Engineering with Computers (2019) 35:537–550
https://doi.org/10.1007/s00366-018-0614-6

ORIGINAL ARTICLE

ALBATROS: adaptive line-based sampling trajectories for sequential measurements

Tom Van Steenkiste[1] · **Joachim van der Herten**[1] · **Dirk Deschrijver**[1] · **Tom Dhaene**[1]

Received: 24 November 2017 / Accepted: 7 May 2018 / Published online: 16 May 2018

Abstract
Measurements in 2D or 3D spaces are ubiquitous among many fields of science and engineering. Often, data samples are gathered via autonomous robots or drones. The path through the measurement space and the location of the samples is traditionally determined upfront using a one-shot design of experiments. However, in certain cases, a sequential approach is preferred. For example, when dealing with a limited sampling budget or when a quick low-resolution overview is desired followed by a steady uniform increase in sampling density, instead of a slow high-resolution one-shot sampling. State-of-the-art sequential design of experiment methods are point-based and are often used to set up experiments both in virtual (simulation) as well as real-world (measurement) environments. In contrast to virtual experimentation, physical measurements require movement of a sensor probe through the measurement space. In these cases, the algorithm not only needs to optimize the sample locations and order but also the path to be traversed by sampling points along measurement lines. In this work, a sequential line-based sampling method is proposed which aims to gradually increase the sampling density across the entire measurement space while minimizing the overall path length. The algorithm is illustrated on a 2D and 3D unit space as well as a complex 3D space and the effectiveness is validated on an engineering measurement use-case. A computer code implementation of the algorithm is provided as an open-source toolbox.

Keywords Line-based sampling · Sequential design · Design of experiments · Area coverage · Automated measurements · Surrogate modeling

1 Introduction

Many fields of science and engineering rely on measurements in 2D or 3D spaces to capture and map specific values of interest. The measurements are typically performed using a measurement device such as an Unmanned Aerial Vehicle (UAV), Unmanned Ground Vehicle (UGV), or robotic arm. Example applications include fire detection and monitoring [14], cartography [15], vegetation monitoring [3], agricultural field mapping [32], 3D mapping [29], electro-magnetic compatibility (EMC) testing [12], etc.

The location and order of measurements in physical and simulation experiments are determined using Design of Experiments (DoE) methods [27]. In this work, we consider computer-aided DoE. The primary focus of computer-aided DoE is on the space-filling aspect of DoE to cover the entire region [25]. The goal of space-filling methods is to evenly distribute the information gathering across the entire measurement space. In literature, this is known as exploration of the measurement space. Traditionally, DoE methods are one-shot approaches that determine all sample locations upfront. Their downside is the risk of selecting too few data points leading to undersampling (incomplete information) or selecting too much data points leading to oversampling (increased measurement cost), which is both undesired.

The number of sample points that can be gathered and the path length to be traversed can also be limited due to battery drainage or mechanical failure. Mechanical failures are common [6, 26] when the measurement devices work in hazardous conditions such as in nuclear radiation measurements [19]. Finally, in some situations, such as fire detection and monitoring [14], it is desirable to first get a quick general low-resolution overview of the measurement space before initiating more detailed and time-consuming high-resolution

✉ Tom Van Steenkiste
tomd.vansteenkiste@ugent.be

[1] Department of Information Technology, Ghent University-imec, IDLab, Technologiepark-Zwijnaarde 15, 9052 Gent, Belgium

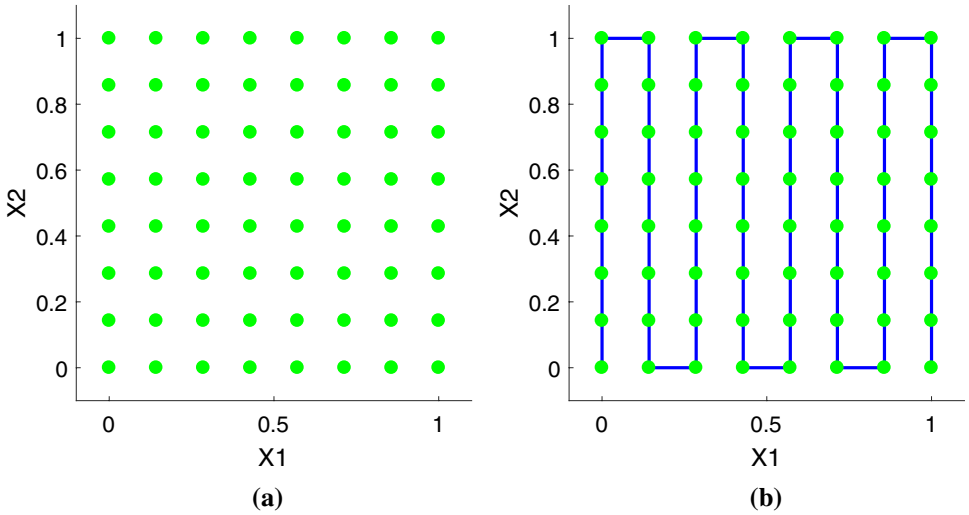


Fig. 1 Two different sampling methods. **a** Point-based sampling. **b** Line-based sampling

measurements. As a solution, sequential DoE strategies have been developed to iteratively extend the sample set during the measurement process.

In sequential DoE, subsequent batches of new sample locations are determined based on previously collected samples. These previous samples can be collected via any sampling scheme. A key advantage of this approach is that the total amount of samples does not need to be prespecified, as it can grow during the experimentation. Termination criteria can be based on physical constraints such as battery status, mechanical failure, or time duration. Alternatively, a measure of information density can be constructed if the samples are used to build a surrogate model [17]. An error measure of the surrogate model can be calculated using cross validation [18].

State-of-the-art sequential DoE strategies are focused on computer simulations. In virtual experimentation, jumping through the design space is often without additional cost. Real measurements, however, are associated with physical movements of a sensor probe through a measurement space which comes at a significant cost. This movement should also be optimized. Instead of sampling individual points, line-based sequential sampling methods are needed that generate optimal sampling lines along which sampling points are chosen. In this case, the path and sample locations are optimized and the DoE consists of 2 parts:

- a coverage path planning algorithm to ensure that the path of given length l covers the area as evenly spread as possible (i.e., space-filling).
- a path sampling algorithm to determine where to perform N measurements along the selected path.

In this work, a combined coverage path and path sampling algorithm is presented to gradually increase the sampling density over the entire measurement space called the Adaptive Line-Based Sampling TRajectOrieS (ALBATROS) algorithm. The algorithm combines point-based sampling methods from DoE with area coverage strategies to generate sampling trajectories. These sampling trajectories constitute a path that is represented as a sequence of line segments. While the line segments are traversed, measurements are performed and the sampling density is gradually increased over the entire measurement space. This ensures that at each time step, a quasi-uniform distribution of the information is obtained.

This paper is organized as follows. In Sect. 2, related work is discussed. Then, in Sect. 3, the ALBATROS algorithm is presented. In Sect. 4, the experimental setup is discussed with the evaluation criteria, and in Sect. 5, the experiments and results are presented. Finally, conclusions are drawn in Sect. 6.

2 Related work

Design of experiments encompasses all methods used to adequately gather information in a measurement space. In this work, we focus on computer-aided design of experiments. These strategies can either be point-based, in which the information consists of samples with a specific location or they can be line-based in which data samples are collected along line segments in the measurement space, and hence, the order of these samples is determined by these lines. Figure 1 demonstrates a point-based sampling scheme and a line-based sampling scheme. Note that in the end, the sample locations represented by the green dots are the same. The sampling order, however, does not necessarily have to be the same. For the point-based sampling, the measurements can jump through the design space, whereas for the line-based sampling, the order follows a strict line pattern.

Table 1 Overview of computer-aided DoE strategies

	Point-based	Line-based
One-shot	Factorial designs [28] Latin hypercube designs [33]	Boustrophedon path [8] Hilbert curve [21]
Sequential	Maximin criterion [23] LOLA-Voronoi [10]	ALBATROS

Traditionally, computer-aided DoE determines the entire data collection upfront in one shot. A more advanced option is to use sequential design in which the data collection strategy is iteratively extended. This strategy can still be computed upfront but can also be stopped early during the data gathering process based on error thresholds or due to other stopping criteria without significantly affecting the sampling distribution.

One can classify all computer-aided DoE strategies in four classes, see Table 1.

2.1 Point-based sampling

Point-based sampling methods can be categorized into one-shot methods and sequential methods:

- Examples of one-shot point-based DoE methods include factorial designs [28] and latin hypercube designs (LHDs) [33]. These algorithms focus on space-fillingness and covering the entire space as evenly as possible. However, one-shot point-based DoE methods require a fixed number of samples (N) to be defined upfront. If N is chosen inadequately, it can lead to oversampling or undersampling. Furthermore, the measurement sensor can be influenced by external factors that hamper data collection, such as mechanical failures [6, 26], unexpectedly altering the value of N during the measurement process.
- On the other hand, sequential point-based DoE methods from the active learning domain [7] transform the sampling procedure into an iterative process [11, 17]. An example of a sequential DoE method is maximin sampling [23] which determines the location of additional sample points based on the maximin criterion (see Sect. 3.3 for details).

2.2 Line-based sampling

A different type of sampling is line-based sampling. Instead of determining individual sampling points, a sampling trajectory is determined along which samples are taken. These line-based DoE methods are known in the literature as coverage path planning algorithms and the additional step of determining the sampling points along the path is often omitted. Similar to the point-based DoE methods, the line-based methods focus on space-fillingness of the lines. Again, these sampling methods can be categorized into one-shot methods and sequential methods:

- An example of a one-shot coverage path algorithm in a trapezoidal space is the Boustrophedon path [8]. When the measurement space is not a trapezoidal, e.g. due to obstacles, more advanced coverage algorithms are used, such as the Boustrophedon cellular decomposition method [8]. Other algorithms incorporate energy expenditure and time duration into the coverage path planning [5]. Another type of space-filling curve is the Hilbert curve [21]. Such space-filling curves have, for example, been proposed as paths for mine-clearing robots [30, 31]. Hilbert curves are fully specified by their order p. For a detailed description of the mathematics of Hilbert curves and properties of other space-filling curves, see [2, 9, 16].
- For sequential coverage path and path sampling, no algorithms focused on space-filling design exist. The ALBATROS algorithm presented in Sect. 3 aims to fill this gap.

3 Adaptive line-based sampling trajectories (ALBATROS)

3.1 Overview

The ALBATROS algorithm is a sequential line-based sampling algorithm. It is designed to gradually and sequentially extend the set of samples across the entire measurement space, taking into account the cost of moving the measurement sensor. The algorithm requires a (small) initial set of samples to be chosen before the sequential loop begins. These first samples can be determined by any classic DoE strategy. Once these initial samples have been gathered, the ALBATROS sequential loop, consisting of three phases, starts, as shown in Fig. 2.

The ALBATROS sequential loop starts with defining a new waypoint, which is the end-point of one complete path extension step. It is chosen at a location where the least amount of information has previously been collected. In the second step, the algorithm computes an efficient path towards that waypoint taking into account: (1) that movements are expensive and (2) that the device can gather extra samples along the way. Finally, once a path has been selected, the algorithm selects samples along the path. Depending on information density criteria, the algorithm can either stop at this point or it can continue the sequential loop by choosing additional waypoints.

Each of the four phases of the ALBATROS algorithm is now discussed in detail in the following parts. In the final

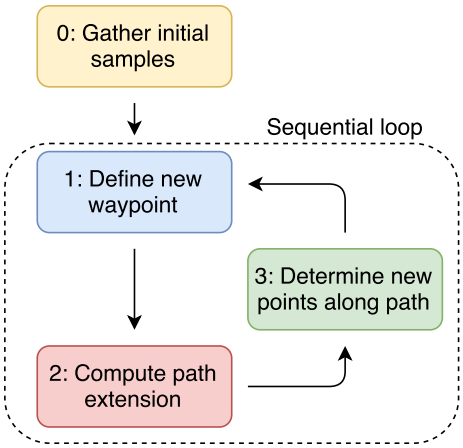


Fig. 2 Four phases of the ALBATROS algorithm

part of this section, the complete algorithm is shown in pseudocode with a detailed step-by-step description.

3.2 Phase 0: gather initial samples

The initial phase of the ALBATROS algorithm is the gathering of initial data samples. In the most basic form, this consists of at least two samples that are collected along a one-shot path that departs from the starting position of the measurement sensor.

However, in certain settings, the set of initial samples can be considerably larger. Consider, for example, a measurement device with a limited battery capacity. Given the design specifications, it is certain that the battery will last at least a specified amount of time. After that, the battery can fail at any moment. In such cases, it is useful to follow an optimal one-shot path (as discussed in Sect. 2) when the battery life is still guaranteed and move over to a sequential sampling strategy when the remaining battery life becomes uncertain.

The typical full one-shot path in a trapezoid measurement space consists of back and forward traversals on a grid known as a Boustrophedon path [8], as shown in Fig. 3a. Another possible initial path is the space-filling Hilbert curve [21], as shown in Fig. 3b.

The goal of these initial paths is to cover the measurement space as evenly as possible while minimizing the overall path length. Other initial paths are also possible. Depending on the use-case, secondary requirements are also important (e.g. minimizing the amount of turns in the path, etc.). In such a case, the Boustrophedon path is preferred over the Hilbert curve.

After the initial path is determined, sample locations are chosen along the path. In the most basic form, measurements are taken at fixed intervals. A more advanced method is to optimize their position along the coverage path. The density of the measurements can vary and has to be chosen by the operator. For a given density, the Boustrophedon path is the optimal shortest path to spread information across the measurement space [22]. The measurement device moves along the path and takes samples along equidistant steps.

3.3 Phase 1: defining new waypoint

After the initial samples have been gathered, the sequential sampling loop of the ALBATROS algorithm moves into phase 1. The first step in this loop is to define the new waypoint. This point should be chosen to maximally increase the amount of information in the measurement space. Hence, this point should be as far away as possible from any other point already gathered as initial samples or during previous iterations of the loop. This requirement can be formally defined as the maximin criterion ϕ over the input space $\mathcal{X}$

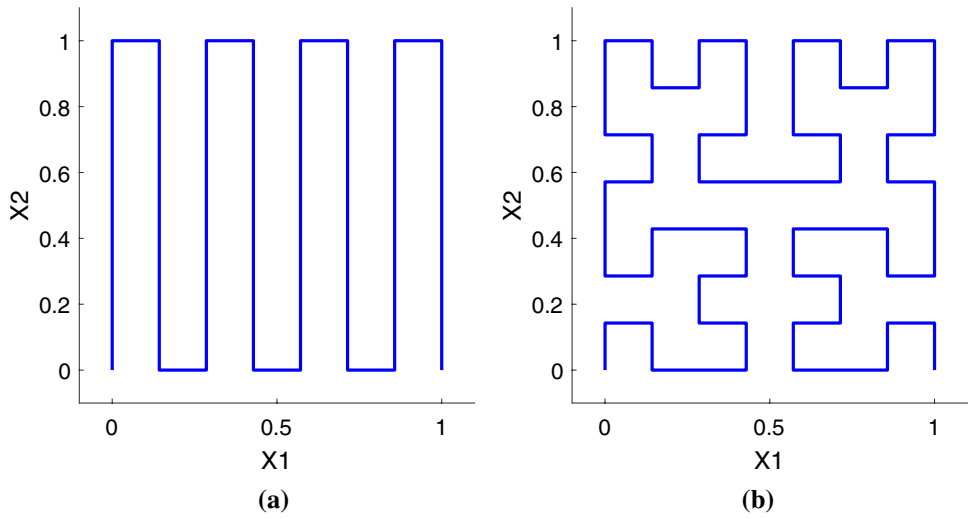


Fig. 3 Examples of initial paths. **a** Boustrophedon path. **b** Hilbert curve of order 3

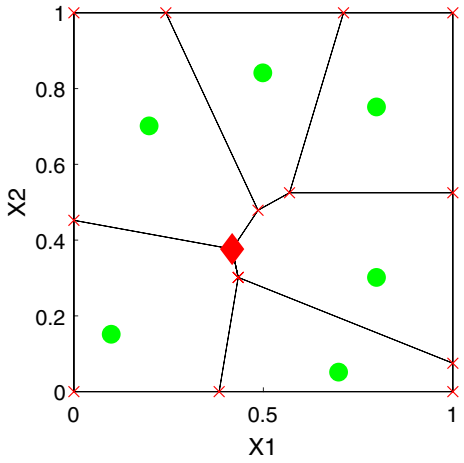


Fig. 4 Determination of maximin point (red diamond) in Voronoi diagram

[23], where X is the set of already gathered samples and $\mathbf{x}_i$ is the candidate point:

$$\phi = \max_{\mathbf{x}_i \in \mathcal{X}} \min_{\mathbf{x}_j \in X} \|\mathbf{x}_i - \mathbf{x}_j\| \tag{1}$$

If two or more candidate points result in the same maximin score, then the mean distance to all other points is used to break ties.

To determine the point in the measurement space with the largest maximin score, a bounded Voronoi diagram is constructed which will also be used in later steps of the algorithm. A Voronoi diagram of a set of points X consists of edges E and vertices V. The edges represent equidistant locations between the two nearest points. Each point is surrounded by multiple edges, composing a Voronoi cell. For details on the construction of a Voronoi diagram, the reader is referred to [1]. This diagram limits the search space for the largest maximin score to the vertices of the Voronoi diagram as these points are furthest away from any center point. This is illustrated in Fig. 4, where the green circles represent the Voronoi centers, the red crosses represent the Voronoi vertices and the red diamond represents the maximin point. The maximin point is determined as the vertex with the largest minimal distance to one of the Voronoi centers.

3.4 Phase 2: compute path extension

In the second step of the ALBATROS loop, a path is generated from the last sampled point of the previous iteration (or of the initial design), to the newly determined maximin waypoint. This is the coverage path planning step. The Voronoi diagram, which has already been constructed during the previous phase, can be re-used. When the Voronoi diagram is generated using all previously selected samples

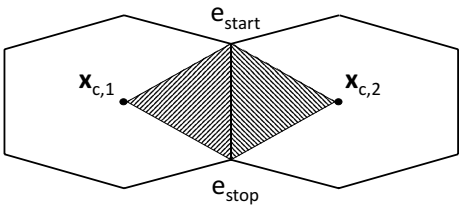


Fig. 5 ALBATROS edge weight metric

as cell centers, the Voronoi edges represent equidistant lines between the two nearest cell centers. Hence, traveling on these lines optimally spreads out the information gathering process. The Voronoi diagram in Fig. 4 can be interpreted as a graph consisting of the red crosses and black edges that connect them. Similar strategies have been used in, e.g., military obstacle avoidance settings [24] or optimal path planning settings [4].

Once the Voronoi diagram is generated, a weighted graph is constructed of the Voronoi diagram with the Voronoi vertices as graph nodes and the edges as graph edges. The weights of the edges are computed using a metric to balance out two conflicting goals. On the one hand, the path should be as short as possible. On the other hand, the path should visit many edges that are far away from previously sampled points represented as Voronoi centers. The weights of these edges are computed based on the distance of each point on the edge to the nearest Voronoi centers, which are on opposite sides of the edge. This can be interpreted as the sum of the integrals of the distance of each point on the edge to the Voronoi centers as shown by

$$\begin{aligned} w(e) &= \sum_{i=1}^{N_e} \int_{e_{\text{start}}}^{e_{\text{stop}}} ||\mathbf{x} - \mathbf{x}_{c,i}|| d\mathbf{x} \\ &= \sum_{i=1}^{N_e} \frac{||e_{\text{stop}} - e_{\text{start}}|| \times ||\mathbf{x}_{c,i,\text{projection}} - \mathbf{x}_{c,i}||}{2} \end{aligned} \tag{2}$$

where e_{start} and e_{stop} represent the beginning and end corner points of the Voronoi edge, respectively, N_e represents the number of points nearest to that edge, $\mathbf{x}_{c,i}$ represents the ith Voronoi center closest to the edge and $\mathbf{x}_{c,i,\text{projection}}$ represents the orthogonal projection of that point onto the edge. In the 2D case, N_e can either be 1 for edges on the boundaries or 2 for the other edges. Note that this metric can easily be computed as the surface of the triangles set up by the points e_{start}, e_{stop} and the $\mathbf{x}_{c,i}$ as illustrated in Fig. 5.

With this setup, the edges having the largest weights are the most beneficial for the space-filling properties to sample, because these edges are furthest away from any previously selected points. To find the optimal path, the shortest

path algorithm of Dijkstra [13] is used on the same graph but with the weights first converted by subtracting them from the maximal weight in the graph. Converting the weights in this way leads to a natural regularization on the number of hops, improving the performance of the algorithm.

3.5 Phase 3: determine new points along the path

In the final step, the path sampling algorithm will determine where the sensor should pause to take a measurement along the path that was determined using Dijkstra's algorithm. Again, the most basic form consists of samples taken equidistantly. The sample interval can also change dynamically across multiple iterations of the sequential loop.

In the ALBATROS algorithm, a heuristic is used to determine the location of the sample points. First, the number of points to be selected is computed based on the length of the selected path and the operator-determined proposed interval between the samples. However, instead of selecting the points at that fixed distance, the points are chosen to be furthest away from any other point. This is achieved by evaluating a large number of random points along the predetermined path and computing the distances of these points to all earlier selected sample points. The point with the largest distance is added to the new sample queue and the algorithm continues selecting new points until the maximum amount of points has been added. When the budget has been depleted, the points in the queue are ordered based on their position along the path to be traversed.

The measurement sensor can now gather these additional samples. The iterative sequential loop is then repeated until a stopping criterion is satisfied, such as, e.g., an information density metric. In other cases, the sequential loop continues until other physical constraints (such as time duration, traveled distance, battery depletion, or mechanical failure) have been reached.

3.6 ALBATROS algorithm

In the previous parts, an overview was given of each of the separate phases of the ALBATROS algorithm. Now, Algorithm 1 provides an overview of the pseudocode for ALBATROS.

- line 1–2: Generate initial points. The measurement sensor moves along the initial path and returns the measured samples.
- line 3: Check if a predetermined stopping criterion has been met. If not, the path is extended with additional points. If the criterion has been met, the algorithm ends.
- line 4–7: Generate a Voronoi diagram based on all sample points. Relevant variables from the Voronoi diagram are extracted.
- line 8–10: Determine the new waypoint and add an edge from the new waypoint to the closest Voronoi vertex.
- line 11–15: Edges from the current (= last) point to the vertices of the closest Voronoi edge are added as well. These added edges will allow a complete path to be formed.
- line 16–24: For each cell in the Voronoi diagram, the center point is computed. Then the influence, as determined by Equation 2, of that point to all its edges is added to the weights of the edges.
- line 25–28: Dijkstra solves a shortest path problem. As such, the weights of the weight matrix are converted to allow for a shortest path algorithm.
- line 29: The Dijkstra shortest path algorithm is used to determine an optimal path.
- line 30–33: The best points to sample along the path are determined and the measurement device is instructed where to execute the measurements. The outputs are added to an output array and the sampled points are added to the sample point array.

Algorithm 1 Adaptive Line-Based Sampling Trajectories

1: X = getInitialPoints()
2: Y = measure(X)
3: **while not** stoppingCriteriumMet(X,Y) **do**
4: voronoiDiagram = generateBoundedVoronoiDiagram(X)
5: E = voronoiDiagram.edges
6: V = voronoiDiagram.vertices
7: C = voronoiDiagram.cells
8: $\mathbf{x}_{new}$ = getMaximinPoint(X,V)
9: $\mathbf{x}_{new,closest}$ = closest point of X to $\mathbf{x}_{new}$
10: $E = E \cup (\mathbf{x}_{new}, \mathbf{x}_{new,closest})$
11: $n = |X|$
12: $\mathbf{x}_{current} = X(n)$
13: $\mathbf{a}, \mathbf{b}$ = vertices of edge in E closest to $\mathbf{x}_{current}$
14: $E = E \cup (\mathbf{x}_{current}, \mathbf{a})$
15: $E = E \cup (\mathbf{x}_{current}, \mathbf{b})$
16: **for all** $c \in C$ **do**
17: $\mathbf{x}_c$ = point in center of cell c
18: **for all** $e \in c$ **do**
19: $\mathbf{x}_{c,projection}$ = projection of $\mathbf{x}_c$ to edge e
20: projectionDistance = $||\mathbf{x}_{c,projection} - \mathbf{x}_c||$
21: edgeDistance = $||e_{start} - e_{stop}||$
22: $w(e) = w(e) + \frac{\text{projectionDistance} \times \text{edgeDistance}}{2}$
23: **end for**
24: **end for**
25: maxWeight = max(w)
26: **for all** $e \in E$ **do**
27: $w(e)$ = maxWeight - $w(e)$ + 1
28: **end for**
29: P = Dijkstra(w,$\mathbf{x}_{current}$,$\mathbf{x}_{new}$)
30: S = samplePath(P)
31: Y_{new} = measure(S)
32: $Y = Y \cup Y_{new}$
33: $X = X \cup S$
34: **end while**
35: **return** Y

4 Experimental setup

The algorithm is implemented in MATLAB®[1] using the Multi-Parametric Toolbox 3 [20] for computing the Voronoi diagram. ALBATROS is provided as an open-source toolbox on http://sumo.intec.ugent.be/ALBATROS.

Two measures are defined to test the performance of the algorithm on two evaluation criteria: the information gathered during the process has to be spread as uniformly as possible at each moment in time and the distance traveled l to capture a specific amount of information has to be minimized.

To quantify the spread of information in the measurement space, a concentration metric is defined in Eq. 3. The metric consists of the standard deviation of the d-dimensional volume $\mathrm{Vol}(C_{\mathbf{x}})$ of the d-dimensional Voronoi cells $C_{\mathbf{x}}$ induced by sample point $\mathbf{x}$:

$$\text{concentration} = \sqrt{\frac{1}{N-1}\sum_{\mathbf{x_i}\in X}\left(\mathrm{Vol}(C_{\mathbf{x_i}}) - \frac{1}{N}\sum_{\mathbf{x_j}\in X}\mathrm{Vol}(C_{\mathbf{x_j}})\right)^2} \tag{3}$$

5 Experiments

First, the ALBATROS algorithm is benchmarked in comparison with other DoE methods on a 2D space and two 3D spaces. Finally, the ALBATROS algorithm is applied to an engineering use-case of electro-magnetic compatibility testing.

5.1 2D comparison benchmark

The first experiment compares the ALBATROS sequential line-based algorithm to the three other categories from Sect. 2. Each of the algorithms is executed in a unit space $\mathcal{X} = [0, 1]^2$. As the algorithms are deterministic, no repetitions are required. Each time the concentration metric and path length l are computed for increasing subsets of gathered samples.

For the one-shot line-based benchmark, the Boustrophedon path is chosen. Along this path, samples are gathered equidistantly. Note that this one-shot line-based benchmark is a specific case of the one-shot point-based method, created by imposing a measurement order on the samples. Hence, we use this generalized experiment to represent both one-shot categories. The one-shot samples are sampled at a fixed distance of 0.1 in both directions. To demonstrate a path extension of the one-shot design, an extra Boustrophedon path is appended to the end of the sampling campaign with a fixed distance of 0.05 in the opposite way. For the sequential point-based algorithm, the maximin algorithm is chosen.

For the sequential line-based ALBATROS algorithm, two different configurations are possible. In the first configuration, the algorithm starts with only two initial points near each other, representing a cold start. In the second configuration, the ALBATROS algorithm continues after an upfront one-shot initial design representing the start of ALBATROS as an addition to another sampling algorithm. In both configurations, the proposed sampling distance is 0.1 as with the one-shot benchmark. In all benchmarks the starting point is [0, 0].

Figure 6 shows the initial steps of the ALBATROS algorithm, starting from a Hilbert Curve Design (HCD) of order 2. The blue lines represent the path, the black lines represent the Voronoi diagram, the green circles represent the previously sampled points which are now the Voronoi centers and the red diamonds represent the path extension. The figure demonstrates the ALBATROS algorithm gradually and uniformly increasing the sampling density. As the initial Hilbert curve already provides a rough outline of the measurement space, the ALBATROS algorithm proceeds to fill in the void in the middle.

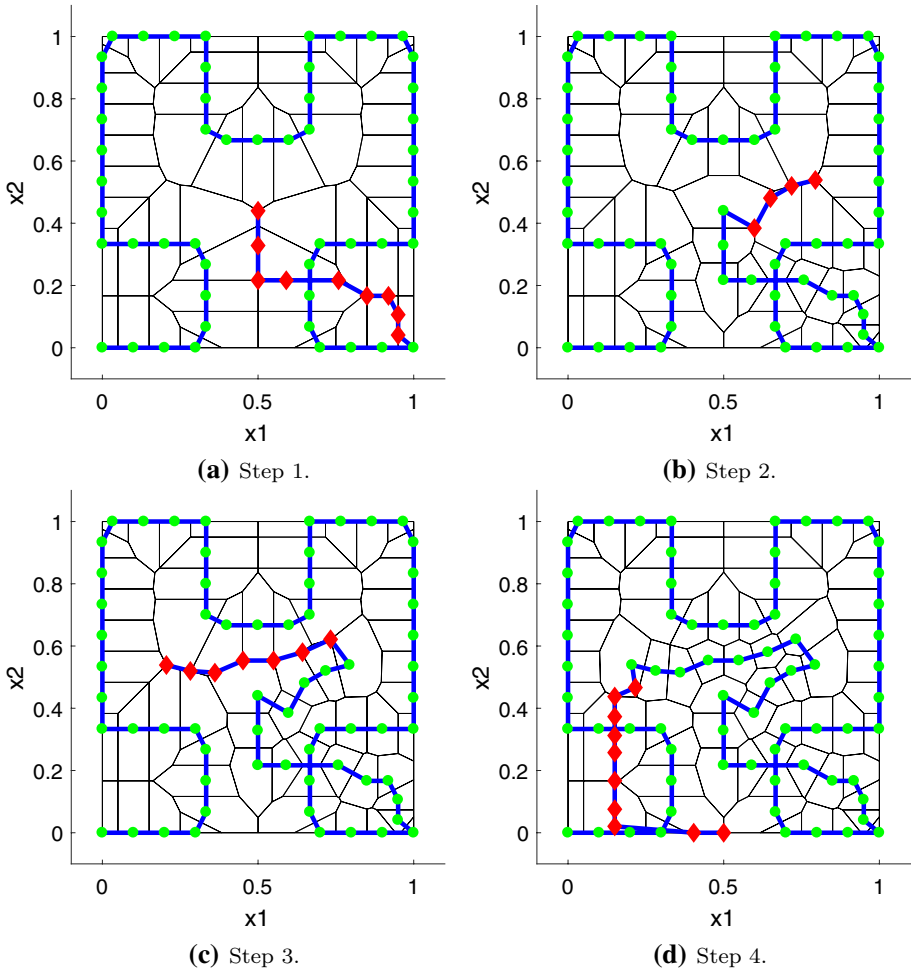
The ALBATROS algorithm can also be initialized starting from only two initial points. This is shown in Fig. 7. This figure demonstrates how the algorithm first gets a broad view of the space by sampling along the boundaries and then fills in the middle parts.

The final paths and sample positions after 256 sample points with each algorithm in the comparison are shown in Fig. 8. This demonstrates the structure in the measurements for the one-shot methods in Fig. 8a. Figure 8b shows how the sequential point-based sampling method continuously travels over the center of the measurement space. Finally, Fig. 8c, d demonstrate how the ALBATROS algorithm spreads out the information right from the start and avoids traversing over the same location multiple times.

In the following results, the ALBATROS algorithm starting from two initial points will be used for further illustration of the sequential line-based sampling algorithm. For all benchmarks, the concentration and path length l metrics discussed in Sect. 4 are computed, as shown in Fig. 9.

Figure 9a shows the concentration as a function of the number of samples. It is clear that the sequential point-based design is most spread if only the number of samples are considered. However, it is shown in Fig. 9b that the length of this corresponding path is substantially higher. When this is taken into account, it is seen from Fig. 9c that the ALBATROS sequential line-based design makes a better tradeoff than other approaches: the spread of measured samples is higher for any given path length. The one-shot design only provides a better spread of the data when the entire grid

[1] MATLAB, The MathWorks, Inc., Natick, Massachusetts, United States.



(a) Step 1. **(b)** Step 2. **(c)** Step 3. **(d)** Step 4.

Fig. 6 ALBATROS path extension steps, starting from HCD

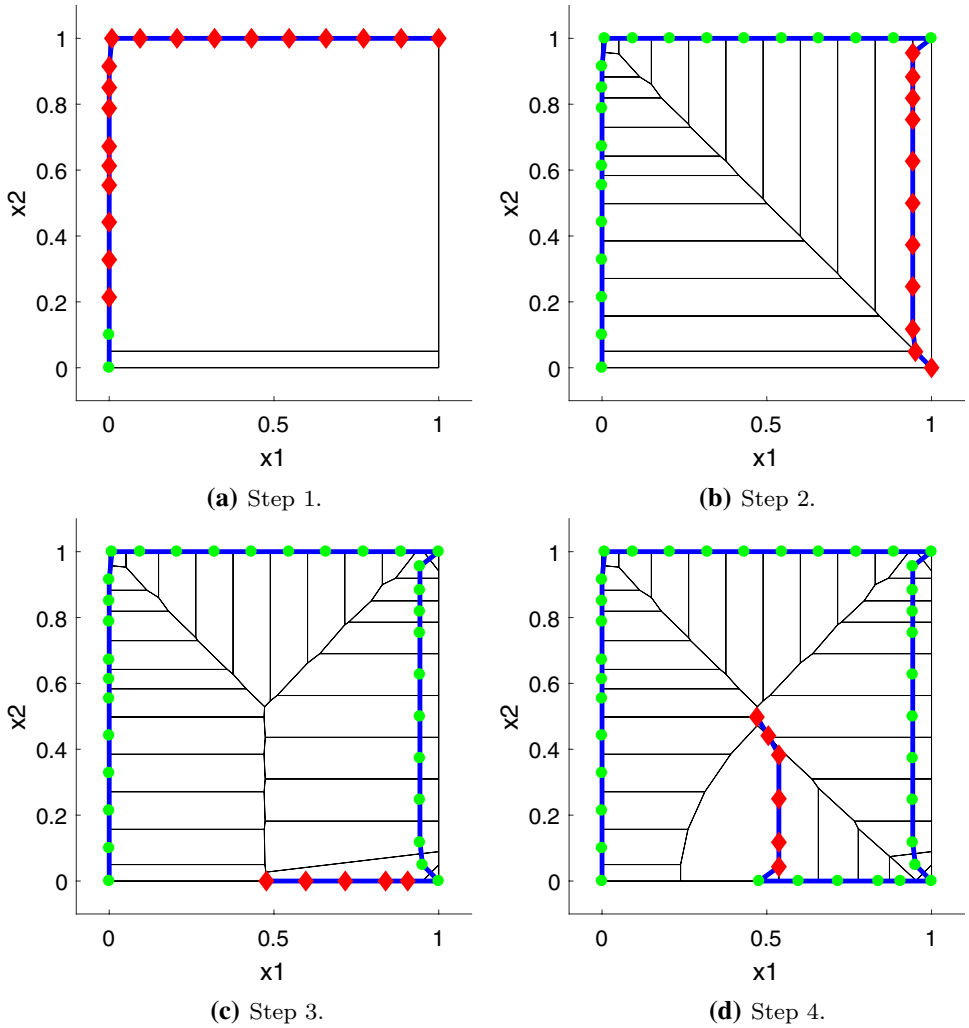
of exactly $11 \times 11 = 121$ samples is measured, which corresponds to a path length of 12.1.

From these graphs, the optimal working conditions for each of the algorithms can be derived. In case of fast movements of the measurement probe or in simulations, in which the path length l incurred by moving through the sampling space does not have to be taken into account, the sequential point-based strategy is the best choice as it provides the maximum amount of information spread out across the measurement space per sample. However, if on the other hand moving through the sampling space is costly and if we have a predefined sampling budget, the one-shot sampling techniques outperform the others as they provide the maximum amount of information for the shortest traveling path at the end of the measurement campaign. However, often, in real-life use-cases, the sampling budget is not fixed or guaranteed and the end of the measurement campaign cannot be predetermined. There are many possible reasons for the sampling budget to be cut short as outlined in Sect. 1. In such cases, the ALBATROS algorithm is the best choice as it efficiently collects information across the measurement space while simultaneously minimizing the path length. The ALBATROS algorithm can also be used when the required sampling budget for sufficient accuracy is difficult to estimate. The algorithm can continue sampling while accuracy metrics indicate if more samples are required.

5.2 3D comparison benchmark

A similar benchmark is performed on a 3D unit space $\mathcal{X} = [0, 1]^3$. For the one-shot line-based benchmark, a 3D Boustrophedon path is chosen with a sampling distance of 0.1. Again, this path is extended in the end with a Boustrophedon path in the opposite direction with a sampling distance of 0.05. This same design is also used to represent the one-shot point-based benchmark. For the sequential line-based benchmark, the ALBATROS algorithm starting from two initial points is chosen with a proposed sampling distance of 0.1. All benchmarks have [0, 0, 0] as starting location. Figure 10 shows the performance metrics.

The metrics again show how the sequential point-based design is the most optimal when compared to the amount



(a) Step 1. **(b)** Step 2.

(c) Step 3. **(d)** Step 4.

Fig. 7 ALBATROS path extension steps, starting from two points

of samples. However, the path length increases much faster than the two other designs. At 121 samples, the concentration metric drops significantly for the one-shot design as it has reached a $11 \times 11 \times 1$ grid in the 3D space. However, when it continues building a $11 \times 11 \times 11$ grid, the metric quickly rises again indicating that some points are more concentrated than others. The ALBATROS sequential point-based design on the other hand, keeps the concentration of points low and the spread high. When the path length that can be traversed is unknown or uncertain, the ALBATROS sequential point-based design is the most optimal choice.

The ALBATROS algorithm works in any 2D or 3D convex space. To demonstrate the algorithm on other shapes than the unit square and unit cube, a benchmark is performed on a 3D space determined by a Buckyball in $[-1, 1]^3$. The resulting evolution of the metrics is shown in Fig. 11. The metrics again show how the ALBATROS sequential line-based design is the most optimal choice when the path length is unknown or uncertain. An animation of the evolution of the sampling path is available at https://www.youtube.com/user/sumolab.

5.3 Use-case: electro-magnetic compatibility testing

To test the algorithm on an engineering problem, an electro-magnetic compatibility (EMC) testing use-case [12] is considered. In this problem, a device under test (DUT) is scanned using a near-field (NF) scanning probe to assess the electro-magnetic compatibility in the x-plane and y-plane. A top view of the DUT is shown in Fig. 12a and consists of a double microstrip bend over a slot. The NF scanning system consists of a computer numerical control milling machine that was rebuilt into an NF scanning system. For a detailed description of the setup, see [12].

In previous experiments, a one-shot dense grid consisting of 3375 scan points was collected of the magnetic

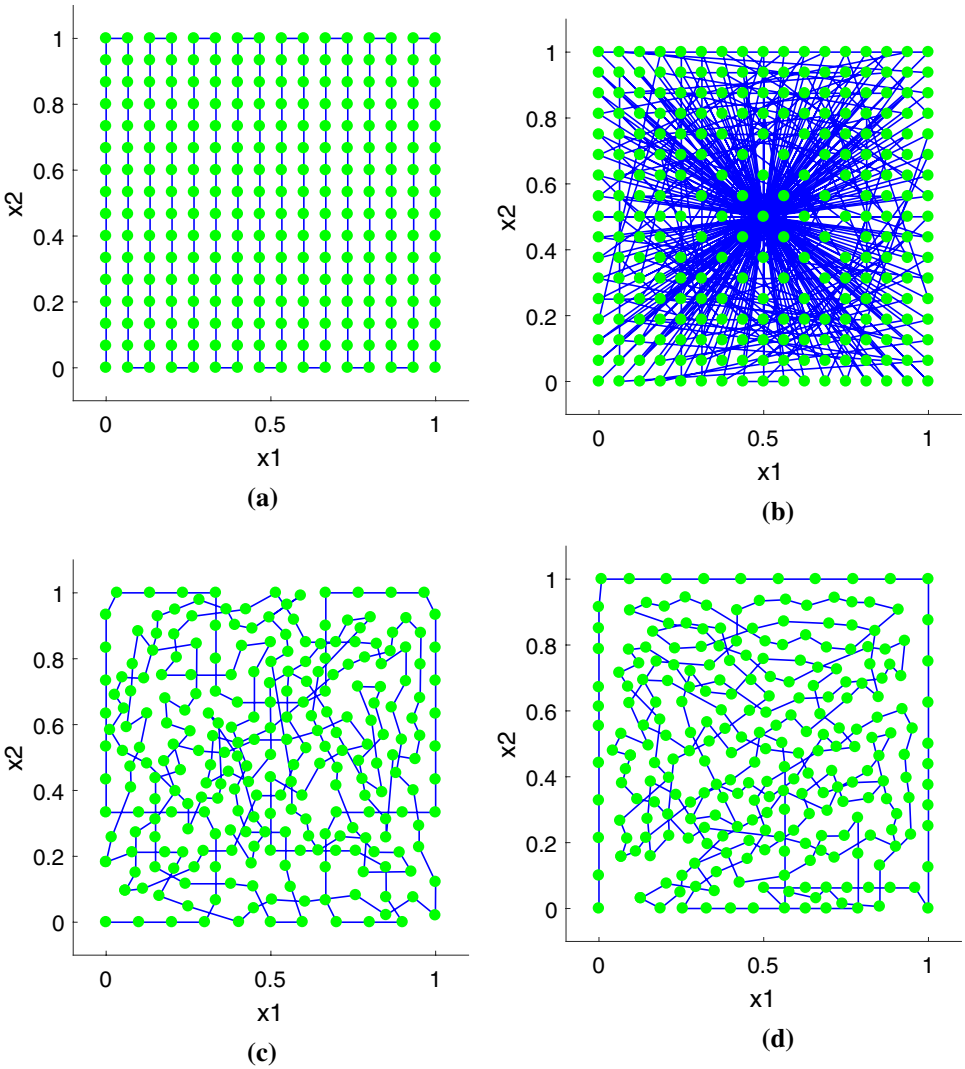


Fig. 8 Final sample path after 256 samples. **a** One-shot line-based and point-based sampling. **b** Sequential point-based sampling. **c** ALBATROS sampling with HCD. **d** ALBATROS sampling with two initial points

field in the x-plane $|H_x|$. This dense dataset, combined with linear interpolation is used in this example. The resulting magnetic field $|H_x|$ is shown in Fig. 12b. The measurement space is not a unit square but a rectangle shape of size $\mathcal{X} = [0, 75] \times [0, 39](cm \times cm)$.

For the one-shot line-based and one-shot point-based algorithms, a Boustrophedon path is chosen with a step size of 5. For the sequential point-based algorithm, the maximin algorithm is chosen. For the sequential line-based algorithm, ALBATROS is used with a proposed sampling distance of 5. All benchmarks have the point [0, 0] as starting point. The resulting metrics are shown in Fig. 13. The same remarks and conclusions can be made as with the other examples.

To evaluate the performance of the algorithms to accurately represent information about the measurement space, the resulting sample sets are compared with the ground truth data of our original dense dataset. To this end, the predicted values at the locations of the original dense grid are determined using nearest neighbor interpolation and nearest neighbor extrapolation. These predicted values $\hat{y}_i$ are then compared to the original values y_i in the root mean squared error (RMSE) metric in Fig. 14.

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2} \tag{4}$$

The RMSE metric in Fig. 14 behaves similar to the concentration metric in the other figures and examples. When only the number of samples is important and the traversed path length can be ignored, the sequential point-based design is the best choice. When a fixed sampling budget is known before, the one-shot design is the best choice, as shown in Fig. 14a. However, when movements through the measurement space are expensive and the path length is unknown or uncertain, the ALBATROS sequential line-based design proves to be the best choice, as shown in Fig. 14b, as it aims to spread out the

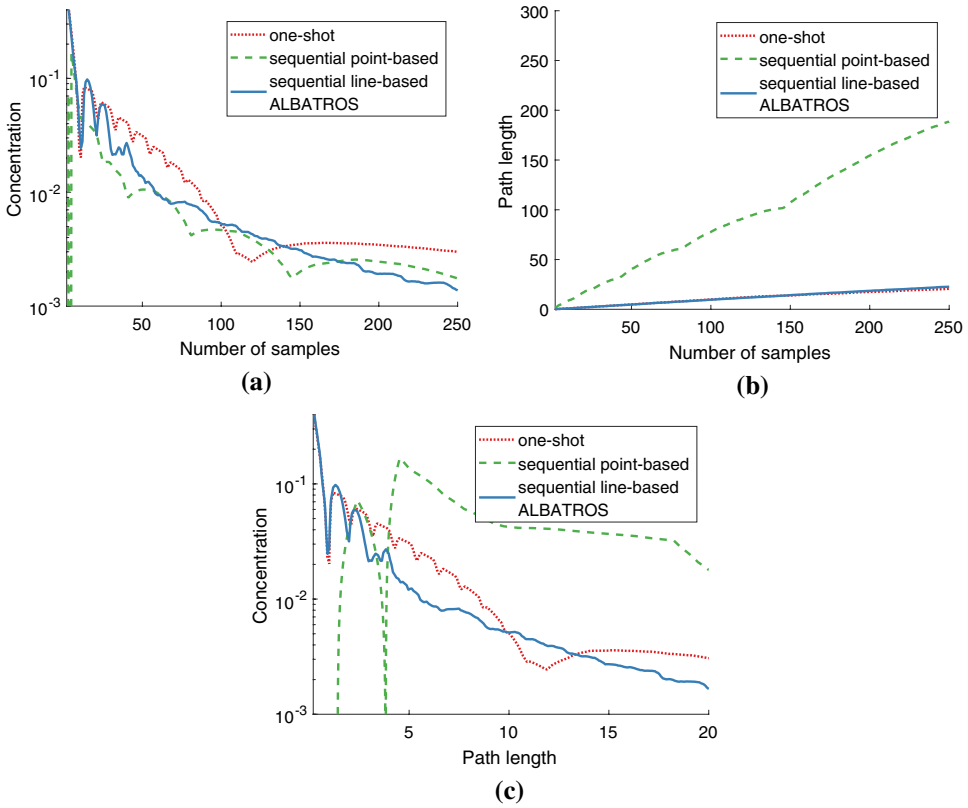


Fig. 9 Comparison of benchmarks in 2D unit square. **a** Concentration vs. number of samples. **b** Path length vs. number of samples. **c** Concentration vs. path length

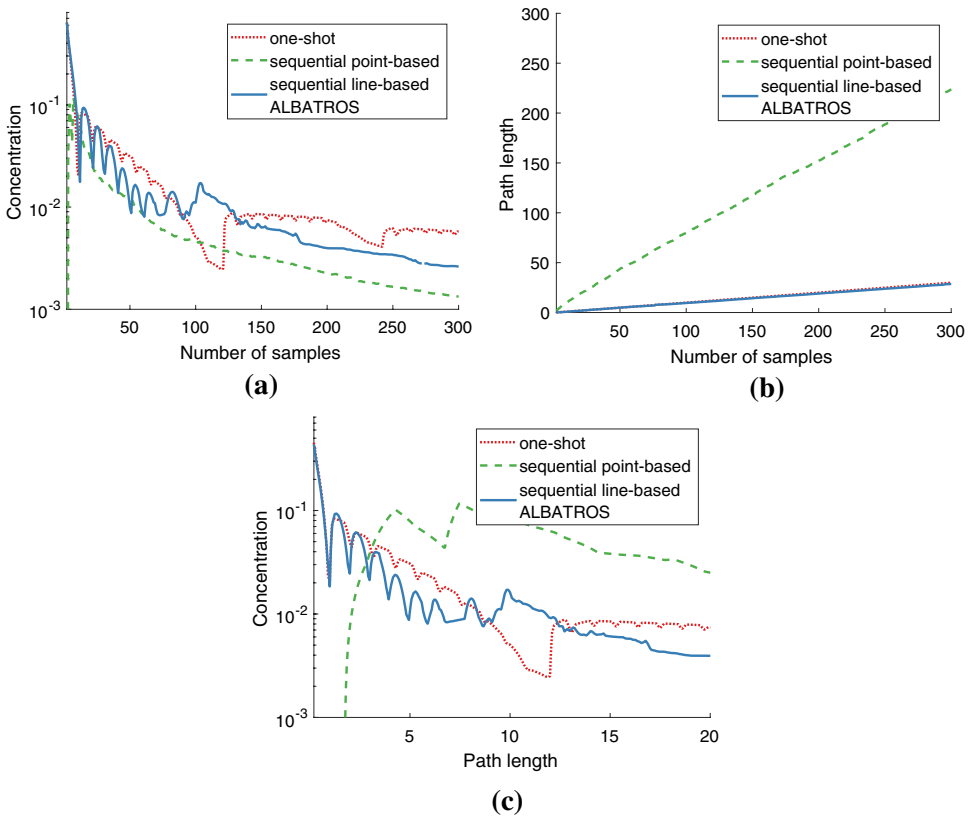
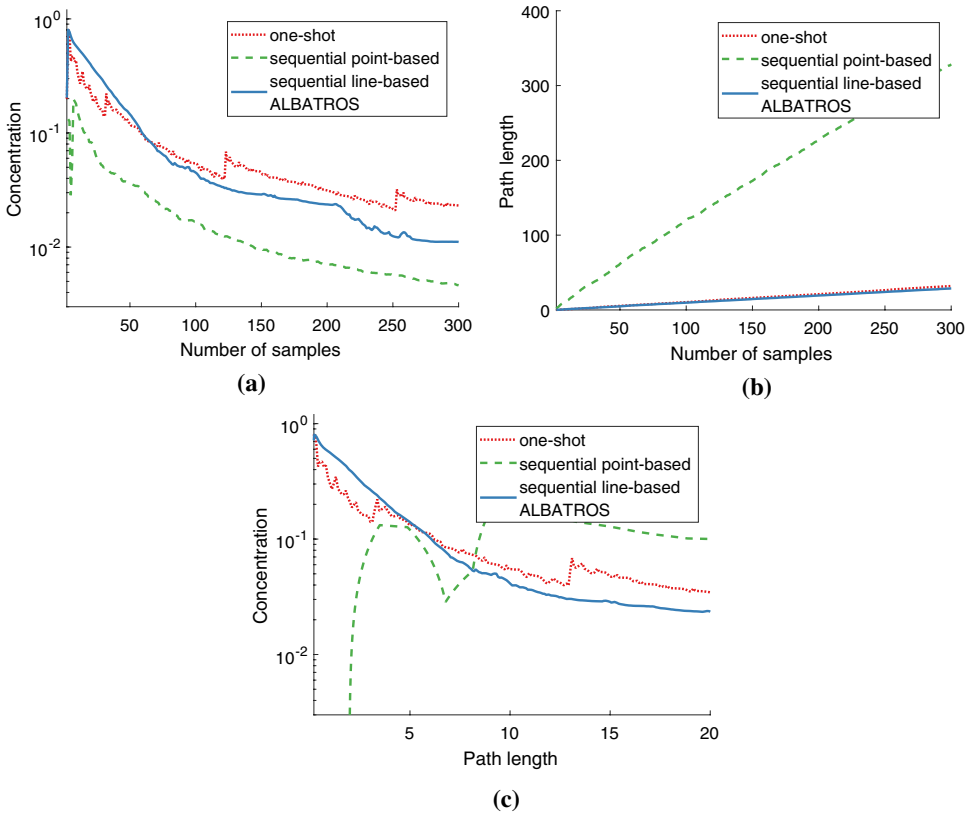


Fig. 10 Comparison of benchmarks in 3D unit cube. **a** Concentration vs. number of samples. **b** Path length vs. number of samples. **c** Concentration vs. path length



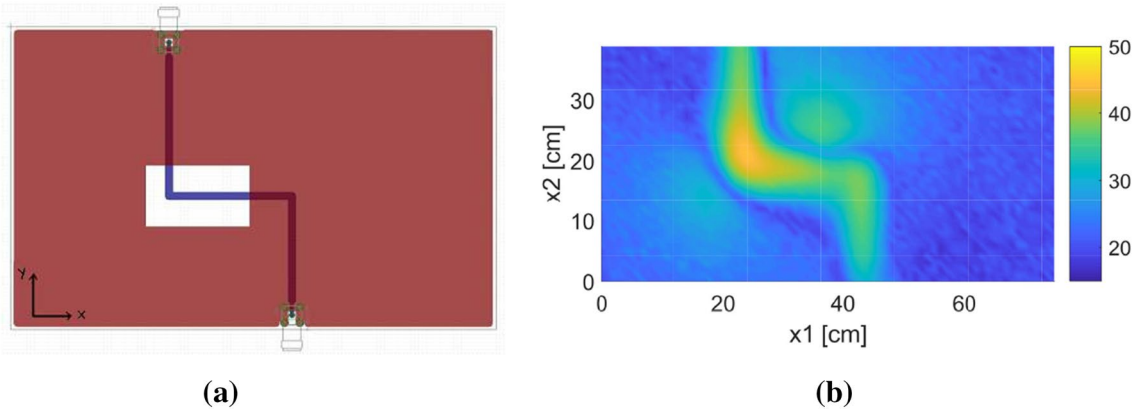


Fig. 11 Comparison of benchmarks in 3D Buckyball. **a** Concentration vs. number of samples. **b** Path length vs. number of samples. **c** Concentration vs. path length

Fig. 12 EMC testing use-case. **a** Device under test: bend microstrip over a slot. **b** Magnetic field $|H_x|$ of DUT

information throughout the measurement space, leading to a more accurate model representation of the data, while at the same time minimising the traversed path length.

6 Conclusion

The Adaptive Line-Based Sampling TRajectOrieS for Sequential Measurements algorithm was introduced. The ALBATROS algorithm is a sequential line-based sampling algorithm designed to gradually and uniformly increase the sampling density across the entire measurement space. It consists of a combined coverage path planning algorithm and path sampling algorithm. Although sequential point-based sampling methods provide the optimal spread of information in terms of the number of samples, these algorithms ignore the incurred costs of moving the sensor through the measurement space. One-shot sampling algorithms on the other hand optimize the path length but lack a gradual spread of information during the sampling process. When the sampling should be robust to sudden failures or when a predefined sampling budget is difficult to estimate upfront, the ALBATROS

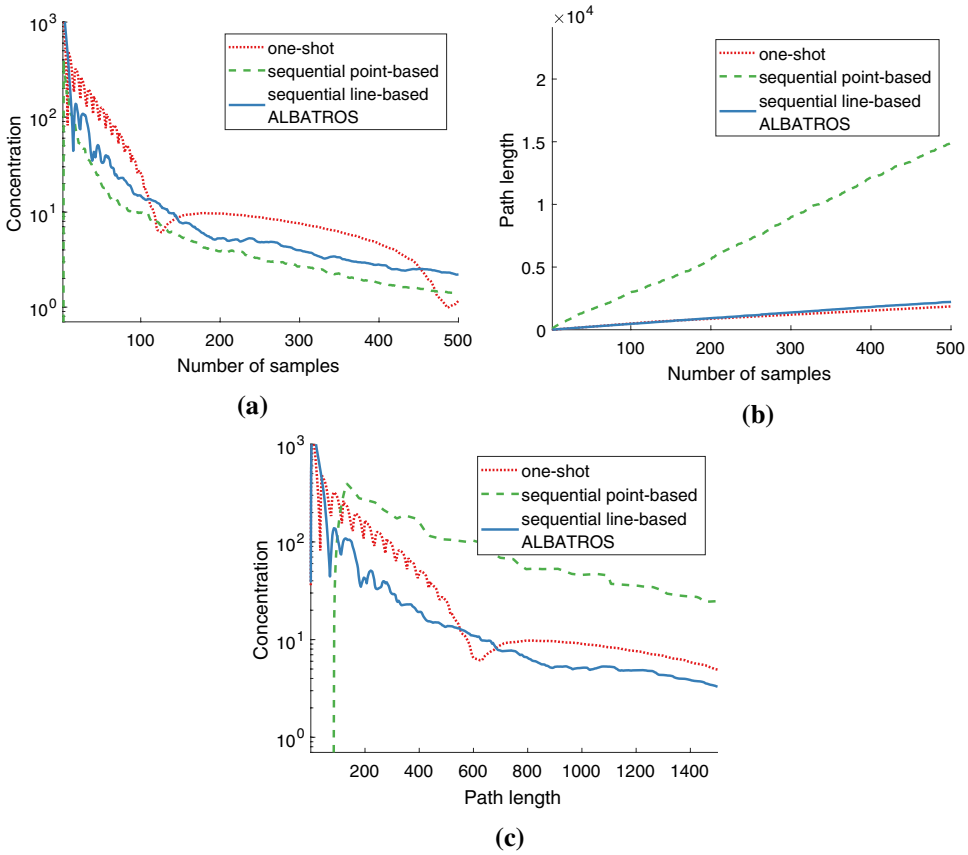


Fig. 13 Comparison of benchmarks in 2D EMC use-case. **a** Concentration vs. number of samples. **b** Path length vs. number of samples. **c** Concentration vs. path length

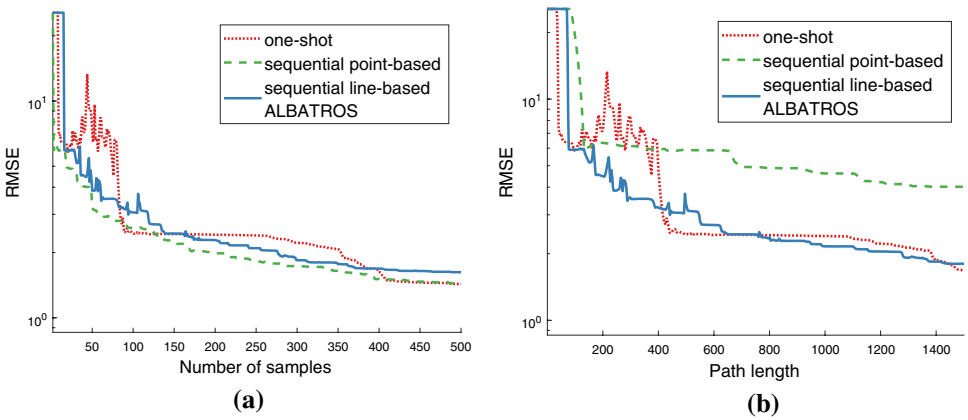


Fig. 14 RMSE comparison of benchmarks in 2D EMC use-case. **a** RMSE vs. number of samples. **b** RMSE vs. path length

algorithm provides a powerful sampling strategy in many real-life use-cases. The effectiveness of the ALBATROS algorithm has been demonstrated on several convex 2D and 3D spaces as well as on a practical engineering use-case.

References

1. Aurenhammer F (1991) Voronoi diagrams—a survey of a fundamental geometric data structure. ACM Comput. Surv. CSUR 23(3):345–405
2. Bader M (2012) Space-filling curves: an introduction with applications in scientific computing, vol 9. Springer, Berlin
3. Berni JA, Zarco-Tejada PJ, Suárez L, Fereres E (2009) Thermal and narrowband multispectral remote sensing for vegetation monitoring from an unmanned aerial vehicle. IEEE Trans Geosci Remote Sens 47(3):722–738
4. Bhattacharya P, Gavrilova ML (2007) Voronoi diagram in optimal path planning. In: Voronoi diagrams in science and engineering, 2007. ISVD'07. 4th international symposium, IEEE, pp 38–47
5. Broderick JA, Tilbury DM, Atkins EM (2014) Optimal coverage trajectories for a ugv with tradeoffs for energy and time. Auton Robots 36(3):257–271
6. Carlson J, Murphy RR (2005) How UGVs physically fail in the field. IEEE Trans Rob 21(3):423–437
7. Castro RM (2008) Active learning and adaptive sampling for non-parametric inference. Ph.D. thesis, Rice University

8. Choset H (2000) Coverage of known spaces: the boustrophedon cellular decomposition. Auton Robots 9(3):247–253
9. Choset H (2001) Coverage for robotics—a survey of recent results. Ann Math Artif Intell 31(1):113–126
10. Crombecq K, Laermans E, Dhaene T (2011) Efficient space-filling and non-collapsing sequential design strategies for simulation-based modeling. Eur J Oper Res 214(3):683–696
11. Deschrijver D, Crombecq K, Nguyen HM, Dhaene T (2011) Adaptive sampling algorithm for macromodeling of parameterized *s*-parameter responses. IEEE Trans Microw Theory Tech 59(1):39–45
12. Deschrijver D, Vanhee F, Pissoort D, Dhaene T (2012) Automated near-field scanning algorithm for the emc analysis of electronic devices. IEEE Trans Electromagn Compat 54(3):502–510
13. Dijkstra EW (1959) A note on two problems in connexion with graphs. Numer Math 1(1):269–271
14. Martínez-de Dios J, Merino L, Caballero F, Ollero A, Viegas D (2006) Experimental results of automatic fire detection and monitoring with UAVs. For Ecol Manag 234(1):S232
15. Gademer A, Mainfroy F, Beaudoin L, Avanthey L, Germain V, Chéron C, Monat S, Rudant JP (2009) Solutions for near real time cartography from a mini-quadrotor UAV. In: SPIE Europe remote sensing, International Society for Optics and Photonics, p 74781G
16. Galceran E, Carreras M (2013) A survey on coverage path planning for robotics. Robot Auton Syst 61(12):1258–1276
17. Gorissen D, Couckuyt I, Demeester P, Dhaene T, Crombecq K (2010) A surrogate modeling and adaptive sampling toolbox for computer based design. J Mach Learn Res 11(Jul):2051–2055
18. Gorissen D, Couckuyt I, Laermans E, Dhaene T (2010) Multi-objective global surrogate modeling, dealing with the 5-percent problem. Eng Comput 26(1):81–98
19. Han J, Xu Y, Di L, Chen Y (2013) Low-cost multi-UAV technologies for contour mapping of nuclear radiation field. J Intell Robot Syst 70(1–4):401–410
20. Herceg M, Kvasnica M, Jones C, Morari M (2013) Multi-parametric toolbox 3.0. In: Proc. of the European control conference, Zurich, pp 502–510. http://control.ee.ethz.ch/~mpt. Accessed 7 Nov 2017
21. Hilbert D (1891) Ueber die stetige abbildung einer line auf ein flächenstück. Math Ann 38(3):459–460
22. Huang WH (2001) Optimal line-sweep-based decompositions for coverage algorithms. In: IEEE international conference on robotics and automation, 2001. Proceedings 2001 ICRA, vol 1. IEEE, pp 27–32
23. Johnson ME, Moore LM, Ylvisaker D (1990) Minimax and maximin distance designs. J Stat Plan Infer 26(2):131–148
24. Judd K, McLain T (2001) Spline based path planning for unmanned air vehicles. In: AIAA guidance, navigation, and control conference and exhibit, Montreal, Canada, p 4238
25. Kennard RW, Stone LA (1969) Computer aided design of experiments. Technometrics 11(1):137–148
26. King DW, Bertapelle A, Moses C (2005) UAV failure rate criteria for equivalent level of safety. In: International helicopter safety symposium, Montreal, p 9
27. Kleijnen JP (2008) Design and analysis of simulation experiments, vol 20. Springer, Berlin
28. Montgomery DC (2017) Design and analysis of experiments. Wiley, Hoboken
29. Nex F, Remondino F (2014) Uav for 3d mapping applications: a review. Appl. Geomat. 6(1):1–15
30. Spires S, Goldsmith S (1998) Exhaustive geographic search with mobile robots along space-filling curves. In: Drogoul A, Tambe M, Fukuda T (eds) Collective robotics. Springer, Berlin, Heidelberg, pp 1–12
31. Spires SV (2003) Exhaustive search system and method using space-filling curves. US Patent 6,636,847
32. Valente J, Sanz D, Del Cerro J, Barrientos A, de Frutos MÁ (2013) Near-optimal coverage trajectories for image mosaicing using a mini quad-rotor over irregular-shaped fields. Precis Agric 14(1):115–132
33. Viana FA, Venter G, Balabanov V (2010) An algorithm for fast optimal latin hypercube design of experiments. Int J Numer Meth Eng 82(2):135–156