0.0707 ± 0.0011
50	0.5781 ± 0.0149	0.0730 ± 0.0022
51	0.5940 ± 0.0187	0.0736 ± 0.0018
52	0.6253 ± 0.0206	0.0748 ± 0.0011
53	0.6449 ± 0.0189	0.0754 ± 0.0013
54	0.6578 ± 0.0149	0.0765 ± 0.0009
55	0.6782 ± 0.0185	0.0791 ± 0.0016
56	0.7003 ± 0.0106	0.0802 ± 0.0023
57	0.7309 ± 0.0152	0.0809 ± 0.0006
58	0.7638 ± 0.0180	0.0833 ± 0.0022
59	0.7695 ± 0.0194	0.0838 ± 0.0008
60	0.7921 ± 0.0146	0.0854 ± 0.0010
61	0.8191 ± 0.0189	0.0864 ± 0.0007
62	0.8423 ± 0.0128	0.0887 ± 0.0011
63	0.9010 ± 0.0285	0.0901 ± 0.0030
64	0.9684 ± 0.0774	0.1207 ± 0.0660
65	0.9602 ± 0.0520	0.0931 ± 0.0021
66	0.9551 ± 0.0361	0.0961 ± 0.0053
67	0.9954 ± 0.0514	0.1004 ± 0.0129
68	1.0532 ± 0.1229	0.0985 ± 0.0021
69	1.1766 ± 0.1948	0.1053 ± 0.0093
70	1.0881 ± 0.0624	0.0999 ± 0.0008
71	1.0842 ± 0.0383	0.1035 ± 0.0035
72	1.1688 ± 0.0738	0.1099 ± 0.0120

$\bar{M}$	T-multi (s)	T-pc (s)
73	1.2672 ± 0.0680	0.1070 ± 0.0029
74	1.2239 ± 0.0520	0.1072 ± 0.0021
75	1.2327 ± 0.0477	0.1076 ± 0.0023
76	1.2502 ± 0.0415	0.1086 ± 0.0012
77	1.3159 ± 0.0655	0.1266 ± 0.0474
78	1.3454 ± 0.0656	0.1207 ± 0.0282
79	1.3574 ± 0.0483	0.1148 ± 0.0032
80	1.4722 ± 0.1050	0.1156 ± 0.0023
81	1.6065 ± 0.1839	0.1405 ± 0.0606
82	1.6306 ± 0.1923	0.1207 ± 0.0027
83	1.6355 ± 0.0854	0.1248 ± 0.0047
84	1.4985 ± 0.0513	0.1210 ± 0.0031
85	1.5492 ± 0.0633	0.1310 ± 0.0267
86	1.5933 ± 0.0545	0.1249 ± 0.0066
87	1.6612 ± 0.0957	0.1263 ± 0.0016
88	1.8259 ± 0.3095	0.1299 ± 0.0052
89	1.8072 ± 0.0985	0.1357 ± 0.0069
90	1.8522 ± 0.1279	0.1340 ± 0.0025
91	1.9945 ± 0.3600	0.1566 ± 0.0670
92	1.8406 ± 0.0281	0.1319 ± 0.0016
93	1.9077 ± 0.1465	0.1366 ± 0.0077
94	1.8975 ± 0.0528	0.1363 ± 0.0033
95	2.2178 ± 0.1667	0.1454 ± 0.0088
96	1.9593 ± 0.0901	0.1403 ± 0.0013
97	2.3015 ± 0.3617	0.1616 ± 0.0304
98	2.3184 ± 0.2112	0.1461 ± 0.0033
99	2.3593 ± 0.1159	0.1590 ± 0.0179

Table A-5: Results of quantization of 2D GMMs with varying number of components $\bar{M}$. M is the number of support locations for each method and T is the computation time. The methods are multi-grid (multi) and per component (pc). The total number of quantization locations is constraint to 100 for each method. For each chosen GMM size the experiment is run 10 times for different GMMs and the average values are recorded along with their standard-deviation.

A-1-3 GMM Dimension Tests

d	$\bar{W}_2$-multi	$\bar{W}_2$-pc	T-multi (s)	T-pc (s)
10	0.689 ± 0.006	0.879 ± 0.006	0.520 ± 0.063	0.108 ± 0.011
11	0.701 ± 0.013	0.885 ± 0.007	0.602 ± 0.120	0.123 ± 0.026
12	0.710 ± 0.018	0.889 ± 0.008	0.603 ± 0.112	0.127 ± 0.031
13	0.719 ± 0.022	0.893 ± 0.010	0.619 ± 0.105	0.131 ± 0.028

d	$\bar{W}_2$-multi	$\bar{W}_2$-pc	T-multi (s)	T-pc (s)
14	0.728 ± 0.026	0.897 ± 0.012	0.627 ± 0.095	0.135 ± 0.029
15	0.736 ± 0.030	0.900 ± 0.013	0.636 ± 0.090	0.138 ± 0.028
16	0.743 ± 0.033	0.903 ± 0.015	0.631 ± 0.094	0.142 ± 0.027
17	0.750 ± 0.036	0.906 ± 0.016	0.627 ± 0.088	0.146 ± 0.028
18	0.757 ± 0.039	0.909 ± 0.017	0.626 ± 0.084	0.150 ± 0.029
19	0.763 ± 0.042	0.911 ± 0.018	0.624 ± 0.080	0.154 ± 0.030
20	0.769 ± 0.044	0.914 ± 0.018	0.631 ± 0.084	0.159 ± 0.033
21	0.774 ± 0.046	0.916 ± 0.019	0.638 ± 0.083	0.163 ± 0.036
22	0.780 ± 0.048	0.918 ± 0.020	0.645 ± 0.083	0.167 ± 0.037
23	0.785 ± 0.049	0.920 ± 0.020	0.654 ± 0.088	0.171 ± 0.039
24	0.789 ± 0.051	0.922 ± 0.021	0.664 ± 0.094	0.175 ± 0.040
25	0.794 ± 0.052	0.924 ± 0.021	0.676 ± 0.102	0.179 ± 0.043
26	0.798 ± 0.053	0.925 ± 0.021	0.690 ± 0.115	0.185 ± 0.048
27	0.802 ± 0.054	0.927 ± 0.022	0.702 ± 0.123	0.189 ± 0.049
28	0.806 ± 0.055	0.928 ± 0.022	0.725 ± 0.161	0.199 ± 0.073
29	0.810 ± 0.056	0.929 ± 0.022	0.748 ± 0.187	0.205 ± 0.077
30	0.813 ± 0.057	0.931 ± 0.022	0.758 ± 0.188	0.209 ± 0.077
31	0.816 ± 0.058	0.932 ± 0.023	0.769 ± 0.190	0.213 ± 0.077
32	0.819 ± 0.058	0.933 ± 0.023	0.779 ± 0.192	0.217 ± 0.077
33	0.823 ± 0.059	0.934 ± 0.023	0.789 ± 0.194	0.221 ± 0.079
34	0.825 ± 0.059	0.935 ± 0.023	0.800 ± 0.197	0.226 ± 0.080
35	0.828 ± 0.060	0.936 ± 0.023	0.812 ± 0.203	0.230 ± 0.082
36	0.831 ± 0.060	0.937 ± 0.023	0.823 ± 0.207	0.235 ± 0.085
37	0.834 ± 0.061	0.938 ± 0.024	0.834 ± 0.211	0.240 ± 0.087
38	0.836 ± 0.061	0.939 ± 0.024	0.845 ± 0.215	0.244 ± 0.088
39	0.838 ± 0.062	0.940 ± 0.024	0.856 ± 0.220	0.249 ± 0.090
40	0.841 ± 0.062	0.941 ± 0.024	0.867 ± 0.225	0.253 ± 0.092
41	0.843 ± 0.062	0.942 ± 0.024	0.878 ± 0.230	0.257 ± 0.093
42	0.845 ± 0.063	0.943 ± 0.024	0.889 ± 0.235	0.262 ± 0.095
43	0.847 ± 0.063	0.943 ± 0.024	0.900 ± 0.241	0.266 ± 0.098
44	0.849 ± 0.063	0.944 ± 0.024	0.912 ± 0.246	0.271 ± 0.100
45	0.851 ± 0.063	0.945 ± 0.024	0.924 ± 0.254	0.275 ± 0.102
46	0.853 ± 0.063	0.946 ± 0.024	0.936 ± 0.260	0.280 ± 0.104
47	0.855 ± 0.064	0.946 ± 0.024	0.948 ± 0.267	0.284 ± 0.106
48	0.857 ± 0.064	0.947 ± 0.024	0.960 ± 0.274	0.289 ± 0.108
49	0.859 ± 0.064	0.948 ± 0.024	0.979 ± 0.296	0.295 ± 0.113
50	0.860 ± 0.064	0.948 ± 0.024	1.000 ± 0.326	0.301 ± 0.120
51	0.862 ± 0.064	0.949 ± 0.024	1.013 ± 0.333	0.306 ± 0.122
52	0.864 ± 0.064	0.950 ± 0.024	1.026 ± 0.339	0.310 ± 0.124
53	0.865 ± 0.064	0.950 ± 0.024	1.038 ± 0.345	0.315 ± 0.126
54	0.867 ± 0.064	0.951 ± 0.025	1.051 ± 0.351	0.319 ± 0.128
55	0.868 ± 0.064	0.951 ± 0.025	1.064 ± 0.358	0.324 ± 0.130
56	0.870 ± 0.064	0.952 ± 0.025	1.076 ± 0.364	0.328 ± 0.133
57	0.871 ± 0.065	0.952 ± 0.025	1.088 ± 0.370	0.333 ± 0.135
58	0.872 ± 0.065	0.953 ± 0.025	1.102 ± 0.378	0.337 ± 0.137

d	$\bar{W}_2$-multi	$\bar{W}_2$-pc	T-multi (s)	T-pc (s)
59	0.874 ± 0.065	0.953 ± 0.025	1.115 ± 0.385	0.341 ± 0.139
60	0.875 ± 0.065	0.954 ± 0.025	1.127 ± 0.391	0.346 ± 0.141

Table A-7: Quantization results of GMMs over different dimensions d varying from 10 to 60. The table shows the mean and standard deviation for the $\bar{W}_2$ distances and runtimes T (in seconds) for multi-grid (multi) and per-component (pc) methods. The total number of quantization locations is constraint to 100 for each method (for 20 components this results in 97 locations for the multi-grid method and 80 for the per-component method). For each chosen dimension the experiment is run 10 times for different GMMs and the average values are recorded along with their standard-deviation.

A-1-4 GMM's Variance Tests

Var.	$\bar{W}_2$-multi	$\bar{W}_2$-pc	T-multi (s)	T-pc (s)
0.1	0.345332	0.371729	0.3127	0.0777
0.2	0.438195	0.484801	0.2728	0.0751
0.3	0.497161	0.553788	0.2720	0.0756
0.4	0.536809	0.601526	0.2707	0.0769
0.5	0.566485	0.636866	0.2772	0.0763
0.6	0.589815	0.664208	0.2723	0.0790
0.7	0.607942	0.686045	0.2710	0.0766
0.8	0.623268	0.703914	0.2699	0.0745
0.9	0.635924	0.718821	0.3394	0.0813
1.0	0.646751	0.731453	0.2880	0.0828
1.1	0.656266	0.742299	0.2905	0.0813
1.2	0.666016	0.751715	0.2857	0.0786
1.3	0.670989	0.759969	0.2834	0.0791
1.4	0.677216	0.767264	0.2769	0.0764
1.5	0.683084	0.773760	0.2784	0.0768
1.6	0.687963	0.779581	0.2720	0.0810
1.7	0.694172	0.784828	0.2715	0.0734
1.8	0.697145	0.789582	0.2687	0.0758
1.9	0.700648	0.793910	0.2693	0.0759
2.0	0.703883	0.797866	0.2702	0.0732
2.1	0.707658	0.801497	0.2660	0.0757
2.2	0.710006	0.804841	0.2687	0.0739
2.3	0.713064	0.807931	0.2677	0.0757
2.4	0.715708	0.810795	0.2704	0.0778
2.5	0.717784	0.813458	0.3033	0.0770
2.6	0.719201	0.815938	0.3583	0.0988
2.7	0.720431	0.818256	0.2910	0.0775

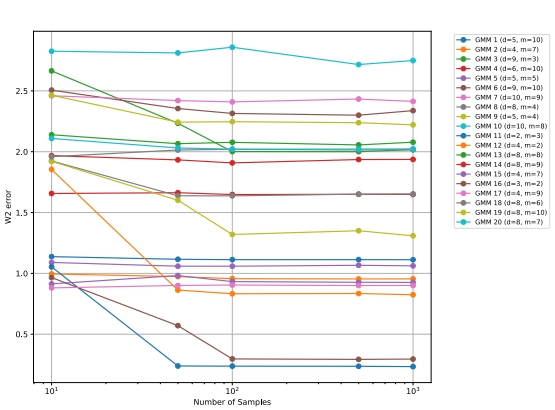
Var.	$\bar{W}_2$-multi	$\bar{W}_2$-pc	T-multi (s)	T-pc (s)
2.8	0.722776	0.820425	0.2743	0.0741
2.9	0.724237	0.822461	0.2695	0.0755
3.0	0.726357	0.824375	0.2710	0.0733
3.1	0.728056	0.826177	0.2768	0.0767
3.2	0.730070	0.827877	0.2699	0.0758
3.3	0.732029	0.829484	0.2702	0.0744
3.4	0.731506	0.831006	0.2713	0.0746
3.5	0.733225	0.832447	0.2951	0.0922
3.6	0.735336	0.833816	0.2810	0.0769
3.7	0.736119	0.835117	0.2681	0.0756
3.8	0.736367	0.836355	0.2695	0.0793
3.9	0.737403	0.837535	0.2830	0.0748
4.0	0.739085	0.838660	0.2758	0.0742
4.1	0.740282	0.839735	0.2741	0.0776
4.2	0.740257	0.840762	0.2780	0.0765
4.3	0.742761	0.841745	0.2815	0.0787
4.4	0.742319	0.842687	0.2918	0.0824
4.5	0.743642	0.843589	0.2826	0.0775
4.6	0.743471	0.844455	0.2737	0.0737
4.7	0.744435	0.845287	0.2686	0.0768
4.8	0.744913	0.846087	0.2701	0.0742
4.9	0.745442	0.846856	0.2699	0.0749
5.0	0.747536	0.847596	0.2688	0.0760
5.1	0.746833	0.848309	0.2715	0.0750
5.2	0.746975	0.848996	0.3369	0.0872
5.3	0.748328	0.849659	0.2816	0.0752
5.4	0.749745	0.850299	0.2714	0.0771
5.5	0.749445	0.850917	0.3018	0.0776
5.6	0.749715	0.851514	0.2765	0.0737
5.7	0.750556	0.852092	0.4699	0.0833
5.8	0.751368	0.852650	0.3074	0.1432
5.9	0.750433	0.853191	0.2925	0.0804
6.0	0.752715	0.853715	0.2809	0.0852
6.1	0.751875	0.854222	0.2897	0.0777
6.2	0.752604	0.854714	0.2727	0.0741
6.3	0.752545	0.855191	0.2747	0.0760
6.4	0.753442	0.855654	0.2700	0.0765
6.5	0.753125	0.856103	0.2707	0.0732
6.6	0.753414	0.856540	0.2674	0.0764
6.7	0.754923	0.856964	0.2689	0.0747
6.8	0.753983	0.857376	0.2717	0.0793
6.9	0.755062	0.857777	0.2738	0.0779
7.0	0.754878	0.858167	0.2859	0.0825
7.1	0.755536	0.858546	0.2823	0.0788

Var.	$\bar{W}_2$-multi	$\bar{W}_2$-pc	T-multi (s)	T-pc (s)
7.2	0.756188	0.858916	0.2715	0.1508
7.3	0.756449	0.859275	0.3150	0.0776
7.4	0.756149	0.859626	0.2771	0.0814
7.5	0.757642	0.859967	0.2836	0.0747
7.6	0.757353	0.860300	0.2890	0.0867
7.7	0.757755	0.860625	0.2821	0.0760
7.8	0.757232	0.860942	0.2753	0.0749
7.9	0.756800	0.861251	0.2699	0.0868
8.0	0.758343	0.861552	0.3985	0.0783
8.1	0.759212	0.861847	0.2919	0.0988
8.2	0.758164	0.862134	0.4566	0.0795
8.3	0.759341	0.862415	0.2743	0.0805
8.4	0.759605	0.862690	0.2741	0.0748
8.5	0.759801	0.862958	0.2720	0.0764
8.6	0.760451	0.863220	0.2715	0.0773
8.7	0.759097	0.863477	0.2743	0.0745
8.8	0.760642	0.863728	0.2726	0.0756
8.9	0.761210	0.863973	0.2710	0.0762
9.0	0.760941	0.864213	0.2704	0.0745
9.1	0.761142	0.864448	0.2735	0.0755
9.2	0.762033	0.864679	0.2707	0.0769
9.3	0.760956	0.864904	0.2712	0.0742
9.4	0.762716	0.865125	0.2740	0.0766
9.5	0.761270	0.865341	0.3562	0.0762
9.6	0.763420	0.865553	0.2731	0.0746
9.7	0.761295	0.865761	0.2750	0.0776
9.8	0.762802	0.865964	0.2760	0.0767
9.9	0.763543	0.866164	0.2719	0.0766
10.0	0.761849	0.866360	0.2711	0.0770

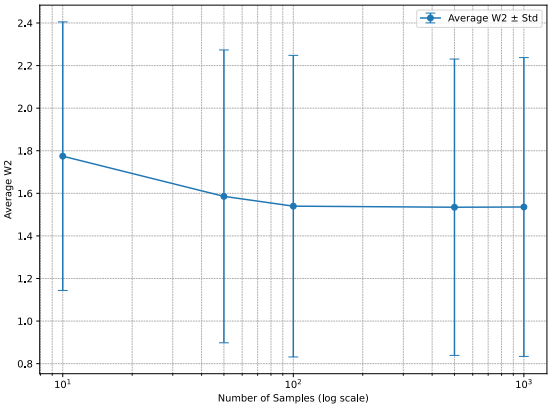
Table A-9: Results of quantization of a 15D-GMM with 10 components, where the variance is scaled for 100 runs, such that it increases by $0.1 + \texttt{Iterations} * 0.1$ in each dimension. $\bar{W}_2$ is the normalized Wasserstein-2 distance and T is the computation time for each method. The methods are multi-grid (multi) and per component (pc). The total number of quantization locations is constraint to 100 for each method.

A-2 Figures

Manually tuning the number of samples for Algorithm 2 based on the Wasserstein distance of the final quantization.



(a) Wasserstein distance across 20 GMMs as a function of sample count.



(b) Average Wasserstein distance vs. samples per component during DBSCAN.

Figure A-1: Influence of the number of samples used in Algorithm 2 on the total Wasserstein distance of the quantization. (a) Per-GMM breakdown over 20 instances. (b) Mean Wasserstein distance across GMMs as a function of samples per component.

Bibliography

[1] Tiba Zaki Abdulhameed, Suhad A Yousif, Venus W Samawi, and Hasnaa I Al-Shaikhli. Ss-dbscan: Semi-supervised density-based spatial clustering of applications with noise for meaningful clustering in diverse density data. *IEEE Access*, 2024.

[2] Steven Adams, Patanè, Morteza Lahijanian, and Luca Laurenti. Finite neural networks as mixtures of gaussian processes: From provable error bounds to prior selection. *cs.LG*, 7 2024.

[3] Luca Ambrogioni, Umut Guclu, and Marcel van Gerven. Wasserstein variational gradient descent: From semi-discrete optimal transport to ensemble variational inference. *stat.ML*, 11 2018.

[4] Devin Balkcom, Andrei Furtuna, and Weifu Wang. The dubins car and other arm-like mobile robots. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 380–386. IEEE, 2018.

[5] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509517, September 1975.

[6] Miguel A Carreira-Perpinán and Christopher KI Williams. On the number of modes of a gaussian mixture. In *International Conference on Scale-Space Theories in Computer Vision*, pages 625–640. Springer, 2003.

[7] Satish Chander and P. Vijaya. *Unsupervised learning methods for data clustering.* Academic Press, 1 2021.

[8] Julie Delon and Agnes Desolneux. A Wasserstein-Type distance in the space of Gaussian mixture models. *SIAM Journal on Imaging Sciences*, 13(2):936–970, 1 2020.

[9] Xiwang Dong, Bocheng Yu, Zongying Shi, and Yisheng Zhong. Time-varying formation control for unmanned aerial vehicles: Theories and applications. *IEEE Transactions on Control Systems Technology*, 23(1):340–348, 2015.

[10] Martin Ester, Hans-Peter Kriegel, Jiirg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. *AAAI*, 1996.

[11] Absalom E. Ezugwu, Abiodun M. Ikotun, Olaide O. Oyelade, Laith Abualigah, Jeffery O. Agushaka, Christopher I. Eke, and Andronicus A. Akinyelu. A comprehensive survey of clustering algorithms: State-of-the-art machine learning applications, taxonomy, challenges, and future research prospects. *Engineering Applications of Artificial Intelligence*, 110:104743, 2022.

[12] Eduardo Figueiredo, Steven Adams, Peyman Mohajerin Esfahani, and Luca Laurenti. Efficient uncertainty propagation with guarantees in wasserstein distance. *arXiv preprint arXiv:2506.08689*, 2025.

[13] Eduardo Figueiredo, Andrea Patane, Morteza Lahijanian, and Luca Laurenti. Uncertainty propagation in stochastic systems via mixture models with error quantification. *eess.SY*, 3 2024.

[14] Rémi Flamary, Nicolas Courty, Alexandre Gramfort, Mokhtar Z. Alaya, Aurélie Boisbunon, Stanislas Chambon, Laetitia Chapel, Adrien Corenflos, Kilian Fatras, Nemo Fournier, Léo Gautheron, Nathalie T.H. Gayraud, Hicham Janati, Alain Rakotomamonjy, Ievgen Redko, Antoine Rolet, Antony Schutz, Vivien Seguy, Danica J. Sutherland, Romain Tavenard, Alexander Tong, and Titouan Vayer. Pot: Python optimal transport. *Journal of Machine Learning Research*, 22(78):1–8, 2021.

[15] Siegfried Graf and Harald Luschgy. *Foundations of quantization for probability distributions*. Springer Science & Business Media, 2000.

[16] Daniel Landgraf, Andreas Völz, Felix Berkel, Kevin Schmidt, Thomas Specker, and Knut Graichen. Probabilistic prediction methods for nonlinear systems with application to stochastic model predictive control. *Annual Reviews in Control*, 56:100905, 2023.

[17] Jesse Levinson, Jake Askeland, Jan Becker, Jennifer Dolson, David Held, Soeren Kammel, J. Zico Kolter, Dirk Langer, Oliver Pink, Vaughan Pratt, Michael Sokolsky, Ganymed Stanek, David Stavens, Alex Teichman, Moritz Werling, and Sebastian Thrun. Towards fully autonomous driving: Systems and algorithms. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 163–168, 2011.

[18] Owen Lockwood and Mei Si. A review of uncertainty for deep reinforcement learning, 2022.

[19] Geoffrey McLachlan and David Peel. *Finite mixture models*. Annual Reviews, 10 2000.

[20] Ali Mesbah. Stochastic model predictive control: An overview and perspectives for future research. *IEEE Control Systems Magazine*, 36(6):30–44, 2016.

[21] Gloriana Monko and Masaomi Kimura. Enhanced ss-dbscan clustering algorithm for high-dimensional data. *Data Science*, 2024.

[22] Todd K Moon. The expectation-maximization algorithm. *IEEE Signal processing magazine*, 13(6):47–60, 1996.

[23] Steven J. Nowlan and Geoffrey E. Hinton. Simplifying neural networks by soft weight-sharing. *Neural Computation*, 4:473–493, 1992.

[24] Ton van den Boom and Bart De Schutter.

[25] Tejas Pawar. Gaussian Mixture Models explained: Applying GMM and EM for effective data clustering. 11 2024.

[26] Gabriel Peyre and Marco Cuturi. *Computational Optimal transport*. Foundations and Trends(r) in M, 2 2019.

[27] Nadia Rahmah and Imas Sukaesih Sitanggang. Determination of optimal epsilon (eps) value on dbscan algorithm to clustering data on peatland hotspots in sumatra. In *IOP conference series: earth and environmental science*, volume 31, page 012012. IoP Publishing, 2016.

[28] Sachinsoni. Clustering Like a Pro: A Beginners Guide to DBSCAN. 12 2023.

[29] Terence Tao. *An introduction to measure theory*, volume 126. American Mathematical Soc., 2011.

[30] Gabriel Terejanu, Puneet Singla, Tarunraj Singh, and Peter D Scott. Uncertainty propagation for nonlinear dynamic systems using gaussian mixture models. *Journal of guidance, control, and dynamics*, 31(6):1623–1633, 2008.

[31] Roman Vershynin. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press, 2018.

[32] Cédric Villani et al. *Optimal transport: old and new*, volume 338. Springer, 2009.

[33] Florian Weissel, Marco F Huber, and Uwe D Hanebeck. Stochastic nonlinear model predictive control based on gaussian mixture approximations. In *Informatics in Control, Automation and Robotics: Selected Papers from the International Conference on Informatics in Control, Automation and Robotics 2007*, pages 239–252. Springer, 2009.

[34] D. Xu and Y. Tian. A comprehensive survey of clustering algorithms. *Annals of Data Science*, 2(2):165–193, 2015.

[35] Rui Xu and D. Wunsch. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, 2005.

[36] Hui Yin, Amir Aryani, Stephen Petrie, Aishwarya Nambissan, Aland Astudillo, and Shengyuan Cao. A rapid review of clustering algorithms. *arXiv preprint arXiv:2401.07389*, 2024.

[37] Guoshen Yu, Guillermo Sapiro, and Stéphane Mallat. Solving inverse problems with piecewise linear estimators: From gaussian mixture models to structured sparsity. *IEEE Transactions on Image Processing*, 21(5):2481–2499, 2012.

[38] Chidong Zhang, Brian E. Mapes, and Brian J. Soden. Bimodality in tropical water vapour. *Quarterly Journal of the Royal Meteorological Society*, 129, 2003.

[39] Marco Zio, Ugo Guarnera, and Roberto Rocci. A mixture of mixture models for a classification problem: The unity measure error. *Computational Statistics & Data Analysis*, 51:2573–2585, 02 2007.