Semester Project at Idiap
Fall 2024

Initialization methods to an ergodic control problem for trajectory optimization

---

Author:

Evangelisti Chiara, 368672

Professor:

Sylvain Calinon

Supervisor:

Guillaume Clivaz

Contents

1 Introduction to ergodic control

Ergodic theory is concerned with the relation between the time-averaged and space-averaged behaviors of a dynamical system. When applied to robotics ergodic control differs from conventional tracking problems since it provides a probability distribution to the robot instead of a single target to reach, with the aim that the latter will cover it efficiently enabling exploration. This results in the robot moving in the environment focusing on different areas proportionally to their density in the underlying spatial distribution. As a matter of fact, in this context, ergodicity refers to the deviation of the time-averaged statistics of the agent's trajectory and the target distribution it aims to explore. Thus minimizing this difference natural exploration trajectories are obtained by the controller [2].
The original formulation of ergodic control, which is also used in this work is the spectral multiscale coverage (SMC) objective, which can be solved through iLQR: a general overview of the two is presented in the following.

1.1 Spectral-based ergodic control (SMC)

In Spectral-based ergodic control (SMC) the tracking problem is translated into the frequency domain: matching the frequency components is exploited to generate the desired trajectory from the target distribution. The primary objective is to minimize the difference between the spectral representation of the target distribution and the robot's trajectory, which is quantified by their Fourier coefficients. These coefficients are weighted to focus on low-frequency components initially, allowing a crude exploration, and progressively refine higher frequencies for detailed coverage, and a cost function compares the two decompositions [2].
The spatial distribution, $g(x)$, is approximated as a weighted decomposition of basis functions, chosen them appropriately it becomes its Fourier Series:

$$g(x) = \sum_{k=-K+1}^{K-1} w_k \phi_k(x),$$

where $\phi_k(x)$ are Fourier basis functions, and w_k are the corresponding coefficients, which are derived from the reconstructed distribution as:

$$\hat{w}_k = \int_{x \in \mathcal{X}} \hat{g}(x) \phi_k(x) \, \mathrm{d}x.$$

The aim is to match these coefficients with those derived from the robot's trajectory:

$$w_k = \frac{1}{T} \int_0^T \phi_k(x(t)) \, dt.$$

A cost function is then defined as a weighted sum of differences between these coefficients:

$$\epsilon = \frac{1}{2} \sum_{k=0}^{K-1} \Lambda_k \left(w_k - \hat{w}_k \right)^2$$

$$\Lambda_k = \left(1 + k^2 \right)^{-1}$$

where Λ_k forces to focus on minimizing the low frequency error first and progressively higher frequency components only when the first approaches 0.
Finally the controller calculates the optimal control command $u(t)$ at each time step by solving a constrained optimization problem:

$$\hat{u}(t) = \arg\min_{u(t)} \epsilon \left(x(t + \Delta t) \right), \quad \text{s.t.} \quad \dot{x}(t) = f\left(x(t), u(t) \right), \quad \|u(t)\| \leq u_{\max}.$$

1.2 Planning formulation of SMC with iLQR

Integrating Spectral-based Ergodic Control (SMC) with iterative Linear Quadratic Regulation (iLQR) allows to combine optimal trajectory planning while guaranteeing alignment with the target spatial distribution. iLQR is a control optimization method that simultaneously account for trajectory tracking and control effort by minimizing a quadratic cost function. It iteratively computes a control updates to optimize the trajectory of a system.
In this approach, the Fourier coefficients w_k of the robot's trajectory and their derivatives are incorporated into the optimization process. The cost function for iLQR is defined as:

$$c(x, u) = \frac{1}{2}\|w - \hat{w}\|_{\Lambda}^2 + u^\top R u,$$

where $\hat{w}$ and Λ follow the previous section notation and R penalizes excessive control inputs to ensure smooth movements. iLQR solves this problem iteratively by linearizing the system dynamics and approximating the cost function with a second-order Taylor expansion. The control command updates u are calculated using the following rule:

$$\Delta u = \left(S_u^\top J(x)^\top Q J(x) S_u + R\right)^{-1} \left(-S_u^\top J(x)^\top Q f(x) - Ru\right),$$

Here, S_u represents the state transition matrix that maps control inputs to changes in state. $J(x)$ is the Jacobian matrix of the system's dynamics with respect to the state. Q and R are the weight matrices for the state and control costs, respectively, and $f(x)$ represents the current system dynamics. The term $-S_u^\top J(x)^\top Q f(x)$ captures the gradient of the cost function with respect to the state, while $-Ru$ penalizes excessive control efforts to ensure smooth movements. The inverse term $(S_u^\top J(x)^\top Q J(x) S_u + R)^{-1}$ combines the state and control weights to compute the optimal control updates Δu.
Additional components can be added to the cost function and the gradient update to drive the trajectory to respect further constraints or attain desired result, more details concerning the one adopted in this work will follow in the next sections.
By integrating SMC with iLQR, the controller achieves trajectories that cover the target distribution efficiently. This combination is particularly useful for applications in exploration, active sensing, and coverage tasks in multi-dimensional spaces [2].

In the following various initialization techniques have been investigated for trajectory planning problems in both single-agent and multi-agent settings in a 2-dimensional case. The code concerning the formulation of the ergodic control problem is adapted from [1], and can be foud at the following repo: Ergodic control manipulation, Chiara Evangelisti.

2 Single-agent problem

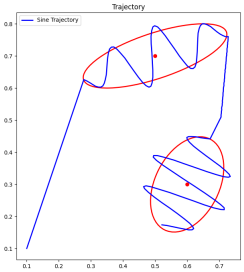
In iLQR (Iterative Linear Quadratic Regulator), the process begins with an initial guess for the control trajectory, which is a sequence of actions or control inputs over time. This initial trajectory serves as a starting point for the optimization process and does not need to be optimal, however it should be a reasonable approximation of the desired behavior. iLQR iteratively refines this initial trajectory to optimize a cost function that balances trajectory tracking and control effort. Therefore in this setting the initialization step involves setting up this initial trajectory based on system dynamics and task requirements. Through iterative optimization, iLQR adjusts the trajectory to better align with the desired objectives, ergodicity in this case.
Throughout this work the underlying distribution is considered to be a Gaussian Mixture Model (GMM), therefore most initialization methods proposed are tailored to it.

2.1 Methods

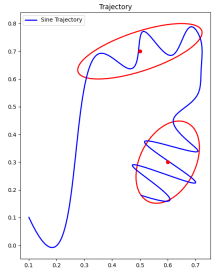
In this single-agent setting the aim was to implement and test different initialization methods to analyze their effect on the resulting trajectory after optimization. Namely two patterned trajectories are chosen to verify, in addition to their overall performance two main questions. The first concerns whether a more structured initialization would allow a less strict cost on the control input, whereas the second aims at observing if and under which considerations the initial trajectory pattern is reflected into the final one. The adopted patterns are a modulated sine trajectory and a spiral trajectory, where each pattern is repeated and tailored to each Gaussian in the distribution.
In the considered setting the system is a simple integrator $u(t) = \dot{x}(t)$ therefore the initial input to the iLQR optimization is obtained simply by differentiating the initial trajectory created.

2.1.1 Sine pattern trajectory

The trajectory is composed of various segments: straight segments are used to connect areas corresponding to different GMM components, and for each of them a modulated sine wave covering the distribution (the ellipse given by the isoline at equivalent probabilities) is created. When a smoother trajectory is required (see *Final results*) a B-spline parametrization can be exploited, with a smoothing factor to control how closely the updated trajectory fits the original one. The two resulting trajectories can be observed in Fig. 1:



(a) Sine pattern example



(b) Smooth sine pattern example, smoothing factor = 0.07

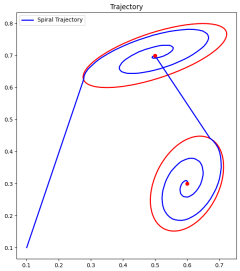
Figure 1: Example of modulated sine pattern trajectory for single-agent initialization

To create such a trajectory first the overall number of points available is subdivided to each of the required segments. Then for each Gaussian the sine wave is obtained as a function of the results of eigendecomposition: the eigenvalues give the length of the ellipse axes , while the eigenvectors give the required orientation which is used to rotate the sine wave. The wave has a

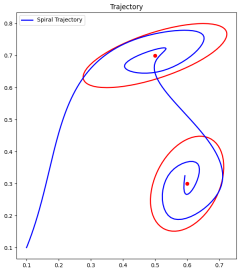
varying amplitude, made of a fixed part (function of the eigenvalue corresponding to the minor axis) and a component following a sine wave such that the resulting amplitude is higher at the center and smaller further along the major axis, to efficiently cover the elliptical shape of the distribution. After rotation to match the Gaussian orientation the wave is also translated by the center of the distribution, then unless it is the last Gaussian a linear segment is added to connect it to the next one.

2.1.2 Spiral pattern trajectory

The same principles exploited in the previous section adopted to define the spiral-based trajectory in this section, whose results can be observed in Fig. 2



(a) Spiral pattern example



(b) Smooth spiral pattern example, smoothing factor = 0.03

Figure 2: Example of spiral pattern trajectory for single-agent initialization

Similar to before different segments are constructed to make up the different parts of the path. Then for each Gaussian the spiral is determined as a combination of a sine and a cosine wave which provide the circular patterns, and progressively reducing the radius of curvature and the angle to finalize the ingoing spiral winding. The sign of the x component of the trajectory can be inverted to have the spiral enveloping in the opposite direction. Eigendecomposition is again exploited to scale and orient the spirals: the eigenvectors matrix is scaled by the eigenvalue magnitude before application to the trajectory to ensure both spatial orientation and adaptation to the distribution shape. As previously the wave is also translated by the center of the distribution and then if it is not the last Gaussian a linear segment is added to connect it to the next one.

2.1.3 Cost components

The iLQR algorithm optimizes a trajectory by minimizing a cost function that combines multiple objectives. Each term in the cost function represents a specific constraint or objective relevant to the task and is weighted to reflect its relative importance. In the considered setting, the cost function includes the following components:

Ergodicity Cost: as mentioned in Section 1, the ergodicity cost ensures that the trajectory covers the target spatial distribution as uniformly as possible. This is achieved by minimizing the difference between the Fourier coefficients of the desired distribution ($\hat{w}$) and those of the trajectory (w), with Q being the precision matrix:

$$\epsilon_{\text{erg}} = (w - \hat{w})^\top Q (w - \hat{w}),$$

Control Effort Cost

The control effort cost penalizes excessive control inputs to avoid large, abrupt changes in velocity. It is defined as:

$$\epsilon_{\text{control}} = u^\top R u,$$

where: u represents the control inputs and R is a diagonal matrix that weights the importance of minimizing control effort. This term ensures efficient motion, avoiding unnecessary input expenditure.

Boundary and Domain Constraints

To ensure the trajectory remains within valid spatial boundaries, a domain cost is implemented using a bounded constraint function. Specifically, the function $f^{\text{cut}}(x)$ is defined to enforce boundary constraints by setting residuals to zero within the bounds and penalizing deviations outside. The bounded domain is determined by $\text{sz} = \frac{\text{param.xlim[1]}}{2}$, where $f^{\text{cut}}(x)$ is given as:

$$f_i^{\text{cut}}(x) = \begin{cases} x_i - \text{sz}, & \text{if } x_i > \text{sz}, \\ x_i + \text{sz}, & \text{if } x_i < -\text{sz}, \\ 0, & \text{if } -\text{sz} \leq x_i \leq \text{sz}. \end{cases}$$

The Jacobian of $f^{\text{cut}}(x)$, $J^{\text{cut}}(x)$, is also used to ensure proper gradient computation during optimization:

$$J_i^{\text{cut}}(x) = \begin{cases} 1, & \text{if } x_i > \text{sz}, \\ 1, & \text{if } x_i < -\text{sz}, \\ 0, & \text{if } -\text{sz} \leq x_i \leq \text{sz}. \end{cases}$$

This approach ensures that the cost function penalizes states that leave the allowed domain while maintaining smooth behavior within the boundaries.

2.2 Early testing

In order to verify the performance of the aforementioned initialization methods some tests were carried out. For all the results described in this section, if not otherwise specified the parameters used are listed in Table 6.

Table 1: Parameters Used in iLQR Optimization

Parameter	Value
`param.nbData`	200/500 (Number of datapoints)
`param.nbVarPos`	2 (Position space dimension)
`param.nbDeriv`	1 (Number of static and dynamic features)
`param.nbVarX`	`param.nbVarPos` × `param.nbDeriv` (State space dimension)
`param.nbFct`	8 (Number of Fourier basis functions)
`param.nbStates`	2 (Number of Gaussians for spatial distribution)
`param.nbIter`	50 (Maximum iLQR iterations)
`param.nbPoints`	1 (Number of viapoints to reach)
`param.dt`	1×10^{-2} (Time step length)
`param.qd`	1×10^{0} (Bounded domain weight term)
`param.qr`	0×10^{4} (Reach target weight term)
`param.r`	1×10^{-8} (Control weight term)
`param.Mu_reach`	$[0.3, 0.9, 0, \ldots, 0]^T$ (Target mean position)
`param.xlim`	$[0, 1]$ (Domain limit)

First the resulting final trajectories, the cost trend and the Fourier coefficients reconstruction can be observed in Fig 3. Then the analysis focused on three main features: the cost comparison between different techniques, the impact of the cost on the input and the effect of the number of Fourier basis functions.

2.2.1 Cost comparison

To verify their usefulness and their efficiency the considered patterned initialization were compared in terms of iLQR cost to other available approaches:

- **zero-input**: the input is initialized to 0 for all trajectories data-points. This means that the velocity is null at all times resulting in a degenerate initial trajectory where the agent simply does not move from the starting point.
- **random-input**: the input is initialized randomly within a bounded range, the resulting trajectory oscillates a lot and does not cover the space in a uniform or efficient way.

- **myopic ergodic control**: at each step of the trajectory the control command is computed to move the agent in a direction which reduced the difference between the current and the desired Fourier coefficients. This is a local update based on its current position and local error. The resulting trajectory already covers the space in an exploratory way.

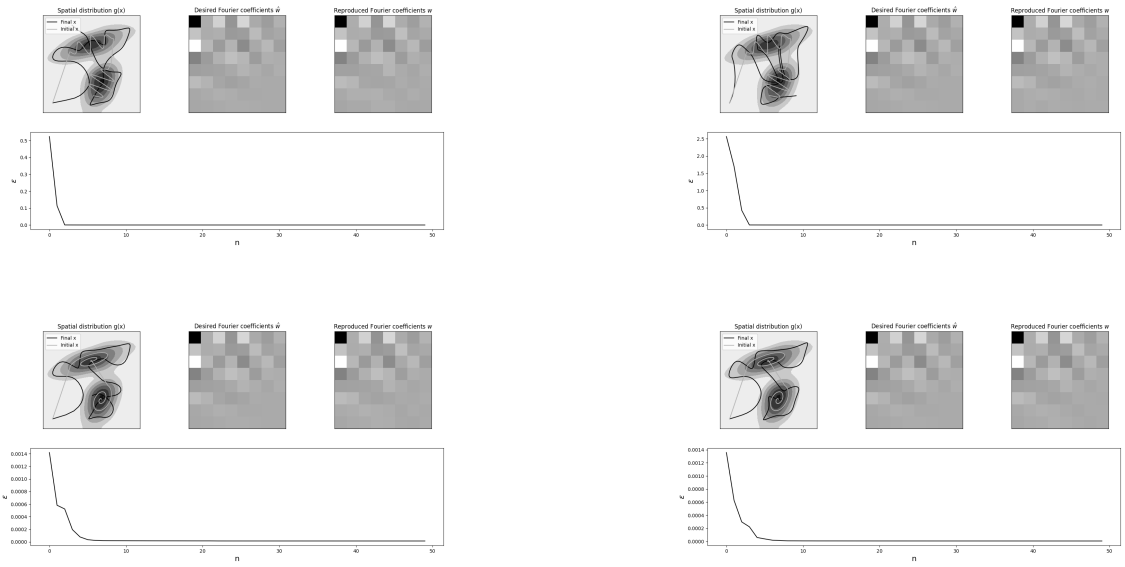


Figure 3: Initial and resulting trajectory, desired and reconstructed Fourier Coefficients and resulting cost for sine and spiral initialization, with 200 (left) and 500 (right) data-points with parameters in Table 1

The comparison is carried out for two trajectories of different length, namely 200 and 500 data-points and the cost trends can be visualized in Fig. 4.

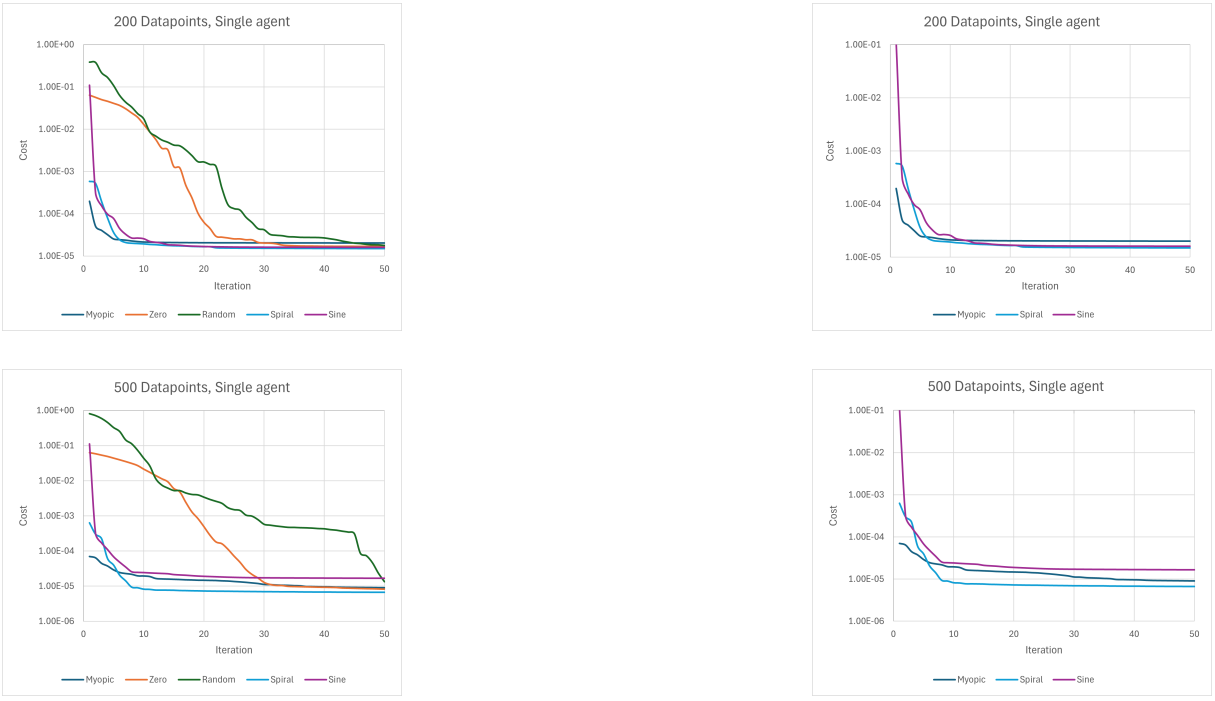
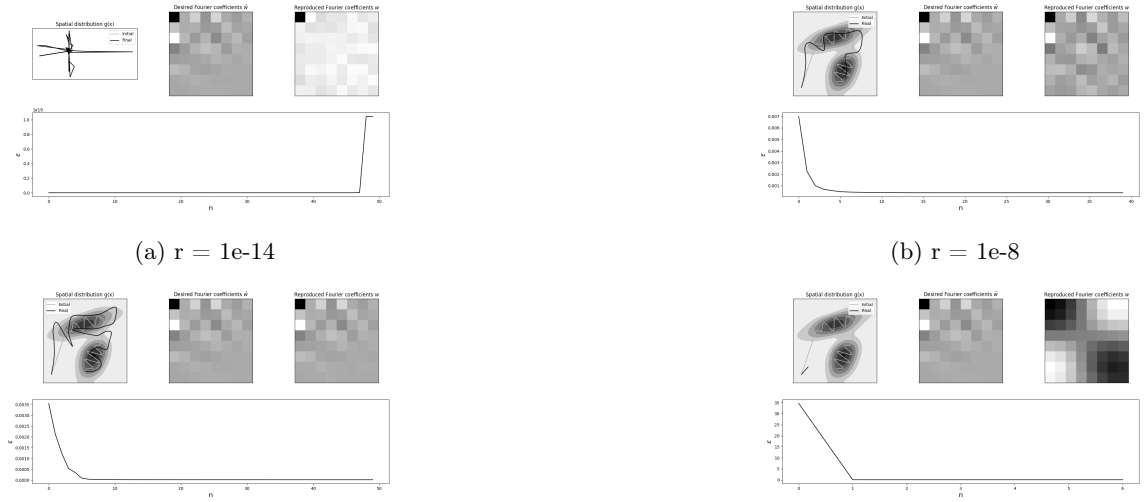


Figure 4: Cost comparison for different initialization methods in logarithmic scale for 200 and 500 data-points1

In the case of a single agent with standard parameters, myopic ergodic control initialization results in the lowest starting cost compared to other methods. While the pattern-based initializations (such as spiral or sine) also achieve lower initial costs than zero or random initialization, they do not perform as well as myopic initialization at the beginning, but generally have similar or better performance at convergence. It is likely that, within the same order of magnitude, the cost of the patterned initialization might slightly vary, depending on how well the parameters chosen for the waves fits the current distribution. This would also explain the minor difference between the sine and spiral pattern. Although the final costs are similar across all initialization methods, both pattern-based and myopic initializations show a steeper cost descent, requiring fewer iterations to converge. This indicates a more efficient optimization process and potentially a lower cumulative cost, which may be meaningful in certain applications.

2.2.2 Cost on input

The r parameter, which penalizes control inputs, significantly influences the trajectory's behavior. With a very high r, the control inputs are effectively set to zero, causing the trajectory to remain close to the starting point. Conversely, when r is too low, the trajectory tends to deviate from the area of interest, as there is little penalty for moving further away, additionally numerical instability can arise, further affecting the trajectory's reliability. Optimal, mid-range values of r guarantee a balance, enabling effective exploration while maintaining focus on the target region. See Fig.5



(a) r = 1e-14

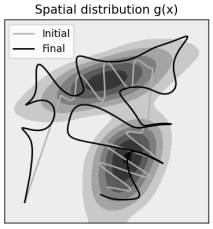
(b) r = 1e-8

(c) r = 1e-6

(d) r = 1e-2

Figure 5: Trajectories from sine initialization for increasing values of param.r

Furthermore, for what concerns the question arising in *Methods*, comparing the sine pattern initialization with the zero-input one, it is confirmed that an initial control command covering the distribution in a suitable way allows to have better performance even with lower costs on the input and guarantees convergence for a wider range of $param.r$ values. Indeed incrementally decreasing the param.r values the resulting final trajectory in the zero-input case shows unstability and non-meaningful exploration at early stages compared to the sine pattern one, see Fig 6.



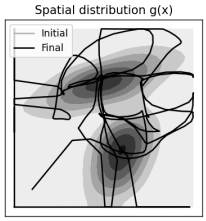
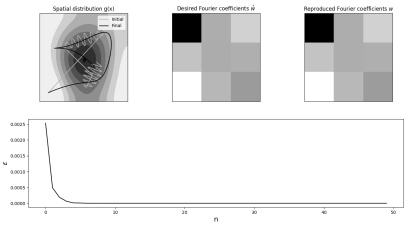


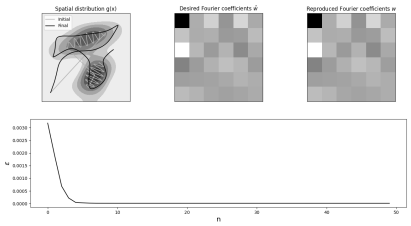
Figure 6: Sine (on the left) and zero-input (on the right) initializations, for param.r = 1e-10

2.2.3 Number of Fourier basis functions

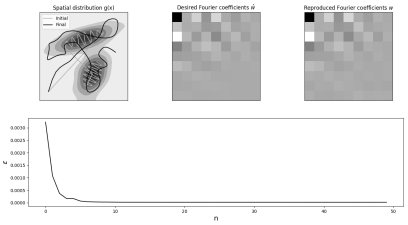
As the number of Fourier basis functions increases, the trajectories become more complex, and the reconstructed distribution more closely resembles the original target distribution, however also the computational complexity increases and the algorithm becomes slower. For instance, in the case where only three basis functions are used, the reconstruction is relatively imprecise, capturing a single Gaussian instead of two. However, despite this limitation, the initialization still aligns sufficiently with the original distribution to guide the ergodic search towards the correct regions of interest. This proves the robustness of the initialization process, even with a coarse representation of the target distribution.



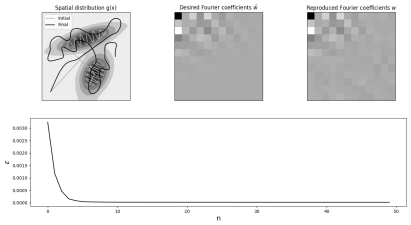
(a) Number of Fourier Basic Functions : 3



(b) Number of Fourier Basic Functions : 6



(c) Number of Fourier Basic Functions : 9



(d) Number of Fourier Basic Functions : 12

Figure 7: Trajectories from sine initialization for increasing values of number of Fourier basic functions

2.3 Additional testing

Further in the project the same problem was tested for an additional feature: a **curvature cost** was introduced. The curvature cost penalizes sharp deviations and enforces smooth trajectories by considering a reference trajectory's curvature. The curvature at each point is computed as:

$$\kappa = \frac{\dot{x}\ddot{y} - \dot{y}\ddot{x}}{\|\dot{x}\|^3},$$

where $\dot{x}$ and $\dot{y}$ are the velocities along the trajectory, $\ddot{x}$ and $\ddot{y}$ are the accelerations. The cost function minimizes deviations from a desired curvature (κ_{ref}), extracted from a reference trajectory:

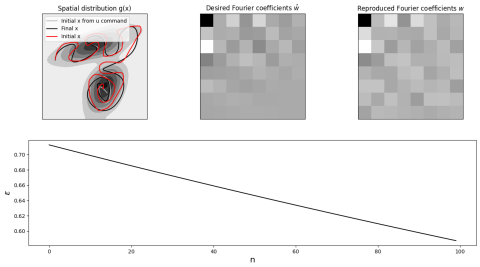
$$\epsilon_{\text{curv}} = \|\kappa - \kappa_{\text{ref}}\|^2.$$

Similar setting as previously are used, with two additional parameters concerning the curvature and slight variations in the cost weights, the updated parameters are displayed in Table 2

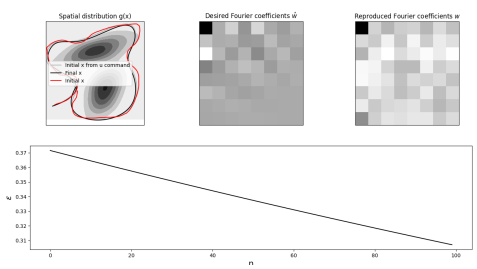
Table 2: Parameters Used in iLQR Optimization (2)

Parameter	Value
`param.nbData`	200
`param.nbVarPos`	2 (Position space dimension)
`param.nbDeriv`	1 (Number of static and dynamic features)
`param.nbDerivCurv`	3 (Number of derivatives for curvature computation)
`param.nbVarX`	`param.nbVarPos` $\times$ `param.nbDeriv` (State space dimension)
`param.nbFct`	8 (Number of Fourier basis functions)
`param.nbStates`	2 (Number of Gaussians for spatial distribution)
`param.nbIter`	50 (Maximum iLQR iterations)
`param.nbPoints`	1 (Number of viapoints to reach)
`param.dt`	1×10^{-2} (Time step length)
`param.qd`	1×10^{0} (Bounded domain weight term)
`param.qr`	1×10^{0} (Reach target weight term)
`param.qc`	1×10^{-6} (Curvature weight term)
`param.r`	1×10^{-6} (Control weight term)
`param.Mu_reach`	$[0.3, 0.9, 0, \ldots, 0]^T$ S(Target mean position)
`param.xlim`	$[0, 1]$ (Domain limit)

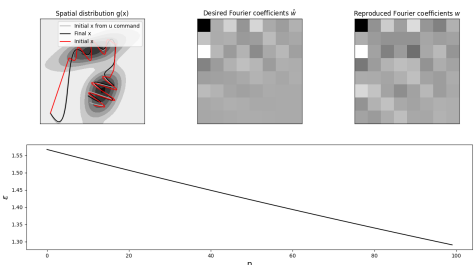
The results presented in Fig. 8 show that the curvature cost successfully maintains the resulting trajectory similar to the input one. In the plots the red trajectory is the designed one which can either be a patterned initialization or be set by the user tracing in with the mouse on the plot of the given distribution. The gray trajectory is the one smoothed out: since the curvature calculation relies on the first and second derivative of the trajectory sharp turns must be avoided to guarantee convergence. Finally the black trajectory is the resulting one, it can be noticed that in all cases it shows a high degree of fidelity to the smooth one provided as initialization. Being the cost on curvature very different from one pattern to the other the costs of the optimizations are not really comparable anymore, however it is possible to observe that they are all steadily decreasing as the trajectory gets refined. Similarly, since the trajectory is forced to be close to the one provided as input, the quality of the reconstruction of the Fourier coefficients is also highly dependent on how well the initialization trajectory covers the desired distribution. For instance, if one considers the example in Fig.8b, where the initial trajectory is mostly traced far from the underlying distribution (represented by the variance isolines), the reconstructed Fourier coefficients are much further from the desired ones, compared to Fig.8a, which uses the same method (input from the mouse), but traces a trajectory covering the GMM.



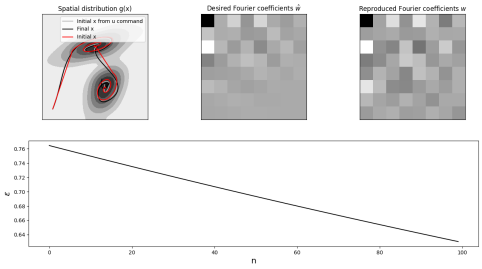
(a) Mouse centered



(b) Mouse not centered



(c) Sine



(d) Spiral

Figure 8: Comparison of Curvature Trajectories

3 Electrostatic halftoning and comparable methods

When multiple agents are considered to jointly cover the space in an ergodic control problem, the initialization is more concerned by how the agents should be distributed in the space to guarantee efficient exploration. A method to distribute particles in the domain, which is explored in this work as initialization technique for a multi-agent exploration problem is **electrostatic halftoning**, an algorithm based on Coulomb attraction presented in [3].

3.1 Electrostatic Halftoning Algorithm [3]

The electrostatic halftoning algorithm models particles as charged entities interacting in a 2D domain under electrostatic forces. This approach ensures that the particles are distributed in a manner that reproduces a given grayscale target image while avoiding excessive clustering. Below, the main principles and pseudocode for the algorithm are outlined.

3.1.1 Principles of the Algorithm

- **Particles:** A fixed number of particles $|\mathcal{P}|$ are used to approximate the target grayscale image. Each particle has an associated charge and position. The number of particles required to guarantee equilibrium and appropriate reproduction, given a discretization of the domain Γ is $|\mathcal{P}|$ defined as:

$$\Gamma := \{(i,j)^\top \mid i \in \{1,\dots,n_x\}, j \in \{1,\dots,n_y\}\},$$

$$|\mathcal{P}| := \text{round}\left(\sum_{x\in\Gamma}(1-u(x))\right),$$

- **Repulsive Forces:** The particles repel each other due to Coulomb forces (in 2D), ensuring they are evenly distributed. The cumulative repulsive force between for particle at position $\mathbf{p}_n$ is given by:

$$\mathbf{F}_{r,n} = -\sum_{\substack{m\in\mathcal{P}\\ m\neq n}} \frac{k\cdot q_n\cdot q_m}{\|\mathbf{p}_m-\mathbf{p}_n\|}\mathbf{e}_{n,m}.$$

  where k is a constant, q_m and q_n are the charges, and $\mathbf{p}_m$ and $\mathbf{p}_n$ are the positions of the particles and $\mathbf{e}_{n,m} := \frac{\mathbf{p}_m-\mathbf{p}_n}{\|\mathbf{p}_m-\mathbf{p}_n\|}$

- **Attraction to Target Image:** Each particle subject to an attractive force determined by the gray-scale intensity of the target image. This attraction ensures that the particle distribution aligns with the underlying image.

$$\mathbf{F}_{a,n} = \sum_{\substack{x\in\Gamma\\ x\neq\mathbf{p}_n}} \frac{k\cdot q_n\cdot(1-u(x))\cdot q}{\|\mathbf{x}-\mathbf{p}_n\|}\mathbf{e}_{n,x}.$$

  Similarly to before $\mathbf{e}_{n,x}$ denotes the unit vector, $\mathbf{x}$ each grid point in the discretized image.

- **Equilibrium:** The algorithm ensures that particles are distributed across the entire image domain by achieving **electrical neutrality**. This neutrality is realised when all particles have equal charges, defined as: $\forall n \in \mathcal{P} : q_n := q := 1$. Thus the total force acting on each particle combining the attractive image forces repulsive inter-particle forces becomes:

$$\mathbf{F}_n = k\cdot\left(\sum_{\substack{x\in\Gamma\\ x\neq\mathbf{p}_n}} \frac{1-u(x)}{\|\mathbf{x}-\mathbf{p}_n\|}\mathbf{e}_{n,x} - \sum_{\substack{m\in\mathcal{P}\\ m\neq n}} \frac{1}{\|\mathbf{p}_m-\mathbf{p}_n\|}\mathbf{e}_{n,m}\right).$$

In regions with constant grayscale values, particles maximize their spacing due to the balance of forces. In textured or boundary regions, attractive forces dominate, tying particles to darker areas.Neutrality ensures that if a region has the correct number o f particles proportional to its grayscale, it neither attracts nor repels external particles, anchoring the particle distribution to the image domain (no need to bound the particles' position updates).

- **Updates:** An artificial time evolution is proposed, leading to equilibrium by iteratively repositioning particles based on the current force field. The update equation for particle positions is given by:

$$\mathbf{p}_n^{k+1} = \mathbf{p}_n^k + \tau \cdot \left(\sum_{\substack{x \in \Gamma \\ x \neq \mathbf{p}_n}} \frac{1 - u(x)}{\|\mathbf{x} - \mathbf{p}_n\|} \mathbf{e}_{n,x} - \sum_{\substack{m \in \mathcal{P} \\ m \neq n}} \frac{1}{\|\mathbf{p}_m - \mathbf{p}_n\|} \mathbf{e}_{n,m} \right),$$

where: p_n^k is the position of particle n at timestep k, τ is the artificial timestep size (e.g., $\tau = 0.1$ by the authors), which incorporates constants such as point charges and Coulomb's constant. This approach reduces the imbalance of forces incrementally, ensuring particles are repositioned towards equilibrium in a stable manner.

- **Convergence:** The algorithm iteratively updates the positions of the particles until a convergence criterion is met, such as a negligible change in the particle positions.

3.1.2 Pseudocode

Algorithm 1: Electrostatic Halftoning Algorithm

Input: Target image I, number of particles N, maximum iterations T, time step Δt
Output: Particle positions $\{\mathbf{p}_i\}_{i=1}^N$

1 **Initialization:**

- Precompute the force field induced by the input image I:

$$\text{forcefield}_p = \sum_{g \neq p} \text{force}(p, g) \quad \forall\, p \in \Gamma.$$

- Distribute particles across the image domain. For faster convergence, initialize positions proportionally to the grayscale intensity:
  1. Select a random pixel x.
  2. If x is unoccupied and a random value $\in [0, \text{max\_grey}]$ exceeds $I(x)$, place a particle at x.

Evolution: **for** $t = 1$ *to* T ***or until convergence*** **do**
  for *each particle* i **do**
    Compute the total force on particle i:

$$\mathbf{F}_i = \text{bilinear\_interpolation}(\text{forcefield}, \mathbf{p}_i) + \sum_{j \neq i} \mathbf{F}_{ij}.$$

  for *each particle* i **do**
    Update position with a timestep τ:

$$\mathbf{p}_i^{t+1} = \mathbf{p}_i^t + \tau \mathbf{F}_i.$$

    Optionally apply random shaking for simulated annealing:

$$\mathbf{p}_i^{t+1} = \mathbf{p}_i^{t+1} + \text{random\_shaking}(t).$$

  Check convergence: if particle positions stabilize, break.

3.2 Algorithm adaptation and examples

As mentioned in the previous section, the number of particles required by the algorithm to guarantee reaching an equilibrium is fixed by the underlying image and the discretization adopted (and it usually is a large number > 200 even for a low resolution of the image). However for the given application, since this method is used to determine particle position in the domain, the number of particles needs to be fixed as an input to the algorithm to correspond to the number of agents. In order to do so and still guarantee balance between attractive and repulsive forces and convergence, the attraction to the target image is scaled by a scaling factor $s = \frac{number\ agents}{required\ particles}$.This was tested introducing a number of particles both smaller and larger than the required number, achieving successful behaviour in both cases.
Furthermore the artificial step parameter τ and the maximum displacement threshold defined for convergence need to be tailored to the application. A smaller τ and threshold allow for a fine-grained but slow convergence, whereas if they are larger a coarser and faster convergence can be achieved.
Finally last step to adapt the algorithm to the given application is the scaling of the agent positions to the domain bounds. The algorithm was tested on a general but more complex image to verify its functioning and on an image corresponding to the considered application, they can be observed in Fig.9 and Fig.10 respectively.

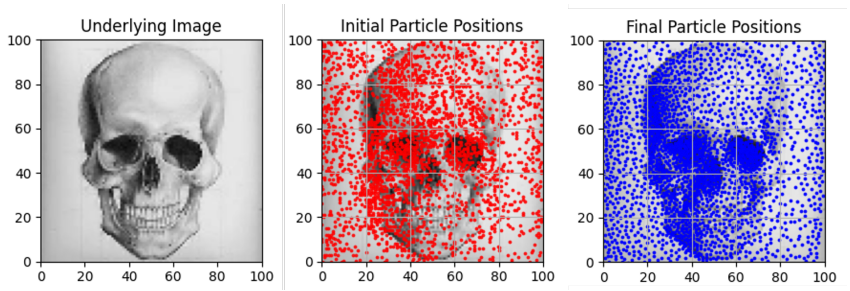


Figure 9: Electrostatic halftoning example 1

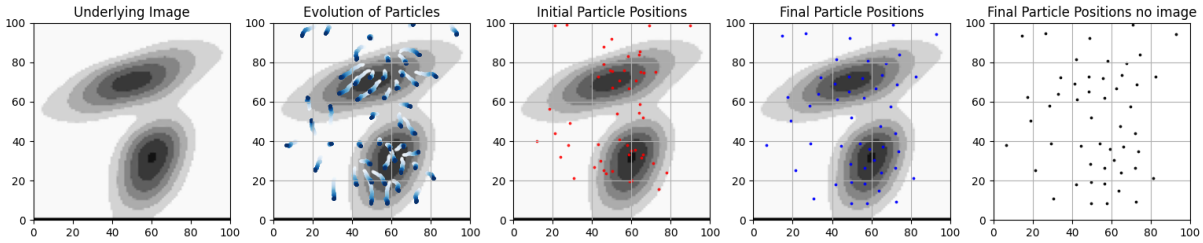


Figure 10: Electrostatic halftoning example 2

A variation of the initialization phase was also considered: the particles were distributed uniformly throughout the image grid before evolution, but the algorithm was able to achieve the same kind of convergence (see Fig. 11)

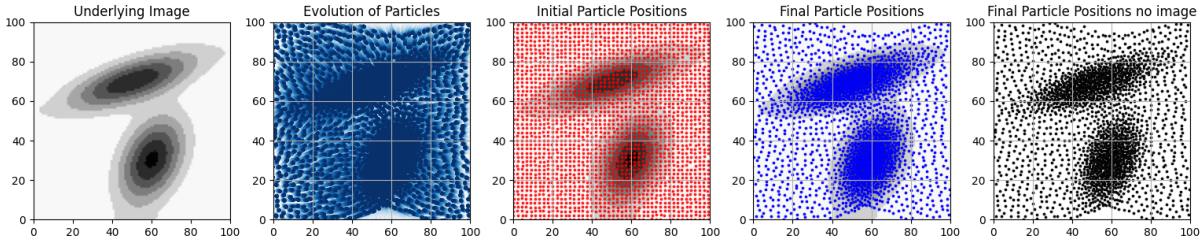


Figure 11: Electrostatic halftoning example 3: s >1 , uniform initialization

3.3 Comparison and Parallelism: Electrostatic Halftoning and Ergodic Control

The Electrostatic Halftoning Algorithm and the Ergodic Control Formulation described in the previous sections share common principles and objectives, but they approach the problem of spatial coverage differently. This section explores their similarities, differences, and conceptual links.

3.3.1 Similarities

Both methods share the following key characteristics:

- **Objective of Spatial Coverage:** Both aim to achieve **coverage** over a domain based on a target distribution:
  - *halftoning*: particles cover the 2D domain according to a grayscale input image.
  - *ergodic control*: the trajectory of an agent ensures that the time-averaged distribution matches the target spatial distribution.
- **Balancing Forces:** Both frameworks use a balance of forces:
  - *halftoning*: attractive forces pull particles toward darker image regions, while repulsive forces ensure even spacing between particles and non-clustered exploration.
  - *ergodic control*: control inputs updates direct the agent toward under-represented regions of the target distribution (obtained from spectral decomposition) while minimizing unnecessary control effort.
- **Iterative Optimization:** Both employ iterative methods:
  - *halftoning*: updates concerns particle positions based on current forces at each step.
  - *ergodic control*: adjusts control inputs iteratively to minimize a spectral cost function.

3.3.2 Differences

Despite their similarities, the two approaches differ significantly in their implementation and scope:

- **Nature of Movement:**
  - *halftoning*: deals with static particle positions, updated in a discrete way until equilibrium is reached.
  - *ergodic control*: focuses on dynamic trajectories over time, where agents move continuously in the state space.
- **Cost Function:**
  - *halftoning*: the cost is implicit in the balance of forces (minimizing imbalance).
  - *ergodic control*: a well-defined spectral cost function quantifies the mismatch between the trajectory and the target distribution.
- **Time Dependency:**
  - *halftoning*: operates in a time-discrete, space-continuous framework with no explicit temporal dynamics.
  - *ergodic control*: explicitly accounts for time dynamics, where the time spent in different regions matters. Indeed it takes into account the entire trajectory at every iteration.
- **Dimensionality of Control:**
  - *halftoning*: operates in a fixed 2D domain.
  - *ergodic control*: can generalize to higher-dimensional spaces.

Halftoning could be interpreted as a static, discrete version of ergodic control, where particles instead of trajectories are used to approximate a target distribution. While their implementations differ, both methods ensure non-clustering and convergence toward a desired spatial pattern or behavior, with ergodic control adding the complexity of temporal dynamics.

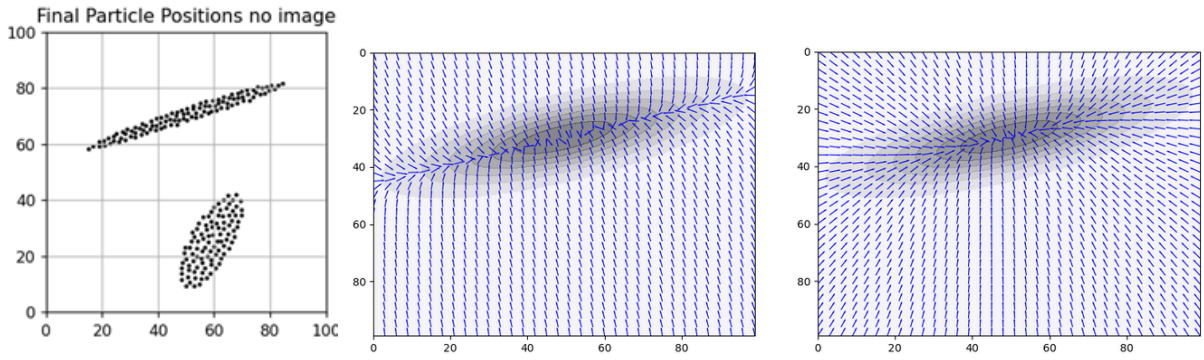
3.4 Alternative comparable methods

The electrostatic halftoning algorithm produced promising results but showed significant limits due to its high computational complexity. The primary bottleneck was the time required for computation, especially during the force field construction. Even with an optimized vectorized implementation, memory limitations prevented testing resolutions higher than 100×100 pixels for the target image.

3.4.1 Gradient-based method

To address these issues, alternative methods were explored. The first alternative focused on exploiting the nature of Gaussian Mixture Models (GMMs), specific to this application. By taking advantage of the **properties of Gaussian distributions**, the gradient of the probability density function at every point was used as the attractive component of the force field. While this approach successfully reduced computational complexity, it introduced its own challenges and was not able to achieve comparable results:

- *Scaling Issues:* The forces required careful scaling to ensure equilibrium and was not longer guaranteed by the homogeneous nature of the forces combined in the original algorithm. This scaling needed to be fine-tuned for each application, depending on input factors like image resolution and the number of agents.
- *Force Field Shape:* The resulting force field and thus the results of the algorithm (Fig.12a) differed significantly from the one computed using the original electrostatic method. Specifically the electrostatic force field tended to orient more towards the center of each Gaussian component (Fig. 12c) while the gradient-based force field was more elongated, following sharply the shape of the Gaussian distribution itself (Fig. 12b)



(a) Image 1 (b) Image 2 (c) Image 3

Figure 12: Three images displayed in the same row.

3.4.2 Diffusion-based approach

Since the gradient-based method did not provide satisfactory results, a **diffusion-based approach** was investigated. This alternative aimed to balance computational efficiency with accurate force field generation, more resembling the particle distribution in space offered by electrostatic halftoning, and thus offering a potential solution to the limitations of both the electrostatic

and gradient-based methods.
The diffusion-based approach provides an iterative method for distributing particles according to a target intensity distribution. It operates by simulating heat diffusion over the image domain, cooling regions around each placed particle to prevent overlapping and clustering, while ensuring the distribution aligns with the target image.

Principles of the Approach

- *Heat Diffusion:* The algorithm uses the heat equation to modify the target distribution iteratively. This ensures that particles are placed in areas of higher intensity while gradually reducing the local intensity to avoid placing multiple particles in the same region.
- *Cooling Kernels:* After each particle is placed, the region around it is "cooled" using a kernel function, which reduces the intensity values in the surrounding area.

The algorithm starts by computing a normalized target distribution derived from the grayscale intensity values of the input image. The heat equation is applied to this distribution to diffuse intensities iteratively. Particles are then placed in regions with the highest remaining intensity. Compared to the electrostatic halftoning approach, this method does not include any stochastic step, therefore for the same set of parameters the result is the same across multiple trials.

Parameters:

- *Number of particles (N)*
- *Resolution:* tested up to 1000*1000, has a significant impact on the speed of the algorithm
- *Diffusion strength* : controls the strength of the diffusion
- *Diffusion coefficient (α):* Controls the rate of diffusion
- *Diffusion repetition* : defines how many times the diffusion should be performed before placing a particle. Increasing this parameter results in a resulting distribution less defined compared to the target image. Satisfying results were obtained simply setting it to 1.
- *Time step (Δt):* Dynamically calculated to ensure numerical stability, based on grid spacing and diffusion coefficients.

These parameters play a crucial role in the resulting distribution, especially the diffusion strength and rate must be tailored to the input resolution to guarantee efficient spatial coverage and avoid empty areas in the domain.

Cooling Kernels
Two kernel types were implemented for the cooling step:

1. *Circular Kernel:* A first implementation uses a circular cooling kernel based on Euclidean distance:
$$d = \sqrt{(x - x_i)^2 + (y - y_i)^2}.$$
The cooling effect is modeled as a Gaussian function:
$$\text{coverage\_density} = \exp\left(-\frac{d^2}{r^2}\right),$$
where r is the particle radius.
2. *Elliptical Kernel:* An elliptical kernel was introduced to achieve anisotropic cooling and thus distribution. The elliptical distance formula is:
$$d = \frac{(x - x_i)^2}{2\sigma_x^2} + \frac{(y - y_i)^2}{2\sigma_y^2},$$
where σ_x and σ_y are the standard deviations along the x- and y-axes, respectively. The cooling effect is defined as:
$$\text{coverage\_density} = \exp(-d).$$

x_i, y_i give the position of the particle placed at the given iteration, corresponding to the current maximum heat location. The computed coverage is normalized, divided by the number of particles and then subtracted to the distribution to simulate the cooling process. The cycle is then repeated for every particle that needs to be placed. The obtained coordinates for the particles are then scaled to match the application domain.

Experiments and examples

The effect of the different parameters was tested to understand a possible tuning for the given application and verify the validity of the approach.
Some reference parameters are considered (see Table 3) and then one at a time is modified to observe its specific impact on the result.

Table 3: Parameters used in the diffusion-based approach.

Parameter	**Value**
Number of Particles	2000
Resolution	200
Grid Spacing (`dx`)	1
Repeat Diffusion	1
Diffusion strength	0.001
Diffusion coefficient (`alpha`)	$[1, 1] \times$ `diffusion`
Kernel typw	Circular
Agent Radius	0.0003

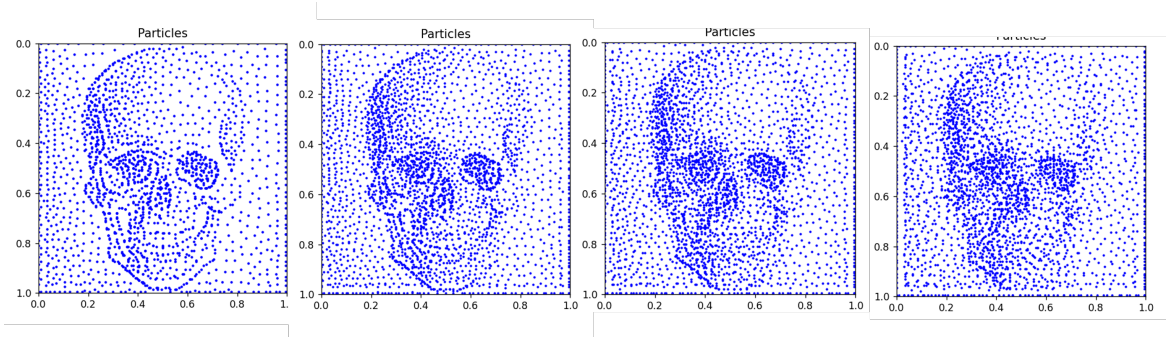


Figure 13: Effect of varying the *diffusion parameter*, from left to right: reference value, 0.002, 0.004, 0.01. Increasing the diffusion strength results in a loss of definition relative to the target image shape.

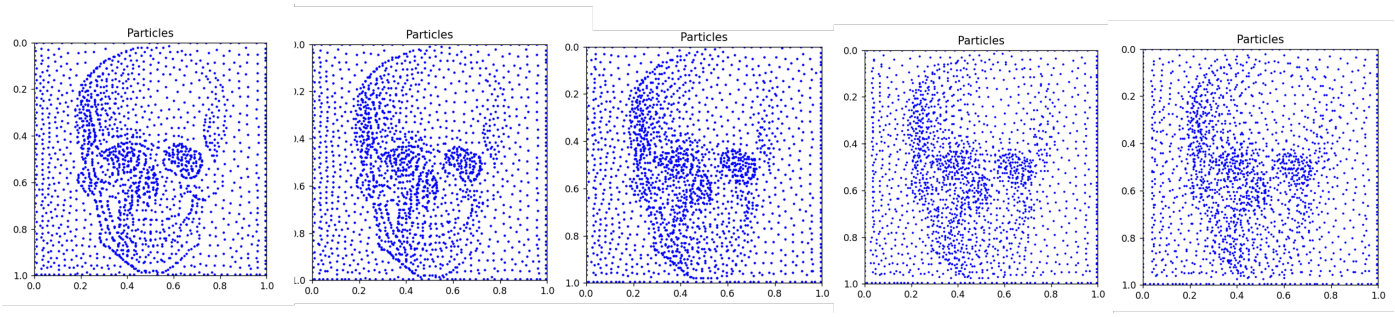


Figure 14: Effect of varying the *repeat diffusion* variable, from left to right: reference value, 0.002, 0.004, 0.01. As the previous image, increasing the diffusion repetition results in a loss of definition relative to the target image shape.

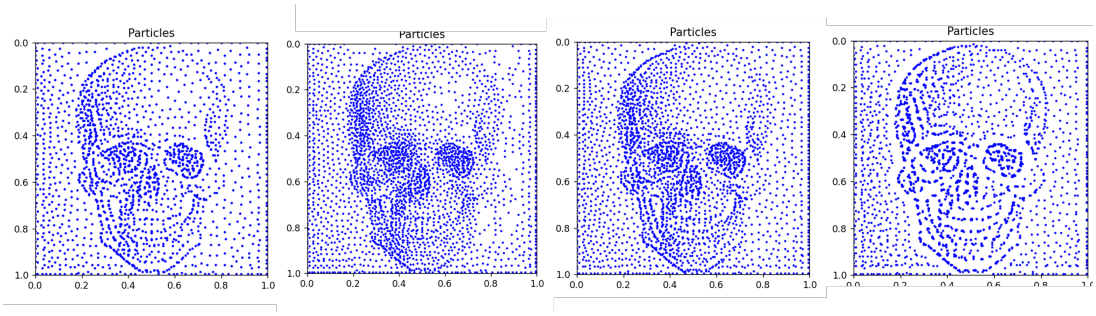


Figure 15: Effect of varying the *agent radius* variable, from left to right: reference value, 0.0001, 0.0002, 0.0006. Decreasing the agent radius allows to place particle closer to one another, filling the domain space more homogeneously. However if the radius is too small (r = 0.001 in this setting), it might lead to empty areas in the image domain.

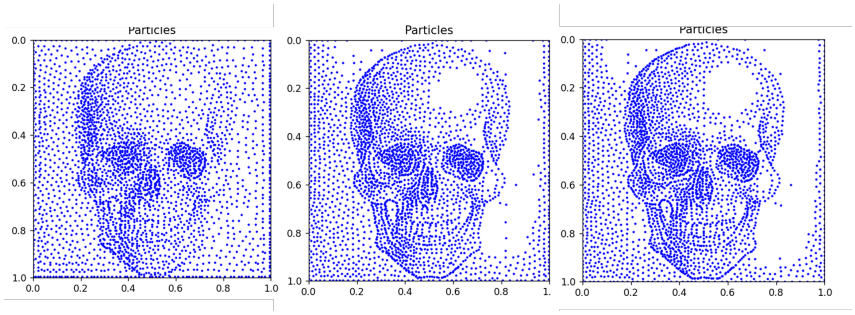


Figure 16: Effect of varying the *resolution* variable, from left to right: reference value, 500, 1000. Here other parameters were modified as well from the reference ones, but are common to the three experiments: $\alpha = 0.99 * [1, 1] \times$ `diffusion`, agent radius = 0.0001. This show the need to relate the diffusion parameters to the input values from the algorithm, to ensure efficient particle placing for all settings.

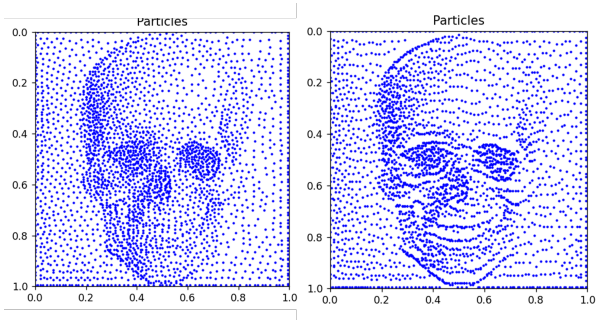


Figure 17: Effect of varying the *Kernel type*: circular (on the left), elliptical (on the right). For the circular one, modification to α and r as in Fig. 16. For the elliptical kernel, $\sigma_x = 0.005$, $\sigma_y = 0.015$: this leads to larger spacing between particles in the y directions and closer in the x direction.

Tentative tuning Due to the issue of having some empty areas in the distribution when modifying parameters a tentative tuning was put in place. Almost all parameters regulating the diffusion have been set as input to the algorithm: the diffusion strength, the agent radius (or covariance), the image resolution and the number of particles. As a consequence the only parameter which can be regulated for this tuning is the diffusion coefficient α. For the considered image it has been hand tuned as:

$$\alpha = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \cdot \text{diffusion\_strength} \cdot \left(\text{resolution}^{0.75} / \left(\text{num\_particles} \right)^{0.7} \right) \cdot \left(\frac{\text{agent\_radius}}{\text{reference\_radius}} \right)^{0.7}$$

This definition generally provides good results for the ranges b following in Table 4.
However when considering wider ranges for each of the parameters, finding a unique definition satisfying all possible combinations becomes quite complex. Moreover, also using an image with

Table 4: Parameters range for diffusion coefficient tuning

Diffusion Strength	**Agent Radius**	**Image Resolution**	**Number of Particles**
[0.001,0.003]	[0.001, 0.005]	[300,900]	[1300.2700]

a different overall greyscale value and distribution would hinder the validity of this definition. Therefore depending on the target image and the desired results the parameters can be set (or the coefficients present in the formula above can be modified) following these general principles:

- *Diffusion Strength*: Scales the diffusion directly, providing the base intensity.
- *Resolution* (resolution$^{0.75}$): Higher resolution requires stronger diffusion to spread particles effectively, with 0.75 ensuring a moderate scaling and a stronger effect for larger values.
- *Number of Particles* ((num_particles)$^{0.7}$): Fewer particles require stronger diffusion to avoid empty spaces, while 0.7 tempers the relationship to avoid excessive reduction.
- *Agent Radius* ((agent_radius/reference_radius)$^{0.7}$): Smaller radii require more diffusion to spread particles since the cooling is more concentrated, normalized to a reference scale to avoid excessive values.

This formula ensures α adapts dynamically to varying grid sizes, particle counts, and domain scales, providing balanced particle distribution without excessive overlap or gaps.

4 Multi-agent problem

As mentioned in the previous section when multiple agents are used for ergodic exploration initialization becomes the matter of defining suitable starting positions for efficient exploration. Prior sections described the electrostatic halftoning method and its diffusion alternative as well as the cost terms terms adopted in the iLQR update. See *Methods* for a reminder.

4.1 Methods

In this setting the methods described in the previous section are tested as initialization techniques for a multi-agent ergodic control trajectory optimization and their performance is compared to some naive alternatives:

- *Center*: all the agents are initialized at the center of the domain
- *Corner*: all agents are placed at the lower left corner of the domain
- *Random*: each agent is assigned a random location within the domain

The cost components are the same as described in the corresponding part in Section 2

4.2 Testing

The trajectory optimization for an ergodic control problem implemented is the same as for Section 2, and therefore requires setting the same parameters. The values adopted for this testing are showed in Table 5, while Table 6 and Table 7 present the ones chosen for the diffusion and the electrostatic halftoning methods respectively. More pictures showing the trajectories and the algorithms results for each trials are available in the Appendix.

Table 5: Parameters Used in iLQR Optimization (3)

Parameter	**Value**
`param.nbAgents`	10, 50, 300, 1000 (Number of datapoints)
`param.nbData`	80, 50, 30, 10 (Number of datapoints)
`param.nbVarPos`	2 (Position space dimension)
`param.nbDeriv`	1 (Number of static and dynamic features)
`param.nbVarX`	`param.nbVarPos` $\times$ `param.nbDeriv` (State space dimension)
`param.nbFct`	8 (Number of Fourier basis functions)
`param.nbStates`	2 (Number of Gaussians for spatial distribution)
`param.nbIter`	50 (Maximum iLQR iterations)
`param.nbPoints`	1 (Number of viapoints to reach)
`param.dt`	1×10^{-2} (Time step length)
`param.qd`	1×10^{0} (Bounded domain weight term)
`param.qr`	0×10^{4} (Reach target weight term)
`param.r`	1×10^{-7} (Control weight term)
`param.Mu_reach`	$[0.3, 0.9] *$ `param.nbAgents` (Target mean position)
`param.xlim`	$[0, 1]$ (Domain limit)

Table 6: Parameters for testing with diffusion

Diffusion Strength	**Diffusion Coefficient**	**Repeat diffusion**	**Agent Radius**	**Kernel type**	**Image Resolution**
0.001	$[1, 1] \times$ `diffusion`	1	0.005	Circular	100

Table 7: Parameters for testing with electrostatic halftoning

Iterations	τ	**Displacement threshold**	**Initialization type**	**Image Resolution**
300	0.05	2.5E-3	Random	100

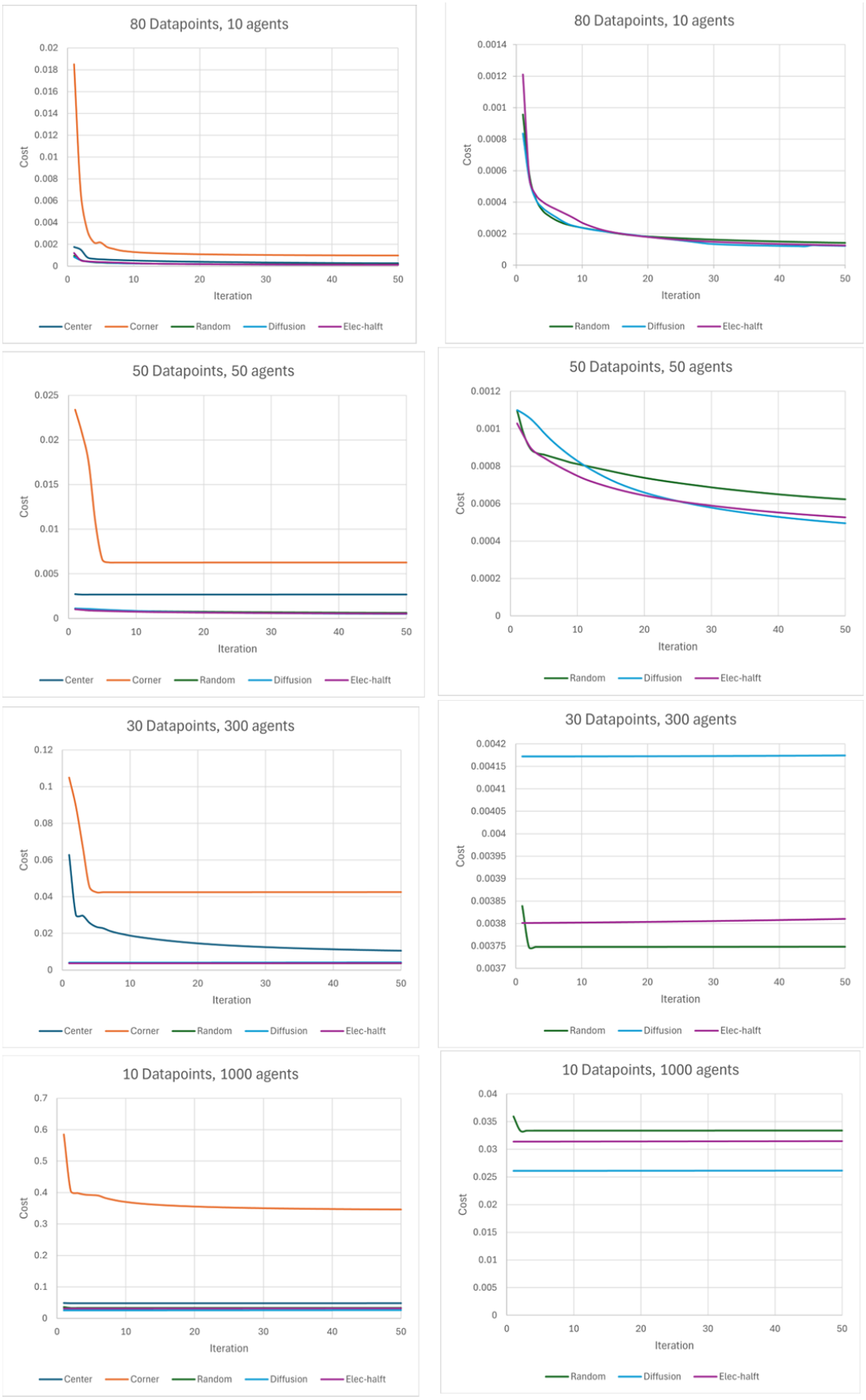


Figure 18: Cost trend comparison between the different techniques described in *Methods* for a varying number of agents and trajectory length

From Fig. 18, it is evident that the corner implementation results in higher costs. This is because starting from a corner requires agents to travel longer distances to cover the domain, given the same number of data points in the trajectory. As a result, this approach is inefficient for general distributions, as it fails to take advantage of spreading agents across the domain effectively.
The center implementation performs better since, for the given distribution, most of the area to be covered is close to the center. However, this method is limited and highly dependent on whether the distribution is centered or symmetric. For other distributions, it might perform poorly. Placing all agents in the same initial position, such as the center, is not an efficient strategy for space coverage as it neglects the advantage of distributing agents across the domain. This approach might only be suitable for specific cases with radial symmetry.
Random initialization shows comparable performance to electrostatic halftoning and diffusion-based methods, sometimes performing slightly better or worse. This similarity is likely due to the simplicity of the current distribution. For more complex or highly concentrated distributions, random initialization may perform worse, while electrostatic halftoning and diffusion methods are better suited as they place agents in more advantageous starting positions. For 30 agents, diffusion initialization is slightly worse than random and electrostatic halftoning. However, the difference is minimal and likely influenced by the stochastic nature of these methods, so it should not be considered significant over multiple trials.
When the number of agents is very high, the cost becomes almost constant. This is because the limited domain is already well-covered, and each agent has only a short trajectory, leaving little room for further optimization.
This trend shows that with enough agents, all methods can achieve nearly complete domain coverage, and the differences in their performance become negligible. These observations suggest that diffusion-based and electrostatic halftoning methods are particularly useful for complex or uneven distributions, while random initialization might be a simpler, adequate solution for less complex cases. However, the choice of method should consider the specific requirements of the task, including the complexity of the distribution and the number of agents available.

5 Inverse kinematics

The final purpose of the exploration problem analyzed throughout this project would be the application to a robot manipular for exploration in a manipulation task. A step in this direction is performing least square inverse kinematics for a manipulator given the viapoints to be followed. The ultimate goal of the ergodic exploration considered in this project is to enable a robotic manipulator to effectively perform exploration tasks in manipulation scenarios. A crucial step towards this objective involves solving the Inverse Kinematics (IK) problem, allowing the manipulator to follow a predefined trajectory represented by a set of via-points in space. Therefore, part of the project was focused on implementing and testing a least-squares-based Inverse Kinematics algorithm for a manipulator, leveraging its kinematic structure defined by Denavit-Hartenberg (DH) parameters. The IK computation ensures that the end-effector of the robot closely follows the desired trajectory while meeting both position and orientation constraints.

5.1 Implementation

A least-squares-based inverse kinematics (IK) solver was adapted from [1] for a robotic manipulator defined using Denavit-Hartenberg (DH) parameters, either in standard or modified convention. The goal of the solver is to compute the joint configurations required for the manipulator's end-effector to follow a predefined trajectory consisting of viapoints in 3D space. These viapoints include positional constraints ($\mathbf{p} \in \mathbb{R}^3$) and orientation constraints ($\mathbf{q} \in \mathbb{S}^3$, represented as unit quaternions).
The IK problem is formulated as an optimization task to minimize the error between the desired end-effector pose ($\mathbf{f}_{\text{desired}}$) and the current pose ($\mathbf{f}_{\text{current}}$) at each viapoint. This error is defined in the tangent space of the combined position and orientation manifold $\mathbb{R}^3 \times \mathbb{S}^3$ using a logarithmic map. The least-squares approach minimizes a quadratic cost function:

$$\mathcal{L} = \frac{1}{2}\mathbf{e}^\top \mathbf{Q}\mathbf{e} + \frac{\alpha}{2}\|\Delta\mathbf{x}\|^2,$$

where $\mathbf{Q}$ is a weighting matrix for positional and rotational constraints, and α is a regularization term to ensure numerical stability. The solver iteratively updates the joint configuration $\mathbf{x}$ using:

$$\Delta\mathbf{x} = -\left(\mathbf{J}^\top \mathbf{Q}\mathbf{J} + \alpha\mathbf{I}\right)^{-1} \mathbf{J}^\top \mathbf{Q}\mathbf{e},$$

where $\mathbf{J}$ is the Jacobian of the forward kinematics function, computed numerically.
The viapoints are defined as a trajectory in 3D space and the solver computes the joint configurations for each point while ensuring that the end-effector maintains the required pose.

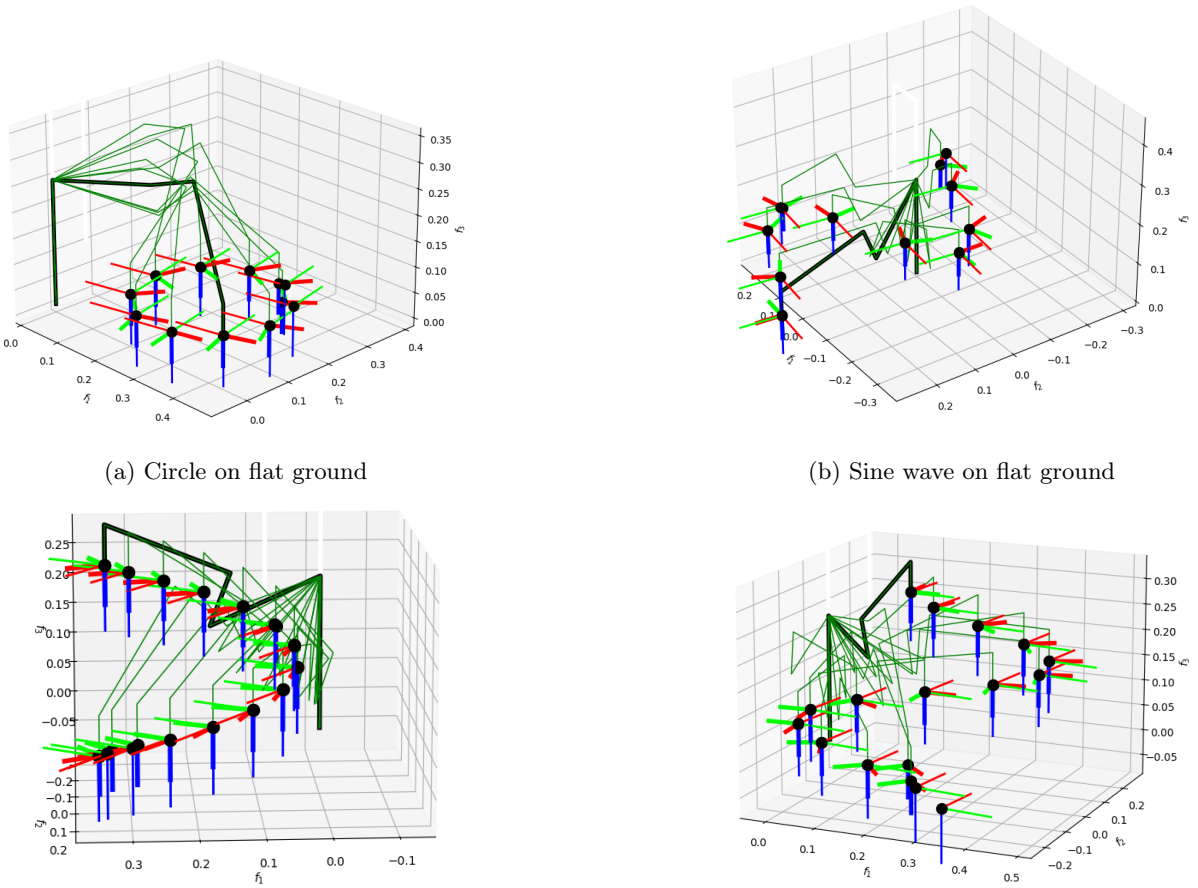
5.2 Application-specific details

The imagined application for this project, aligned with the work done on the ergodic control problem, would be a robot tracing a trajectory with a pencil on an horizontal table. This is achieved by simply setting a very small value (0.001) as z coordinate to viapoints obtained from a 2D trajectory and setting their orientation quaternion to be perpendicular to the xy plane and in opposite direction compared to the positive z direction (the manipular comes from the top of the table) A key simplification in this implementation is assigning zero weight to the rotation around the z-axis in $\mathbf{Q}$, as this degree of freedom is not critical for the task, indeed pencils are generally symmetrical and therefore the angle of rotation around the z-axis is not relevant.

5.3 Examples and verification

The approach was tested hand-designing different trajectories: Fig.19a and Fig.19b are representative of the actual application on an horizontal surface, whereas Fig.19c and Fig.19d show

the applicability of the viapoint definition and least square formulation also in a 3D setting.



(a) Circle on flat ground

(b) Sine wave on flat ground

(c) Circle with increasing z coordinate

(d) Sine with increasing z coordinated

Figure 19: Inverse kinematics visualization for different trajectories: in white the initial manipulator position, in black the final one, in the green the positions at the viapoints

From Fig.19 it can be noticed that the manipulator successfully maintain the desired orientation under different settings. Furthermore to verify the weighting matrix, in the viapoint definition, for each of them a random rotation around the z axis has been defined. It is especially visible from Fig.19a that setting to null the Q component corresponding to the rotation around the z axis attains the desired results. The z axis of the manipulator matches the one of the viapoint, ensuring the correct orientation with respect to the x and y axes, but the x and y axes of the two are not matching, proving that the least- square optimization is not concerned with it.

6 Conclusions and further work

This work explored innovative methods for initialization in ergodic control problems, focusing on their application in single-agent, multi-agent, and trajectory-based scenarios. By combining spectral-based ergodic control (SMC) and iterative Linear Quadratic Regulation (iLQR), the study investigated structured initialization approaches and alternative methods to enhance trajectory optimization for exploration tasks.

In the single-agent case, structured initialization methods such as sine and spiral patterns demonstrated their ability to reduce the ergodic cost and optimize trajectories, ensuring better coverage of the target distribution. These methods offered significant advantages over random or zero-input initialization, including faster convergence and higher trajectory quality. The inclusion of curvature costs further refined these trajectories, ensuring smoothness while preserving fidelity to the initial patterns.

For multi-agent scenarios, a comparison of initialization methods was performed, including naive approaches such as center, corner, and random initialization, as well as more advanced techniques like electrostatic halftoning and diffusion-based initialization. The results showed that advanced methods provide better space coverage and lower costs, particularly for more complex or highly concentrated distributions. Random initialization performed comparably for simpler distributions but lacked the advantageous placement offered by electrostatic halftoning and diffusion techniques. These findings highlight the need of using suitable initialization techniques depending on the problem's complexity and the distribution being covered, especially taking into account that the more advanced methods are also more computationally expensive and require hand-tuning, thus might be unnecessary for simple tasks.

The electrostatic halftoning algorithm effectively distributed agents across a domain in accordance with target intensity distributions, balancing the requirements of attracting agents to target areas and maintaining separation between them, modeled as electrostatic Coulomb forces. To address computational challenges, a diffusion-based alternative was introduced, which used heat diffusion and localized cooling to achieve effective particle distribution. While this method proved scalable and efficient, it required careful parameter tuning, pointing out an inevitable trade-off between simplicity and accuracy.

The project also explored inverse kinematics (IK) for manipulators to ensure trajectory-following capabilities. By employing a least-squares formulation, the IK implementation achieved adherence to positional and orientation constraints. The simplification of neglecting rotation around the z-axis tailored the solution to planar exploration tasks, demonstrating the flexibility of the IK solver in both 2D and 3D scenarios.

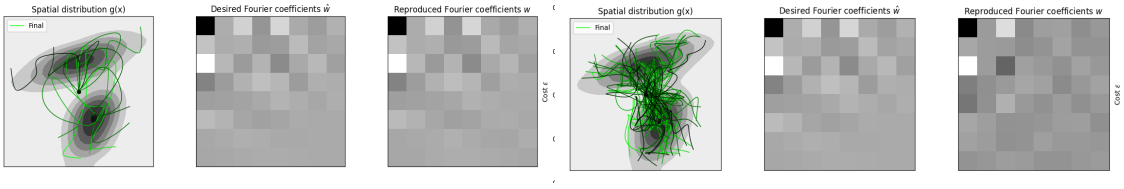
Despite these achievements, the methods developed in this project leave room for improvement. One key area of future work would involve testing these initialization techniques in real-world manipulation tasks. This could include repeating the comparisons made in simulation to evaluate their performance under real-life conditions and verifying the methods on physical robots. Such tests would help assess their robustness in the presence of sensor noise, actuation delays, and hardware constraints. Additionally, exploring these initialization techniques for a wider variety of target distributions would provide interesting insights into their general applicability and limitations.

Furthermore, the structured initialization techniques and diffusion-based methods relied heavily on parameter tuning, such as diffusion coefficients and agent radii. Future work could focus on adaptive parameter optimization to achieve robustness across various environments and use cases. Extending these methods to higher-dimensional spaces would also broaden their applicability to more complex tasks.

In conclusion, this work has provided valuable insights into initialization strategies and optimization techniques for ergodic control problems. The proposed methods and their potential improvements offer a foundation for further investigation.

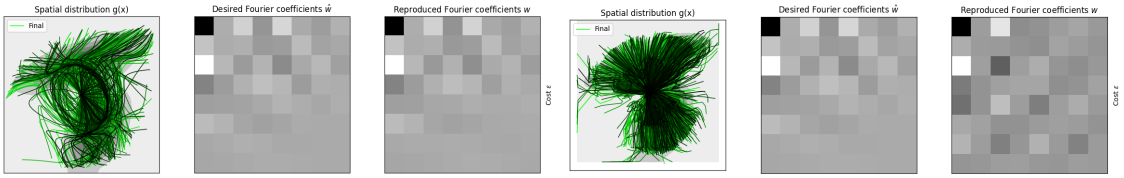
7 Appendix

8 Additional figures concerning multi-agent testing



(a) 10 agents, 80 datapoints

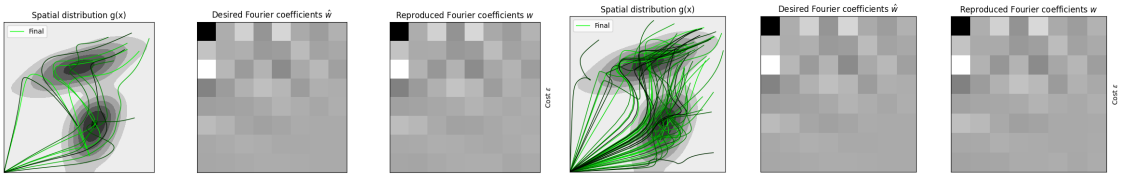
(b) 50 agents, 50 datapoints



(c) 300 agents, 30 datapoints

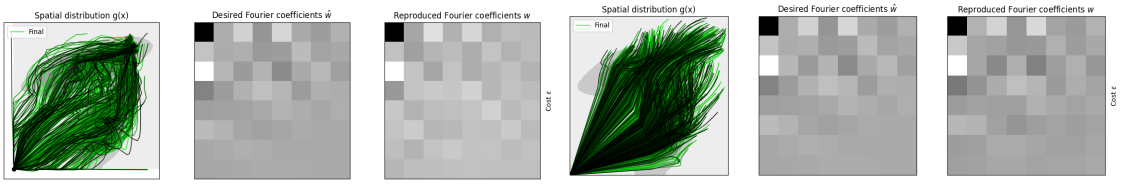
(d) 1000 agents, 10 datapoints

Figure 20: Resulting trajectories, desired and reconstructed Fourier coefficients for *centered* initialization



(a) 10 agents, 80 datapoints

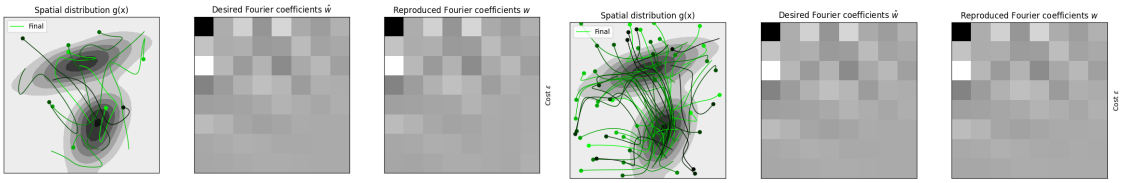
(b) 50 agents, 50 datapoints



(c) 300 agents, 30 datapoints

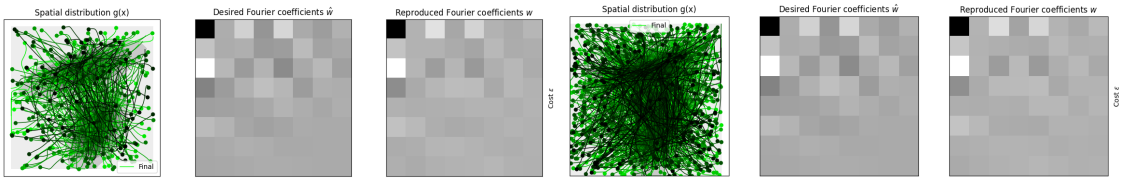
(d) 1000 agents, 10 datapoints

Figure 21: Resulting trajectories, desired and reconstructed Fourier coefficients for *corner* initialization



(a) 10 agents, 80 datapoints

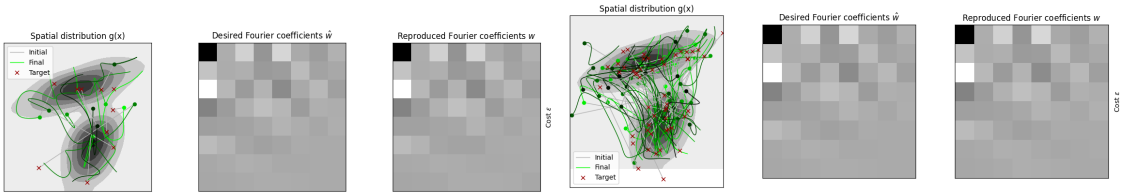
(b) 50 agents, 50 datapoints



(c) 300 agents, 30 datapoints

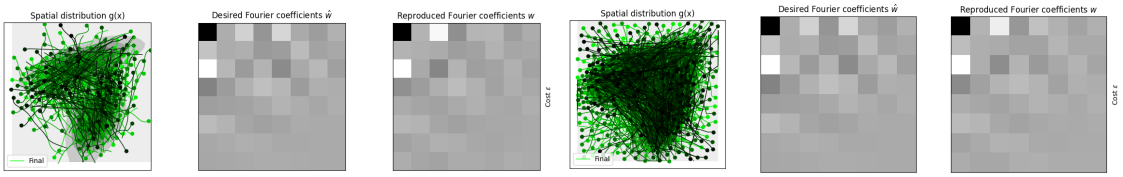
(d) 1000 agents, 10 datapoints

Figure 22: Resulting trajectories, desired and reconstructed Fourier coefficients for *random* initialization



(a) 10 agents, 80 datapoints

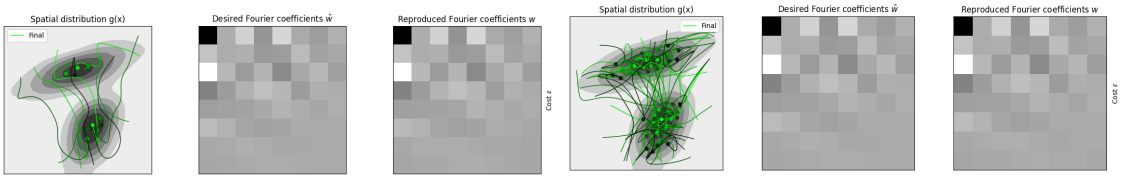
(b) 50 agents, 50 datapoints



(c) 300 agents, 30 datapoints

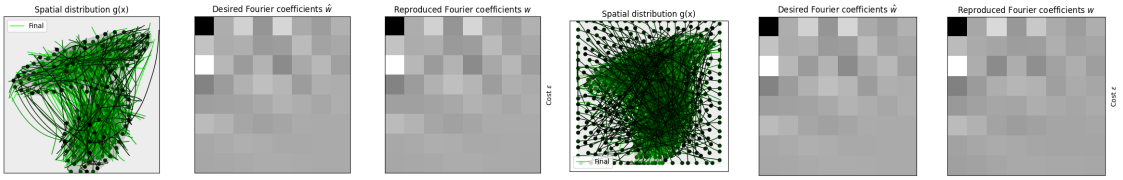
(d) 1000 agents, 10 datapoints

Figure 23: Resulting trajectories, desired and reconstructed Fourier coefficients for *electrostatic halftoning* initialization



(a) 10 agents, 80 datapoints

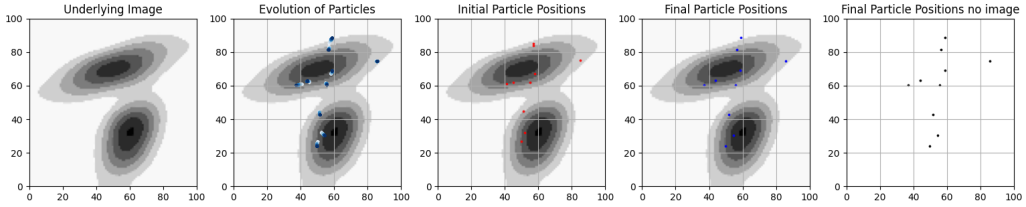
(b) 50 agents, 50 datapoints



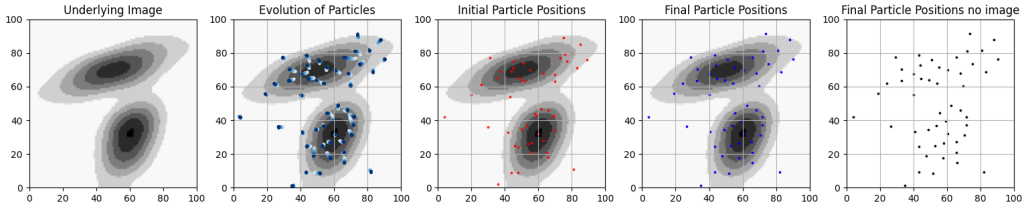
(c) 300 agents, 30 datapoints

(d) 1000 agents, 10 datapoints

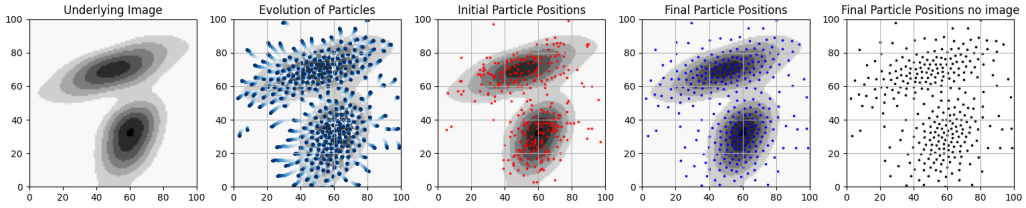
Figure 24: Resulting trajectories, desired and reconstructed Fourier coefficients for *diffusion-based* initialization



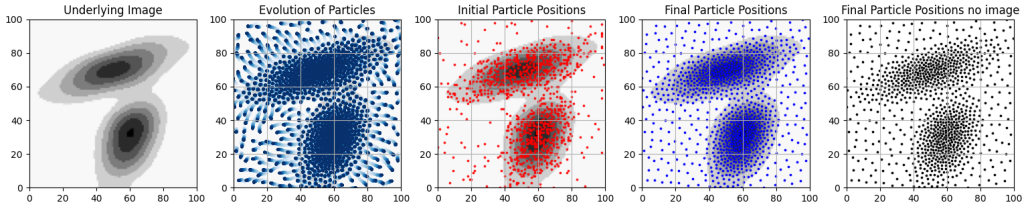
(a) 10 particles



(b) 15 particles

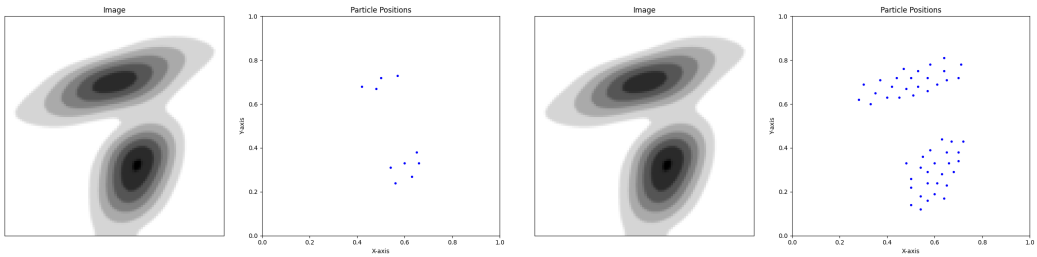


(c) 300 particles



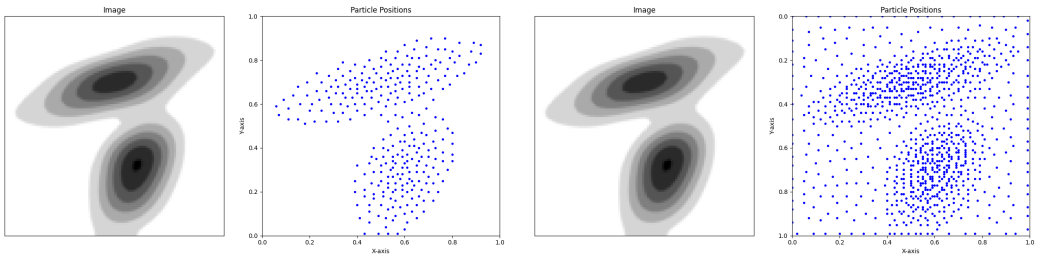
(d) 1000 particles

Figure 25: Electrostatic halftoning result for particle placement, used in initialization to the ergodic control problem.



(a) 10 particles

(b) 50 particles



(c) 300 particles

(d) 1000 particles

Figure 26: Diffusion result for particle placement, used in initialization to the ergodic control problem.

List of Figures

List of Tables

References

[1] Idiap Research Institute. *Robotics Codes From Scratch*. GitLab repository. 2024. URL: https://gitlab.idiap.ch/rli/robotics-codes-from-scratch.

[2] Idiap research institute Sylvain Calinon. *A Math Cookbook for Robot Manipulation*. Last accessed: 2024-12-30. 2024. URL: https://rcfs.ch/doc/rcfs.pdf.

[3] Christian Schmaltz et al. "Electrostatic Halftoning". In: *Computer Graphics Forum* 29.8 (2010), pp. 2313–2327. DOI: https://doi.org/10.1111/j.1467-8659.2010.01716.x. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-8659.2010.01716.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2010.01716.x.