Article

The International Journal of Robotics Research
2020, Vol. 39(9) 1155–1177
© The Author(s) 2020
Article reuse guidelines:
sagepub.com/journals-permissions
DOI: 10.1177/0278364920921941
journals.sagepub.com/home/ijr

FSMI: Fast computation of Shannon mutual information for information-theoretic mapping

Zhengdong Zhang[1], Theia Henderson[1], Sertac Karaman[2] and Vivienne Sze[1]

Abstract

Exploration tasks are embedded in many robotics applications, such as search and rescue and space exploration. Information-based exploration algorithms aim to find the most informative trajectories by maximizing an information-theoretic metric, such as the mutual information between the map and potential future measurements. Unfortunately, most existing information-based exploration algorithms are plagued by the computational difficulty of evaluating the Shannon mutual information metric. In this article, we consider the fundamental problem of evaluating Shannon mutual information between the map and a range measurement. First, we consider 2D environments. We propose a novel algorithm, called the fast Shannon mutual information (FSMI). The key insight behind the algorithm is that a certain integral can be computed analytically, leading to substantial computational savings. Second, we consider 3D environments, represented by efficient data structures, e.g., an OctoMap, such that the measurements are compressed by run-length encoding (RLE). We propose a novel algorithm, called FSMI-RLE, that efficiently evaluates the Shannon mutual information when the measurements are compressed using RLE. For both the FSMI and the FSMI-RLE, we also propose variants that make different assumptions on the sensor noise distribution for the purpose of further computational savings. We evaluate the proposed algorithms in extensive experiments. In particular, we show that the proposed algorithms outperform existing algorithms that compute Shannon mutual information as well as other algorithms that compute the Cauchy–Schwarz quadratic mutual information (CSQMI). In addition, we demonstrate the computation of Shannon mutual information on a 3D map for the first time.

Keywords

Mapping robot, Bayes methods, mutual information, range sensing robot, robot control

1. Introduction

Robot exploration tasks are embedded and essential in several applications of robotics, including disaster response and space exploration. The problem has received a large amount of attention over the past few decades, resulting in a rich literature.

On the one hand, *geometry-based frontier exploration algorithms* approach this problem with heuristics that typically navigate the robot to the frontier of the well-known portion of the environment (Ya mauchi, 1997). Researchers investigated various objective functions (Burgard et al., 2005; González-Banos and Latombe, 2002), and Holz et al. (2011) surveyed their performance in practical scenarios. These heuristics are very efficient from a computational point of view. However, they lack any rigorous reasoning about information, which makes them relatively inefficient in terms of the path spanned by the robot while exploring the environment (Bourgault et al., 2002; Cassandra et al., 1996; Elfes, 1996; Moorehead et al., 2001). In addition, it is hard to extend the geometry that they rely on to three-dimensional environments (Shen et al., 2012).

On the other hand, *information-based mapping and exploration* techniques consider paths that aim to maximize principled information-theoretic metrics to actively maximize the information collected by the robot. The Shannon mutual information (MI) between a perspective scan and the occupancy grid is used for exploration in Bourgault

[1]Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA, USA
[2]Department of Aeronautics and Astronautics, MIT, Cambridge, MA, USA

Corresponding author:
Zhengdong Zhang, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Room 38-260, Cambridge, MA 02139, USA.
Email: zhangzd@csail.mit.edu

et al. (2002). Yu and Gupta (2004) and Wang and Gupta (2002, 2003, 2006) studied entropy reduction in the absence of sensor noise in the C-space, which reduces to physical space for a point robot. Visser and Slamet (2008) introduced an objective function that balances the MI and the moving cost. Charrow et al. (2014) developed an approximated MI representation for efficient multi-robot control. This MI metric is also widely used in many other related applications. Kollar and Roy (2008) proposes an information-theoretic objective for simultaneous localization and mapping (SLAM). Marchant and Ramos (2014) used it to perform continuous path planning. Julian et al. (2014) established a rigorous theory and algorithms for evaluating Shannon MI between a measurement and the map where the sensor follows a known noisy model. While information-based mapping algorithms using Shannon MI provide guarantees on the exploration of the environment, the evaluation of Shannon MI, e.g., by the algorithm provided by Julian et al. (2014), is computational demanding. The run time of the algorithm scales quadratically with the spatial resolution of the occupancy grid and linearly with the numerical integration resolution of the range measurement due to the absence of an analytical solution. It has been pointed out that the speed at which MI is evaluated can limit the planning frequency, which in turn limits the velocity of the robot and the exploration speed of the environment (Nelson and Michael, 2015).

Towards designing algorithms that are computationally more efficient, Charrow et al. (2015b) proposed the use of an alternative information metric, called Cauchy–Schwarz quadratic mutual information (CSQMI). They showed that the integrations in CSQMI can be computed analytically. In addition, they showed a close approximation of CSQMI can be evaluated in time that scales linearly with respect to the spatial resolution of the occupancy grid. It is reported by Charrow et al. (2015b) that CSQMI can be computed substantially faster than Shannon MI, and it behaves similarly to Shannon MI in experiments. Several other works adopted CSQMI as the information metric for exploration. For instance, Charrow et al. (2015a) proposed a hybrid method with global and local trajectory optimization based on CSQMI. Nelson and Michael (2015) proposed an adaptive occupancy grid compression algorithm and used CSQMI to design the planner on the compressed map. Tabib et al. (2016) combined CSQMI with trajectory optimization and compression to build an energy-efficient cave exploration system with a drone.

In this article, we focus on the fundamental problem of computing the Shannon MI metric between a range measurement and the map. We propose a new class of algorithms for efficient evaluation of this metric.

First, we propose the fast Shannon mutual information method, also called the FSMI algorithm, which evaluates Shannon MI exactly for Gaussian distributed sensor noise characteristics as commonly assumed in the literature. The key idea behind the FSMI algorithm is to analytically evaluate a certain integral, which leads to substantial computational savings. Second, we propose two variants of this algorithm, called Approx-FSMI and Uniform-FSMI, which approximate the evaluation of the same metric by truncating the Gaussian distribution (which was first introduced by Charrow et al. (2015b)) and by assuming a uniform sensor noise distribution, respectively. Third, we propose the variants of these algorithms, called FSMI-RLE, Approx-FSMI-RLE, and Uniform-FSMI-RLE, that can handle measurements represented in run-length encoding (RLE). The RLE is a certain kind of compression, which is particularly efficient for working with three-dimensional maps. The time complexity of the proposed algorithms are presented in Table 1. For instance, the time complexity of the FSMI algorithm is $O(n^2)$, where n is the number of cells in the map that the range measurement intersects. In contrast, the time complexity of the existing algorithm for computing Shannon MI is $O(n^2 \lambda_z)$, where λ_z is resolution for a certain numerical integral. The FSMI algorithm avoids this numerical integral, and various approximations of sensor noise characteristics and encodings of the measurement vector provide even more efficiency.

We demonstrate this theoretical computational efficiency in experiments. In particular, we present a comparison of various methods for computing MI in a computational study, in simulations, and in experiments involving a ground robot equipped with a planar laser scanner. In addition, we present mapping in a three-dimensional environment involving a ground robot equipped with a three-dimensional Velodyne VLP-16 laser scanner.

2. Preliminaries

This section is devoted to the introduction of preliminaries and our notation that we use throughout the article. We briefly review the occupancy grid in Section 2.1, the Shannon MI metric and its computation in Section 2.2, and the CSQMI metric and its computation in Section 2.3. Finally, in Section 2.4, we formulate the single-beam MI evaluation problem considered in this article.

2.1. The occupancy grid

We use the occupancy grid to model the environment. Following standard convention (Charrow et al., 2015a,b; Julian et al., 2014; Yamauchi, 1997), we assume that all the occupancy cells are independent and that a Bayesian filter is used to update the occupancy probabilities.

The occupancy grid is denoted by the random variables $M = \{M_1, \dots, M_K\}$, where K is the number of cells and $M_i \in \{0, 1\}$ is the binary random variable that indicates the occupancy of the ith cell. In this case, $M_i = 0$ indicates an empty cell, $M_i = 1$ indicates an occupied cell. The realization of the random variable M_i is denoted by m_i.

The robot is equipped with a range measurement sensor. Let the perspective range measurement be denoted by the random variable Z. The realization of the random variable

Table 1. The time complexity of all six algorithms proposed in this article, where n is the number of cells that the measurement intersects, Δ is the truncation length of Gaussian truncation, n_r is the number of classes of cells in a RLE compression of the measurement, and H is the width of the uniform distribution as measured by the number of cells that the uniform distribution intersects. The algorithms are classified depending on two factors: (i) their assumptions of sensor noise; and (ii) their encoding of measurements. The FSMI algorithm is an exact method that assumes Gaussian sensor noise, similar to existing algorithms in the literature. The Approx-FSMI algorithm is an approximate method that approximates Gaussian sensor noise by truncating the Gaussian distribution. The Uniform-FSMI algorithm approximates by assuming uniform sensor noise. The FSMI-RLE, Approx-FSMI-RLE, and Uniform-FSMI-RLE are their versions that represent measurements in run-length encoding, which is essential when working with large-scale maps for 3D environments. The table shows how various approximations (going down in the table) and RLE (going right in the table) reduce time complexity.

Time complexity of proposed algorithms	No encoding of measurement (2D)	Time complexity	RLE of measurement (3D)	Time complexity
Exact method assuming Gaussian sensor noise	FSMI *(Section 3.1)*	$O(n^2)$	FSMI-RLE *(Section 4.1)*	$O(n_r^2)$
Approximate method via Gaussian truncation	Approx-FSMI *(Section 3.4)*	$O(n\Delta)$	Approx-FSMI-RLE *(Section 4.2)*	$O(n_r\Delta)$
Exact method assuminguniform sensor noise	Uniform-FSMI *(Section 3.6)*	$O(n)$	Uniform-FSMI-RLE *(Section 4.3)*	$O(n_r H)$

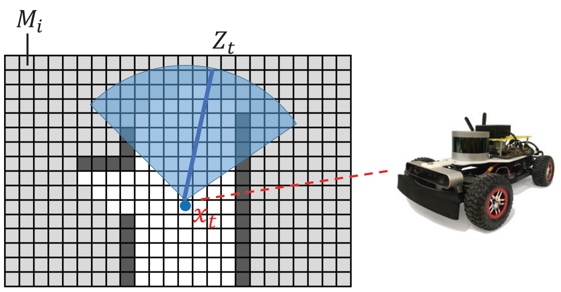

Fig. 1. A 2D environment is represented by an occupancy grid. Each cell in the map is associated with a binary random variable M_i indicating the occupancy of the cell. At time t, a vehicle can scan at location x_t with perspective range measurement being Z_t.

Z is denoted by z. The perspective range measurement at time t is denoted by Z_t, and its realization is denoted by z_t. The measurements obtained up to time t is denoted by $z_{1:t}$. Typically, the robot acquires multiple range measurements at the same time, e.g., measuring range in various directions. The sequence of range measurements obtained at time t is denoted by $z_t = (z_t^1, \ldots, z_t^{n_z})$ where n_z is the number of beams in a scan. Unless explicitly stated otherwise, we assume that the noise distribution of the sensor is a Gaussian with standard deviation σ regardless of the travel distance of the beam.

The state of the robot is denoted by x. In this article, the state variable is the pose of the robot. We denote the state at time t by x_t. We denote the sequence of states from the initial time through time t by $x_{1:t}$. The measurements obtained by the robot are a stochastic function of the map, the sensor model, and the state of the robot at that time as shown in Figure 1.

Assumption 1. **(**The standard independence assumption among the occupancy grid cells (Yamauchi, 1997)). *The occupancy cells are conditionally independent given the measurements, i.e.*,

$$P(M_1 = m_1, \ldots, M_K = m_k | z_{1:t}, x_{1:t}) = \prod_{1 \leq i \leq K} P(M_i = m_i | z_{1:t}, x_{1:t})$$

where $P(\cdot)$ is the probability function.

We assume that the robot has no prior information on the environment, that is, $P(M_i = 1) = P(M_i = 0) = 0.5$ for all $M_i \in M$. Once a measurement is obtained, the standard Bayesian filter can be used to update the occupancy grid (Julian et al., 2014):

$$\frac{P(M_i = 1|z_{1:t}, x_{1:t})}{P(M_i = 0|z_{1:t}, x_{1:t})} = \frac{P(M_i = 1|z_{1:(t-1)}, x_{1:(t-1)})}{P(M_i = 0|z_{1:(t-1)}, x_{1:(t-1)})} \frac{P(M_i = 1|z_t, x_t)}{P(M_i = 0|z_t, x_t)} \quad (1)$$

We denote the probability of occupancy of the ith cell by

$$o_i = P(M_i = 1|z_{1:t}, x_{1:t})$$

In addition, we denote the odds ratio of a cell by

$$r_i = o_i/(1 - o_i)$$

The Bayesian filter given by Equation (1) essentially updates r_i for the cells related to the acquired range measurements.

2.2. The Shannon MI metric

A key step of an information-theoretic mapping system is to select a series of scan locations that maximally reduce the conditional entropy of the map given the prospective measurements. In information theory, it is well known that the conditional entropy relates to the MI by

$$H(M|Z) = H(M) - I(M;Z) \tag{2}$$

Therefore, minimizing the conditional entropy $H(M|Z)$ is equivalent to maximizing the MI $I(M;Z)$.

The Shannon MI metric between the map M and the measurement Z is defined as follows:

$$I(M;Z) = \sum_{m \in \{0,1\}^K} \int_{z \geqslant 0} P(Z=z, M=m) \log \frac{P(Z=z, M=m)}{P(Z=z)P(M=m)} dz \tag{3}$$

In this section, we review the algorithm proposed by Julian et al. (2014) that computes the Shannon MI for single range measurement, such as a LiDAR beam. Throughout the article, we use the word beam and the phrase range measurement interchangeably to describe this measurement.

Strictly speaking, the prospective range measurements of two adjacent beams that have overlaps in the intersected occupancy cells are not independent, making the evaluation of the MI between all the prospective range measurements and the map prohibitively expensive. To tackle this, Julian et al. (2014) makes the following approximation among the occupancy cells and range measurements.

Assumption 2. (Decomposition of MI onto beams). *It is assumed that*

$$I(M; Z_\tau | x_{1:t}, z_{1:t}) \approx \sum_{j=1}^{n_z} I(M_i; Z_\tau^j | x_{1:t}, z_{1:t}) \tag{4}$$

This assumption allows Julian et al. (2014) to only focus on the MI evaluation for a single beam, $I(M_i; Z_\tau^j | x_{1:t}, z_{1:t})$.

For notational simplicity, we omit the conditional probability terms $x_{1:t}$ and $z_{1:t}$. Moreover, we use $M' = (M_1, \ldots, M_n)$ to represent the cells that this single range measurement intersects. By a slight abuse of notation, we use Z from now on to represent a prospective range measurement instead of all measurements in a scan. Note that a beam tells no information about the cells in M that it does not intersect. Therefore, $I(M';Z) = I(M;Z)$. We further assume that cells in M' are listed in ascending order by their distance from the sensor.

Julian et al. (2014) made another assumption to further simplify the computation.

Assumption 3. (Decomposition of MI onto occupancy cells). *It is assumed that*

$$I(M';Z) = \sum_i I(M_i;Z) \tag{5}$$

Under this assumption, Julian et al. (2014) derived an algorithm to compute the MI for a cell, $I(M_i;Z)$. Through accumulating $I(M_i;Z)$, they were able to evaluate the MI for a single beam, $I(M';Z)$.

Specifically, let $\delta_i(z)$ approximate the odds ratio of the inverse sensing model (Thrun et al., 2005) for the cell M_i:

$$\delta_i(z) = \begin{cases} \delta_{occ} & z \text{ indicates } M_i \text{ is occupied} \\ \delta_{emp} & z \text{ indicates } M_i \text{ is empty} \\ 1 & \text{otherwise} \end{cases} \tag{6}$$

where $\delta_{occ} > 1$ and $\delta_{emp} < 1$ are hyper-parameters. As shown by Julian et al. (2014), the MI between the beam and a single cell M_i is

$$I(M_i;Z) = \int_{z \geqslant 0} P(Z=z) f(\delta_i(z), r_i) dz \tag{7}$$

where $P(Z=z)$ is the measurement prior

$$P(Z=z) = P(e_0)P(Z=z|e_0) + \sum_{j=1}^{n} P(e_j)P(Z=z|e_j) \tag{8}$$

and $f(\delta_i(z), r_i)$ is the following function

$$f(\delta, r) = \log\left(\frac{r+1}{r+\delta^{-1}}\right) - \frac{\log \delta}{r\delta + 1} \tag{9}$$

Here with a slight abuse of notation, we define $P(e_j)$ to denote the probability that the jth cell is the first occupied cell on the beam and all the cells before the jth cell are empty. Similarly, we let $P(e_0)$ represent the probability that all cells are empty:

$$\begin{aligned} P(e_0) &= \prod_{l=1}^{n} (1 - o_l) \\ P(e_j) &= o_j \prod_{l<j} (1 - o_l), \quad \text{for all } j \geqslant 1 \end{aligned} \tag{10}$$

The function $P(Z=z|e_j)$, as a function of z, denotes the probability distribution of the range measurement, if the beam passes through the occupancy cells before the jth cell and is blocked by the jth cell. It is determined by the distance between the jth cell and the sensor as well as the previously discussed sensor noise model. For notational simplicity, we abbreviate $P(Z=z|e_j)$ as $P(z|e_j)$ in the discussion that follows, particularly in Section 3.2 where we prove the correctness of the main result.

As Equation (7) does not have a known analytical solution, Julian et al. (2014) evaluated it numerically by discretizing z:

$$I(M_i;Z) = \sum_z P(Z=z) f(\delta_i(z), r_i) \lambda_z^{-1} \tag{11}$$

where λ_z is the resolution for numerical integration. The MI then is computed as $\sum_{i=1}^{n} I(M_i;Z)$. The time complexity of this algorithm is $O(n^2 \lambda_z)$ (Julian et al., 2014). Keep in mind that this algorithm includes many approximations induced by Assumptions 1, 2, and 3. Nevertheless, we omit the label "approx" and refer to this algorithm as the *original MI* algorithm, because this algorithm is the target to optimize and to further approximate in this article.

2.3. The CSQMI metric

In addition to the Shannon MI metric, there exist other metrics that measure the dependency between two random variables; the CSQMI (Charrow et al., 2015b; Principe, 2010) is one of them.

Let $P(Z, M)$ be the joint probability distribution of Z and M, and $P(Z), P(M)$ be the probability distribution of Z and M, respectively. The CSQMI between M and Z is defined as

$$I_{CS}(M;Z) = D_{CS}(P(Z,M), P(Z)P(M)) \tag{12}$$

where $D_{CS}(\cdot,\cdot)$ is the Cauchy–Schwarz divergence between two probability distributions P_1 and P_2, defined as follows:

$$D_{CS}(P_1(X), P_2(X)) = -\frac{1}{2}\log\frac{\left(\int_x P_1(X=x)P_2(X=x)\,dx\right)^2}{\left(\int_x P_1(X=x)^2\,dx\right)\left(\int_x P_2(X=x)^2\,dx\right)} \tag{13}$$

Both the Shannon MI and CSQMI measure the difference between $P(M,Z)$ and $P(M)P(Z)$; Charrow et al. (2015b) showed that the MI map and the CSQMI map of the same occupancy grid look visually similar in practical robotics applications. Julian et al. (2014) established a theoretical guarantee that a robot will eventually explore the whole environment if it greedily chooses the viewpoint to greedily maximize the Shannon MI between the prospective measurement and the occupancy grid. To the best of the authors' knowledge, no such theoretical work has been done for CSQMI.

To evaluate the CSQMI efficiently, Charrow et al. (2015b) assumed Assumption 1, 3, and partially assumed Assumption 2 with a different approximation.

Under these assumptions, Charrow et al. (2015b) has shown that CSQMI for a beam that intersects with n cells can be evaluated analytically in $O(n^2)$ time. This is lower than the $O(n^2\lambda_z)$ complexity of Shannon MI, because numerical integration is avoided in CSQMI. In this article, FSMI is compared against CSQMI. For completeness, we reiterate the formula to compute CSQMI by (Charrow et al., 2015b)

$$I_{CS}(M;Z) = \log\sum_{l=0}^{C} w_l\mathcal{N}(0, 2\sigma^2) + \log\left(\prod_{i=1}^{n}(o_i^2 + (1-o_i)^2)\right.$$
$$\left.\sum_{j=0}^{n}\sum_{l=0}^{n} P(e_j)P(e_l)\mathcal{N}(\mu_l - \mu_j, 2\sigma^2)\right)$$
$$-2\log\sum_{j=0}^{n}\sum_{l=0}^{n} P(e_j)w_l\mathcal{N}(\mu_l - \mu_j, 2\sigma^2) \tag{14}$$

where $\mathcal{N}(x, \sigma^2)$ is the probability density function (PDF) at x of a normal distribution of zero mean and standard derivation of σ, μ_l is the distance from the center of the lth cell to the range sensor on the robot and

$$w_l = P(e_l)^2\prod_{j<l}(o_j^2 + (1-o_j)^2)$$

Charrow et al. (2015b) proposed a close approximation to CSQMI that truncates the tails of a Gaussian distribution. This enables CSQMI to be computed approximately in $O(n)$ time. Specifically, each double sum in Equation (14) can be approximated as follows:

$$\sum_{j=0}^{n}\sum_{l=j-\Delta}^{j+\Delta}\alpha_{j,l}\mathcal{N}(\mu_l - \mu_j, 2\sigma^2) \tag{15}$$

where $\alpha_{j,l}$ represents the corresponding coefficient in the double sum, and Δ is a small constant such as $\Delta = 3$.

2.4. Problem formulation

A typical information-theoretic exploration strategy is to generate a set of potential trajectories, evaluate MI along each of trajectory, and choose the one with the highest MI per travel cost (Charrow et al., 2015a,b; Nelson and Michael, 2015; Tabib et al., 2016).

In order to evaluate the MI along a trajectory, the trajectory is typically discretized in the state space, and MI is computed at each state and then summed, as shown in Figure 2.

It has been shown in previous research (Julian et al., 2014) that the fundamental problem of information-based exploration using range sensing is to efficiently evaluate the MI between the map and single range measurement. This problem is solved several times in typical MI-based exploration algorithms. We call this the *single-beam MI computation* problem.

When we tackle this problem, we follow Assumptions 1 and 2, but we only partially follow Assumption 3, which

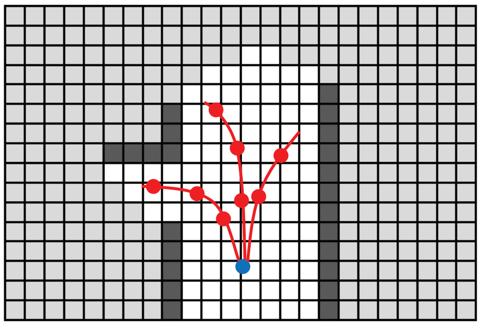

Fig. 2. Each candidate trajectory (marked by red) is discretized into a set of states. Shannon MI is evaluated between the perspective scan measurements at each state and the map, then summed up along the trajectory.

considers two adjacent beams independent even if they overlap with each other significantly on the occupancy grid. Instead, we follow the practice of Charrow et al. (2015b) and filter the beams to only keep those that do not overlap significantly with each other.

Under these assumptions, we consider a range measurement sensor, e.g., using a LiDAR beam emanating from the sensor. Again, we let $M' = (M_1, \ldots, M_n)$ denote the vector of binary random variables representing all the occupancy cells that the beam intersects. Let the random variable Z denote the corresponding range measurement. We wish to measure the dependence between these random variables using Shannon MI. The MI between the range measurement Z and a single cell M_i can be computed using Equation (7). Therefore, the MI between the prospective range measurement Z and all the cells in M is

$$\begin{aligned} I(M';Z) &= \sum_{i=1}^{n} I(M_i;Z) \\ &= \sum_{i=1}^{n} \int_{z \geqslant 0} P(Z=z) f(\delta_i(z), r_i)\, dz \\ &= \sum_{i=1}^{n} \int_{z \geqslant 0} P(Z=z) \left(\log\left(\frac{r+1}{r+\delta^{-1}} \right) - \frac{\log \delta}{r\delta + 1} \right) dz \end{aligned} \tag{16}$$

where $P(Z=z)$ is the measurement prior defined in Equation (8).

This article focuses on efficient algorithms and efficient approximations for computing the quantity defined in Equation (16). We first discuss the algorithms in two dimensions. Then, we propose an extension of the algorithms to the 3D mapping problem that further reduces the computation complexity using the structure of a 3D map.

3. The FSMI algorithm for 2D mapping

This section is devoted to the presentation of the FSMI algorithm for mapping in the 2D plane. The FSMI algorithm is presented in Section 3.1. The correctness of the algorithm is proved in Section 3.2, and its computational complexity is analyzed in Section 3.3. Efficient, practical implementations of the FSMI enabled by tabulation and approximation techniques are discussed in Section 3.4. Finally, efficient implementations for the case when the sensor noise follows a uniform distribution, instead of a Gaussian distribution, are discussed in Section 3.6.

3.1. The FSMI algorithm

In this section, we present the FSMI algorithm. The key idea behind the FSMI algorithm is the following: instead of performing the summation in Equation (16) directly, FSMI computes $I(M;Z)$ (same as $I(M';Z)$) holistically and analytically evaluates one of the resulting integrals, which leads to substantial computational savings.

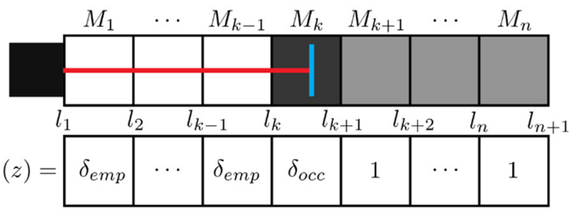

Fig. 3. Illustration of the key idea behind the proof of Lemma 1. The sensor beam (red) hits an obstacle (blue) in cell M_k. The value of the odds ratio inverse sensor model is shown at the bottom of the figure.

Algorithm 1 summarizes this procedure with subroutines in Algorithms 2 and 3. To describe the algorithms, let us first present our notation. Let l_i be the distance from the beam's origin to the ith cell where $l_1 = 0$ and l_i is monotonically increasing, as shown in the middle of Figure 3. Let us denote the center of the ith cell by $\mu_i = (l_i + l_{i+1})/2$. Just as in Equation (10), $P(e_j)$ is used to denote the probability that the jth cell is the first non-empty cell in M', i.e.,

$$P(e_j) = o_j \prod_{i<j} (1 - o_i)$$

Let $P(e_0)$ be the probability that all cells are empty. Here $\Phi(\cdot)$ denotes the standard normal cumulative distribution function (CDF). We also define

$$C_k = f(\delta_{occ}, r_k) + \sum_{i<k} f(\delta_{emp}, r_i) \tag{17}$$

where δ_{occ} and δ_{emp} are from the inverse sensor model defined in Equation (6) and

$$G_{j,k} = \int_{l_k}^{l_{k+1}} P(z|e_j)\, dz \tag{18}$$

In Algorithm 1, the total MI is initialized to zero (line 1). Then, Algorithms 2 and 3 are used to compute $P(e_j)$ for all $0 \leqslant j \leqslant n$ (line 2) and C_k for all $1 \leqslant k \leqslant n$ (line 3), respectively. Then, Algorithm 1 enumerates j, k through 0 to n, and for each pair (k, j) it computes $G_{k,j}$ and accumulates $P(e_j) C_k G_{k,j}$ into the total MI (lines 4–7).

Algorithm 1. The FSMI algorithm

```
Require: $\sigma$ and $l_i, o_i$ for $1 \leqslant i \leqslant n$.
1: $I \leftarrow 0$
2: Compute $P(e_j)$ for $0 \leqslant j \leqslant n$ with Algorithm 2.
3: Compute $C_k$ for $1 \leqslant k \leqslant n$ with Algorithm 3.
4: for $j = 0$ to $n$ do
5:     for $k = 0$ to $n$ do
6:         $G_{k,j} \leftarrow \Phi((l_{k+1} - \mu_j)/\sigma_j) - \Phi((l_k - \mu_j)/\sigma_j)$
7:         $I \leftarrow I + P(e_j) C_k G_{k,j}$
8: return $I$
```

Algorithm 2. Evaluate $P(e_j)$ for $0 \leqslant j \leqslant n$

```
Require: $o_i$ for $1 \leqslant i \leqslant n$
1: $E_0 \leftarrow 1$
2: for $j = 1$ to $n$ do
3:     $E_j \leftarrow E_{j-1}(1 - o_j)$
4:     $P(e_j) \leftarrow E_{j-1} o_j$
5: $P(e_0) = E_n$
6: return $P(e_j)$ for $0 \leqslant j \leqslant n$
```

Algorithm 3. Evaluate C_k for $1 \leqslant k \leqslant n$

```
Require: $\delta_{emp}, \delta_{occ}$ and $r_i$ for $1 \leqslant i \leqslant n$
1: $q_0 = 0$
2: for $k = 1$ to $n$ do
3:     $q_k = q_{k-1} + f(\delta_{emp}, r_k)$
4:     $C_k = q_{k-1} + f(\delta_{occ}, r_k)$
5: return $C_k$ for $1 \leqslant k \leqslant n$
```

3.2. Correctness of the FSMI algorithm

The following theorem states the correctness of the FSMI algorithm, that is, the FSMI algorithm indeed returns $I(M'; Z)$, i.e., the Shannon MI between the measurement Z and M', the part of the map the beam intersects with.

Theorem 1. (Correctness of FSMI). *The Shannon MI between the range measurement of the beam and all of the cells in M' is*

$$I(M'; Z) = \sum_{j=0}^{n} \sum_{k=1}^{n} P(e_j) C_k G_{k,j} \tag{19}$$

To prove this theorem, we need the following intermediate result regarding the structure of $f(\delta, r)$.

Let $F(z) = \sum_{i=1}^{n} f(\delta_i(z), r_i)$. The following lemma states that $F(z)$ is a piecewise-constant function.

Lemma 1. (Piecewise constant summation). *The function $F(z)$ is piecewise constant. In particular, if z lies in the kth cell, i.e., $l_k \leqslant z < l_{k+1}$, then $F(z) = C_k$ where $C_k = f(\delta_{occ}, r_k) + \sum_{i<k} f(\delta_{emp}, r_i)$.*

Proof. For $i < k$, a measurement of z implies that the beam has passed through M_i. Therefore, M_i should be empty and $\delta_i(z) = \delta_{emp}$. By definition, $\delta_k(z) = \delta_{occ}$. A measurement of z also indicates that the beam stops at cell k, which gives no information about cells M_j for $j > k$, so $f(\delta_j(z), r_j) = f(1, r_j) = 0$. Therefore, each term of $F(z)$ is constant for $l_k \leqslant z \leqslant l_k + 1$ and the sum is equal to the desired C_k, proving the lemma. □

Lemma 1 shows that the function $F(z)$ changes its value only at the cell boundaries, as z increases. (See Figure 3 for an illustration.) If we compute the MI between a range measurement and all the cells it can intersect at once, we can take advantage of this property to turn the integration in Equation (7) into a sum.

Using Lemma 1, we prove Theorem 1 as follows.

Proof Theorem 1. We begin with the total Shannon MI for one range measurement as stated in Equation (16). We substitute the definition of Shannon MI provided in Equation (7), and rearrange to reveal the sum described in Lemma 1. We arrive at the following expression:

$$\begin{aligned} I(M'; Z) &= \sum_{i=1}^{n} I(M_i; Z) = \sum_{i=1}^{n} \int_z P(z) f(\delta_i(z), r)\, dz \\ &= \sum_{i=1}^{n} \int_z \sum_{j=0}^{n} P(e_j) P(z|e_j) f(\delta_i(z), r)\, dz \\ &= \sum_{j=0}^{n} P(e_j) \int_z P(z|e_j) \left(\sum_{i=1}^{n} f(\delta_i(z), r_i) \right) dz \\ &= \sum_{j=0}^{n} P(e_j) \int_z P(z|e_j) F(z)\, dz \end{aligned} \tag{20}$$

Inspired by the result of Lemma 1, we divide the integration over z into a sum of multiple integration intervals across each cell boundary. This allows us to isolate the described term that is constant across each cell as follows:

$$\begin{aligned} I(M'; Z) &= \sum_{j=0}^{n} P(e_j) \sum_{k=1}^{n} \int_{l_k}^{l_{k+1}} P(z|e_j) C_k\, dz \\ &= \sum_{j=0}^{n} \sum_{k=1}^{n} P(e_j) C_k \int_{l_k}^{l_{k+1}} P(z|e_j)\, dz \\ &= \sum_{j=0}^{n} \sum_{k=1}^{n} P(e_j) C_k G_{k,j} \end{aligned} \tag{21}$$

This completes the proof. □

3.3. Computational complexity of the FSMI algorithm

We study the time complexity of Algorithm 1 with respect to n, the number of cells that a single range measurement intersects. The result is stated in the following theorem.

Theorem 2. *The time complexity of Algorithm 1 is $O(n^2)$.*

The proof of Theorem 2 is straightforward with the following intermediate results.

Lemma 2. *We can compute $P(e_j)$, $0 \leqslant j \leqslant n$, altogether in $O(n)$ with Algorithm 2.*

Lemma 3. *We can compute C_k, $1 \leqslant k \leqslant n$, altogether with Algorithm 3.*

Lemma 4. *The standard normal CDF, $\Phi(\cdot)$, can be evaluated with a look-up table. We can evaluate $G_{k,j}$ in $O(1)$ as follows*:

$$G_{k,j} = \Phi\left(\frac{l_{k+1} - \mu_j}{\sigma}\right) - \Phi\left(\frac{l_k - \mu_j}{\sigma}\right) \tag{22}$$

Note that, unlike the algorithm proposed in Julian et al. (2014), the FSMI algorithm does not perform any numerical integration. As a result, the complexity of FSMI outperforms the algorithm in Julian et al. (2014) by a factor of λ_z, the integration resolution.

Remark 1. *FSMI has the same time complexity as the exact version of CSQMI.*

Furthermore, because we assume the noise distribution has constant σ, we can directly precompute $\Phi(\frac{\cdot}{\sigma})$ to avoid one division operation per query.

Note that in addition to tabulating $\Phi(\cdot)$, we also build look-up tables for $f(\delta_{occ}, r_i)$ and $f(\delta_{emp}, r_i)$ for all i rather than evaluate them based on their definition as that can be computationally expensive. Specifically, we precompute $f(\delta_{occ}, r_i)$ and $f(\delta_{emp}, r_i)$ for a discrete set of values of r_i and store the results in a look-up table. We set $\delta_{emp}\delta_{occ} = 1$, following Julian et al. (2014); because $f(\delta_{emp}, r_i) = f(\delta_{occ}, 1/r_i)$, a single look-up table suffices.

3.4. Efficient implementations via Gaussian truncation

Charrow et al. (2015b) proposed to approximate the CSQMI metric by setting the tail of the Gaussian noise distribution to zero. We apply their technique to FSMI. Specifically, let Δ be the truncation width. We approximate Equation (19) as follows:

$$I(M'; Z) = \sum_{j=0}^{n} \sum_{k=j-\Delta}^{j+\Delta} P(e_j) C_k G_{k,j} \tag{23}$$

We refer to the variation of FSMI that applies the approximation described above as *Approx-FSMI*. The complexity of evaluating Approx-FSMI is $O(n\Delta)$. The value of Δ can be as small as 3 in practical problem instances, similar with the CSQMI algorithm (Charrow et al., 2015b).

3.5. Comparison against CSQMI

We refer to the corresponding CSQMI algorithm with the Gaussian truncation approximation as *Approx-CSQMI* (Charrow et al., 2015b). Both Approx-CSQMI and Approx-FSMI have the same complexity, $O(n\Delta)$. Nevertheless, this section argues that Approx-FSMI has two main advantages against Approx-CSQMI.

3.5.1. Computational speed. First, we argue that the constant factor of the two algorithms are different: the Approx-FSMI algorithm can be implemented so that it requires fewer computations than the Approx-CSQMI algorithm. Intuitively, this is because our computation in Equation (23) has only one double summation whereas the approximate version of CSQMI in Equation (14) has two similarly structured double summations of the same size.

We formalize this argument by counting the number of *floating point operations* (FLOPS) (IEEE, 2019) in both algorithms. Without loss of generality, we approximate the number of FLOPS by the number of multiplications in this comparison and omit operations such as additions, because they are significantly cheaper than multiplications on both general-purpose central processing units (CPUs) and field-programmable gate arrays (FPGAs) (Hennessy and Patterson, 2011; Rabaey et al., 2002). Other more expensive operations, such as $\log(\cdot)$, occur only a constant number of times in Equation (23) and the approximate version of Equation (14); hence, they do not contribute much to the evaluation time of either algorithm.

Theorem 3. (Number of multiplications). *Evaluating Approx-FSMI and Approx-CSQMI require* $(\Delta + 3)n$ *and* $(2\Delta + 9)n$ *multiplications, respectively.*

Note that the number of FLOPS is only an approximate metric of the computation speed. In practice, owing to the compiler optimizations such as single instruction, multiple data (SIMD) as well as memory caching, the relative speed between the Approx-FSMI algorithm and the Approx-CSQMI algorithm can vary. Nevertheless, having significantly smaller number of FLOPS empirically makes Approx-FSMI run faster; this is supported by the experimental results in Section 5.1.

The capability of computing MI twice as fast enables robots to make exploration decisions faster. Alternatively, it can also enable a robot to evaluate more exploration paths in a fixed amount of time, possibly leading to a shorter exploration path, as shown in Li et al. (2019).

3.5.2. Generalization to other noise models. Both the FSMI algorithm and the Approx-FSMI algorithm are derived assuming Gaussian noise model. Nevertheless, the proof of Theorems 1 and 2 suggest that the FSMI algorithm only requires the evaluation of the CDF of the underlying noise model. Hence, the FSMI algorithm generalizes to other noise distributions such as a Gaussian mixture distribution as long as the CDF of the distribution can be evaluated in $O(1)$. To the best of the authors' knowledge, the CSQMI algorithm is only derived for the Gaussian noise distribution.

3.6. Shannon MI under a uniform measurement noise model

Recall that the asymptotic time complexity of Approx-FSMI is $O(n\Delta)$. We observe that if the sensor noise can be approximated by a uniform distribution rather than a Gaussian distribution, under reasonable technical assumptions, the Shannon MI can be evaluated in $O(n)$, independently of Δ. Note that approximating a sensor noise model by a uniform distribution is not always practical; nevertheless we believe there is merit in presenting the algorithm for the uniform noise model because of the nice properties it has.

Algorithm 4. The Uniform-FSMI algorithm

```
Require: $H$ and $r_i$ for $1 \leqslant i \leqslant n$.
1: $I \leftarrow 0$
2: Compute $P(e_j)$ for $0 \leqslant j \leqslant n$ with Algorithm 3.1.
3: Compute $C_k$ for $1 \leqslant k \leqslant n$ with Algorithm 3.1.
4: $D_k \leftarrow 0$
5: for $k = 1$ to $n$ do
6:     $D_k \leftarrow D_{k-1} + C_k$
7: for $j = 0$ to $n$ do
8:     $I \leftarrow I + P(e_j)\frac{D_{\min(n,j+H)} - D_{\max(0,j-H-1)}}{2H+1}$
9: return $I$
```

Theorem 4. (Uniform-FSMI). *Suppose that cells have constant width. Again, let the boundary of the ith cell being l_i and l_{i+1} and for all i, $l_{i+1} - l_i = \Delta L$, which is a constant. Suppose that the sensor noise model is uniform and that the limits are quantized onto cell boundaries*:

$$P(Z|e_i) \sim U[l_i - H\Delta L,\ l_{i+1} + H\Delta L] \tag{24}$$

for $H \in \mathbb{Z}^+$ is a constant for the beam.

Let $D_i = \sum_{j \leqslant i} C_j$ *for* $1 \leqslant i \leqslant n$ *and* $D_i = 0$ *otherwise. Then, the Shannon MI between the beam and all the cells it intersects is*

$$I(M'; Z) = \sum_{j=0}^{n} P(e_j)\frac{D_{j+H} - D_{j-H-1}}{2H+1} \tag{25}$$

Proof. This proof follows the proof of Theorem 1 until Equation (20). There, we plug in the PDF of the uniform distribution:

$$\begin{aligned} I(M'; Z) &= \sum_{j=0}^{n}\sum_{k=1}^{n} P(e_j) C_k \int_{l_k}^{l_{k+1}} P(z|e_j)\,dz \\ &= \sum_{j=0}^{n} P(e_j) \sum_{k=j-H}^{j+H} \frac{C_k}{2H+1} \\ &= \sum_{j=0}^{n} P(e_j)\frac{D_{j+H} - D_{j-H-1}}{2H+1} \end{aligned} \tag{26}$$

This completes the proof. □

Remark 2. *Theorem 4 assumes all cells on a beam have the same width. In practice, owing to the discrete nature of the occupancy grid map, a beam that does not align with the axis of the grid map will intersect with different portions of different cells. This is typically handled by a ray-tracing algorithm, such as the Bresenham algorithm (Bresenham, 1977), which discretizes a beam into a series of occupancy cells of the same width. Although this is still an approximation, the loss is small and we can consider the assumption of Theorem 4 to be valid.*

Remark 3. *If the sensor does not strictly follow a uniform distribution, we can approximate it with a uniform distribution by matching the mean and variance of the two distributions. For example, if $P(z|e_i) \sim \mathcal{N}(\mu_i, \sigma)$, we can set $H = round\left(\sqrt{3}\sigma - 1/2\right)$.*

The algorithm to compute Uniform-FSMI is summarized in Algorithm 4. Its complexity is stated in the following theorem.

Theorem 5. (Time complexity of the Uniform-FSMI algorithm). *The time complexity of Algorithm 4 is $O(n)$.*

Proof. Line 2 computes all of $P(e_j)$ in $O(n)$. Line 3 computes all of C_k in $O(n)$ as well. The for-loop from line 5 to line 6 finishes in $O(n)$ time. The last for-loop from line 7 to line 8 computes the Shannon MI in $O(n)$. Therefore, the total time complexity of Algorithm 4 is $O(n)$. □

The time complexity of the Uniform-FSMI algorithm outperforms Approx-FSMI and Approx-CSQMI by a factor of Δ. In the experiments presented in Section 5, we demonstrate how this translates to an additional speedup of the Shannon MI computation in practice.

4. FSMI for a 3D environment using RLE

The algorithms and analysis provided in Section 3 focused on 2D environments. In this section, we are concerned with algorithms and representations that can handle 3D environments. A natural extension of the 2D occupancy map is a 3D voxel map (Moravec, 1996; Roth-Tabak and Jain, 1989). Indeed, the algorithms and analysis presented in Section 3 readily extend to 3D mapping problems using the voxel map. However, in most applications, the memory requirements for the voxel map exceed what is typically available on embedded computers today, which can be mounted on robots of smaller form factors. Hence, owing to the size of the map, the algorithms based on voxel maps will be relatively inefficient when used as-is for 3D mapping of large-scale environments.

Fortunately, in most applications, the 3D space that a robot navigates has a special structure: the empty spaces are typically large and continuous. The OctoMap data structure in Hornung et al. (2013) takes advantage of this structure. It compresses the map by using voxels of varying sizes. See Figure 4(a) for an illustration. Large homogeneous regions that would contain thousands of small cells can be represented with a single cell in an OctoMap representation. Thus, OctoMap representations have become widely used in mapping and exploration in 3D environments (Burri et al., 2015; Endres et al., 2014; Whelan et al., 2015).

To the best of the authors' knowledge, there exists no prior work that studies the computation of MI between range measurements and the environment represented by an OctoMap. For efficient computation, we represent each measurement using the RLE (Robinson and Cherry, 1967). Consider a measurement beam, as shown in Figure 4(a). Suppose we project this beam onto equally-sized "virtual cells," as shown in Figure 4(b). These virtual cells correspond to the real occupancy cells if we do not use the OctoMap to compress the occupancy map; OctoMap is

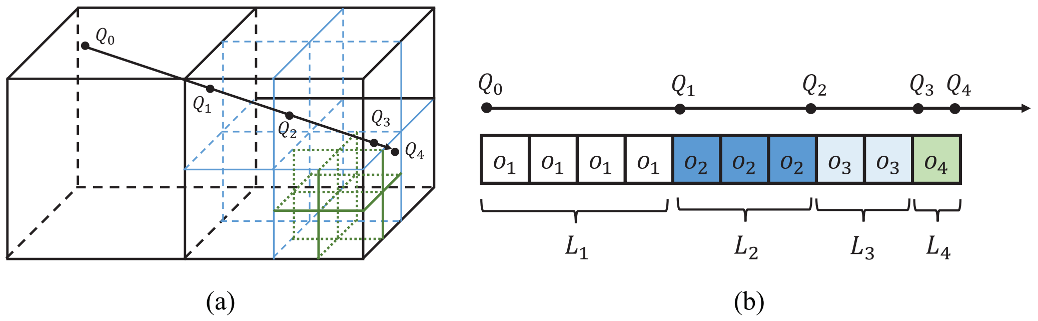

Fig. 4. An illustrative example of how an OctoMap representation adaptively represents an environment (a) and how ray-tracing on the OctoMap representation results in an occupancy sequence represented by RLE format (b).

after all a compression and the computation should always be defined on the uncompressed occupancy map. Then, the RLE encoding is a sequence of numbers that encodes each occupancy value together with the number of consecutive virtual cells with the same occupancy value. This simple compression method is valuable for representing measurements on the OctoMap structure in a way that we can apply the analysis presented in the previous section.

This section proposes a class of algorithms to compute the Shannon MI directly on this compressed sequence. We show that the time complexity of this algorithm is linear with respect to the length of the compressed vector. In practice, this translates to significant savings in computation time.

This section is organized as follows. Section 4.1 formalizes the problem of adapting FSMI to RLE and presents an efficient tabulation-based solution, namely the FSMI-RLE Algorithm. Section 4.2 discusses a numerical issue with the proposed solution and resolves it using Gaussian truncation, which we call the Approx-FSMI-RLE algorithm. Finally, Section 4.3 introduces a Uniform-FSMI-RLE algorithm for mapping in three dimensions when the sensor noise follows a uniform distribution. Readers who are more interested in using the algorithms can skip the proofs, the theorems, and the lemmas.

4.1. FSMI-RLE algorithm: Shannon MI on occupancy sequences compressed by RLE

In this section, we present the FSMI-RLE algorithm which computes the Shannon MI between a sensor beam and an array of cells whose occupancy values are compressed by RLE. After introducing our notation, Theorem 6 presents the main result. Algorithm 7 summarizes the computation according to the theorem.

Suppose the sequence consists of n total number of virtual cells, divided into n_r groups, each consisting of consecutive virtual cells with the same occupancy value. The ith group contains cells with the same occupancy probability, $o_i \in (0, 1)$. We define the number of cells in the ith group by $L_i \in \mathbb{Z}^+$. The total number of cells, n, relates to L_i by $\sum_{i=1}^{n_r} L_i = n$. Thus, the vector of occupancy values for the range measurement is

$$\{ \underbrace{o_1, \ldots, o_1}_{\text{repeated } L_1 \text{ times}}, \ldots, \underbrace{o_{n_r}, \ldots, o_{n_r}}_{\text{repeated } L_{n_r} \text{ times}} \} \tag{27}$$

Let $s_u = \sum_{i<u} L_i$ denote the index of the first cell in the uth group of cells. Let $P_E(u) = \prod_{i<u}(1 - o_i)^{L_i}$ denote the probability that all the cells on the beam before the first cell of the uth group of cells are empty. Also define $D_E(u) = \sum_{i<u} L_i f(\delta_{emp}, r_i)$. In addition, define the following:

$$\alpha[x, L_u, L_v] = \sum_{j=0}^{L_u-1}\sum_{k=0}^{L_v-1} x^j \exp\left(-\frac{(j-k)^2}{2\sigma'^2}\right) \tag{28}$$

$$\beta[x, L_u, L_v] = \sum_{j=0}^{L_u-1}\sum_{k=0}^{L_v-1} k \cdot x^j \exp\left(-\frac{(j-k)^2}{2\sigma'^2}\right) \tag{29}$$

where the variable $x \in [0, 1]$ is an occupancy probability, and the variables L_u, L_v represent the block sizes of the uth and vth groups of cells. Here α, β are auxiliary terms used for calculating the MI for sensor beams that are occluded by a cell in the uth group of cells but the measurements suggest that the beams fall in the vth group of cells owing to the sensor noise.

Figure 5 illustrates this notation. Define $\bar{o}_u = 1 - o_u$. Let w denote the width of a virtual cell, and define $\sigma' = \sigma/w$, where σ is the standard deviation of the noise distribution.

Theorem 6. (Shannon MI in RLE). *The Shannon MI $I(M'; Z)$ of a measurement represented in RLE can be expressed as follows:*

$$I(M'; Z) = \sum_{u=1}^{n_r}\sum_{v=1}^{n_r} \frac{P_E(u) o_u}{\sqrt{2\pi}\sigma'}((D_E(v) + f(\delta_{occ}, r_v))$$
$$A[\bar{o}_u, L_u, L_v, s_u - s_v] + f(\delta_{emp}, r_v) B[\bar{o}_u, L_u, L_v, s_u - s_v]) \tag{30}$$

where

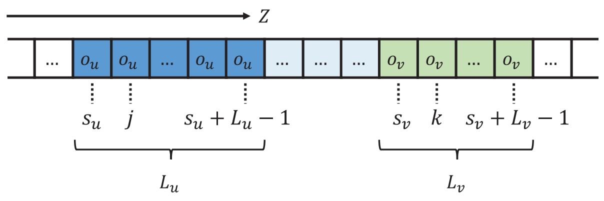

Fig. 5. Notation for FSMI computation on an occupancy sequence compressed by RLE.

$$A[x, L_u, L_v, t] = \sum_{j=0}^{L_u-1}\sum_{k=0}^{L_v-1} x^j \exp\left(-\frac{(j+t-k)^2}{2\sigma'^2}\right)$$
$$B[x, L_u, L_v, t] = \sum_{j=0}^{L_u-1}\sum_{k=0}^{L_v-1} k \cdot x^j \exp\left(-\frac{(j+t-k)^2}{2\sigma'^2}\right) \quad (31)$$

In addition, $A[x, L_u, L_v, t]$ *and* $B[x, L_u, L_v, t]$ *can be computed as*

$$A[x, L_u, L_v, t] = \begin{cases} x^{-t}(\alpha[x, L_u + t, L_v] - \alpha[x, t, L_v]), & \text{if } t \geqslant 1 \\ \alpha[x, L_u, L_v], & \text{if } t = 0 \\ (\alpha[x, L_u, L_v - t] - \alpha[x, L_u, -t]), & \text{if } t \leqslant -1 \end{cases} \quad (32)$$

and

$$B[x, L_u, L_v, t] = \begin{cases} x^{-t}(\beta[x, L_u + t, L_v] - \beta[x, t, L_v]) & \text{if } t \geqslant 1; \\ \beta[x, L_u, L_v] & \text{if } t = 0; \\ \beta[x, L_u, L_v - t] - \beta[x, L_u, -t] & \\ \quad + t \cdot A[x, L_u, L_v, t] & \text{if } t \leqslant -1. \end{cases} \quad (33)$$

Proof. First we prove the statement in Equation (30). We reorganize Equation (19) into double sum over groups of cells defined in Equation (27):

$$I(M'; Z) = \sum_{u=1}^{n_r}\sum_{v=1}^{n_r}\sum_{j=0}^{L_u-1}\sum_{k=0}^{L_v-1} P(e_{s_u+j}) C_{s_v+k} G_{s_u+j, s_v+k} \quad (34)$$

where

$$P(e_j) = o_j \prod_{l<j}(1 - o_l)$$

$$C_k = \sum_{l<k} f(\delta_{emp}, r_l) + f(\delta_{occ}, r_k)$$

$$G_{k,j} = \frac{1}{\sqrt{2\pi}\sigma'} \exp\left(-\frac{(k-j)^2}{2\sigma'^2}\right)$$

For a fixed u, for all j, we have

$$P(e_{s_u+j}) = P_E(u)(1 - o_u)^j o_u \quad (35)$$

Similarly,

$$C_{s_v+k} = D_E(v) + k \cdot f(\delta_{emp}, r_v) + f(\delta_{occ}, r_v) \quad (36)$$

Substituting Equations (35) and (36) into Equation (34) proves the statement in Equation (30).

Second, we prove the statement in Equation (32). When $t \geqslant 1$,

$$\begin{aligned} A[x, L_u, L_v, t] &= \sum_{j=0}^{L_u-1}\sum_{k=0}^{L_v-1} x^j \exp\left(-\frac{(j+t-k)^2}{2\sigma'^2}\right) \\ &= \sum_{j=0}^{L_u-1}\sum_{k=0}^{L_v-1} x^{-t} x^{j+t} \exp\left(-\frac{(j+t-k)^2}{2\sigma'^2}\right) \\ &= x^{-t}\sum_{j=0}^{L_u-1}\sum_{k=0}^{L_v-1} x^{j+t} \exp\left(-\frac{(j+t-k)^2}{2\sigma'^2}\right) \\ &= x^{-t}\sum_{j=t}^{L_u+t-1}\sum_{k=0}^{L_v-1} x^{j} \exp\left(-\frac{(j-k)^2}{2\sigma'^2}\right) \\ &= x^{-t}(\alpha[x, L_u + t, L_v] - \alpha[x, t, L_v]) \end{aligned}$$

Similarly, when $t \leqslant -1$,

$$\begin{aligned} A[x, L_u, L_v, t] &= \sum_{j=0}^{L_u-1}\sum_{k=0}^{L_v-1} x^j \exp\left(-\frac{(j+t-k)^2}{2\sigma'^2}\right) \\ &= \sum_{j=0}^{L_u-1}\sum_{k=-t}^{L_v-t-1} x^j \exp\left(-\frac{(j-k)^2}{2\sigma'^2}\right) \\ &= \alpha[x, L_u, L_v - t] - \alpha[x, L_u, -t]. \end{aligned}$$

Third, we prove the statement in Equation (33). The case for $t = 0$ is trivial. The case for $t \geqslant 1$ is similar to the case of A, above. When $t \leqslant -1$, we have

$$\begin{aligned} B[x, L_u, L_v, t] &= \sum_{j=0}^{L_u-1}\sum_{k=0}^{L_v-1} k \cdot x^j \exp\left(-\frac{(j+t-k)^2}{2\sigma'^2}\right) \\ &= \sum_{j=0}^{L_u-1}\sum_{k=0}^{L_v-1} (k - t + t) \cdot x^j \exp\left(-\frac{(j+t-k)^2}{2\sigma'^2}\right) \\ &= \sum_{j=0}^{L_u-1}\sum_{k=0}^{L_v-1} (k - t) \cdot x^j \exp\left(-\frac{(j-(k-t))^2}{2\sigma'^2}\right) \\ &\quad + \sum_{j=0}^{L_u-1}\sum_{k=0}^{L_v-1} t \cdot x^j \exp\left(-\frac{(j+t-k)^2}{2\sigma'^2}\right) \\ &= \sum_{j=0}^{L_u-1}\sum_{k=-t}^{L_v-t-1} k \cdot x^j \exp\left(-\frac{(j-k)^2}{2\sigma'^2}\right) \\ &\quad + \sum_{j=0}^{L_u-1}\sum_{k=0}^{L_v-1} t \cdot x^j \exp\left(-\frac{(j+t-k)^2}{2\sigma'^2}\right) \\ &= \beta[x, L_u, L_v - t] - \beta[x, L_u, -t] \\ &\quad + t \cdot A[x, L_u, L_v, t] \end{aligned}$$

□

Theorem 6 motivates an efficient algorithm that balances time complexity and space complexity. Specifically, it is realized by the *FSMI-RLE* algorithm, presented in Algorithms 5, 6, and 7, which can rapidly evaluate Shannon MI with a time complexity that is independent of the number of virtual cells, n, and without requiring a large amount of memory. In summary, the algorithm pre-computes and stores the functions $\alpha(x, L_u, L_v)$ and $\beta(x, L_u, L_v)$ in look-up tables, and uses Equation 30 to combine these values to produce the Shannon MI. When tabulating the functions $\alpha(x, L_u, L_v)$ and $\beta(x, L_u, L_v)$ into look-up tables, the algorithm quantizes the occupancy probability variable $x \in [0, 1]$ into a finite set of values,[1] which we denote by $\mathcal{X}$.

It is easy to see that the functions $\alpha(x, L_u, L_v)$ and $\beta(x, L_u, L_v)$ are polynomials in x, hence continuous in x; thus, they can be approximated arbitrarily well by taking the resolution of $\mathcal{X}$ fine enough. Hence, the algorithm is still an exact algorithm for computing Shannon MI, as it can approximate the result with arbitrary accuracy.

The correctness of Algorithm 7 is provided in the following.

Theorem 7. (Correctness of FSMI-RLE). *Algorithm 7 computes $I(M'; Z)$ in Equation (30).*

Proof. Algorithms 5 and 6 correctly computes $P_E(u)$ and $D_E(v)$; they are called in lines 5 and 6 in Algorithm 7 to calculate $P_E(u)$ and $D_E(v)$ for all of $u, v, 1 \leqslant u, v \leqslant n_r$. Then in Algorithm 7, lines 7 and 9 enumerate u, v from 1 to n_r; inside the loop, the computation follows Equation (30) except for the normalization, which happens outside the loop in line 13. This completes the proof. □

The time and space complexity of the algorithm are provided in the following.

Theorem 8. (Time complexity of FSMI-RLE). *The time complexity of the FSMI-RLE algorithm is $O(n_r^2)$.*

Proof. Algorithms 5 and 6 only loops through 1 to n_r once; both have a complexity of $O(n_r)$. Although we put the tabulation of α and β inside Algorithm 7 for completeness, in the actual implementation the table is precomputed once outside Algorithm 7. The rest of the algorithm is a double for-loop to compute Equation (30), resulting in a time complexity of $O(n_r^2)$. Therefore, the overall time complexity is $O(n_r^2)$. □

Theorem 9. (Space complexity of FSMI-RLE). *The space complexity of the FSMI-RLE algorithm is $O(|\mathcal{X}|n^2)$.*

Proof. Except for the two tables α and β, all the other variables in Algorithm 7 store at most n_r real numbers. Each of α and β tabulates against x, L_u, L_v. Note that x is quantized into $|\mathcal{X}|$ entries and $1 \leqslant L_u, L_v \leqslant n_r$; hence, α and β each stores exactly $|\mathcal{X}|n_r^2$ real values. Therefore, the space complexity of Algorithm 7 is $O(|\mathcal{X}|n_r^2)$. This completes the proof. □

First, note that the time complexity depends only on the number of groups of cells n_r, i.e., the size of the RLE, but not on the number of virtual cells n. Note that n_r is significantly smaller than n because of the RLE compression. Hence, reducing the time complexity from $O(n^2)$ to $O(n_r^2)$ translates to substantial savings in MI computation when the OctoMap achieves reasonable compression of the 3D environment.

Second, let us note that closed-form solutions to A and B in Equations (6) and (31) may reduce space complexity of the algorithm. However, the authors were unable to find closed-form solutions without any approximations. With approximation, we are able to derive closed-form solutions. Unfortunately, despite their constant time and space complexity the closed-form solutions are so complex that its runtime exceeds that of the tabulated algorithms in all practical scenarios that we considered. See the Appendix for our closed-form solutions. Owing the inefficiency of the closed-form solutions, we do not include use them in FSMI-RLE and opt for the tabulated algorithms. Finding closed-form solutions that can be evaluated more efficiently remains an open problem.

Third, unfortunately, the FSMI-RLE algorithm suffers from numerical issues in some problem instances. Consider a case when there are two groups each with a very large number of virtual cells, say L_u and L_v are very large numbers for some u and v, $u>v$. Recall that $A[\bar{o}_u, L_u, L_v, t]$ is the product of $(\bar{o}_u)^{-t}$ and $\alpha[\bar{o}_u, L_u + t, L_v] - \alpha[\bar{o}_u, t, L_v]$. As $t = s_u - s_v \geqslant L_v$, the variable t also becomes large. Hence, when $\bar{o}_u$ is close to zero, $\bar{o}_u^{-t}$ becomes a very large number. At the same time, $(\alpha[\bar{o}_u, L_u + t, L_v] - \alpha[\bar{o}_u, t, L_v])$ becomes a very small number. As a result, in some cases, the multiplication of these numbers cannot be completed correctly owing to the precision limitations of multiplication on computing hardware. The authors believe that the numerical issues can be avoided by tabulating the functions A and B into look-up tables, instead of the functions α and β. However, unfortunately, tabulating A and B requires orders of magnitude more memory, owing to the additional integer variable t that they encompass. Instead, in the next section, we propose an approximate algorithm for computing Shannon MI, based on truncating of the Gaussian distribution, which avoids the numerical issues while maintaining computational efficiency. The key insight behind the numerical stability of the approximation is that the variable t never becomes too large as a result of the truncation.

4.2. Approx-FSMI-RLE algorithm: approximating Shannon MI in RLE via truncation

In this section, we present an approximate version of the algorithm presented in the previous section. The approximation is achieved by truncating the Gaussian distribution,

Algorithm 5. Evaluate $P_E(u)$ for $1 \leqslant u \leqslant n_r$

```
Require: $o_u, L_u$ for $1 \leqslant u \leqslant n_r$
1: $P_E(1) \leftarrow 1$
2: for $u = 2$ to $n_r$ do
3:     $P_E(u) \leftarrow P_E(u-1)(1 - o_{u-1})^{L_{u-1}}$
4: return $P_E(u)$ for $1 \leqslant u \leqslant n_r$
```

Algorithm 6. Evaluate $D_E(v)$ for $1 \leqslant v \leqslant n_r$

```
Require: $r_v$, $L_v$ for $1 \leqslant v \leqslant n_r$
1: $D_E(1) \leftarrow 0$
2: for $v = 2$ to $n_r$ do
3:     $D_E(v) \leftarrow D_E(v-1) + L_{v-1} f(\delta_{emp}, r_{v-1})$
4: return $D_E(v)$ for $1 \leqslant v \leqslant n_r$
```

Algorithm 7. The FSMI-RLE algorithm

```
Require: Noise standard derivation $\sigma$, cell width $w$ and $s_i, L_i, o_i, r_i$ for $1 \leqslant i \leqslant n_r$.
1: $I \leftarrow 0$
2: $\sigma' = \sigma / w$
3: Compute $P_E(u)$ for $1 \leqslant u \leqslant n_r$ with Algorithm 5.
4: Compute $D_E(v)$ for $1 \leqslant v \leqslant n_r$ with Algorithm 6.
5: Tabulate $\alpha[x, L_u, L_v]$.
6: Tabulate $\beta[x, L_u, L_v]$.
7: for $u = 1$ to $n_r$ do
8:     $x \leftarrow \text{floor}((1 - o_u)/o_{res}) o_{res}$
9:     for $v = 1$ to $n_r$ do
10:        Compute $A[x, L_u, L_v, t]$ based on Equation (32).
11:        Compute $B[x, L_u, L_v, t]$ based on Equation (33).
12:        $I \leftarrow I + P_E(u) o_u ((D_E(v) + f(\delta_{occ}, r_v))A + f(\delta_{emp}, r_v))B)$
13: $I \leftarrow I / (\sqrt{2\pi}\sigma')$
14: return $I$
```

similar to the approximation presented in Section 3.4. In brief, we wish to obtain an efficient algorithm that evaluates Shannon MI as in Equation (23), which was obtained from Equation (19) by setting $G_{k,j} = 0$ for all k, j with $|k - j| > \Delta$.

In the rest of this section, we first state the truncated parallel of Theorem 6. Then, we present the Approx-FSMI-RLE algorithm, which approximates the FSMI-RLE algorithm via Gaussian truncation. Next, we prove the correctness of the Approx-FSMI-RLE algorithm and then analyze its computational complexity.

Recall that, for every two groups of virtual cells, say u and v, the variables s_u, s_v denote the index for the first virtual cell of its group and the variables L_u, L_v denote the number of virtual cells in that group. When we evaluate Shannon MI using Equation (30), most terms in A and B will be equal to zero owing to Gaussian truncation. There are three cases to consider. First, two groups u and v might be further than Δ away, in which case all terms in A and B evaluate to zero. See Figure 6 for an illustration. Second, two groups are close to each other and long enough, so that only a subset of the terms are zero. See Figure 7(a) for an illustration. In this case, we consider their sub-groups with non-zero elements. We denote the starting index of the sub-groups by s'_u, s'_v, and we denote the length of these sub-groups by L'_u, L'_v. Third, the two groups closer and shorter than Δ. See Figure 7(b) for an illustration. In this case, all terms are non-zero.

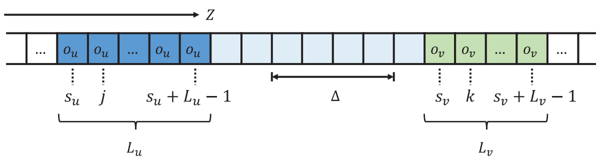

Fig. 6. When the condition in Lemma 5 is violated, Gaussian truncation sets the Shannon MI contribution between these two blocks to zero; otherwise, the Shannon MI contribution is non-zero.

The following theorem presents the main result of this section. This result establishes a concise formula to compute the Shannon MI after the truncation of the Gaussian distribution. Most importantly, it saves computation by eliminating terms that evaluate to zero. In particular, it provides the formulas to compute the variables s'_u, s'_v, L'_u, and L'_v, which we referenced previously.

Theorem 10. (Approx-FSMI-RLE algorithm). *If $G_{k,j} = 0$ when $|k - j| > \Delta$, the Shannon MI can be evaluated as*

$$I(M'; Z) = \frac{I_{u<v} + I_{u>v} + I_{u=v}}{\sqrt{2\pi}\sigma'} \tag{37}$$

where

$$I_{u<v} = \sum_{u=1}^{n_r} \sum_{\substack{v>u \\ s_v < s_u + L_u + \Delta}} P_E(u)(1 - o_u)^{s'_u(u,v) - s_u} o_u((D_E(v) f(\delta_{emp}, r_v) \\ + f(\delta_{occ}, r_v))A[x, L'_u(u,v), L'_v(u,v), s_u - s_v] \\ + f(\delta_{emp}, r_v)B[x, L_{u'}(u,v), L_{v'}(u,v), s_u - s_v]) \tag{38}$$

$$I_{u>v} = \sum_{u=1}^{n_r} \sum_{\substack{v<u \\ s_u < s_v + L_v + \Delta}} P_E(u) o_u((D_E(v) + (s'_v(u,v) - s_v) f(\delta_{emp}, r_v) \\ + f(\delta_{occ}, r_v))A[x, L'_u(u,v), L'_v(u,v), s_u - s_v] \\ + f(\delta_{emp}, r_v)B[x, L_{u'}(u,v), L_{v'}(u,v), s_u - s_v]) \tag{39}$$

and

$$I_{u=v} = \sum_{u=1}^{n_r} P_E(u) o_u (D_E(u) + f(\delta_{occ}, r_u))\theta[x, L_u] \\ + f(\delta_{emp}, r_u))\gamma[x, L_u]) \tag{40}$$

where

$$\theta[x, L] = \alpha[x, L, L]$$

$$\gamma[x, L] = \beta[x, L, L]$$

The $L'_u(u,v), L'_v(u,v), s'_u(u,v), s'_v(u,v)$ that appeared in the above equations are defined as

$$s'_u(u,v) = \begin{cases} \max(s_u, s_v - \Delta) & u < v \\ s_u & u > v \end{cases} \tag{41}$$

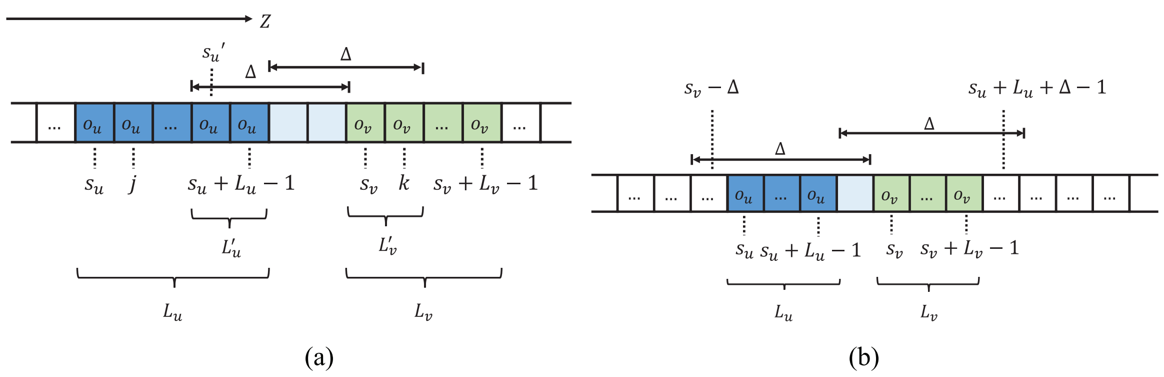

Fig. 7. Gaussian truncation may change the start and the length of a pair of blocks of cell, depending on their relative distance and group length. (a) Start location and length of the blocks changed by Gaussian truncation. (b) Start location and length of the blocks unaffected by Gaussian truncation.

$$L'_u(u,v) = \begin{cases} s_u + L_u - s'_u(u,v) & u<v \\ \min(L_u, s_v + L_v + \Delta - s_u) & u>v \end{cases} \tag{42}$$

$$s'_v(u,v) = \begin{cases} s_v & u<v \\ \max(s_v, s_u - \Delta) & u>v \end{cases} \tag{43}$$

$$L'_v(u,v) = \begin{cases} \min(L_v, s_u + L_u + \Delta - s_v) & u<v \\ s_v + L_v - s'_v(u,v) & u>v \end{cases} \tag{44}$$

To prove this theorem, we first establish a necessary and sufficient condition for $A[x, L_u, L_v, t] \neq 0$ and $B[x, L_u, L_v, t] \neq 0$ to hold after the truncation.

Lemma 5. (Non-zero MI block pair after Gaussian truncation). *Let $t = s_u - s_v \neq 0$. If $G_{k,j} = 0$ when $|k-j| > \Delta$, then $A[x, L_u, L_v, t] \neq 0$ if and only if $u = v$, or $u<v, s_v < s_u + L_u + \Delta$ or $v<u, s_u < s_v + L_v + \Delta$ The same holds for $B[x, L_u, L_v, t]$.*

Proof. Figure 6 illustrates the case when $u<v$. The case of $u>v$ is symmetric. When $u=v$ Gaussian truncation cannot set all terms in A and B to zero. This completes the proof. □

This lemma enables us to skip computations for any pair of (u, v) that does not satisfy the above condition. Based on this lemma, now we prove Theorem 10.

Proof of Theorem 10. We first prove the statement in Equations (41)–(44). As u and v are symmetric, we only discuss the case of $u<v$. See Figure 7 for an illustration.

The formula for $s'_u(u,v)$, the new start position of the left block, is different depending on whether the two blocks belong to the case of Figure 7(a) or Figure 7(b). In the former case $s'_u(u,v) = s_v - \Delta$, whereas in the latter case $s'_u(u,v) = s_u > s_v - \Delta$. Combining them into one formula, we have $s'_u(u,v) = \max(s_u, s_v - \Delta)$.

With $s'_u(u,v)$, to get $L'_u(u,v)$ we simply subtract the number of truncated cells, $s'_u(u,v) - s_u(u,v)$, from L_u, i.e., $L'_u(u,v) = L_u - (s'_u(u,v) - s_u) = s_u + L_u - s'_u(u,v)$.

Gaussian truncation does not affect the start position of the group of cells further away from the scanning position. As in this case $u<v$, the start position of the vth group stays the same, i.e., $s'_v(u,v) = s_v$.

The derivation for $L'_v(u,v)$ is similar to the derivation for $s'_u(u,v)$. If it is the case of Figure 6, we have $L'_v(u,v) = s_u + L_u + \Delta - s_v$. Otherwise, $L'_v(u,v) = L_v < s_u + L_u + \Delta - s_v$. Combining the two cases, we get $L'_v(u,v) = \min(L_v, s_u + L_u + \Delta - s_v)$.

The same proof applies to the case of $u>v$. Hence, we have proven Equations (41)–(44).

The proof for the top part of this theorem follows Theorem 6. After the Gaussian truncation, s_u, s_v, L_u, L_v becomes $s'_u(u,v), s'_v(u,v), L'_u(u,v), L'_v(u,v)$. As a result, $P_E(u)$ will change when $s'_u \neq s_u$; let us denote the updated value by $P'_E(u)$. Similarly, the value of $D_E(v)$ also changes when $s'_v \neq s_v$ and we denote it by $D'_E(v)$.

For $P'_E(u)$, we note that $s'_u(u,v) = s_u$ when $u>v$ according to Equation (41); hence, we have

$$P'_E(u) = \begin{cases} P_E(u)(1-o_u)^{s'_u(u,v)-s_u} & u<v \\ P_E(u) & u \geqslant v \end{cases} \tag{45}$$

Similarly, for $D'_E(v)$, we derive the following equation based on Equations (43) and (44):

$$D'_E(v) = \begin{cases} D_E(v) & u \leqslant v \\ D_E(v) + (s'_v(u,v) - s_v) f(\delta_{emp}, r_v) & u>v \end{cases} \tag{46}$$

Substituting Equations (45) and (46) into Equation (30) and separating the cases for $u<v$, $u=v$ and $u>v$, we prove the theorem. □

Theorem 10 motivates the Approx-FSMI-RLE algorithm, which we present in Algorithm 8. The algorithm creates look-up tables for the functions α, β, θ, and γ using Algorithms 9 and 10 in lines 5 and 6. Then, the algorithm proceeds with computing Shannon MI based on Equation (37) in lines 7–27. Line 8 handles the case when $u=v$ by applying the formula in Equation (40). Lines 9–17 handle the case when $u<v$ by applying the formula in Equation

(38). Finally, lines 18–26 handle the case when $u > v$ by applying the formula in Equation (39).

The following theorem establishes the correctness of the Approx-FSMI-RLE algorithm. The proof of the algorithm is evident from the previous description.

Theorem 11. (Correctness of Approx-FSMI-RLE). *Algorithm 8 computes $I(M'; Z)$ in Equations (37)–(39).*

The following two theorems establish the computational complexity of the Approx-FSMI-RLE algorithm.

Theorem 12. (Time complexity of the Approx-FSMI-RLE algorithm). *The time complexity of the Approx-FSMI-RLE algorithm is $O(\Delta n_r)$.*

Proof. First, the computation of $P_E(u)$ in line 3 and $D_E(v)$ in line 4 of Algorithm 8 takes $O(n_r)$ in total to complete. Then we note that the tabulation of $\alpha, \beta, \theta, \gamma$ in lines 5 and 6 in practice takes place outside the algorithm for a single beam; these tables are filled only once at the start of an exploration task so that they will not be computed again for the evaluation of Shannon MI on any single beam once the exploration starts.

The for-loop over u from line 7 to line 26 enumerates u from 1 to n_r. The computation consists of three parts, corresponding to $I_{u=v}$, $I_{u<v}$, and $I_{u>v}$ in Theorem 10. Line 8 handles the case of $v = u$ and it costs $O(1)$. Lines 9–17 handle the case of $u < v$ and loop through all $v \in \{v | v > u \,\text{and}\, s_v < s_u + L_u + \Delta\}$. Note that $s_u + L_u$ is the index of the start of the next group of cells right of the uth group of cells. Thus, we have $s_v \geqslant s_u + L_u$. As $s_v < s_u + l_u + \Delta$, the number of valid v is bounded by $(s_u + L_u + \Delta) - (s_u + L) = \Delta$. The computation inside the for-loop in lines 9–17 is $O(1)$, hence, the complexity of this for-loop is $O(\Delta)$. Similarly the complexity for the for-loop in lines 18–26 that handle the case of $u > v$ is also $O(\Delta)$. Thus, the overall complexity for the for-loop in lines 7–27 is $O(\Delta n_r)$.

Therefore, the overall complexity of the whole algorithm is $O(n_r + \Delta n_r) = O(\Delta n_r)$. □

Theorem 13. (Space complexity of Approx-FSMI-RLE algorithm). *The space complexity of the Approx-FSMI-RLE algorithm is $O\big(|\mathcal{X}|(n + \Delta^2)\big)$.*

Proof. The main memory of Algorithm 8 is used to store the tables $\alpha[x, L_u, L_v], \beta[x, L_u, L_v]$ and $\theta[x, L_u], \gamma[x, l_u]$. Compared with it, the memory for all other variables has at most $O(n_r)$ space complexity, which is negligible. For both α and β, we have $1 \leqslant L_u, L_v \leqslant \Delta$ because of Gaussian truncation. For θ, γ, we have $1 \leqslant L_u \leqslant n$. As $x \in \mathcal{X}$, the overall space complexity of Algorithm 8 is $O\big(|\mathcal{X}|(n + \Delta^2)\big)$. □

Algorithm 8. The Approx FSMI-RLE algorithm

Require: Noise standard derivation σ, cell width w and s_i, L_i, o_i, r_i for $1 \leqslant i \leqslant n_r$, quantization resolution o_{res}, Gaussian truncation width Δ, maximal length of a group of cell L_M.

```
 1: $I \leftarrow 0$
 2: $\sigma' = \sigma / w$
 3: Compute $P_E(u)$ for $1 \leqslant j \leqslant n_r$ with Algorithm 5.
 4: Compute $D_E(v)$ for $1 \leqslant k \leqslant n_r$ with Algorithm 6.
 5: Tabulate $\alpha[x, L_u, L_v]$ and $\theta[x, L_u]$ with Algorithm 9 where $L_{bound} = \Delta$ and $L_{max} = L_M$.
 6: Tabulate $\beta[x, L_u, L_v]$ and $\gamma[x, L_u]$ with Algorithm 10 where $L_{bound} = \Delta$ and $L_{max} = L_M$.
 7: **for** $u = 1$ **to** $n_r$ **do**
 8:     $I \leftarrow I + P_E(u) o_u Z(D_E(u) + f(\delta_{occ}, r_v))\theta[x, L_u] + f(\delta_{emp}, r_u))\gamma[x, L_u]$
 9:     **for** $v \in \{v | v > u \,\text{and}\, s_v < s_u + L_u + \Delta\}$ **do**
10:         $s'_u \leftarrow \max(s_u, s_v - \Delta)$
11:         $L'_u \leftarrow s_u + L_u - s_u(u, v)'$
12:         $s'_v \leftarrow s_v$
13:         $L'_v \leftarrow \min(L_v, s_u + L_u + \Delta - s_v)$
14:         $t \leftarrow s'_u - s'_v$
15:         $A = x^{-t}(\alpha[x, L'_u + t, L'_v] - \alpha[x, t, L'_v])$
16:         $B = x^{-t}(\beta[x, L'_u + t, L'_v] - \beta[x, t, L'_v])$
17:         $I \leftarrow I + P_E(u)(1 - o_u)^{s'_u - s_u} o_u((D_E(v) + f(\delta_{occ}, r_v))A + f(\delta_{emp}, r_v)B)$
18:     **for** $v \in \{v | v < u \,\text{and}\, s_u < s_v + L_v + \Delta\}$ **do**
19:         $s'_u \leftarrow s_u$
20:         $L'_u \leftarrow \min(L_u, s_v + L_v + \Delta - s_u)$
21:         $s'_v \leftarrow \max(s_v, s_u - \Delta)$
22:         $L'_v \leftarrow s_v + L_v - s'_v$
23:         $t \leftarrow s'_u - s'_v$
24:         $A = x^{-t}(\alpha[x, L'_u + t, L'_v] - \alpha[x, t, L'_v])$
25:         $B = x^{-t}(\beta[x, L'_u + t, L'_v] - \beta[x, t, L'_v])$
26:         $I \leftarrow I + P_E(u) o_u((D_E(v) + (s'_v - s_v) f(\delta_{emp}, r_v) + f(\delta_{occ}, r_v))A + f(\delta_{emp}, r_v))B)$
27:     $I \leftarrow I / (\sqrt{2\pi}\sigma')$
28:     **return** $I$
```

Algorithm 9. Tabulating $\alpha[x, L_u, L_v]$ and $\theta[x, L_u]$

```
Require: $x_{res}$, $\sigma'$, $L_{bound}$, $L_{max}$
 1: $N_x \leftarrow \text{floor}(1/x_{res})$
 2: for $i = 0$ to $N_x$ do
 3:     $x \leftarrow i \cdot x_{res}$
 4:     $\alpha[x, 1, 1] \leftarrow 1$
 5:     for $L_u = 2$ to $L_{bound}$ do
 6:         $\alpha[x, L_u, 1] = \alpha[x, L_u - 1, 1] + \exp(-\frac{(L_u-1)^2}{2\sigma'^2})$
 7:     for $L_v = 2$ to $L_{bound}$ do
 8:         $\alpha[x, 1, L_v] = \alpha[x, 1, L_v - 1] + x^{L_v-1}\exp(-\frac{(1-L_v)^2}{2\sigma'^2})$
 9:     for $L_u = 2$ to $L_{bound}$ do
10:         for $L_v = 2$ to $L_{bound}$ do
11:             $\alpha[x, L_u, L_v] = \alpha[x, L_u, L_v - 1] + \alpha[x, L_u - 1, L_v] - \alpha[x, L_u - 1, L_v - 1] + x^{L_v-1}\exp(-\frac{(L_u - L_v)^2}{2\sigma'^2})$
12:     for $L = 1$ to $L_{bound}$ do
13:         $\theta[x, L] = \alpha[x, L, L]$
14:     for $L = L_{bound} + 1$ to $L_{\max}$ do
15:         $\theta[x, L] \leftarrow \theta[x, L - 1]$
16:         for $i = 1$ to $L - 1$ do
17:             $\theta[x, L] \leftarrow \theta[x, L] + x^{L-1}\exp(-\frac{(i-L)^2}{2\sigma'^2})$
18:             $\theta[x, L] \leftarrow \theta[x, L] + x^{i-1}\exp(-\frac{(i-L)^2}{2\sigma'^2})$
19:         $\theta[x, L] \leftarrow \theta[x, L] + x^{L-1}$
20: return $\alpha$ and $\theta$
```

Algorithm 10. Tabulating $\beta[x, L_u, L_v]$ and $\gamma[x, L_u]$

```
Require $x_{res}$, $\sigma'$, $L_{bound}$, $L_{max}$
 1: $N_x \leftarrow \text{floor}(1/x_{res})$
 2: for $i = 0$ to $N_x$ do
 3:     $x \leftarrow i \cdot x_{res}$
 4:     $\beta[x, 1, 1] \leftarrow 0$
 5:     for $L_u = 2$ to $L_{bound}$ do
 6:         $\beta[x, L_u, 1] \leftarrow 0$
 7:     for $L_v = 2$ to $L_{bound}$ do
 8:         $\beta[x, 1, L_v] = \beta[x, 1, L_v - 1] + (L_v - 1)\exp\left(-\frac{(1-L_v)^2}{2\sigma'^2}\right)$
 9:     for $L_u = 2$ to $L_{bound}$ do
10:         for $L_v = 2$ to $L_{bound}$ do
11:             $\beta[x,L_u,L_v] = \beta[x,L_u,L_v-1] + \beta[x,L_u-1,L_v] - \beta[x,L_u-1,L_v-1] + (L_v-1)x^{L_u-1}\exp\left(-\frac{(L_u-L_v)^2}{2\sigma'^2}\right)$
12:     for $L = 1$ to $L_{bound}$ do
13:         $\gamma[x, L] = \beta[x, L, L]$
14:     for $L = L_{bound} + 1$ to $L_{\max}$ do
15:         $\gamma[x, L] \leftarrow \gamma[x, L - 1]$
16:         for $i = 1$ to $L - 1$ do
17:             $\gamma[x, L] \leftarrow \gamma[x, L] + (i - 1)x^{L-1}\exp\left(-\frac{(i-L)^2}{2\sigma'^2}\right)$
18:             $\gamma[x, L] \leftarrow \gamma[x, L] + (L - 1)x^{i-1}\exp\left(-\frac{(i-L)^2}{2\sigma'^2}\right)$
19:         $\gamma[x, L] \leftarrow \gamma[x, L] + (L - 1)x^{L-1}$
20: return $\beta$ and $\gamma$
```

4.3. Uniform-FSMI-RLE algorithm: Shannon MI in RLE assuming a uniform distribution for sensor noise

In this section, we discuss an algorithm that computes Shannon MI on measurements in RLE assuming uniform distribution for the sensor noise. Unfortunately, the resulting algorithm is not substantially better than the Approx-FSMI-RLE algorithm. However, we still describe the key ideas behind the algorithm for the purposes of completeness.

Let $\mathbb{1}(\cdot)$ be the indicator function. We also employ the notation defined in Theorem 4. The following theorem presents the main result.

Theorem 14. (Uniform FSMI-RLE). *If the sensor measurement noise follows uniform distribution, i.e.,*

$$P(Z|e_i) \sim U[l_i - Hw,\ l_{i+1} + Hw] \tag{47}$$

for $H \in \mathbb{Z}^+$ and the width of the virtual cell w, then the Shannon MI between Z and M' can be evaluated as

$$\begin{aligned} I(M'; Z) = \sum_{u=1}^{n_r}\sum_{v=1}^{n_r}\frac{P_E(u)o_u}{2H+1}((D_E(v) + f(\delta_{occ}, r_v)) \\ \sum_{j=0}^{L_u-1}\sum_{k=0}^{L_v-1} k \cdot \bar{o}_u^j 1(|j + t - k)| \leq H) \\ + f(\delta_{emp}, r_v))\sum_{j=0}^{L_u-1}\sum_{k=0}^{L_v-1}\bar{o}_u^j 1(|j + t - k| \leq H)) \end{aligned} \tag{48}$$

This motivates us to focus on the computation of the following two terms:

$$\begin{aligned} F[x, L_u, L_v, t] &= \sum_{j=0}^{L_u-1}\sum_{k=0}^{L_v-1} x^j 1(|j + t - k)| \leq H) \\ G[x, L_u, L_v, t] &= \sum_{j=0}^{L_u-1}\sum_{k=0}^{L_v-1} k \cdot x^j 1(|j + t - k)| \leq H) \end{aligned} \tag{49}$$

We apply the same tabulation techniques and algorithms presented in Section 4.1, except we tabulate F and G and we use Equation (48) to compute Shannon MI. We call the resulting algorithm Uniform-FSMI-RLE.

The following theorem states the time complexity of the Uniform-FSMI-RLE algorithm.

Theorem 15. (Time complexity of Uniform-FSMI-RLE). *If $F[x, L_u, L_v, t]$ and $G[x, L_u, L_v, t]$ can be evaluated in $O(1)$, the time complexity of the Uniform-FSMI-RLE algorithm is $O(n_r H)$.*

Note that the support of the uniform distribution itself only spans $2Hw$ in length; therefore, its effect on the Shannon MI computation is similar to that of the Gaussian truncation, and H is similar to Δ despite their definitions being different (H is the radius of the uniform distribution, whereas Δ is the truncation of the Gaussian distribution). Therefore, the proof of Theorem 15 is trivial given the time complexity of the Approx-FSMI-RLE algorithm, so we skip it in this article.

In practice, the magnitude of the value of H is also comparable with Δ; hence, the time complexity of the Uniform-FSMI-RLE algorithm, $O(n_r H)$ is not lower than the time complexity of the Approx-FSMI-RLE algorithm, which is $O(n_r \Delta)$.

5. Experimental results

This section is devoted to our experiments. In Section 5.1, we demonstrate the FSMI and Approx-FSMI algorithms in computational experiments with randomly generated occupancy values. Then, in Section 5.2, we demonstrate the effectiveness of these algorithms for 2D mapping in a synthetic environment. In Section 5.3, we demonstrate the same algorithms in an experiment involving a 1/10-scale car-like robot. In all cases, we compare the proposed algorithms with the existing algorithms for computing the Shannon MI metric and existing algorithms for computing the CSQMI metric.

Then, we turn our attention to the proposed algorithms for run-length encoded occupancy sequences. In Section 5.4, we demonstrate the Approx-FSMI-RLE algorithm in computational experiments and show the speedup compared with the FSMI algorithm, which does not support RLE. Following, in Section 5.5, we showcase the Approx-FSMI-RLE algorithm in a scenario involving a 1/10-scale car-like robot equipped with a Velodyne 16-channel laser range finder. All computational experiments use one core of an Intel Xeon E5-2695 CPU. All experiments involving the 1/10-scale car use a single core of the ARM Cortex-A57 CPU on the NVIDIA Tegra X2 platform.

5.1. Computational experiments for 2D MI algorithms

The first experiment studies the accuracy and the throughput of evaluating MI in computational experiments. We consider a scenario where the length of the beam is 10 m and the resolution of the occupancy grid is 0.1 m. The occupancy values are generated at random. We set $\delta_{occ} = 1/\delta_{emp} = 1.5$. We set the sensor's noise to be a normal distribution with a constant $\sigma = 0.05$ m regardless of the travel distance of the beam.

First, we compared the run time and accuracy of following three algorithms: (i) the existing Shannon MI computation algorithm by Julian et al. (2014) with integration resolution parameter set to $\lambda_z = 0.01$ m, which we call the SMI algorithm; (ii) the FSMI algorithm; and (iii) the Approx-FSMI algorithm with truncation parameter set to $\Delta = 3$. To measure the accuracy of the algorithms, we compute the ground truth using the algorithm by Julian et al. (2014) with integration resolution parameter set to $\lambda_z = 10$ μm. The results are summarized in Figures 8(a) and 8(b). We observe that the FSMI algorithm computes Shannon MI more accurately than the SMI algorithm (with integration parameter set to $\lambda_z = 0.01$ m) while running more than three orders of magnitude faster. This can be explained by the low numerical integration resolution when $\lambda_z = 0.01$ m. The run time of Approx-FSMI is an additional seven times faster than FSMI at the cost of a small reduction accuracy. Still, the Approx-FSMI algorithm is more accurate than the SMI algorithm with $\lambda_z = 0.01$ m.

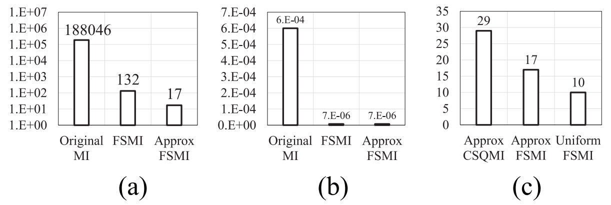

Fig. 8. Speed and relative error of different MI algorithms on a single beam: (a) mean time (μs); (b) mean relative error; (c) mean time (μs).

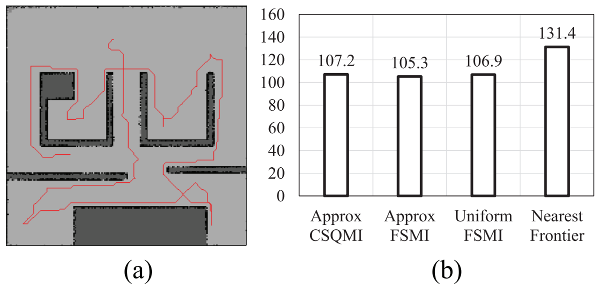

Fig. 9. Synthetic 2D environment exploration experiment results: (a) map and trajectory; (b) trajectory length.

Second, we compared the run time of the following three algorithms: (i) the Approx-CSQMI algorithm with truncation parameter set to $\Delta = 3$; (ii) the Approx-FSMI algorithm with truncation parameter set to $\Delta = 3$; (iii) the Uniform-FSMI algorithm. The results are shown in Figure 8(c). We find that Approx-FSMI is 1.7 times faster than Approx-CSQMI, and Uniform-FSMI is 3 times faster than Approx-CSQMI. The acceleration is not as large as predicted in Theorem 3 owing to compiler optimizations applied to Approx-CSQMI.

5.2. Simulated scenario for 2D mapping algorithms

In this section, we consider mapping in a synthetic environment, shown in Figure 9(a). The environment is $18 \times 18\text{m}^2$, and it is represented by an occupancy grid with resolution 0.1 m. The virtual robot that can measure range in all directions with 2° resolution, thus emulating a laser range finder with 180 beams. We compare the following four algorithms: (i) the Approx-FSMI algorithm with truncation parameter set to $\Delta = 3$; *(ii)* the Approx-CSQMI algorithm with truncation parameter set to $\Delta = 3$; (iii) the Uniform-FSMI Algorithm; and (iv) the nearest frontier exploration method. In the first three cases, the robot uses the Dijkstra's algorithm (Cormen et al., 2009) to compute paths to all free cells and chooses a path that maximizes the ratio between the MI gain if a robot scans at the last cell of the path and the travel distance to the last cell. In the fourth case, we use the nearest frontier algorithm discussed in Charrow et al. (2015b), where the robot travels to the closest cluster of frontier cells. The nearest frontier algorithm is a widely used baseline for information-theoretic mapping frameworks (Charrow et al., 2015b; Julian et al., 2014) and has been carefully compared against both the original MI algorithm and the CSQMI algorithm. Interested readers should refer to these prior works for more information about the comparison; we report results for this method mainly for completeness. In all cases, the exploration terminates when the reduction of the entropy of the map representation falls below a constant threshold. All algorithms are run three times and the results are averaged.

The results are shown in Figure 9. An example map generated by the Approx-FSMI algorithm as well as the path spanned by the robot are shown in Figure 9(a). The average path lengths spanned by the four exploration strategies are shown in Figure 9(b). We find that the first three exploration algorithms, all based on information-theoretic metrics, execute paths with approximately the same length. The frontier exploration method executes paths that are on average at 22% longer than the information-based methods.

Throughout the experiments, the average time spent computing MI was recorded. The Approx-CSQMI algorithm takes 12.4μs per beam, the Approx-FSMI algorithm takes 8.3μs per beam, and the Uniform-FSMI takes 4.9μs per beam, on average. The ranking of the algorithms is consistent with our results reported in Section 5.1. In these experiments, the average beam length is roughly half of what was used in Section 5.1, hence these evaluation times are roughly half as well.

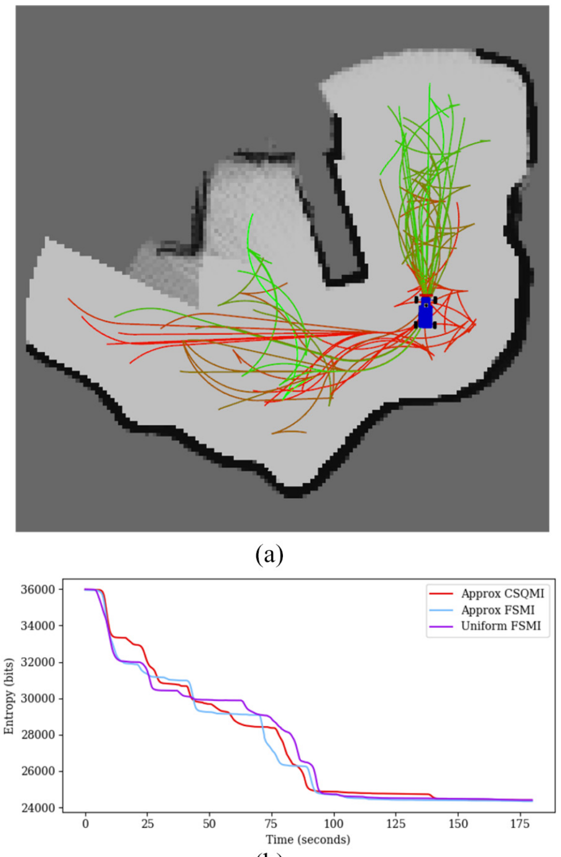

Figure 10. Real experiments with a car in a 2D environment. (a) Paths computed with the RRT* based planner are colored based on their potential information gain. Paths that obtain a the most MI per distance are colored green. (b) Representative samples of the map's entropy over the course of exploration.

5.3. Real-world scenario for 2D mapping algorithms

In this section, we consider a mapping scenario involving a 1/10th-scale car-like robot equipped with a Hokoyo UST-10LX LiDAR. In our experiments, we limited the field of view of the LiDAR to 230° and its range to 3 m. The sensor provides 920 range measurements within this field of view. The robot is placed in the 8×8 m^2 environment shown in Figure 11(h). The location of the robot is obtained in real time using an OptiTrack motion-capture system. Path planning is accomplished using the RRT* algorithm (Karaman and Frazzoli, 2011), with Reeds–Shepp curves (Reeds and Shepp, 1990) as the steering function. In our experiments, we evaluate MI along each path in the RRT* tree at 0.2 m intervals with 50 equally spaced beams. We find the path that maximizes the ratio between the MI along the path and the total path length. A visualization of these potential paths and their rankings is shown in Figure 10(a). Once the car reaches the end of the selected trajectory, it computes a new trajectory using the same algorithm. This procedure is repeated until the entropy of the map drops below a constant threshold. In all of these experiments, the resolution of the occupancy grid map is set to 0.05 m, and the sensor parameters are set to $\sigma = 0.05$ m and $\delta_{occ} = 1/\delta_{emp} = 1.5$.

In this setting, we compare the following three algorithms: (i) the Approx-CSQMI Algorithm with truncation parameter set to $\Delta = 3$; (ii) the Approx-FSMI Algorithm with truncation parameter set to $\Delta = 3$; and (iii) the Uniform-FSMI algorithm. We found that all three algorithms perform similarly in terms of how quickly they reduce the entropy of the map as shown in Figure 10(b). We measured Approx-CSQMI to take 422.7μs, Approx-FSMI to take 148.7μs, and Uniform-FSMI to take 111.4μs per beam, on average. The ranking of these timings is consistent with Sections 5.1 and 5.2. Differences in scaling may be due to the differences in compiler optimizations on the ARM Cortex-A57 CPU present in the NVIDIA Tegra X2 platform. These optimizations include memory caching optimizations, the usage of SIMD, and many others; most of them depend on the underlying hardware and associated compiler. We also provide a timelapse of the exploration with the Approx-FSMI algorithm in Figure 11.

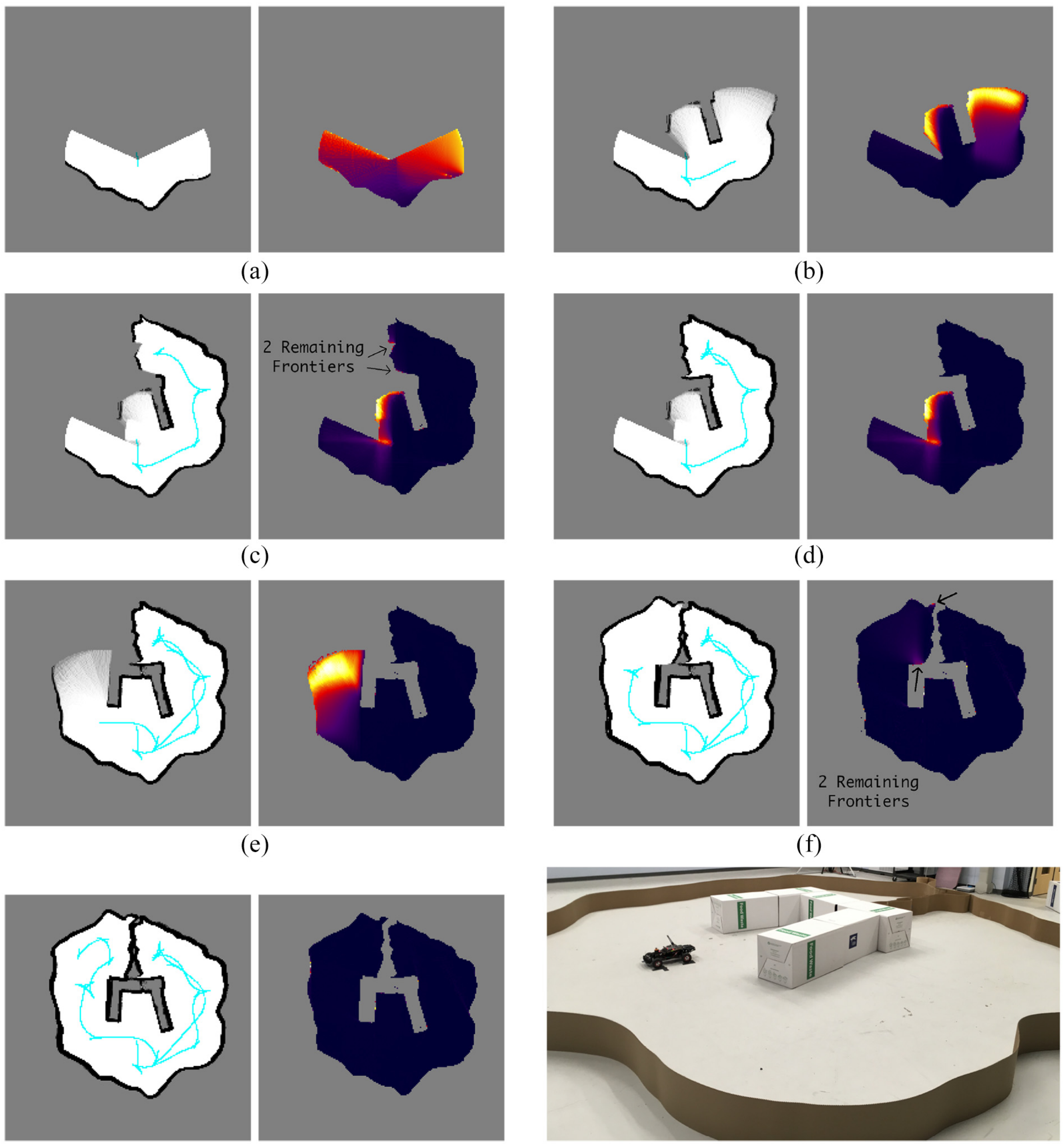

Fig. 11. A timelapse of a 2D exploration experiment. (a) The car begins exploration, revealing explorable areas on either side of the map. (b) The car moves towards the larger of the two frontiers. (c) The car reaches the end of this branch with two small frontiers occluded by its field of view. (d) The car turns around to see an occluded corner, completing the right side of the map. (e) The car turns around to reveal the small inlet and the other side of the map. (f) The car reaches the end of the branch, again revealing all but two frontiers. (g) The car maneuvers to explore the last two corners, competing the map. (h) The real world environment and the car in its starting position. On the left, the black and white figures are occupancy maps where the dark pixels are the most likely to be occupied. The blue line shows the path taken by the car. On the right the heat map figures are the corresponding MI surfaces with bright colors representing scanning locations with high MI. The accompanying video is available at: https://youtu.be/6Ia0conjKMQ

5.4. Computational experiments for the Approx-FSMI-RLE algorithm

In this section, we evaluate the Approx-FSMI-RLE algorithm, which calculates the Shannon MI of occupancy sequences represented using the RLE. We compare the following two algorithms in terms of run time: (i) the Approx-FSMI-RLE algorithm with truncation parameter set to $\Delta = 3$ executing on the compressed measurement using RLE; and *(ii)* the Approx-FSMI algorithm with truncation

Table 2. The run time of the Approx-FSMI-RLE algorithm versus that of the FSMI-RLE algorithm (μs). The baseline algorithm, the Approx-FSMI algorithm that operates on the uncompressed occupancy vector, takes 56.1 μs to complete.

	$L=1$	$L=2$	$L=4$	$L=8$	$L=16$	$L=32$	$L=64$	$L=128$
Run time of Approx-FSMI-RLE	240.9	79.4	31.5	12.3	7.6	4.9	3.4	2.3
Run time ratio of the two algorithms	0.2	0.7	1.8	4.6	7.4	11.2	16.5	24.4

parameter set to $\Delta = 3$ executing on the uncompressed measurement.

We consider a synthetic beam passing through $n = 256$ occupancy cells divided into groups of L cells. Within each group, the cells are given the same occupancy values. RLE compresses this sequence by a factor of L, so L can be interpreted as the compression ratio. We run each algorithm 1,000 times with randomly generated occupancy values.

The results are presented in Table 2. We find that the Approx-FSMI-RLE algorithm is slower than the Approx-FSMI algorithm when $L < 4$. However, it achieves an order of magnitude speedup for $L \geqslant 32$. To avoid the overhead of the Approx-FSMI-RLE algorithm when $L < 4$, we can adaptively switch between using the Approx-FSMI algorithm on the decompressed sequence and using Approx-FSMI-RLE algorithm on compressed sequence depending on whether $L < 4$ or not.

5.5. *Real-world scenario for the Approx-FSMI-RLE algorithm*

In this section, we evaluate the Approx-FSMI-RLE algorithm using the same 1/10-scale car-like robot described in Section 5.3. In these experiments, the car is now equipped with a Velodyne Puck LiDAR with 16 channels with a vertical field of view of $\pm 15°$. The horizontal field of view is limited to 270° owing to occlusions on the frame of the car. Although the robot travels on the 2D plane, this range sensor allows it to measure the environment in three dimensions.

We use the Approx-FSMI-RLE algorithm to evaluate MI in a 3D mapping task. We use the OctoMap software package (Hornung et al., 2013) to represent the map, which naturally returns run-length encoded occupancy sequences. Once again we obtain location information from the OptiTrack motion capture system. We utilize the RRT* algorithm as described in Section 5.3 to generate a set of possible trajectories and then choose to travel along the one that maximizes the ratio between the MI along the path and the length of the path. We evaluate the MI along each path at 0.2 m intervals using the Approx-FSMI-RLE algorithm. For this purpose, we randomly sample 100 beams within the LiDAR's 270° horizontal field of view and $\pm 15°$ vertical field of view.

An example experiment including the resulting map, the MI surface, and path are shown in Figure 12 at various intervals. As compared with Section 5.3, where the MI gain tended to be highest near boundaries, we observe that MI in these 3D experiments is higher in the center of unknown regions. We conclude that owing to the car's limited vertical vision this phenomenon exists to enforce the car to "take a step back" in order to get a better view.

For comparison, we repeated the same 3D experiments using the Approx-FSMI algorithm. We observed that, on average, the RLE compresses occupancy sequences by roughly 18 times. We found that this compression enables the Approx-FSMI-RLE algorithm to be 8 times faster than the Approx-FSMI in terms of run time, even when we exclude the time to decompress the RLE representation coming from the OctoMap to the uncompressed representation used by the Approx-FSMI algorithm.

6. Conclusion

In this article, we have introduced FSMI, an algorithm for computing the Shannon MI between potential future measurements and an occupancy grid.

For 2D information-theoretic mapping scenarios, we have introduced three algorithms: FSMI, Approx-FSMI which approximates FSMI with arbitrary precision, and Uniform-FSMI which computes exact Shannon MI under the assumption that the measurement noise is uniformly distributed.

We have also extended the algorithms to 3D mapping tasks when an OctoMap data structure is used to represent the map. We have introduced the FSMI-RLE algorithm that accelerates the FSMI algorithm with a RLE compression technique. To address numerical issues inherent to the FSMI-RLE algorithm, we have proposed the Approx-FSMI-RLE algorithm, which utilizes Gaussian truncation. We have also discussed the Uniform-FSMI-RLE algorithm, which parallels the Uniform-FSMI algorithm.

We have rigorously proved guarantees on the correctness and the computational complexity of the proposed algorithms.

In our computational experiments, we have shown that the FSMI algorithm achieved more than three orders of magnitude computational savings when compared with the original Shannon MI computation algorithm described in (Julian et al., 2014), while maintaining higher accuracy. We have shown that the Approx-FSMI runs 7 times faster than FSMI with negligible precision loss. We have then shown that the Approx-FSMI algorithm has the same asymptotic computational complexity as the Approx-CSQMI algorithm. We have also shown that the Approx-FSMI algorithm has a smaller constant factor than the Approx-CSQMI algorithm, which is measured by the number of

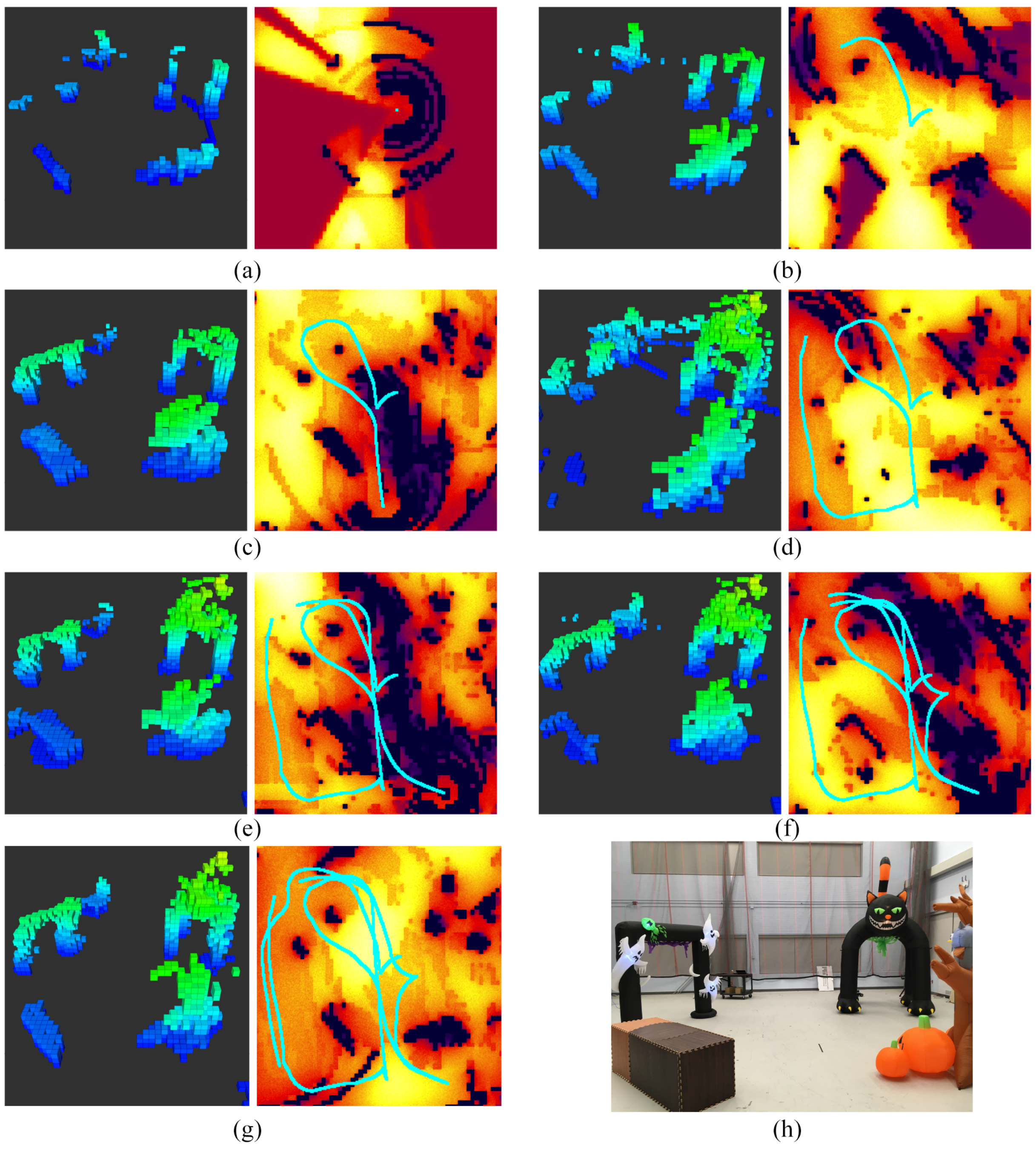

Fig. 12. A timelapse of a 3D exploration experiment in the environment pictured in (h). (a) The car begins exploration, revealing areas with high MI in the top and bottom of the map. (b) The car moves to the top left revealing a large amount of free space (not visible in the 3D map) as well as the top of the tree in the bottom right. The regions in the lower half of the map now have the highest MI. (c) The car moves to the bottom of the map revealing the top of the arch in the top left and the backs of the box in the bottom left and tree in the bottom right. (d) The car moves to a point in the far top left with, perhaps because it will reveal the top of the cat in the top right. In doing so, the arch occludes it from the motion capture cameras, distorting the map. The break in the trajectory is visible in the next frame. (e) The car moves to the bottom right and recovers from most of the distortion. (f) The car moves to the top left and recovers some information about the back of the arch and cat (not visible in these figures). (g) The car moves to the bottom left, improving the top of the cat as well as the box. (h) The environment featuring an arch, a giant cat, a box, and a tree shown from the same perspective. The tallest obstacle, the cat, is 4 m tall, 20 $\times$ taller than the car. On the left of each figure is a 3D OctoMap where color indicates the height of a voxel. On the right, is a 2D projection of the corresponding MI surface. For each point in the image, we compute the MI from a scan consisting of beams are randomly sampled from the LiDAR's 360° horizontal field of view and $\pm 15°$ vertical field of view. Localization is done using motion capture cameras as in the 2D experiments; however, this is occasionally lost in the case where the 3D obstacles occlude the camera view.

multiplication operations, In our computational experiments, we have observed that indeed the FSMI algorithm runs twice as fast as the Approx-CSQMI algorithm. Moreover, we have shown that if the measurement noise follows a uniform distribution, the Shannon MI can be computed *exactly* in linear time with respect to occupancy resolution by the Uniform-FSMI algorithm.

We have also conducted synthetic and real-world experiments to demonstrate the performance of the Approx-FSMI-RLE algorithm over the Approx-FSMI. Our experiments show 8 times of acceleration in practice.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was partially funded by the AFOSR YIP FA9550-16-1-0228, by the NSF CAREER 1350685 and NSF CPS 1837212.

Note

1. Let us note that, in many practical implementations of mapping algorithms, the occupancy values o_u is quantized into a finite set of values from $[0, 1]$ for computational efficiency (see, e.g., Hess et al., 2016).

ORCID iD

Zhengdong Zhang https://orcid.org/0000-0001-9014-4646

References

Bourgault F, Makarenko AA, Williams SB, Grocholsky B and Durrant-Whyte HF (2002) Information based adaptive robotic exploration. In: *2002 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vol. 1. IEEE, pp. 540–545.

Bresenham J (1977) A linear algorithm for incremental digital display of circular arcs. *Communications of the ACM* 20(2): 100–106.

Burgard W, Moors M, Stachniss C and Schneider FE (2005) Coordinated multi-robot exploration. *IEEE Transactions on robotics* 21(3): 376–386.

Burri M, Oleynikova H, Achtelik MW and Siegwart R (2015) Real-time visual–inertial mapping, re-localization and planning onboard MAVs in unknown environments. In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 1872–1878.

Cassandra AR, Kaelbling LP and Kurien JA (1996) Acting under uncertainty: Discrete Bayesian models for mobile-robot navigation. In: *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'96)*, Vol. 2. IEEE, pp. 963–972.

Charrow B, Kahn G, Patil S, et al. (2015a) Information-theoretic planning with trajectory optimization for dense 3D mapping. In: *Robotics: Science and Systems*, Vol. 6.

Charrow B, Kumar V and Michael N (2014) Approximate representations for multi-robot control policies that maximize mutual information. *Autonomous Robots* 37(4): 383–400.

Charrow B, Liu S, Kumar V and Michael N (2015b) Information-theoretic mapping using Cauchy–Schwarz quadratic mutual information. In: *Proceedings IEEE International Conference on Robotics and Automation*.

Cormen TH, Leiserson CE, Rivest RL and Stein C (2009) *Introduction to Algorithms*. Cambridge, MA: MIT Press.

Elfes A (1996) Robot navigation: Integrating perception, environmental constraints and task execution within a probabilistic framework. In: *Reasoning with Uncertainty in Robotics*. New York: Springer, pp. 91–130.

Endres F, Hess J, Sturm J, Cremers D and Burgard W (2014) 3-D mapping with an RGB-d camera. *IEEE Transactions on Robotics* 30(1): 177–187.

González-Banos HH and Latombe JC (2002) Navigation strategies for exploring indoor environments. *The International Journal of Robotics Research* 21(10–11): 829–848.

Hennessy JL and Patterson DA (2011) *Computer Architecture: A Quantitative Approach*. Amsterdam: Elsevier.

Hess W, Kohler D, Rapp H and Andor D (2016) Real-time loop closure in 2D LiDAR SLAM. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1271–1278.

Holz D, Basilico N, Amigoni F, et al. (2011) A comparative evaluation of exploration strategies and heuristics to improve them. In: *Proceedings of ECMR*, pp. 25–30.

Hornung A, Wurm KM, Bennewitz M, Stachniss C and Burgard W (2013) OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*. DOI: 10.1007/s10514-012-9321-0.

IEEE (2019) IEEE Standard for Floating-Point Arithmetic. *IEEE Standard 754-2019 (Revision of IEEE 754-2008)*.

Julian BJ, Karaman S and Rus D (2014) On mutual information-based control of range sensing robots for mapping applications. *The International Journal of Robotics Research* 33(10): 1375–1392.

Karaman S and Frazzoli E (2011) Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research* 30(7): 846–894.

Kollar T and Roy N (2008) Efficient optimization of information-theoretic exploration in SLAM. In: *Proceedings of AAAI*, Vol. 8, pp. 1369–1375.

Li P, Zhang Z, Karaman S and Sze V (2019) High-throughput computation of Shannon mutual information on chip. In: *Robotics: Science and Systems (RSS)*.

Marchant R and Ramos F (2014) Bayesian optimisation for informative continuous path planning. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 6136–6143.

Moorehead SJ, Simmons R and Whittaker WL (2001) Autonomous exploration using multiple sources of information. In: *Proceedings 2001 IEEE International Conference on Robotics and Automation (ICRA 2001)*, Vol. 3. IEEE, pp. 3098–3103.

Moravec H (1996) Robot spatial perception by stereoscopic vision and 3D evidence grids. *Techical Report CMU-RI-TR-96-34*, Robotics Institute, Carnegie Mellon University.

Nelson E and Michael N (2015) Information-theoretic occupancy grid compression for high-speed information-based exploration. In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 4976–4982.

Principe JC (2010) *Information Theoretic Learning: Renyi's Entropy and Kernel Perspectives*. Springer Science & Business Media.

Rabaey JM, Chandrakasan AP and Nikolic B (2002) *Digital Integrated Circuits*, Volume 2. Englewood Cliffs, NJ: Prentice Hall.

Reeds J and Shepp L (1990) Optimal paths for a car that goes both forwards and backwards. *Pacific Journal of Mathematics* 145(2): 367–393.

Robinson A and Cherry C (1967) Results of a prototype television bandwidth compression scheme. *Proceedings of the IEEE* 55(3): 356–364.

Roth-Tabak Y and Jain R (1989) Building an environment model using depth information. *Computer* 22(6): 85–90.

Shen S, Michael N and Kumar V (2012) Stochastic differential equation-based exploration algorithm for autonomous indoor 3D exploration with a micro-aerial vehicle. *The International Journal of Robotics Research* 31(12): 1431–1444.

Tabib W, Corah M, Michael N and Whittaker R (2016) Computationally efficient information-theoretic exploration of pits and caves. In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE.

Thrun S, Burgard W and Fox D (2005) *Probabilistic Robotics*. Cambridge, MA: MIT Press.

Visser A and Slamet BA (2008) Balancing the information gain against the movement cost for multi-robot frontier exploration. In: *European Robotics Symposium 2008*. Berlin: Springer, pp. 43–52.

Wang P and Gupta K (2002) Computing C-space entropy for view planning based on beam sensor model. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3. IEEE, pp. 2389–2394.

Wang P and Gupta K (2003) Computing C-space entropy for view planning with a generic range sensor model. In: *2003 IEEE International Conference on Robotics and Automation*, Vol. 2. IEEE, pp. 2406–2411.

Wang P and Gupta K (2006) A configuration space view of view planning. In: *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 1291–1297.

Whelan T, Kaess M, Johannsson H, Fallon M, Leonard JJ and McDonald J (2015) Real-time large-scale dense RGB-d slam with volumetric fusion. *The International Journal of Robotics Research* 34(4–5): 598–626.

Yamauchi B (1997) A frontier-based approach for autonomous exploration. In: *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA'97)*. IEEE, pp. 146–151.

Yu Y and Gupta K (2004) C-space entropy: A measure for view planning and exploration for general robot–sensor systems in unknown environments. *The International Journal of Robotics Research* 23(12): 1197–1223.

Appendix. Analytical solution to the summation

This section derives the closed-form solution to the following two terms

$$\begin{aligned} A[x, L_u, L_v, t] &= \sum_{j=0}^{L_u-1} \sum_{k=0}^{L_v-1} x^j \exp\left(-\frac{(j-k+t)^2}{2\sigma^2}\right) \\ B[x, L_u, L_v, t] &= \sum_{j=0}^{L_u-1} \sum_{k=0}^{L_v-1} k \cdot x^j \exp\left(-\frac{-(j-k+t)^2}{2\sigma^2}\right) \end{aligned} \tag{50}$$

As stated previously, we have not found an analytic way to evaluate $A[x, L_u, L_v, t]$ and $B[x, L_u, L_v, t]$. Fortunately, if we allow for slight approximation, the closed-form solutions to both terms show up.

Specifically, we approximate the summation $A[x, L_u, L_v, t] = \sum_{j=0}^{L_u-1} \sum_{k=0}^{L_v-1} x^j \exp\left(-\frac{(j-k+t)^2}{2\sigma^2}\right)$ by integration:

$$\int_{-\frac{1}{2}}^{L_u-\frac{1}{2}} \int_{-\frac{1}{2}}^{L_v-\frac{1}{2}} x^j \exp\left(-\frac{(j-k+t)^2}{2\sigma^2}\right) dj\,dk \tag{51}$$

To derive the closed-form solution to this integration, we need the following lemma.

Lemma 6. *Let* erfc$(\cdot)$ *be the complementary error function. Let* $0 < x < 1$. *We have*

$$\begin{aligned} \theta_{\sigma,x}(a_1, a_2, t) &= \int_{a_1}^{\infty} \int_{a_2}^{\infty} x^j \exp\left(-\frac{(j-k+t)^2}{2\sigma^2}\right) dj\,dk \\ &= \frac{1}{\log x}\left(\sqrt{\frac{\pi}{2}} x^{-t} \sigma' \left(x^{a_1+t}\left(-2 + \operatorname{erfc}\left(\frac{a_1 - a_2 + t}{\sqrt{2}\sigma}\right)\right)\right.\right. \\ &\quad \left.\left.\left. - e^{\frac{1}{2}\sigma^2 \cdot \log^2 x} x^{a_2} \operatorname{erfc}\left(\frac{a_1 - a_2 + t - \sigma^2 \log^2 \bar{o}_u}{\sqrt{2}\sigma}\right)\right)\right)\right) \end{aligned} \tag{52}$$

In addition, $\theta_{\sigma,x}(a_1, a_2, t)$ *can be evaluated in* $O(1)$.

Based on the lemma, the analytic solution is as follows.

Theorem 16. (Closed-form solution to the approximation). *Equation* (51) *can be evaluated in* $O(1)$ *as follows*:

$$\begin{aligned} &\int_{-\frac{1}{2}}^{L_u-\frac{1}{2}} \int_{-\frac{1}{2}}^{L_v-\frac{1}{2}} x^j \exp\left(-\frac{(j+t-k)^2}{2\sigma^2}\right) dj\,dk \\ &= \theta_{\sigma,x}\left(-\frac{1}{2}, -\frac{1}{2}, t\right) - \theta_{\sigma,x}\left(L_u - \frac{1}{2}, -\frac{1}{2}, s_u - s_v\right) \\ &\quad - \theta_{\sigma,x}\left(-\frac{1}{2}, L_v - \frac{1}{2}, t\right) + \theta_{\sigma,x}\left(L_u - \frac{1}{2}, L_v - \frac{1}{2}, t\right) \end{aligned} \tag{53}$$

Although this solution is closed form and has $O(1)$ time complexity, evaluating it on the fly requires eight evaluation of the complementary error function and tens of expensive operations including exponential and square root.

Similarly, we approximate $B[x, L_u, L_v, t] = \sum_{j=0}^{L_u-1} \sum_{k=0}^{L_v-1} k \cdot x^j \exp\left(-\frac{(j-k+t)^2}{2\sigma^2}\right)$ by

$$\int_{-\frac{1}{2}}^{L_u-\frac{1}{2}} \int_{-\frac{1}{2}}^{L_v-\frac{1}{2}} k \cdot x^j \exp\left(-\frac{(j-k+t)^2}{2\sigma^2}\right) dj\,dk. \tag{54}$$

We can also show that Equation (54) has a closed-form solution that can be evaluated in $O(1)$. However, the solution is much longer than the solution for Equation (51) requiring far more FLOPS. In addition, these solutions are not used in the FSMI-RLE algorithm. Therefore, we do not include it in this article.