Collective Sampling of Environmental Features under Limited Sampling Budget

Yara Khaluf*, Pieter Simoens

IDLab, Department of Information Technology, Ghent University - imec
Technologiepark 126, B-9052 Gent, Belgium

Abstract

Exploration of an unknown environment is one of the most prominent tasks for multi-robot systems. In this paper, we focus on the specific problem of how a swarm of simulated robots can collectively sample a particular environment feature. We propose an energy-efficient approach for collective sampling, in which we aim to optimize the statistical quality of the collective sample while each robot is restricted in the number of samples it can take. The individual decision to sample or discard a detected item is performed using a voting process, in which robots vote to converge to the collective sample that reflects best the inter-sample distances. These distances are exchanged in the local neighbourhood of the robot. We validate our approach using physics-based simulations in a 2D environment. Our results show that the proposed approach succeeds in maximizing the spatial coverage of the collective sample, while minimizing the number of taken samples.

Keywords: Swarm robotics, Spatial sampling, Collective behavior, Collective decision-making, Environment sampling

*Corresponding author
Email addresses: yara.khaluf@ugent.be (Yara Khaluf), pieter.simoens@ugent.be (Pieter Simoens)

1. Introduction

Robot swarms are gaining importance as the scope of their applications is getting wider [1, 2, 3, 4, 5]. Different from other types of multi-robot systems, in swarm robotics no central coordination or knowledge is assumed. Instead, robots only execute a set of simple behavioral rules and communicate with their local neighbors [6]. Impressive collective behaviors can emerge from these simple rules, while the solution stays fault-tolerant and scalable [7, 8, 9]. Due to the large number of robots in a swarm, the system can keep functioning even when a few individuals get damaged. Scalability also results from the fact that robots only exploit information from their local neighbourhood.

One of the key tasks, in which swarm robotics offers a potentially cost-efficient and robust solution is analyzing and mapping of large environments [10], in which they can cover large areas within limited time periods. Environment analysis is a fundamental task for different applications. For instance, in the agriculture domain, environment analysis may be used to map soil quality. [11], A robot swarm robot can be used to build a spatial sample of the distribution of some plant features (e.g. leave color) across a large field, which then can be used as an indicator for particular crops.

In this paper, we tackle the problem of gathering information about the spatial distribution of a specific environmental feature using a simulated swarm of robots[1]. Robots, in this study, operate as mobile sensors, which perform a random walk and decide autonomously on whether to discard or to sample and upload the locations of detected items to a central system for statistical analysis. The information gathered by the swarm is referred to as the *collective sample*. Note that the central system is used only for the statistical analysis of the collective sample, but has no impact on the individual robot's decision-making process nor on the constitution of the collective sample that is generated by the swarm.

[1] For simplicity, we will refer to the simulated robot as robot throughout the paper.

Real robots are associated with limited on-board batteries and hence the number of energy-expensive operations such as a (wireless) upload need to be limited. We, therefore, address the problem of collective sampling under the constraint of a *limited sampling budget* (LSB). Our optimization goal focuses on the cost associated with uploading information about the sampled data items rather than the cost associated of the robot's travel. In particular, we aim to design a behavioral decision model for the individual robot that results in a collective sample of maximal statistical quality with a minimum number of samples. Neither the actual locations nor the spatial distribution from which these locations were sampled are known by the robots. This makes the generation of a high-quality sample that covers all regions a challenging task. In this paper, we study the most stringent LSB: we allow each robot to upload only one item.

Similar to any statistical sampling, collective sampling needs to maximize the coverage of the problem space (i.e. a 2D physical environment), so that the statistical distance between the actual item distribution and the distribution estimated from the collective sample is minimized. Increasing the number of uploads improves the statistical quality of the sample but needs to be traded off with the LSB constraint. Our approach relies on covering the largest set of inter-sample distances by adopting a *local voting mechanism* that allows to collectively decide which robots will upload the location of a detected item. Inter-sample distances represent a key parameter in several sampling applications, e.g., the analysis of the T-Cell Receptor Repertoires [12], or gene expressions [13].

The rest of the paper is organized as follows. In Section 2 we describe the problem of collective sampling using a homogeneous swarm of robots under the constraint of limited sampling budget (LSB) and the performance measure we use to evaluate our results. The behavior of the individual robot is presented in Section 3, in which we propose a novel approach for efficient exploration and information exploitation in sampling unknown environments. Our experimental configurations are described in Section 4, and the results are discussed in Section 5. We conclude our paper in Section 6.

2. Problem Description

We consider a system of N homogeneous robots that explore a large and unknown 2D environment to sample a particular feature, denoted by Ω. The feature Ω is discrete, thus, consists of a finite number of items M, which are scattered across the environment following a particular spatial distribution $P(\mathbf{x})$. We define Ω as a static feature, i.e. it doesn't change over time in any of its properties such as its spatial distribution, quality level, or others. When a robot encounters a sample of Ω, it might decide to upload the spatial coordinates of its current location $\mathbf{x}$. The upload decision is governed by one of the behaviours defined in the next section. The collective sample uploaded by the robot swarm is denoted by $\mathbf{S_{coll}}$.

$$\mathbf{S_{coll}} = \{\mathbf{x_K}\} \qquad K = \|\mathbf{S_{coll}}\|; K \leq min(M, N) \tag{1}$$

where, $\mathbf{x_K}$ is the coordinate vector of the Kth uploaded item of the feature Ω and we impose that each robot cannot upload more than one item. Please note that allowing the robot to sample more than one item can only improve the statistical quality of the collective sample. Hence, we have selected the most challenging setting by limiting the number of samples to one per robot and benefit from this condition on obtaining an energy-efficient approach in terms of the uploading process.

The collective sample $\mathbf{S_{coll}}$ is used to estimate the parameters of the spatial distribution of the feature Ω. In particular, $\mathbf{S_{coll}}$ is used to estimate the mean $\boldsymbol{\mu_g}$ and (diagonal) co-variance $\boldsymbol{\Sigma_g}$ of a multi-modal bivariate Gaussian distribution with G modes:

$$P\left(\mathbf{x}|\left\{\boldsymbol{\mu_g}, \boldsymbol{\Sigma_g}\right\}\right) = \frac{1}{G}\sum_{g=1}^{G}\mathcal{N}\left(\boldsymbol{\mu_g}, \boldsymbol{\Sigma_g}\right) \tag{2}$$

In this paper, we will restrict the analysis to $G = 1$ and $G = 2$ and diagonal co-variance $\boldsymbol{\Sigma_g}$. The main goal of our study is to produce a collective sample $\mathbf{S_{coll}}$ that allows the most accurate estimation of the feature distribution, while minimizing the total number of uploads. We define the following measures to evaluate the efficiency of the collective sampling process at the swarm level:

- **The statistical quality of the collective sample $\mathbf{S_{coll}}$**: the uploaded samples are a subset of the M items ($M \sim P(\mathbf{x})$), which are distributed over the environment. In order to evaluate the statistical correctness of the distribution $Q(\mathbf{x})$ estimated from the collective sample $\mathbf{S_{coll}}$, we use the Kullback-Leibler (KL-) divergence [14]. KL-divergence is a well-known measure in information theory [15, 16] that is given by:

$$D_{KL}(P||Q) = \int_{-\infty}^{\infty} p(x) log \frac{p(x)}{q(x)} dx \tag{3}$$

where P is the actual spatial distribution of the data items and Q is the distribution that is estimated based on the sample uploaded by the robots. The distance is a non-negative measure that is zero when the two distributions are identical.

- **The upload percentage:** is a global measure that emerges from the autonomous decisions of the individuals whether to upload or to discard items, and it is computed as follows:

$$\delta_{\mathbf{S_{coll}}} = \frac{K}{M} \tag{4}$$

where K is the number of uploads, and M is the total number of items of feature Ω scattered over the environment ($K \leq min(M, N)$). The M items in the environment are a sample of the original distribution P. Therefore, the best estimate Q that the swarm can possibly obtain, is by sampling all of the M items. The upload percentage is used to indicate the percentage of knowledge obtained about the feature by reporting the number of uploaded items over the total number of items. This metric can be used as an indicator of the efficiency of the swarm performance in terms of energy. Since in practical applications, such uploading operations are generally expensive in terms of energy, limiting the number of uploads is a desired energy goal.

3. The individual uploading behavior

In our proposed approach, to make an upload/discard decision, each robot exploits the limited information sensed while exploring the environment, in addition to the information shared by its local neighborhood. The local neighborhood includes all robots that are within the communication range and are in line-of-sight. The upload decision is made probabilistically, since no individual has a complete knowledge of the environment nor of its current conditions. In our study, robots have no knowledge on the parameters of the spatial distribution and on the size of the environment. Yet, the collective sample S_{coll} needs to (i) provide the widest possible spatial coverage of sample points, and (ii) reflect the distribution of inter-sample distances. In the following, we propose a Collective Sampling Controller (CSC) that aims to achieve these requirements under a Limited Sampling Budget (LSB) of one item per robot. We also present two simpler variants of this controller that we will use as benchmarks in our experiments

Collective Sampling Controller (CSC): The CSC operates in three phases:

1. *The exploration phase*: in which robots explore the environment for a period of δ_e using a diffusion behavior that allows the swarm to maximize its coverage. To help first reaching a maximum coverage, robots are not allowed to stop on data items in case they detect any during this phase. The diffusion behavior is inspired by diffusion models of gas particles, in which the particles move from spaces with high concentration to spaces with low concentration and hence tend to fill the whole space [17]. In our algorithm, obstacles are understood as gas particles and the robot tends to move away towards spaces with lower concentrations. This tendency results in two outputs: obstacle avoidance and maximum coverage. We implement the diffusion behavior as follows. In each simulation step, the robot accumulates the vectors extracted from the readings of its proximity

sensors[2]. Each reading has a value and an angle to indicate both the relative distance and the relative angle to the obstacle perceived through that sensor. The accumulated vector is used as an indicator towards the most free direction—i.e. the space with lowest concentration. Different from localized motion models such as Brownian motion [18], we preserve the same direction and move in a straight line when no obstacles (or other robots) are sensed. This allows the robots to diffuse and increase the swarm exploration coverage.

Pseudo-code of the behavior in this phase is shown in Alg. 1.

Algorithm 1: The algorithm followed by the individual robots in our physics-based simulations to explore (diffuse in) their environment.

```
initialize the accumulated proximity vector v = (x = 0, y = 0);
for i=1, i ≤ 24 do
    x = sensor(i).value * cos(sensor(i).angle);
    y = sensor(i).value * sin(sensor(i).angle);
    v.x = v.x + x;
    v.y = v.y + y;
angle = atan2(v.y, v.x);
length = sqrt(v.x*v.x + v.y*v.y) / 24;
if length < threshold then
    keep moving straight with linear speed 5 m/sec;
    else
        turn towards the free direction using the accumulated proximity
         vector v;
```

2. *The detection phase*: robots use this phase to select the data items, for which they will make the decision to upload or discard. After the exploration phase, robots are assumed to have achieved the maximum possible

[2]The robot used in our study has 24 proximity sensors. Details are provided in the next section.

coverage, and therefore are ready to start marking the nearest items for potential upload. This phase lasts for a period of δ_d and in this phase robots continue applying the diffusion behavior that enables obstacle avoidance and spreading-out but they are now allowed to stop on data items.

3. *The exploitation phase*: robots which have detected data items in the previous phase, will take in this phase an individual decision to upload or discard the location $\mathbf{x}$ of its detected item. This decision is taken after a local voting procedure—i.e. across the robot's neighborhood.

   The voting procedure is illustrated in Fig. 1. First, each robot i that has detected a data item computes its distance d_{ij} to all its local neighbors j —i.e. robots within its communication range—that have detected data items as well. We will explain in Sec. 4 how such relative distances are computed in our physics-based simulations.

   As mentioned above, one of the design goals of the voting process is to maximize the coverage over the inter-sample distances. This is achieved by allowing each robot to *uniformly* sample $D \leq ||N_i||$ ($||N_i||$ is the number of robots in the local neighborhood of robot i) values d_x within the range $[min(d_{ij}), max(d_{ij})]$, where d_{ij} denotes the set of all distances between robot i and its neighbours j.

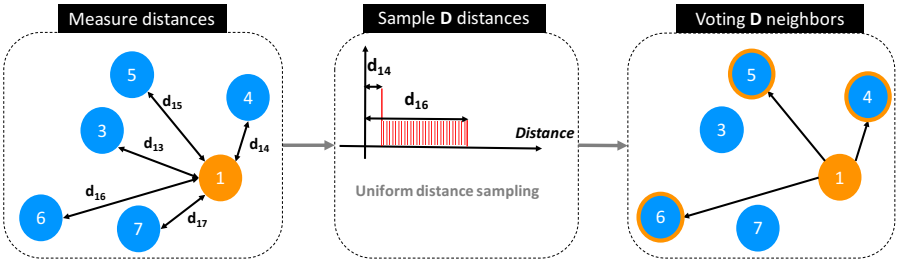


Figure 1: The voting mechanism in the CSC controller. Each robot samples uniformly from the range of distances computed with neighbours, who have detected an item. The robot votes for the neighbours whose distances are the closest match to the sampled distances.

Each robot will now vote for the D neighbors of which the actual distance d_{ij} is most close to one of the values d_x that were uniformly sampled. In particular, each sampled distance d_x is mapped to the neighbor j from the neighborhood N_i of robot i as follows:

$$Map(d_x) = j, \text{ where } \quad |d_x - d_{ij}| < |d_x - d_{ir}| \quad \forall r \in N_i, r \neq j, \qquad (5)$$

By uniformly sampling from the interval of actual distances, all inter-item distances have the same probability to be included in the robot's sample (and hence to be a potential uploaded). Since every robot can upload one data item at most, these sampling and voting mechanisms allow the uploaded data item locations to have a higher chance to represents all inter-sample distances.

The robot sends its votes to its selected neighbors. Upon receiving a vote, the receiver increases its tendency to upload the location of the data item it detected (and stopped on), see Fig. 1.

When the number of votes received by a robot is higher than a predefined threshold ψ, the robot decides to upload, otherwise it doesn't. This mechanism is inspired by the Response Threshold Model (RTM), a well-known approach in robot swarms [19]. In this model, a robot will decide to switch from its current option A to option B if a stimulus value crosses a particular threshold (and vice versa). In our specific case, option A is to not upload the detected item, option B is to upload it, and the stimulus is the number of votes received by the robot. When the number of votes exceeds the threshold, the robot uploads. One of the main challenges in applying RTM is to properly set the threshold. In many studies this threshold is defined to be static. The threshold value results from a series of simulations and fine-tuning processes. Whereas in some other studies the threshold is dynamic and is adapted to the dynamics of the task environment. In our study, we use a static threshold ψ, the value of which we have calibrated using a set of initial simulations. The pseudo code of the CSC is given in

Alg. 2:

Algorithm 2: The algorithm followed by the individual robots in our physics-based simulations to generate the collective sample.

initialize parameters;
while *current_time* $\leq \delta_e$ **do**
    Explore the environment using
    a) Diffusion random walks;
    b) Obstacle avoidance;
while *(current_time* $> \delta_e$*) and (current_time* $\leq \delta_e + \delta_d$*)* **do**
    Seek data items using
    a) Diffusion random walks;
    b) Obstacle avoidance;
    if *a data item is detected* **then**
        Mark data item (stop on it);
        Broadcast a notification to the neighborhood;
if *robot i has detected an item* **then**
    Compute Euclidean distances to all neighbors d_{ij} $\forall j \in N_i$, where N_i is the set of neighbors of robot i, which have detected data items;
    Sample D distances $d_x \sim U\ [min\left(\{d_{ij}\}\right), max(\{d_{ij}\})]$;
    Map each d_x to a neighbor of robot i using Eq. (5);
    Send votes to all selected neighbors;
    Count number of votes from neighbours;
    if *number of votes* $> \psi$ **then**
        Upload data item;

Finally, in case of generalizing our algorithm to allow the robot to sample more than one data item (as mentioned above, this can only improve the statistical quality of the collective sample), the algorithm will iterate over the second and third phases, since the exploration phase is needed only once. Furthermore, our algorithm can be extended easily to 3D environments, because the robot's

upload/discard decisions are taken based on local interactions. Surely, when dealing with 3D environments, the robot's random walk behavior needs to be adapted; in particular, the sampling process of the random angle which the robot uses when performing obstacle avoidance.

3.1. Benchmark

In order to assess the importance of the exploration and exploitation phases, we also define two simpler controllers that we will use as benchmark to evaluate the performance of the CSC controller.

Always-Uploading Controller (AUC): no exploration phase is used. Robots perform a random walk using the diffusion process explained above and upload the first item detected. No information is exchanged with the robot's neighborhood.

The AUC may be useful for tasks in which a minimum amount of data items needs to be uploaded within a specific deadline. The downside of this controller is that most of the robots will upload at the region with a high number of data items that is nearest to their starting location. Under the constraints of LSB and swarm size N, the statistical quality of the collective sample S_{coll} is likely to drop, since the robots will concentrate their uploads at specific spots, rather than enlarging the coverage of their collective sample.

Blocked-by-Neighbor Controller (BNC): this controller exploits the neighborhood's information to maximize the spatial coverage of the collective sample. A robot that detects an item will send a blocking signal to its local neighborhood. A robot will only upload an item if it is not in the local neighborhood of an uploading robot, otherwise, it will continue exploring the environment further. By blocking uploads in the local neighborhood, we reinforce exploration. BNC maximizes the spatial coverage of the collective sample S_{coll} under two limitations (i) the swarm size N and (ii) the duration of the experiment. When the average distance between data items is smaller than the robot's communication range, BNC fails to maximize the spatial coverage of the collective sample S_{coll}, due to the high number of blocking events by the neighbors. This affects

negatively the statistical quality of the collective sample.

4. Experimental Setup

We run simulations with $N = 100$ robots distributed over a rectangular 2D arena. The number of robots $N = 100$ was selected so that the average time robots spend in obstacle avoidance is smaller than the average time spent in other tasks (e.g., exploring, detecting, etc.). While keeping N fixed, we vary the number of items M over the range $[20 - 200]$. This allows us to study the collective dynamics for different ratios of N to the number of data items. We simulate a swarm of Footbots[3] using ARGoS—a state-of-the-art simulator for large-scale swarms that provides a high level of accuracy in simulating the robots' physics and dynamics. The Footbot has 24 proximity sensors to sense obstacles, and we use its range-and-bearing system (sensor and actuator) to (i) exchange messages relying on line-of-sight communications and (ii) to extract the relative distance to the message source (in centimeters)—this is how the robots compute the relative distance to their neighbors as shown in Fig. 1. Differently, the locations of the data items are assumed to be available for the simulated robots. ARGoS doesn't offer any simulated localization system, besides the simulated Footbot is not equipped with any localization feature. However, in practice we assume that an absolute frame reference can be used or other mechanisms can be applied such as light emitting, where a top camera system can be used to extract the different locations of light.

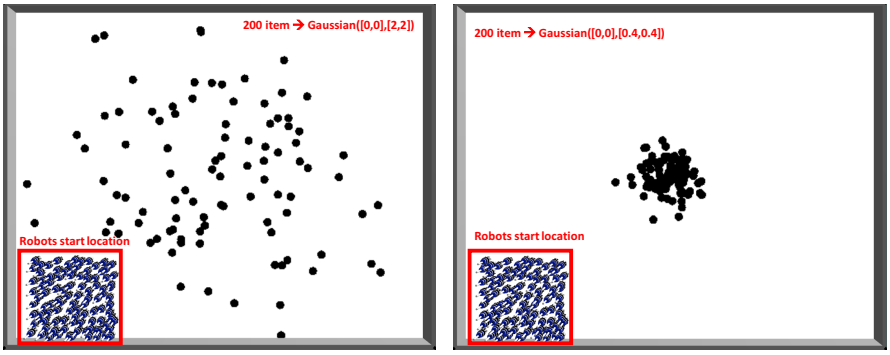
We test the performance of our proposed controllers using a unimodal as well as a bimodal Gaussian distribution to sample the location of the M data items at the beginning of the experiment. The macroscopic performance of the swarm is measured using the KL-divergence Eq. (3) and the upload percentage Eq. (4). Since the exploitation phase involves only communication between

[3] A wheeled robot used in the Swarmanoid project. It is equipped with 24 proximity sensors distributed around its perimeter, camera, and range-and-bearing communication system. It moves in the simulation at a speed of 5 cm/s.

agents, whereas the exploration and the detection phases involve agent motion, and because of the significant difference between communication speed and motion speed, the length of the first two phases is significantly larger than the length of the exploitation phase. The exact split of the phases length is computed based on the duration it may take the robot to travel along the arena diagonally. This time is used to define the duration of the two phases, whereas the rest of the time is assigned to the exploitation phase (details are given in the following sections for each of the two arenas). Our reported results for all experiment configurations are averaged over 30 runs. The feature Ω is represented as a group of colored circles scattered across the arena that can be detected using the color sensors at the bottom of the Footbot. The diameter of the circle is set to 10 cm. The diameter of the Footbot is 17 cm, so only one robot at a time can be over a particular circle. We also set the communication range of the robot to 1 m.

We fit S_{coll} to a multi-modal bivariate Gaussian with equal weights (as in the right-hand side of Eq. (2)). However, we do not assume the algorithm to know the number of Gaussians beforehand. Instead, we first apply k-means clustering [20] on S_{coll}. As we will see in the experiments, in some scenarios our fitting algorithm estimates a single modal distribution rather than a bimodal. The parameters of the multi-modal bivariate Gaussian ($\boldsymbol{\mu_g}$ and $\boldsymbol{\Sigma_g}$) are estimated from the samples S_{coll}. The KL-divergence computes the distance between the actual distribution that was used to distribute the items before the start of the experiment, and a fit of the collective sample.

Unimodal Gaussian: We use a 10×8 m^2 rectangular arena as shown in Fig. 2. We set the length of each experiment with a unimodal Gaussian to 3000 time units (300 simulated seconds). The speed of each simulated robot is 5 cm/s. Since the diagonal of the arena is appx. 1250 cm. It will take the robot appx. 250 simulated second to travel that distance. Hence, we set the duration of the exploration and detection phases to 0.8 of the total experiment time ($0.8 \times 300 = 240$ simulated second), and we split this duration equally between the two phases. Whereas the exploitation phase is assigned the rest of



(a) (b)

Figure 2: Unimodal Gaussian distribution of 200 data items, (a) unclustered configuration, and (b) clustered configuration.

the experiment time—i.e. 0.2 of the experiment time. Table. 1 summarizes the parameters used for the unimodal Gaussian. In all experiments with a unimodal Gaussian, the mean $\boldsymbol{\mu_g}$ is set to (0,0), the center of the arena. The value of the co-variance $\boldsymbol{\Sigma_g}$ (diagonal matrix) will then determine the spread of the M items. In our experiments, we will test two configurations: an unclustered and clustered configuration. For the unclustered configuration, shown in Fig. 2, we set all elements of the 2x2 diagonal matrix $\boldsymbol{\Sigma_g}$ to 2 m. Consequently, according to the 3-σ rule [21], the value of 2 m used for the standard deviation allows the data items to cover a squared area of 6×6 m^2 with a probability of 0.997. Differently, for the clustered configuration of the unimodal Gaussian, we set all diagonal elements of $\boldsymbol{\Sigma_g}$ to $\sigma = 0.4$ m. Hence, the data items cover a squared area of 1.2×1.2 m^2 with a probability of 0.997.

Bimodal Gaussian: we use a 16×16 m^2 square arena as shown in Fig. 3. The arena for the experiments with the bimodal Gaussian is larger than in the case of the unimodal Gaussian because we aim to preserve the inter-sample distances between the data items while keeping the same value for the standard deviation of a single Gaussian mode. The inter-sample distances is a critical parameter that influences directly the intensity of the collisions between the robots while detecting the data items. Therefore, we increased the arena size

Parameter	**Value**
Arena dimensions	$10 \times 8\ m^2$
Experiment time	3000 time steps (300 s)
Total number of items M	20-200 (steps of 10)
Swarm size N	100 robots
Linear speed of robot	5 cm/s
Unclustered Gaussian	$\mu = (0, 0)$ and $\sigma = 2$
Clustered Gaussian	$\mu = (0, 0)$ and $\sigma = 0.4$
Duration of exploration phase δ_e	0.4 of the experiment time
Duration of detection phase δ_d	0.4 of the experiment time
Duration of exploitation phase δ_p	0.2 of the experiment time
Size of voted neighbors D	0.5 of the local neighborhood
Uploading threshold ϕ	0.1 of the neighborhood size

Table 1: Table of parameters for the experiments executed with unimodal Gaussian.

in the bimodal experiment setting. Similar to the computations done for the unimodal Gaussian, the diagonal of the arena in the case of the bimodal Gaussian is appx. 2200 cm. Thus, it will take the robot appx. 440 simulated second to travel that distance (robot's speed is 5 cm/s). Hence, we set the duration of the exploration and detection phases to 0.8 of the total experiment time ($0.8 \times 550 = 440$ simulated second), and we split this duration equally between the two phases. Whereas the exploitation phase is assigned the rest of the experiment time—i.e. 0.2 of the experiment time. In all experiments with bimodal Gaussian distributions, we use a value of $\sigma = 2$ for all elements of the diagonal co-variance matrix $\mathbf{\Sigma_g}$. The means of both modals are placed on a diagonal of the arena. We vary the distance between the means of both modals and discern between a *close-means* and *far-means* scenario. In the latter scenario, the M items will be clustered around the two means. For both scenarios, we introduce two configurations: (i) non-symmetric: the means of the Gaussians

are placed on the diagonal of the arena from the bottom left to the top right corner; (ii) symmetric: the means of the Gaussians are placed on the diagonal from the bottom right to the top left corner. Hence, the spatial distribution of the M items looks symmetric from the deployment location of the robots—i.e. the bottom left corner of the arena. Example settings of all four configurations are shown in Fig. 3. The parameters used in the experiments of the bimodal distribution are given in Table. 2.

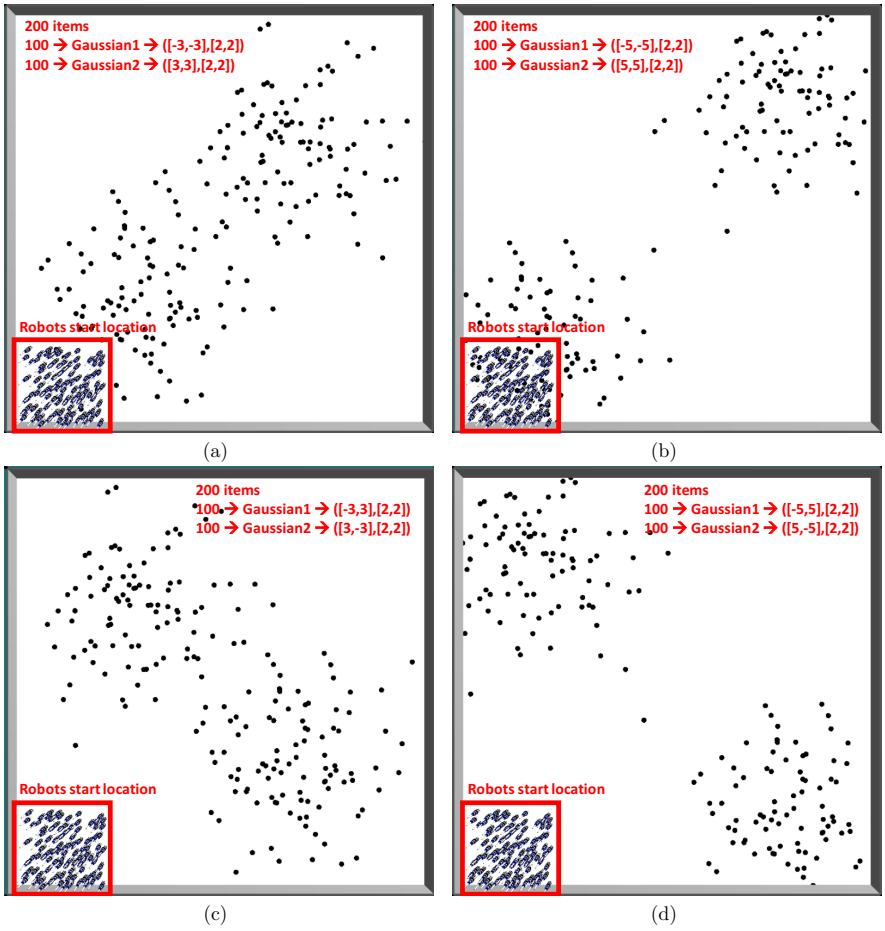


Figure 3: Sample scenarios for the different settings used in our experiments with bimodal Gaussian distributions of Ω. (a) near-means, non-symmetric; (b) far-means, non-symmetric; (c) near-means, symmetric; (d) far-means, symmetric.

Parameter	Value
Arena dimensions	$16 \times 16\ m^2$
Experiment time	5500 time steps (550 s)
Total number of items M	20-200 (steps of 10)
Swarm size N	100 robots
Linear speed of robot	5 cm/s
Near-means non-symmetric	$\mu_1 = (-3, -3)$, $\mu_2(3, 3)$ and $\sigma_1 = 2$, $\sigma_2 = 2$
Far-means non-symmetric	$\mu_1 = (-5, -5)$, $\mu_2(5, 5)$ and $\sigma_1 = 2$, $\sigma_2 = 2$
Near-means symmetric	$\mu_1 = (-3, 3)$, $\mu_2(3, -3)$ and $\sigma_1 = 2$, $\sigma_2 = 2$
Far-means symmetric	$\mu_1 = (-5, 5)$, $\mu_2(5, -5)$ and $\sigma_1 = 2$, $\sigma_2 = 2$
Duration of exploration phase δ_e	0.4 of the experiment time
Duration of detection phase δ_d	0.4 of the experiment time
Duration of exploitation phase δ_p	0.2 of the experiment time
Size of voted neighbors D	0.5% of the local neighborhood
Uploading threshold ϕ	0.1 of the neighborhood size

Table 2: Table of parameters for the experiments executed with bimodal Gaussian.

5. Results and Discussions

5.1. Unimodal distribution

5.1.1. Unclustered configuration

Fig. 4 demonstrates the fit of the distribution extracted from the collective sample S_{coll} that was collected by the three types of controllers and for different item densities (from left to right: $M = \{20, 50, 100, 150, 200\}$). For each controller, the upper graphs show the locations of the robots who decided to upload their detected items. The bottom graph shows a histogram of the locations along the X-axis and the fitted distribution. While Fig. 4 gives a more qualitative insight, in Fig. 5, we plot the two performance measures that were described in section 2.

When $M > 50$, the CSC controller provides the lowest KL divergence. The BNC controller provides slightly worse KL-divergence. Notably, the KL divergence of the AUC controller degrades rapidly with increasing values of M. This observation can be explained in light of the larger number of items available for the robots to sample nearby their starting location. With the AUC controller, robots will stop and sample the first item detected by each robot. Therefore, the collective sample becomes biased with a shifted mean (to negative coordinates). This is clearly visible in Fig. 4b.

The KL-divergence of the fitting in case of the collective sample generated by the BNC controller is significantly better than the KL-divergence of the fittings from the AUC controller, and similar to the fittings of the CSC controller. This performance of BNC can be explained by the large spatial coverage of the collective sample that this controller generates, particularly for an unclustered distribution of the items. Two features of BNC may be responsible for the slightly decrements in performance for higher values of M in comparison to CSC: (i) when inter-item distances are smaller than the robot communication range, those are never sampled by the BNC controller, and such distances are more frequent with higher values of M. (ii) The more items there are, the more neighbors will sample, and thus the more blocking actions are taken. Consequently, robots will spend a long time searching for items that they can sample and upload without being blocked by a neighbor. This may result in the experiment finishing before a large-enough sample is generated.

The upload percentage, as shown in Fig. 5b is slightly higher for the CSC controller than for the BNC controller. The AUC controller results in the highest upload percentage. Remarkably, the AUC upload percentage decreases with higher values of M. This is due to spatial interferences (i.e. obstacle avoidance) between robots that are trying to upload items nearby their starting location, and robots that are trying to move out.

5.1.2. Clustered configuration

The results of this configuration are shown in Fig. 6 and Fig. 7. For this configuration, the AUC and BNC both generate a biased distribution with a shifted mean, specifically for high item densities, see Fig. 6b,c. For AUC, similar to the unclustered configuration: the part of the cluster that is nearer to the robots' starting location will be over-sampled and the mean is shifted to negative values. In the case of BNC, the shifted mean results from the robots approaching first the cluster area near to their starting location. Since the robot's communication range covers a large part of the cluster, these robots will then block other robots who try to sample from other locations of the cluster. Therefore, the area of the cluster that is covered first is the one with the highest probability to be sampled. CSC performs best in the case of clustered item distribution, thanks (i) to the exploration and detection phases, in which the robots detect the cluster of data items, and (ii) to the exploitation phase that balances both the locations and the number of samples collected, see Fig. 6a.

As shown in Fig. 7a, the estimated distributions of the samples generated by the CSC result in the lowest KL-divergence. AUC generates again the worst samples: the KL-divergence decreases almost linearly with the item density. In terms of upload percentage, shown in Fig. 7b, BNC is the most economic controller. However, we can notice that the upload percentage of BNC drops with increasing item density due to the blocking effects. As a result, the size of the collective sample will decrease and the KL-divergence will increase accordingly. Another remarkable result is the stabilization of the upload percentage of CSC around 0.3 for higher values of M. This is a clear indicator of the efficiency of CSC by converging to an adequate sample size even when a large amount of items is available to sample from. Finally, the AUC has the highest upload percentage, but this decreases with M due to spatial interferences between the robots in the small area where the items are clustered.

5.2. Bimodal distribution

5.2.1. Near means, non-symmetric

Fig. 8 illustrates the fitting of the distribution generated for the bi-modal Gaussian based on the estimated means and standard deviations extracted from the collective sample using each of the three controllers. In Fig. 8b, it is obvious that the collective sample generated by the AUC covers mainly the Gaussian near to the robots' starting location ($\mu = (-3, -3)$, $\sigma = 2$) rather than the far Gaussian ($\mu = (3, 3)$, $\sigma = 2$). Hence, the fitting generated for the near Gaussian is significantly better than the far one, and this increases with increasing the item density.

A similar problem, although less significant, is observed for the BNC controller, as shown in Fig. 8c. The CSC provides the best coverage of both Gaussians, as shown in Fig. 8a. Nevertheless, the algorithm (i.e. k-means clustering) that is used to derive the distribution from the collective sample was not efficient enough in distinguishing between the two Gaussians. Instead, it was mostly interpreted as one Gaussian.

Due to all reasons explained above, none of the three controllers has reached a high accuracy in the estimation of the item distribution (Fig. 9c), even when CSC has reached a considerably wider coverage and both BNC and CSC have uploaded significantly fewer items than AUC Fig. 10.

However, CSC shows a better KL-divergence than BNC for the Gaussian furthest from the starting location. Both controllers are better than AUC, see Fig. 9a, for which the KL-divergence even degrades with increasing item density.

For the Gaussian nearest to the robots' starting location, CSC preserves its KL-divergence. AUC is the best controller here, since all robots sample for the Gaussian near to their starting location, see Fig. 9b.

5.2.2. Near means, symmetric

For the symmetric configuration with near means, no bias is found in the sampling process performed by all three controllers. see in Fig. 11. All controllers generate a wide-enough sample of both Gaussians, thanks to the equal

distance of the means from the robots' starting location.

Nevertheless, due to the means being close to each other, the k-means clustering algorithm was not able to recognize the bi-modality in the item distributions, and hence the KL-divergence values are high for all controllers, see Fig. 12. The percentage of uploaded items is similar for CSC and BNC and decreases with higher item densities. AUC results in the highest upload percentage, see Fig. 13. Consequently, for this particular settings of near means with symmetric configuration, BNC outperformed CSC.

It is however important to note that the item distribution of each Gaussian can be categorized as unclustered with respect to the robots' communication range. In case the Gaussian distributions would have had a smaller variance, CSC and not BNC would have been the best controller (see Sec. 5.1).

5.2.3. Far means, non-symmetric

Fig. 14 shows the fitting of the distribution estimated from the collective sample that is delivered by each of the three controllers. Similar to the case of near means, CSC is able to sample both Gaussian distributions. An interesting observation though is that the furthest Gaussian is slightly better covered than the near one. This is due to the fact that by the end of the exploration phase, most robots have reached the furthest Gaussian but there was not enough time to further diffuse and generate the most balanced coverage over the whole arena. Nevertheless, the effect of this parameter setting is not fundamental in the performance of CSC, since wide-enough samples of both Gaussian are attained.

The collective sample delivered by AUC is biased to the Gaussian nearest to the robots' starting location, with a considerably sparse sampling of the furthest Gaussian. This effect becomes even more accentuated for higher item densities: the majority of the robots aggregates at the nearest Gaussian. BNC suffers from the same biased sampling, however to a lesser extent thanks to the blocking process applied by the neighbors, which stimulates a wider dispersion of the robots across the environment.

The CSC controller results in a fairly low KL-divergence values for both the

near and the far Gaussian, see Fig. 15a,b. For AUC, the KL-divergence improves significantly for the near Gaussian due to the over sampling performed at that cluster, see Fig. 15b. BNC also performs the best for the Gaussian nearest to the robots' starting location and positions itself between AUC and CSC. When averaging the KL-divergence over both Gaussians, all controllers perform similarly, in terms of this perfromance measure.

In terms of the upload percentage, CSC has the best performance, even better than BNC. This remarkable result can be explained by the minimization mechanism applied during the exploitation phase of the CSC. For BNC, the blocking by neighbours effect is minimal because the two Gaussians are far-enough from each other. For higher item densities, the blocking intensity increases and hence the upload percentages of CSC and BNC start to converge.

5.2.4. Far means, symmetric

Finally, in this configuration the three controllers generate collective samples that lead to a good and similar fitting, as shown in Fig. 17.

As shown in Fig. 18, all controllers achieve a fairly low KL-divergence for low item densities. For higher item densities, the performance of AUC and BNC drops while CSC sustains its the performance. This rather non-intuitive result is obtained due to the sampling dynamics of AUC and BNC. When increasing the item density, more robots will find an item in the area between the two Gaussian distributions. Thus more items are sampled at that specific area, leading to a more difficult separation of the two Gaussian distributions by the analyzing algorithm, and therefore higher KL-divergence values, for those controllers. In Fig. 17, the reader can indeed notice the higher intensity of sampling generated by AUC and BNC over CSC in the area between the two Gaussian distributions.

Regarding the upload percentage, CSC performs equally to BNC and samples much fewer items than AUC, see Fig. 19.

6. Conclusion

In this study, we have investigated the application of robot swarms in sampling environmental features that are spatially distributed over large-scale unknown environments. This problem is of a high interest when considering future applications of robotics systems in large-scale environments such as search and rescue, precision agriculture, and even in-body cell sampling with no external control. We have addressed the sampling problem under the constrained of a limited sampling budget (LSB), that is associated with the limited on-board capabilities of the robots. We have leveraged our challenge by attempting to maximize the statistical quality of the collective sample (measured using the KL-divergence), while minimizing the number of samples taken and limiting the number of samples to one by each robot. We have proposed a novel controller CSC (collective sampling controller), which relies on three phases: exploration, detection, and exploitation to better search the environment and represent the inter-sample distances. The performance of CSC was compared to two other controllers (AUC and BNC) that were implemented as special cases and benchmark for CSC.

Our results show that both the exploration phase and the voting mechanism used during the exploitation phase facilitate in most cases a highly accurate estimation of the parameters of the feature spatial distribution (verified using KL-divergence measure), and/or a high economic sampling (verified using the percentage of the items sampled). As a future work, we would like to extend CSC so that robots become able to decide for the number of voted neighbors autonomously based on (i) the size of the local neighborhood and (ii) the experiences collected during the exploration phase about the item densities. Furthermore, besides the uploading cost, which we aimed to optimize in this study, other cost functions such as the ones account for robots' travelling costs need to be taken into account.

References

[1] Y. Khaluf, M. Dorigo, Modeling robot swarms using integrals of birth-death processes, ACM Transactions on Autonomous and Adaptive Systems (TAAS) 11 (2) (2016) 8.

[2] E. Şahin, Swarm robotics: From sources of inspiration to domains of application, in: E. Şahin, W. M. Spears (Eds.), Swarm Robotics, Springer, Berlin, Heidelberg, 2005, pp. 10–20.

[3] Y. Khaluf, Edge detection in static and dynamic environments using robot swarms, in: 2017 IEEE 11th International Conference on Self-Adaptive and Self-Organizing Systems (SASO), IEEE, 2017, pp. 81–90.

[4] H. Hamann, Swarm Robotics: A Formal Approach, Springer, 2018.

[5] Y. Khaluf, M. Birattari, F. Rammig, Probabilistic analysis of long-term swarm performance under spatial interferences, in: International Conference on Theory and Practice of Natural Computing, Springer, 2013, pp. 121–132.

[6] M. Brambilla, E. Ferrante, M. Birattari, M. Dorigo, Swarm robotics: a review from the swarm engineering perspective, Swarm Intelligence 7 (1) (2013) 1–41.

[7] H. Hamann, G. Valentini, Y. Khaluf, M. Dorigo, Derivation of a micro-macro link for collective decision-making systems, in: International Conference on Parallel Problem Solving from Nature, Springer, 2014, pp. 181–190.

[8] Y. Khaluf, E. Ferrante, P. Simoens, C. Huepe, Scale invariance in natural and artificial collective systems: a review, Journal of The Royal Society Interface 14 (136).

[9] M. Dorigo, et al., On the definition of self-organizing systems: Relevance of positive/negative feedback and fluctuations, in: Swarm Intelligence: 10th

International Conference, ANTS 2016, Brussels, Belgium, September 7-9, 2016, Proceedings, Vol. 9882, Springer, 2016, p. 298.

[10] E. Masehian, M. Jannati, T. Hekmatfar, Cooperative mapping of unknown environments by multiple heterogeneous mobile robots with limited sensing, Robotics and Autonomous Systems 87 (2017) 188–218.

[11] H. Anil, K. S. Nikhil, V. Chaitra, B. S. G. Sharan, Revolutionizing farming using swarm robotics, in: 2015 6th International Conference on Intelligent Systems, Modelling and Simulation, 2015, pp. 141–147.

[12] R. Yokota, Y. Kaminaga, T. J. Kobayashi, Quantification of inter-sample differences in t-cell receptor repertoires using sequence-based information, Frontiers in immunology 8 (2017) 1500.

[13] W. Huber, A. v. Heydebreck, M. Vingron, Analysis of microarray gene expression data, in: Handbook of Statistical Genetics, John Wiley & Sons, Ltd, 2003.

[14] S. Kullback, R. A. Leibler, On information and sufficiency, The Annals of Mathematical Statistics 22 (1) (1951) 79–86.

[15] T. Van Erven, P. Harremos, Rényi divergence and kullback-leibler divergence, IEEE Transactions on Information Theory 60 (7) (2014) 3797–3820.

[16] D. J. Galas, G. Dewey, J. Kunert-Graf, N. A. Sakhanenko, Expansion of the kullback-leibler divergence, and a new class of information metrics, Axioms 6 (2) (2017) 8.

[17] A. I. Skoulidas, Molecular dynamics simulations of gas diffusion in metal-organic frameworks: argon in cubtc, Journal of the American Chemical Society 126 (5) (2004) 1356–1357.

[18] J. Barraquand, B. Langlois, J.-C. Latombe, Numerical potential field techniques for robot path planning, IEEE transactions on systems, man, and cybernetics 22 (2) (1992) 224–241.

[19] G. Theraulaz, E. Bonabeau, J. Denuebourg, Response threshold reinforcements and division of labour in insect societies, Proceedings of the Royal Society of London B: Biological Sciences 265 (1393) (1998) 327–332.

[20] J. A. Hartigan, M. A. Wong, Algorithm as 136: A k-means clustering algorithm, Journal of the Royal Statistical Society. Series C (Applied Statistics) 28 (1) (1979) 100–108.

[21] F. Pukelsheim, The three sigma rule, The American Statistician 48 (2) (1994) 88–91.

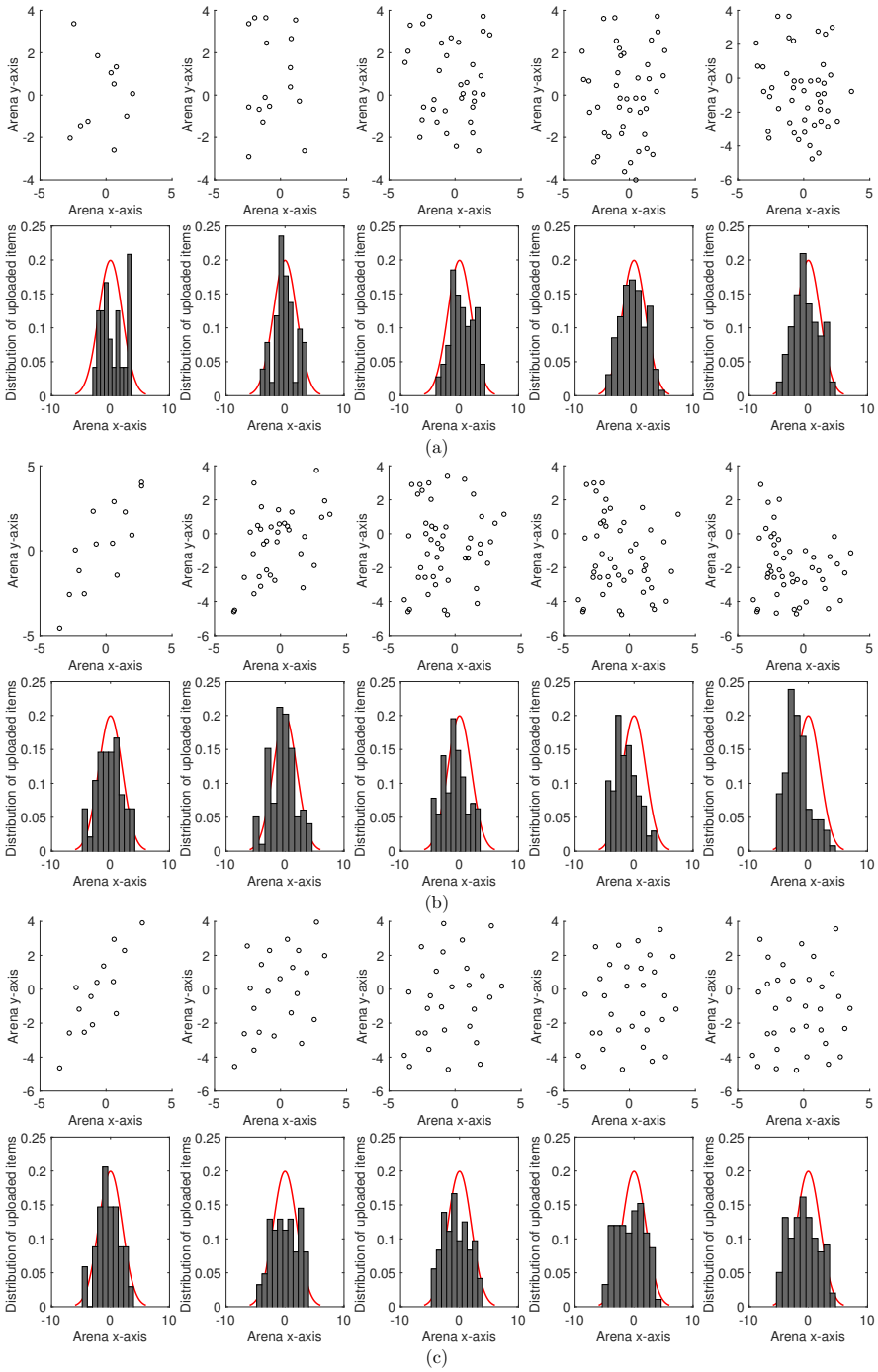
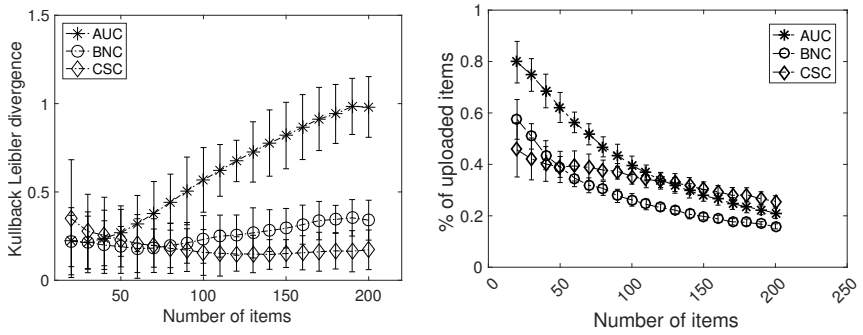


Figure 4: The fitting of the distribution generated from the collective sample associated with the output of one sampling process for (a) CSC, (b) AUC, and (C) BNC.



(a) (b)

Figure 5: Collective sampling performance metrics for the unclustered unimodal distribution: (a) KL-divergence, and (b) the percentage of uploaded items.

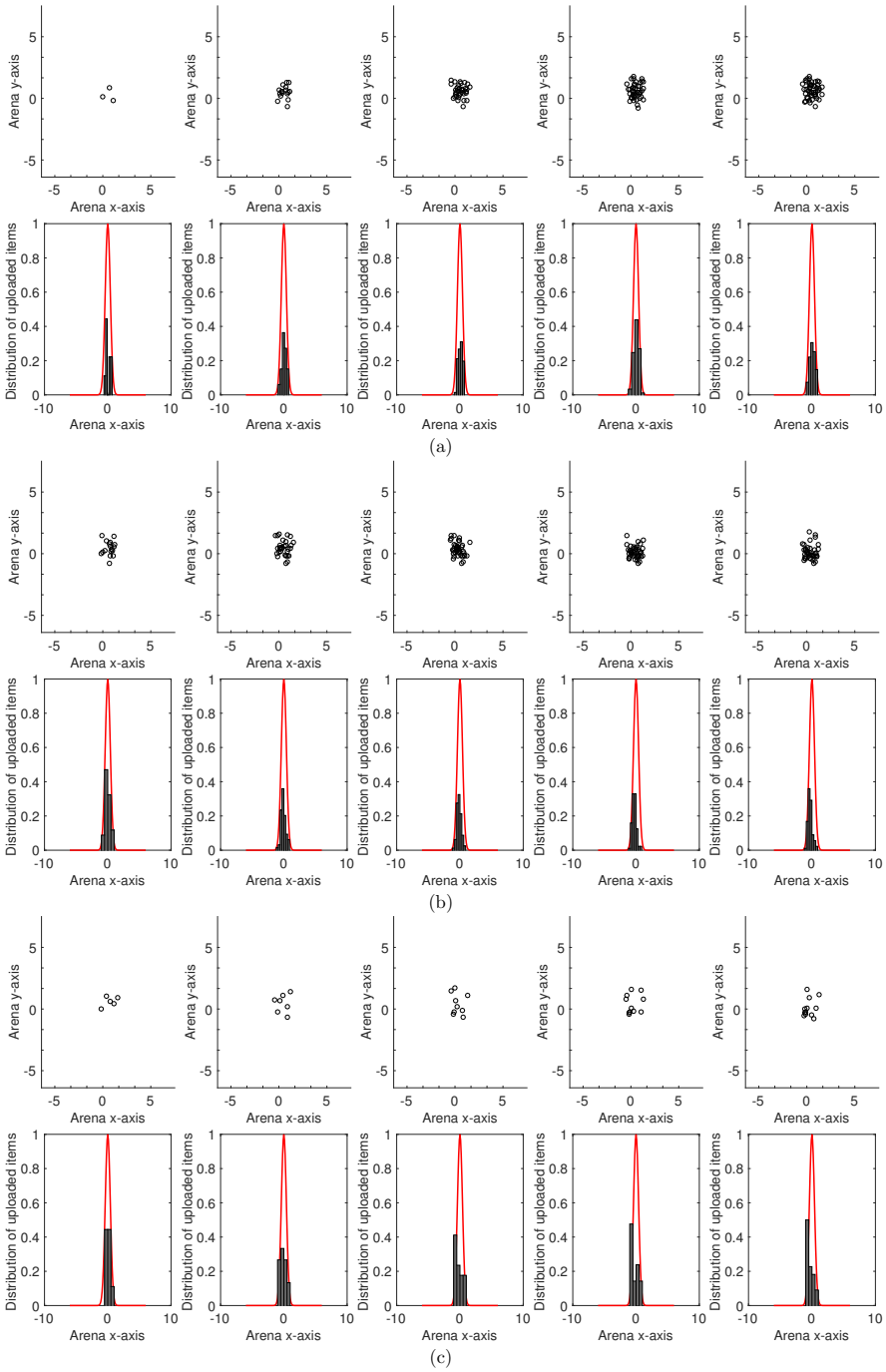
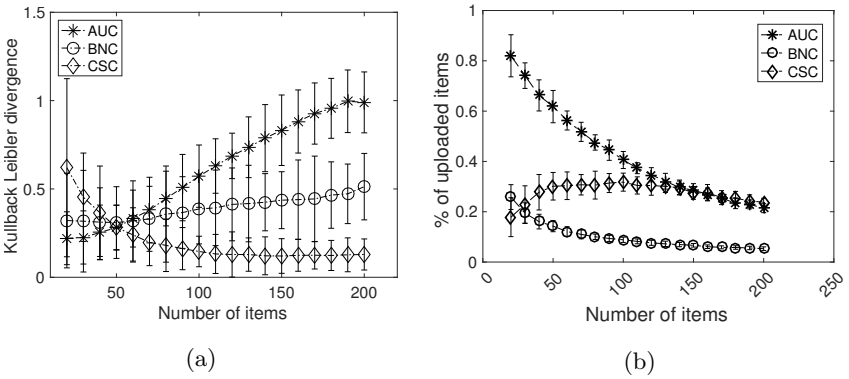


Figure 6: The fitting of the distribution generated from the collective sample associated with the output of one sampling process for (a) CSC, (b) AUC, and (C) BNC.



(a) (b)

Figure 7: Collective sampling performance metrics for the clustered unimodal distribution: (a) KL-divergence, and (b) the percentage of uploaded items.

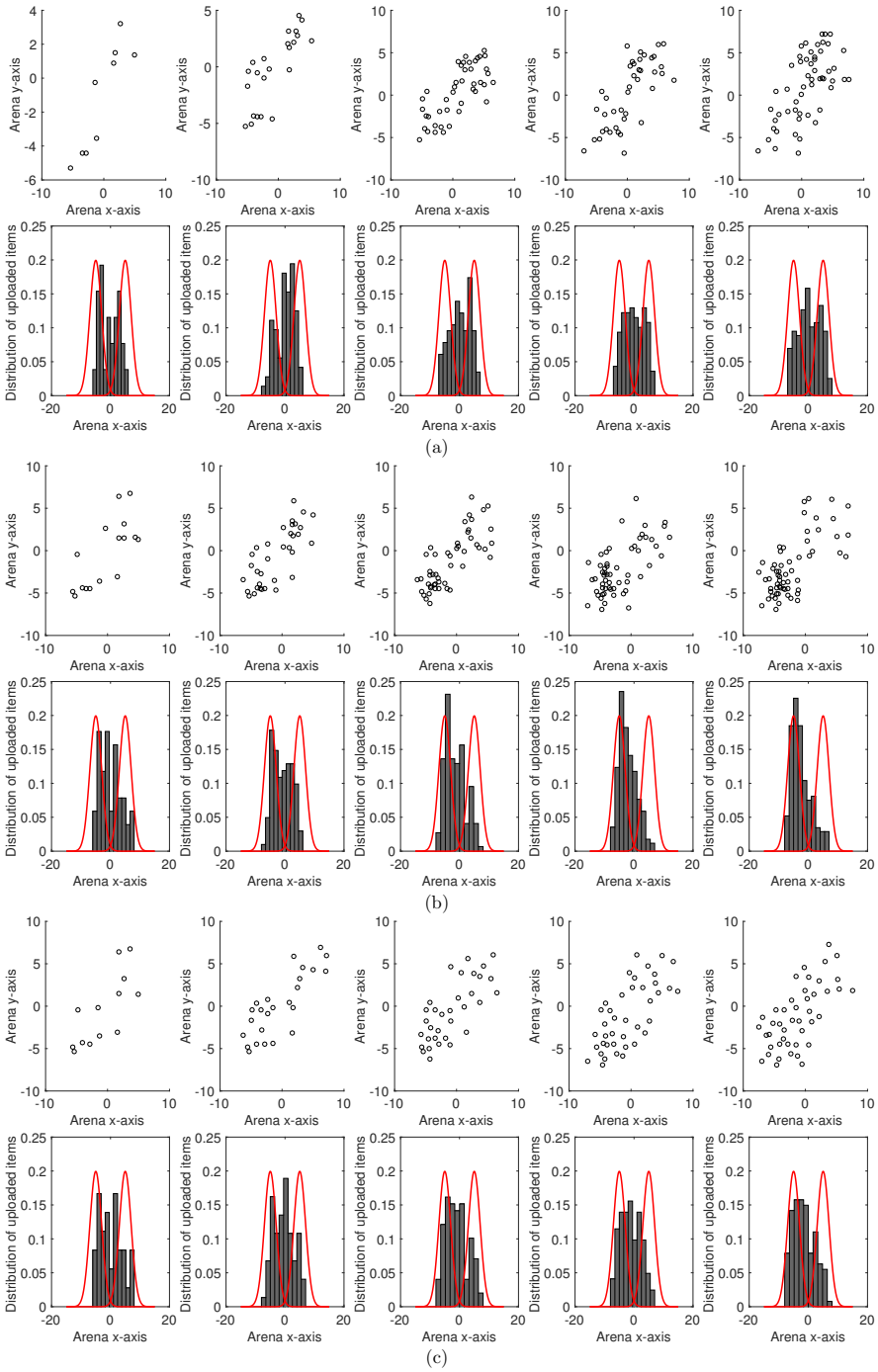


Figure 8: The fitting for bi-modal Gaussian, near means and non-symmetric item distributions, generated by (a) CSC, (b) AUC, and (C) BNC.

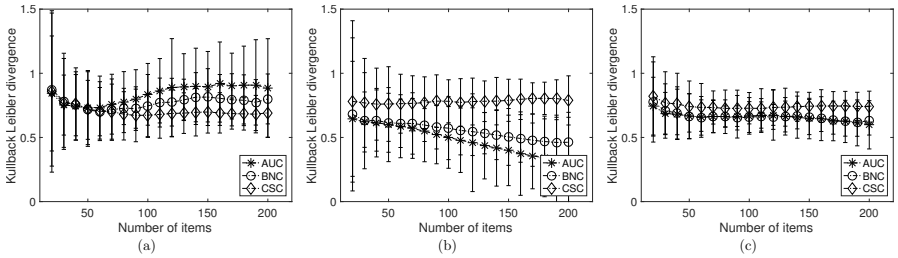


Figure 9: KL-divergence of the bimodal distribution (near-means, non-symmetric) estimated from the collective sample (a) the Gaussian furthest from the robots' starting location, (b) the Gaussian nearest to the robots' starting location, (c) the average of KL-divergence over both Gaussians.

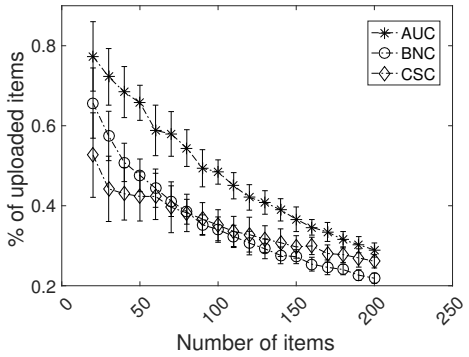


Figure 10: Percentage of uploaded items for the near-means, non-symmetric configuration.

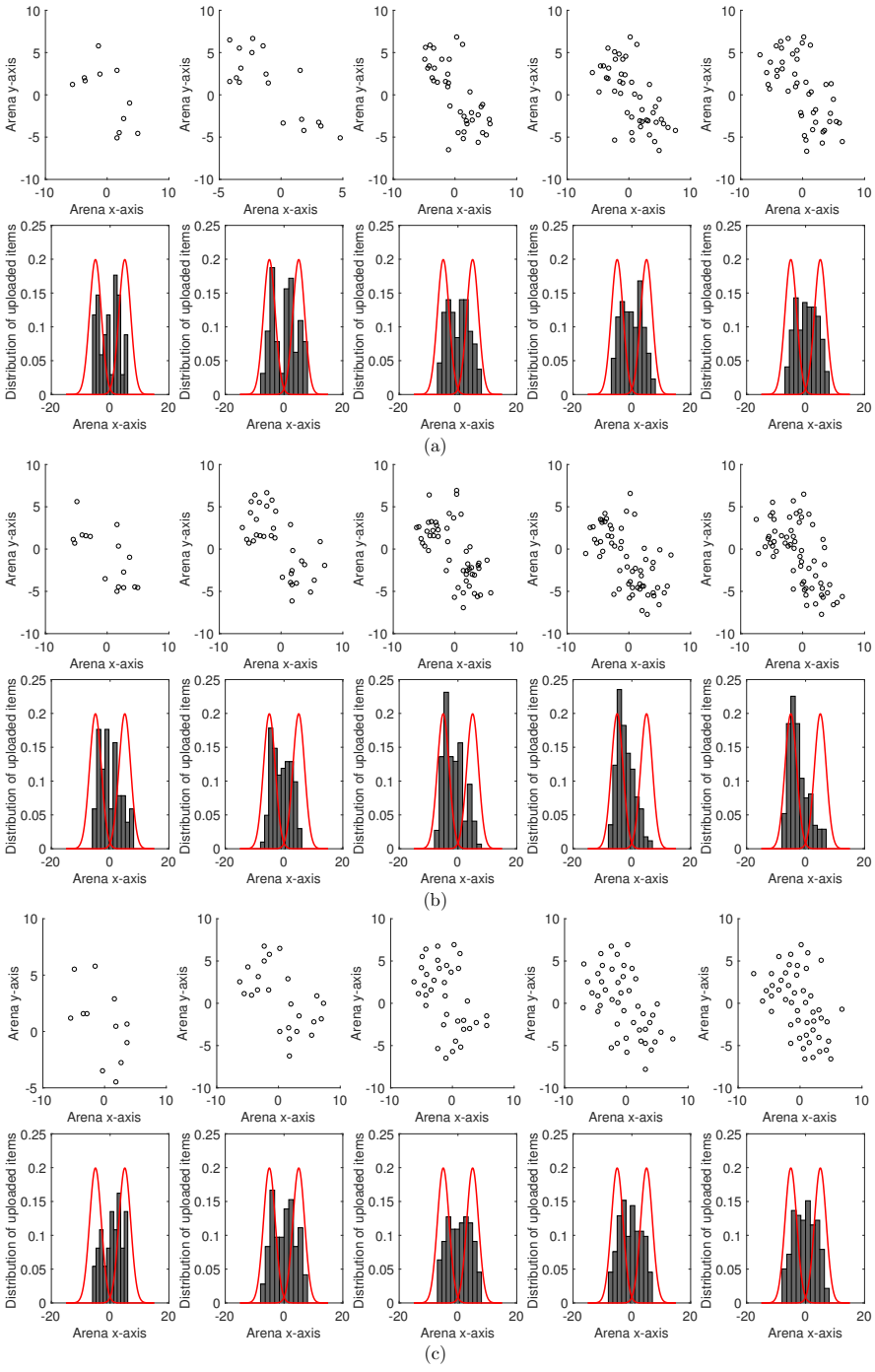


Figure 11: The fitting of the sample collected from a bi-modal Gaussians (near means, symmetric) by (a) CSC, (b) AUC, and (C) BNC.

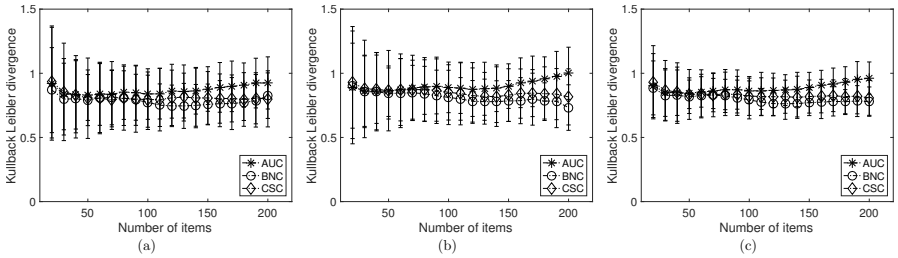


Figure 12: KL-divergence of the bimodal distribution (near-means, symmetric) estimated from the collective sample (a) the Gaussian on the right side of the robots' starting location, (b) the Gaussian on the left of the robots' starting location, (c) the average of KL-divergence over both Gaussians.

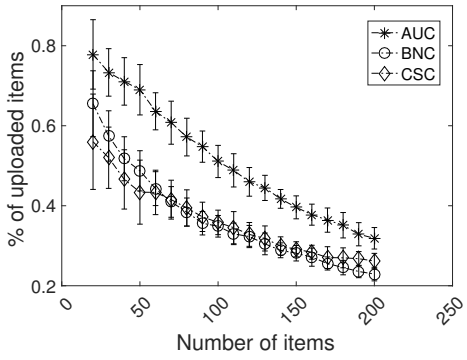


Figure 13: Percentage of uploaded items for the near-means, symmetric configuration.

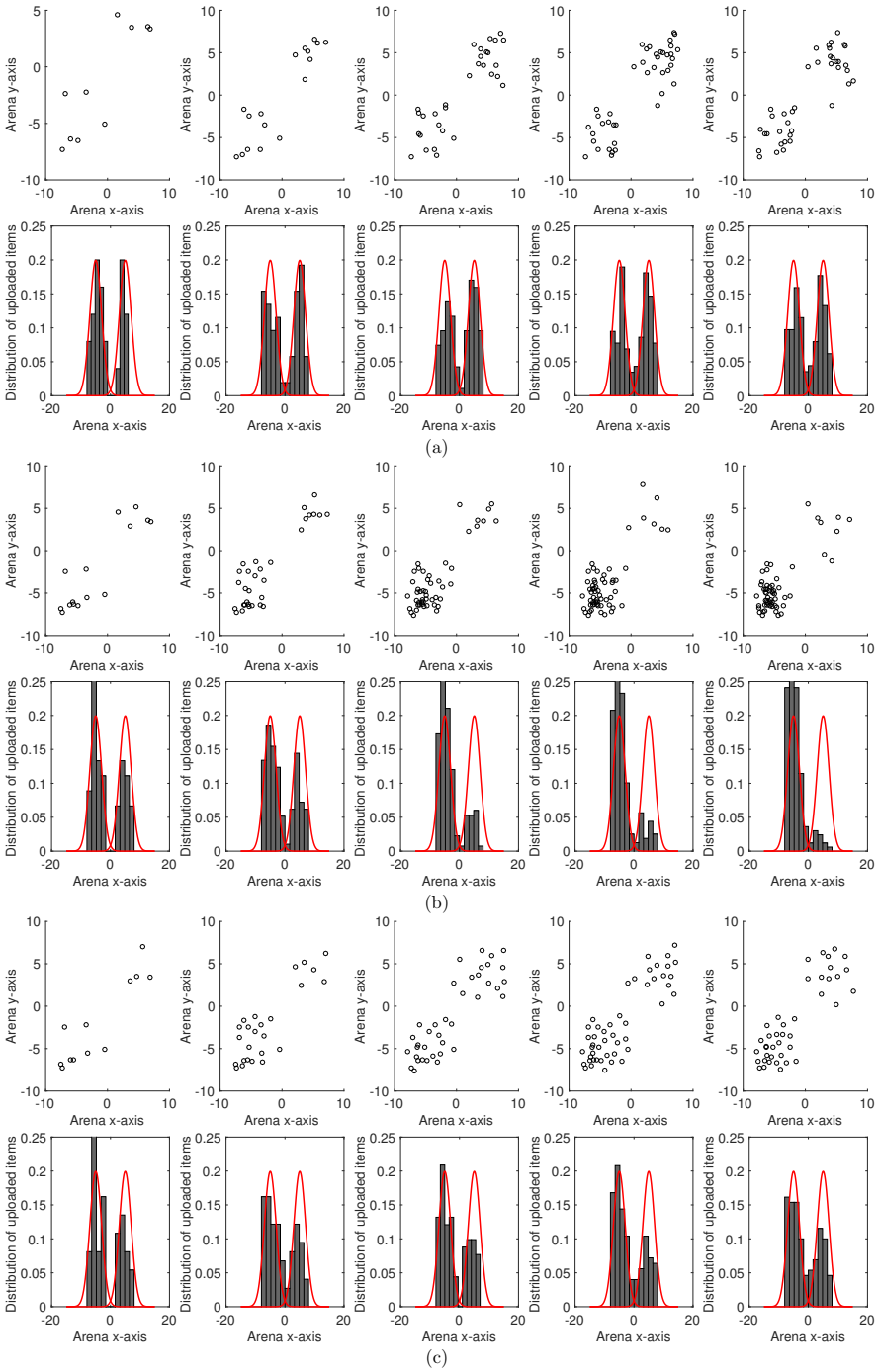


Figure 14: The fitting for bi-modal Gaussian, far means and none-symmetric item distributions, generated by (a) CSC, (b) AUC, and (C) BNC.

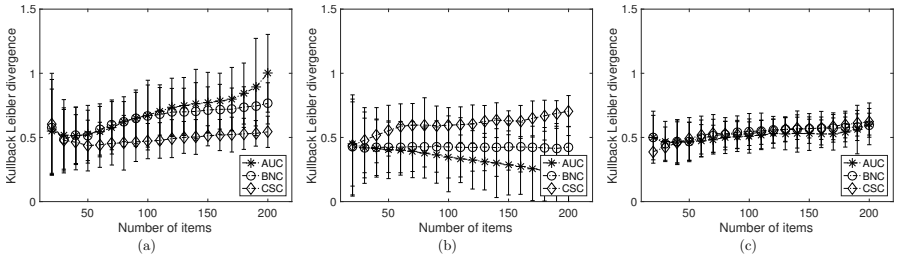


Figure 15: KL-divergence of the bimodal distribution (near-means, symmetric) estimated from the collective sample (a) the Gaussian on the right side of the robots' starting location, (b) the Gaussian on the left of the robots' starting location, (c) the average of KL-divergence over both Gaussians.

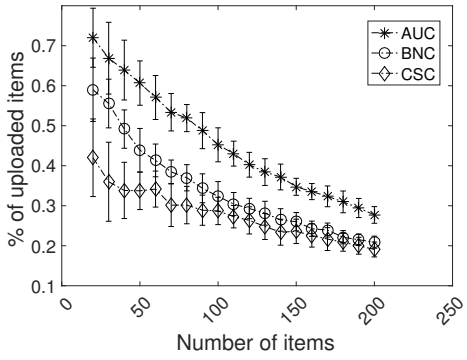


Figure 16: Percentage of uploaded items for the far means, non-symmetric configuration.

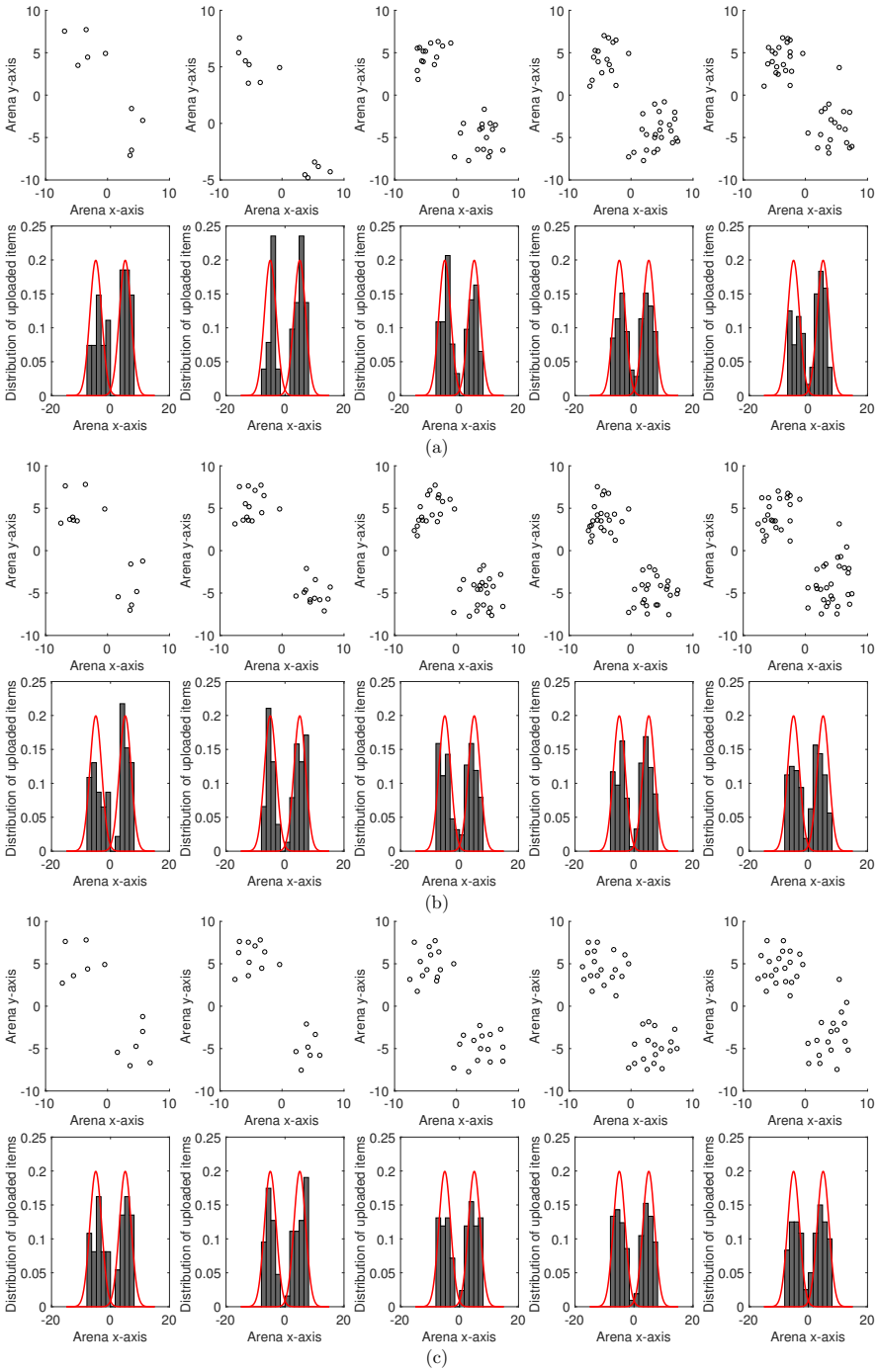


Figure 17: The fitting for bi-modal Gaussian, far means and symmetric item distributions, generated by (a) CSC, (b) AUC, and (C) BNC.

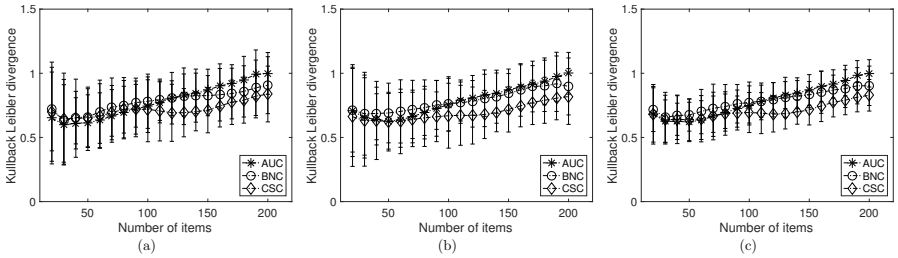


Figure 18: KL-divergence of the bimodal distribution (near-means, symmetric) estimated from the collective sample (a) the Gaussian on the right side of the robots' starting location, (b) the Gaussian on the left of the robots starting location, (c) the average of KL-divergence over both Gaussians.

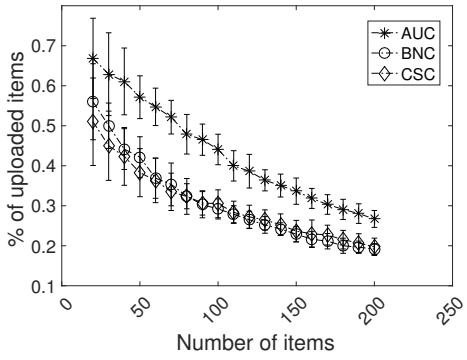


Figure 19: Percentage of uploaded items for the far-means, symmetric configuration.