sensors

Letter

Computationally Efficient Cooperative Dynamic Range-Only SLAM Based on Sum of Gaussian Filter [†]

Jung-Hee Kim [1] and Doik Kim [2,*]

[1] Department of Electronic Engineering, Hanyang University, Seoul 04763, Korea; jhkim.sp.0901@gmail.com
[2] Center for Intelligent and Interactive Robotics, Korea Institute of Science and Technology, Seoul 02792, Korea
[*] Correspondence: doikkim@kist.re.kr

[†] This paper is an extended version of our paper published in Kim, J.-H.; Kim, D. Cooperative Range-only SLAM based on Sum of Gaussian Filter in Dynamic Environments. In Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Macau, China, 3–8 November 2019.

Received: 14 May 2020; Accepted: 8 June 2020; Published: 10 June 2020

Abstract: A cooperative dynamic range-only simultaneous localization and mapping (CDRO-SLAM) algorithm based on the sum of Gaussian (SoG) filter was recently introduced. The main characteristics of the CDRO-SLAM are (i) the integration of inter-node ranges as well as usual direct robot-node ranges to improve the convergence rate and localization accuracy and (ii) the tracking of any moving nodes under dynamic environments by resetting and updating the SoG variables. In this paper, an efficient implementation of the CDRO-SLAM (eCDRO-SLAM) is proposed to mitigate the high computational burden of the CDRO-SLAM due to the inter-node measurements. Furthermore, a thorough computational analysis is presented, which reveals that the computational efficiency of the eCDRO-SLAM is significantly improved over the CDRO-SLAM. The performance of the proposed eCDRO-SLAM is compared with those of several conventional RO-SLAM algorithms and the results show that the proposed efficient algorithm has a faster convergence rate and a similar map estimation error regardless of the map size. Accordingly, the proposed eCDRO-SLAM can be utilized in various RO-SLAM applications.

Keywords: simultaneous localization and mapping (SLAM); range-only SLAM; sum of Gaussian (SoG) filter; cooperative approach

1. Introduction

Simultaneous localization and mapping (SLAM), that is, localizing a robot while building a map of unknown environments at the same time, has been a popular topic. It has been developed in several forms, e.g., visual SLAM [1,2], bearing-only SLAM [3], range-only SLAM (RO-SLAM) [4–9], and their combinations [10,11], etc.

Compared with other types of SLAMs, the RO-SLAM, which uses range sensors only, has some distinguishing features. First, the range measurements are highly ambiguous. The potential location of a node can be anywhere on a ring shape, as shown in Figure 1a. Such ambiguity progressively disappears with the additional range measurements at different locations of a robot, as illustrated in Figure 1b,c. Second, the map itself simply shows the locations of nodes. Therefore, the data association issue, which is one of the main difficulties in other types of SLAMs, does not need to be considered. Due to this simplicity, the RO-SLAM has been applied in a wide range of applications such as submarine autonomous vehicles, search and rescue, etc., [6,12].

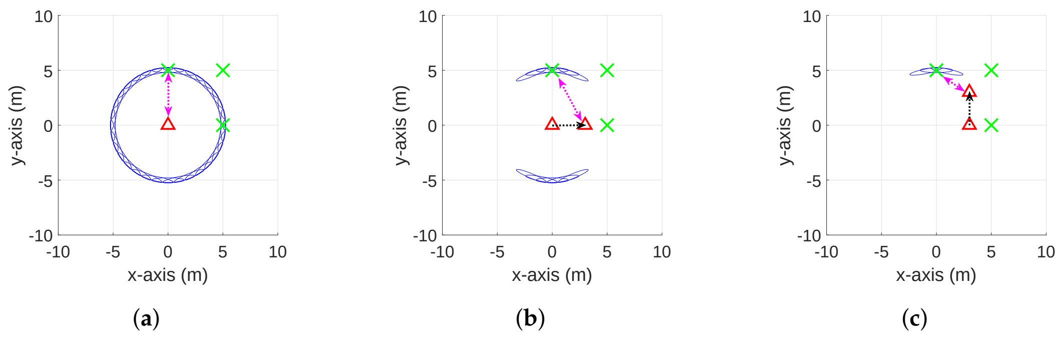


Figure 1. Explanation of the conventional RO-SLAM algorithms. Red triangles and green crosses indicate robots and neighbor nodes, respectively. Black and magenta dotted lines are the movement of the robot and range measurements, respectively. Blue ellipse is a possible location of nodes. The conventional RO-SLAM algorithms involve (**a**) the initial localization when receiving the first measurement, (**b**) movement of a robot to disambiguate the x-axis, and (**c**) movement to disambiguate the y-axis.

The Extended Kalman filter (EKF) has been widely employed as a standard solution to the RO-SLAM problem, along with some beacon initialization methods, e.g., trilateration [4], the probability grid [5], the particle filter [7], and the sum of Gaussian (SoG) filter [5,6]. The trilateration method (or multilateration) is simple, but its performance can be seriously damaged by measurement noise. The probability grid can perform better than the trilateration, but its performance depends on the size and resolution of the grid. The particle filter can represent an arbitrary probability density function (pdf) with a number of particles, such as a non-Gaussian ring shape for the beacon location in the RO-SLAM; however, it has a high computational burden due to the large number of particles. Compared with the particle filter approach, the SoG filter [5,6] can give a similar estimation accuracy in a more computationally efficient way, because it can efficiently cover almost the same area of beacon locations with a small number of Gaussian distributions. In this paper, the multilateration and SoG approaches are utilized to determine the initial position to give a fast and accurate localization result.

To improve the estimation accuracy, several RO-SLAM algorithms employ a cooperative approach, in which there are not only direct measurements between a robot and its neighboring nodes, but also, other inter-node measurements are integrated for localization [8,9]. In Ref. [8], the inter-node measurements were simply incorporated into the standard EKF based framework, and the efficiency and scalability of the large scale sensor network were further improved by using the sparse extended information filter (SEIF) based cooperative algorithm [9,13]. Recently, the SoG filter based cooperative approach, called the cooperative dynamic RO-SLAM (CDRO-SLAM) [14], was introduced by using the advantages of the SoG filter, such as the high estimation accuracy, as mentioned above. In addition, the CDRO-SLAM can cope with moving nodes by appropriately manipulating the SoG filter, unlike other RO-SLAM algorithms which assume that nodes are fixed.

In this paper, the efficient implementation of the CDRO-SLAM (eCDRO-SLAM) is proposed by introducing several efficient implementation techniques to mitigate the high computational burden of the CDRO-SLAM due to the inter-node measurements. Furthermore, its detailed computational complexity analysis is also given, and the results show that the computational efficiency of the eCDRO-SLAM is significantly improved over the CDRO-SLAM, which is shown in the experiment results.

The main properties of the proposed eCDRO-SLAM can be summarized, where the first two properties are obtained by inheriting from the advantages of the CDRO-SLAM:

- Accurate and fast localization results are achieved by incorporating the cooperative approach and the SoG filter.
- Sensor nodes can be applied to dynamic environments where installed nodes are not fixed or moving such as unstable and unstructured disaster sites, moving landmarks, interactive information with human and non-fixed objects to analyze human activity, etc.

- To mitigate the high computational cost induced by the inter-node measurements, several efficient techniques are introduced, which greatly reduces the required computational complexity of the proposed eCDRO-SLAM.

This paper is organized as follows: Section 2 derives the CDRO-SLAM, including a cooperative scheme using inter-node measurements and a tracking scheme of non-fixed nodes. In Section 3, an efficient implementation of the CDRO-SLAM (eCDRO-SLAM) is presented to mitigate the computational burden due to the cooperative scheme. Section 4 verifies the performance of the eCDRO-SLAM with several experimental results, and Section 5 gives the concluding remarks.

2. Cooperative Dynamic RO-SLAM

2.1. Overview

This section summarizes our previous work on the CDRO-SLAM [14], aimed at developing a cooperative RO-SLAM under dynamic environments. The CDRO-SLAM algorithm is composed of two stages: (i) the initialization stage estimates the initial locations of a robot and map by using inter-node measurements, which improves and accelerates the map estimation and (ii) the movement stage further refines the map generated in the initialization stage or tracks nodes in motion.

In more detail, the initialization stage is described in Algorithm 1. (i) Once the initial positions of nodes are guessed with the help of iterative multilateration, it is refined by employing the sum of Gaussian (SoG) filter approach. (ii) The SoG variables are first generated to find the candidate locations of neighbors, and (iii) the SoG variables are updated by exploiting the inter-node measurements to quickly reduce the number of candidates. (iv) The weighted SoG variables are merged into a single Gaussian distribution to estimate the new positions of the nodes. (v) Furthermore, a method to transfer the weight of the SoG variables is applied to resolve the inherent non-convexity in the anchor-free localization. (vi) The above procedures are repeated until the positions of all nodes converge.

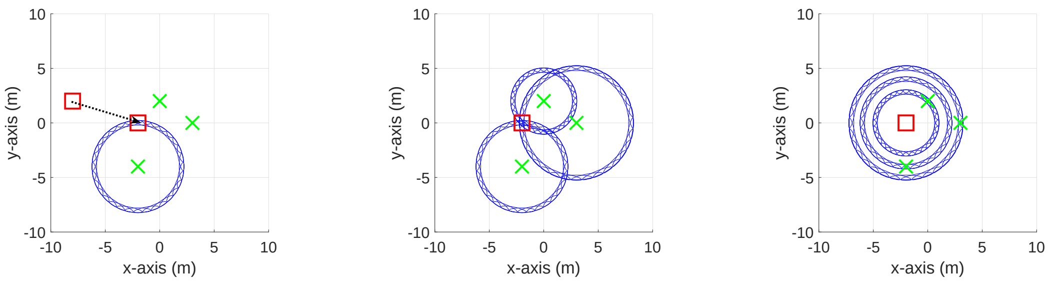
In the movement stage, a robot and/or some nodes can be moved. Firstly, if a robot moves, its motion is updated using raw odometry data. Then, after receiving a new distance from the neighbor node, the locations of the corresponding neighbor node and the robot are further refined by using the EKF procedure, like in the usual RO-SLAM algorithms (see [15] for details). Secondly, for the case where some nodes move, their movement can be tracked by properly manipulating the SoG filter, which is illustrated in Algorithm 2 and Figure 2. (i) If j denotes an index of a moving node, the movement of the jth node is first detected by one of motion detection techniques (e.g., the norm of the accelerometer). (ii) When getting a newly modified distance due to the motion of the jth node, the corresponding SoG variables are reset (see Figure 2a). After that, the new position of the jth node is estimated by (iii) updating and (iv) merging the SoG variables (see Figure 2b). Finally, the SoG variables from the jth node to its neighbor nodes are also recalculated by (v) resetting and (vi) updating them to reflect the newly estimated position of the jth node (see Figure 2c).

Differences between the conventional SoG based RO-SLAM algorithms and the proposed CDRO-SLAM are discussed in terms of the weight update scheme and computational complexity. Figures 1 and 3 show the weight update processes of both approaches, respectively. For both algorithms, the weight update starts when the first measurement between a robot and one of its neighbor nodes is received, and then the potential location is assumed to be anywhere within a ring shape whose radius is equal to the measurement (see Figures 1a and 3a). For conventional algorithms, the ambiguity progressively disappears by moving a robot along with x and y axes, i.e., a non-collinear motion (see Figure 1b,c). On the other hand, the proposed algorithm updates the weight by using inter-node measurements without moving a robot. The ambiguity is gradually removed as more measurements from the non-collinear adjacent nodes arrive (see Figure 3b,c).

The computational cost of the CDRO-SLAM is increased due to the inclusion of the SoG variables for inter-node measurements. To reduce this computational burden, several efficient implementation techniques such as the weight symmetry of the SoG distributions are introduced later in Section 3.

Algorithm 1: Initialization stage of the CDRO-SLAM.
(M for the number of nodes, and i, j, m for the indices of nodes)

```
(i) Iterative multilateration
for $i \leftarrow 0$ to $M-1$ do
    Initialize the location of $i$th node by using (1)–(4).
end
(ii) Generating SoG
for $i, j \leftarrow 0$ to $M-1$, $(j \neq i)$ do
    Generate the mean $\mathbf{m}_{ij}^k$, covariance $\mathbf{C}_{ij}^k$, and weight $w_{ij}^k$ by using (5)–(7).
end
(iii) Updating SoG
Update the mean $\mathbf{m}_{ij}^k$ and covariance $\mathbf{C}_{ij}^k$ using EKF.
/* Algorithms 3 and 4 */
for $i, j, m \leftarrow 0$ to $M-1$, $(j \neq i, m \neq i, m \neq j)$ do
    Compute and normalize the likelihood $l_{ij,m}^k$.
    Update and normalize the weight $w_{ij}^k$.
end
(iv) Merging SoG
for $i \leftarrow 0$ to $M-1$ do
    Compute the new location of the $i$th node with (8)–(9).
end
(v) Transferring weight
for $i, j \leftarrow 0$ to $M-1$, $(j \neq i)$ do
    Transfer the weight $w_{ij}^k$ by using (11).
end
(vi) Determining convergence
for $i \leftarrow 0$ to $M-1$ do
    Determine the convergence of the $i$th node.
end
```

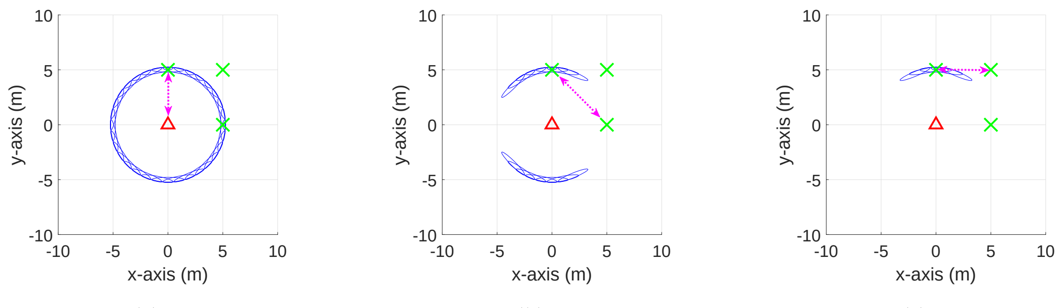


(**a**) (**b**) (**c**)

Figure 2. Description of the updated procedures for tracking moving nodes. (**a**) Reset the SoG variables from the neighbor node to the moving node, (**b**) update and merge the SoG variables, and (**c**) reset the SoG variables from the moving node to its neighbor nodes and update the corresponding weights. The red squares and green crosses represent the moving nodes and neighbor nodes, respectively. The black dotted line represents the movement of a node.

Algorithm 2: Moving stage of the cooperative dynamic range-only simultaneous localization and mapping (CDRO-SLAM).
(M represents the number of nodes, and i, j, m represent the indices of nodes)

```
(i) Detecting movement
Detect whether the $j$th node is moving or not.

(ii) Resetting SoG (Figure 2a)
Reset the mean $\mathbf{m}_{ij}^k$, covariance $\mathbf{C}_{ij}^k$, and weight $w_{ij}^k$ using Equations (5)–(7).

(iii) Updating the SoG (Figure 2b)
Update the mean $\mathbf{m}_{ij}^k$ and covariance $\mathbf{C}_{ij}^k$ using EKF.
/* Algorithms 3 and 4 */
for $i, m \leftarrow 0$ to $M-1$, $(i \neq j, m \neq i, m \neq j)$ do
    Compute and normalize the likelihood $l_{ij,m}^k$.
    Update and normalize the weight $w_{ij}^k$.
end

(iv) Merging the SoG (Figure 2b)
Estimate the new location of the $j$th node.

(v) Resetting the SoG (Figure 2c)
for $i \leftarrow 0$ to $M-1$, $(i \neq j)$ do
    Reset the mean $\mathbf{m}_{ji}^k$, covariance $\mathbf{C}_{ji}^k$, and weight $w_{ji}^k$ using Equations (5)–(7).
end

(vi) Updating the weight (Figure 2c)
for $i, m \leftarrow 0$ to $M-1$, $(i \neq j, m \neq i, m \neq j)$ do
    Compute and normalize the likelihood $l_{ji,m}^k$.
    Update and normalize the weight $w_{ji}^k$.
end
```



(**a**) (**b**) (**c**)

Figure 3. Explanation of the weight update scheme in the CDRO-SLAM. Red triangles and green crosses indicate robots and neighbor nodes, respectively. Black and magenta dotted lines are the movement of the robot and range measurements, respectively. Blue ellipse is a possible location of nodes. The CDRO-SLAM involves (**a**) the initialization of the SoG variables when receiving the first measurement, (**b**) determining the SoG distribution after updating the weight with the first inter-node measurement, and (**c**) the second inter-node measurement.

2.2. Initialization Stage

2.2.1. Iterative Multilateration

The initial estimation is done by employing the modified iterative multilateration method so that errors are not propagated in each iteration. As stated in [16], the first three nodes are supposed to be placed at the origin, on the positive x-axis, and in the upper half plane, respectively, as follows:

$$\mathbf{p}_0(n) = (x_0(n), y_0(n)) = (0, 0) \tag{1}$$

$$\mathbf{p}_1(n) = (x_1(n), y_1(n)) = (d_{01}, 0) \tag{2}$$

$$\begin{aligned}\mathbf{p}_2(n) &= (x_2(n), y_2(n)) \\ &= \left(\frac{d_{01}^2 + d_{02}^2 - d_{12}^2}{2x_1(n)}, \sqrt{d_{02}^2 - x_2^2(n)} \right).\end{aligned} \tag{3}$$

In Equations (1)–(3), $\mathbf{p}_i(n)$ is defined as the position of the ith node at iteration n, and d_{ij} is the distance measurement between the ith and jth nodes. The next node is localized by exploiting the positions of the first two nodes as in $\mathbf{p}_2(n)$, resulting in the following two candidates due to its ambiguity:

$$\begin{aligned}\mathbf{p}_m(n) &= (x_m(n), y_m(n)) \\ &= \left(\frac{d_{01}^2 + d_{0m}^2 - d_{1m}^2}{2x_1(n)}, \pm\sqrt{d_{0m}^2 - x_m^2(n)} \right).\end{aligned} \tag{4}$$

The sign of $y_m(n)$ can be chosen as the one whose estimated distance from $\mathbf{p}_m(n)$ to $\mathbf{p}_2(n)$ is closer to the measurement d_{2m}. Note that $\mathbf{p}_m(n)$ is affected by the position error of $\mathbf{p}_1(n)$, as shown in Equation (4). The above procedures (i.e., Equations (1)–(4)) are used in Algorithm 1.

2.2.2. Generating SoG

After guessing the initial positions of a robot and map, SoG variables are generated as shown in Figure 4. Note that in the conventional SoG based RO-SLAM methods, only the SoG variables between robot and its neighbor nodes are generated, while in the proposed CDRO-SLAM, those between any pair of two nodes are generated. For two different nodes (let us say the ith and jth nodes, $i \neq j$), the mean $\mathbf{m}_{ij}^k$, covariance $\mathbf{C}_{ij}^k$, and weight w_{ij}^k are generated as described in [5,6]:

$$\mathbf{m}_{ij}^k = \begin{bmatrix} x_i(n) + d_{ij}\cos\left(\frac{2\pi k}{N}\right) \\ y_i(n) + d_{ij}\sin\left(\frac{2\pi k}{N}\right) \end{bmatrix}, \tag{5}$$

$$\mathbf{C}_{ij}^k = \begin{bmatrix} \mathbf{v}_r & \mathbf{v}_t \end{bmatrix} \begin{bmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_t^2 \end{bmatrix} \begin{bmatrix} \mathbf{v}_r^T \\ \mathbf{v}_t^T \end{bmatrix}, \tag{6}$$

$$w_{ij}^k = \frac{1}{N}, \tag{7}$$

where $k = 0, 1, \ldots, N-1$, is an index of SoG, and N is number of SoG, respectively. Note that the procedures of generating the SoG variables (i.e., Equations (5)–(7)) is utilized in Algorithms 1 and 2.

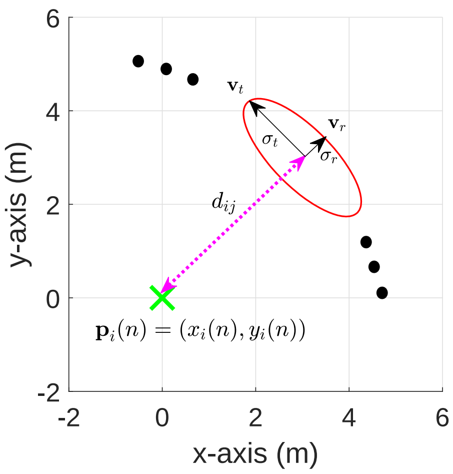


Figure 4. Generation of SoG variables. The dotted magentaline represents the measurement. $\mathbf{v}_r$ and $\mathbf{v}_t$ are, respectively, the radial and tangential unit vectors, and σ_r and σ_t denote, respectively, the radial and tangential standard deviations.

2.2.3. Updating SoG

The update scheme of SoG variables consists of two parts. When collecting the current measurement d_{ij}, the corresponding mean $\mathbf{m}_{ij}^k$ and covariance $\mathbf{C}_{ij}^k$ are first updated using the standard EKF procedure (see [15]). After that, the weights w_{ij}^k between all inter-nodes are updated by using the inter-node measurements to remove the ambiguity.

A detailed update procedure for the weight w_{ij}^k is summarized in Algorithm 3. In order to update w_{ij}^k between the ith and jth nodes, the likelihood $l_{ij,m}^k$ is first computed with respect to the mth nodes connected with the jth node (in lines 2–5 of Algorithm 3), where m is an index of the neighbor nodes. After the normalization of $l_{ij,m}^k$ in lines 6–8, the weight w_{ij}^k is updated in lines 9–11 and normalized in lines 12–14. The above procedures are performed for all possible mth neighbor nodes, i.e., $m \neq i$, $m \neq j$. As more measurements d_{mj} are used, the updated weight w_{ij}^k becomes a larger value if $d_{ij,m}^k$ is close to d_{mj}; otherwise, it will converge rapidly to zero.

Algorithm 3: Weight update procedure.

```
1  for $m \leftarrow 0$ to $M-1$, $m \neq i$, $m \neq j$ do
2      for $k \leftarrow 0$ to $N-1$ do
3          $d_{ij,m}^k = \left\| \mathbf{m}_{ij}^k - \mathbf{p}_m(n) \right\|$
4          $l_{ij,m}^k = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left( -\frac{(d_{ij,m}^k - d_{mj})^2}{2\sigma^2} \right)$
5      end
6      for $k \leftarrow 0$ to $N-1$ do
7          $l_{ij,m}^k = \frac{l_{ij}^k}{\sum_k l_{ij}^k}$
8      end
9      for $k \leftarrow 0$ to $N-1$ do
10         $w_{ij}^k = w_{ij}^k \times l_{ij,m}^k$
11     end
12     for $k \leftarrow 0$ to $N-1$ do
13         $w_{ij}^k = \frac{w_{ij}^k}{\sum_k w_{ij}^k}$
14     end
15 end
```

2.2.4. Merging SoG

After updating the SoG variables, the weighted SoG is merged into a single Gaussian distribution to estimate the new locations of nodes. A closed-form solution [17,18] for the merged SoG is employed for the new location $\mathbf{p}_i(n+1)$ and its covariance $\mathbf{C}_i(n+1)$, as follows:

$$\mathbf{p}_i(n+1) = \frac{1}{\sum_{j,k}\left(w_{ji}^k\right)} \sum_{j,k}\left(w_{ji}^k \mathbf{m}_{ji}^k\right), \tag{8}$$

$$\mathbf{C}_i(n+1) = \frac{1}{\sum_{j,k}\left(w_{ji}^k\right)} \sum_{j,k} w_{ji}^k (\mathbf{C}_{ji}^k + \bar{\mathbf{C}}_{ji}^k), \tag{9}$$

$$\bar{\mathbf{C}}_{ji}^k = \left(\mathbf{m}_{ji}^k - \mathbf{p}_i(n+1)\right)\left(\mathbf{m}_{ji}^k - \mathbf{p}_i(n+1)\right)^T. \tag{10}$$

2.2.5. Transferring Weight

Due to the inherent non-convex nature of anchor-free localization, a local solution can be obtained after merging SoG variables. To prevent falling into a local solution due to a small weight being ignored, a weight transfer method is introduced here by distributing weights around the maximum weight. If $k_{\max}$ indicates an index of the maximum weight from the SoG distribution, the weight can be heuristically distributed as a Gaussian distribution at the center of $k_{\max}$. More specifically, with the mean distance between two indices $k_{\max}$ and k, defined by $d_{k_{\max}-k} = \|\mathbf{m}_{ij}^{k_{\max}} - \mathbf{m}_{ij}^k\|$, the weight is distributed as follows:

$$w_{ij}^k = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{d_{k_{\max}-k}^2}{2\sigma^2}\right), \tag{11}$$

where σ is experimentally determined (e.g., 0.5). Finally, the weight w_{ij}^k is normalized to ensure that its sum is 1.

The effectiveness of the proposed weight transfer technique was verified with a simulation, as shown in Figure 5. In the simulation, five nodes were deployed, where four nodes were placed in a rectangular shape and a robot was moved inside the rectangular shape. The noise variance was set to 0.3–0.8. The RMSE performance was obtained by ensemble averaging over 20 independent trials. As shown in Figure 5, the RMSE performance improved by about 50% with the proposed weight transfer method because it gives a chance to avoid a local minimum.

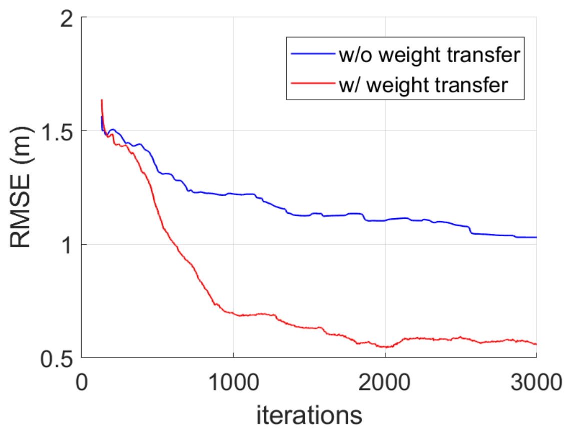


Figure 5. The root-mean-square error (RMSE) performance of map estimation with and without the weight transfer technique.

2.2.6. Convergence

The convergence of each node is determined by using $\tilde{\mathbf{p}}_i(n)$, the time-average of $\mathbf{p}_i(n)$, as follows:

$$\tilde{\mathbf{p}}_i(n+1) = \alpha\tilde{\mathbf{p}}_i(n) + (1-\alpha)\mathbf{p}_i(n). \tag{12}$$

If $\|\tilde{\mathbf{p}}_i(n+1) - \tilde{\mathbf{p}}_i(n)\| < \varepsilon$, then the corresponding node is considered to be converged. Here, ε is a small positive constant threshold. If not converged, the procedure from Section 2.2.3 to Section 2.2.4 is repeated when a new measurement is available.

2.3. Moving Stage

As mentioned in Section 2.1, the movement of any node can be tracked in the moving stage. Unlike the conventional RO-SLAM algorithms, the proposed CDRO-SLAM can deal with the movement of the neighbor nodes as well as the robot's movement. In particular, the tracking procedure during the movement of neighbor nodes was derived in Algorithm 2 by adopting a detection and tracking moving object (DATMO) strategy [19] into a SoG approach. As shown in Algorithm 2, the same procedures as in the initialization stage are recombined; thus, the detailed procedure for the moving stage is not given here. Before concluding this section, it should be noted from Algorithm 2 that the indices of SoG variables in (ii) and (iii) of Algorithm 2 are symmetric counterparts of those in (v) and (vi) (see Figure 2b,c), which will be utilized when developing an efficient implementation, as described in the following section.

3. Efficient Implementation

Due to the extra SoG variables between the inter-nodes, the CDRO-SLAM [14] has a greater computational cost compared with the conventional SoG based RO-SLAM algorithms which use only the direct measurements between a robot and its neighbor nodes. To mitigate this incurred computational burden, several efficient implementation techniques are proposed here.

Efficient implementation can be applied to the corresponding computation parts shown in Algorithms 1–3. Especially, the weight update procedure that causes the main computational burden is summarized in Algorithm 3 for the original one and Algorithm 4 for the efficient implementation to show the computational efficiency.

Algorithm 4: Efficient weight update procedure.

```
1  for $m \leftarrow 0$ to $M-1$, $m \neq i$, $m \neq j$ do
2      for $k \leftarrow 0$ to $N-1$ do
3          $d^k_{ij,m} = \left\|\mathbf{m}^k_{ij} - \mathbf{p}_m(n)\right\|$
4          $l^k_{ij,m} = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(d^k_{ij,m}-d_{mj})^2}{2\sigma^2}\right)$
5          $w^k_{ij} = w^k_{ij} \times l^k_{ij,m}$
6      end
7  end
8  for $k \leftarrow 0$ to $N-1$ do
9      $w^k_{ij} = \frac{w^k_{ij}}{\sum_k w^k_{ij}}$
10 end
11 for $k \leftarrow 0$ to $N-1$ do
12     $r = \mathrm{mod}(k + N/2, N)$
13     $w^r_{ji} = w^k_{ij}$
14 end
```

3.1. Reduced Normalization

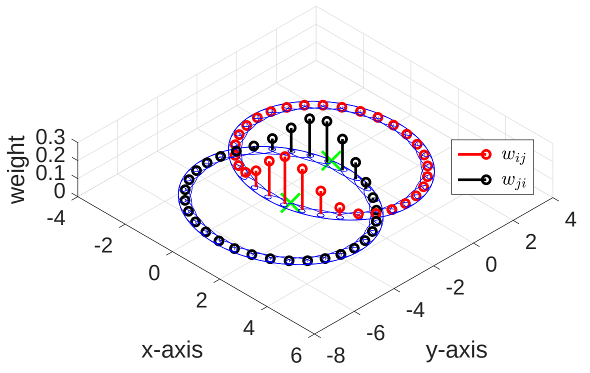
When computing the likelihood $l_{ij,m}^{k}$, its normalization can be omitted without damaging the performance. Similarly, the normalization of the weight w_{ij}^{k} can be carried out only once after the end of for-loop m.

3.2. Symmetric SoG Distribution

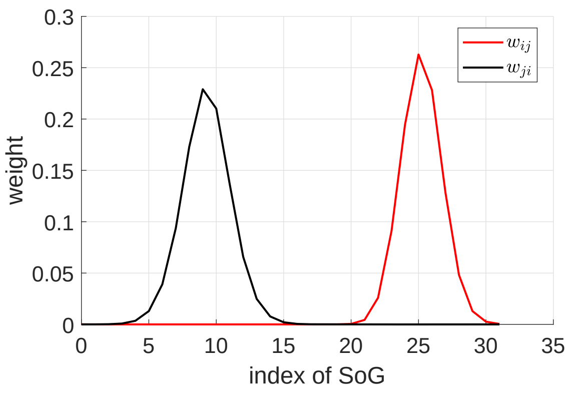
Obviously, a higher weight represents a higher probability of the existence of a node. Let us consider two weights w_{ij}^{k} and w_{ji}^{k} for $i \neq j$. As shown in Figure 6a, the high weight area for each node is symmetrical, i.e., a high w_{ij}^{k} occurs in the direction of a negative y-axis for the w_{ij}^{k} distribution, and a high w_{ji}^{k} occurs in the direction of the positive y-axis for the w_{ji}^{k} distribution.

In Figure 6b, the calculated distribution of two weights w_{ij}^{k} and w_{ji}^{k} is described when $N = 32$ without considering symmetry. If the symmetry property is used, once w_{ij}^{k} is computed (the highest probability occurred in $k = 25$), its counterpart w_{ji}^{k} can be simply copied from w_{ij}^{k} by using the modulo operation (shown in lines 11–14 of Algorithm 4), which gives $k = 9$ for the highest probability and is equal to the index with the calculated distribution of w_{ji}^{k} in Figure 6b.

Furthermore, the symmetry property can be applied in the movement stage. As discussed in the previous section, the SoG distributions of (iii) and (vi) in Algorithm 2 are symmetrical counterparts of each other. Therefore, the weight w_{ji}^{k} of (vi) of Algorithm 2 is simply symmetrically copied from the weight w_{ij}^{k} that was previously computed in (iii) of Algorithm 2.



(**a**)



(**b**)

Figure 6. Weight symmetry between w_{ij}^{k} and w_{ji}^{k}. The green crosses represent the estimated positions of the *i*th and *j*th nodes. (**a**) The SoG variables are distributed in the xy plane, and the weight is shown on the z-axis. (**b**) The weights w_{ij}^{k} and w_{ji}^{k} are depicted against the SoG index *k*.

3.3. Removal of Low Probability

As investigated in several SoG based RO-SLAM algorithms, the merging procedure can be efficiently realized by dropping small Gaussian weights in Equations (8)–(9). The threshold value can be set as small enough, e.g., $0.00001/N$ [20], where N is the number of SoG.

3.4. Efficiency Analysis

In order to show the efficiency, two versions of the proposed algorithm are considered: one is the original CDRO-SLAM and the other is its efficient implementation version (eCDRO-SLAM) with the aforementioned efficient techniques. The computational efficiency of CDRO-SLAM and eCDRO-SLAM is also compared with that of several conventional RO-SLAM algorithms, i.e., the efficient probabilistic RO-SLAM (EPRO-SLAM) [6] and the RBPF-based RO-SLAM (RBPF-RO-SLAM) algorithm [7].

The required computational complexity for updating the weight is considered in terms of the computing likelihood number $l_{ij,m}^{k}$, which is cost-dominant and is summarized in Table 1. Here, M, Q, and N are the numbers of nodes, particles, and SoG, respectively. Note that RBPF-RO-SLAM requires a higher computational cost than EPRO-SLAM, since $Q = \beta \cdot d_{ij}$ is usually determined to be much larger than N, e.g., $Q \geq 1000$ and $N = 32$, where $\beta = [400, 2000]$ [7].

Table 1. Computational complexity of updating the weight.

Algorithms	Likelihood
EPRO-SLAM [6]	N
RBPF-RO-SLAM [7]	Q
CDRO-SLAM	$N(M-2)$
eCDRO-SLAM	$N(M-2)/2$

For CDRO-SLAM, Algorithm 3 requires $N(M-2)$ computations, and for eCDRO-SLAM, Algorithm 4 needs $N(M-2)/2$ computations, on average. The computation of Algorithm 4 is reduced by half compared with Algorithm 3.

It is not easy to compare non-cooperative and cooperative algorithms at the same level because their approaches to estimating the location are totally different, i.e., the cooperative algorithms use many inter-node measurements for the map estimation, but the non-cooperative algorithms use robot movement to measure independent and direct ranges between the robot and its neighbors. As illustrated in Figure 1a–c, the EPRO-SLAM requires several direct measurements at the non-collinear locations to remove the ambiguity in the location of nodes, which results in αN computations in total, where α is the number of required direct measurements at the non-collinear locations. On the other hand, the proposed CDRO-SLAM can yield an accurate location even with one iteration, as described in Figure 3a–c. In this case, the computational burden ratio of the proposed eCDRO-SLAM over EPRO-SLAM is $\frac{(M-2)/2}{\alpha}$ at each iteration. When α is minimal (i.e., $\alpha = 3$ as shown in Figure 1), the computational complexity ratio can be approximately $\frac{M}{6}$. In the case of (at least) $\alpha \approx M$, to achieve a similar ambiguity level to the proposed eCDRO-SLAM, the ratio can be approximately $\frac{1}{2}$.

Furthermore, when updating all nodes, the total computational burden of the proposed eCDRO-SLAM is $NM(M-1)(M-2)/2$, since there are $M(M-1)$ SoG variables between inter-nodes; thus, the total computational burden ratio is about $\frac{NM(M-1)(M-2)/2}{NM\alpha} \approx M/2$ for $\alpha \approx M$, i.e., the total computational burden increases linearly as the number of nodes, M, increases.

4. Experiments

As shown in Figure 7, two sets of experiments were conducted with the same number of nodes ($M = 9$) but different sizes (3.6 m $\times$ 4.8 m for Experiment 1 and 6.6 m $\times$ 8.4 m for Experiment 2). In the experiments, the motion of the robot was arbitrary controlled remotely by a person, and it was represented by using the measured odometry in Figure 7. Also, TurtleBot3 Burger and Pozyx [21] were used in the experiments for the ultra-wide-band (UWB) measurements . For the acquisition of measurements, all the inter-node measurements were initially collected while a robot remained still, and then the direct measurements were collected while the robot moved, in order to clearly show the difference of weight update scheme between the conventional RO-SLAM and the proposed eCDRO-SLAM. The performance of the eCDRO-SLAM with the inter-node measurements was verified with real experimental data, and it was compared with EPRO-SLAM [6] and RBPF RO-SLAM [7], which use direct measurements only. Note that the proposed eCDRO-SLAM was considered only because it gave almost same performance when compared with the CDRO-SLAM, except the computational efficiency. The computational complexity comparison of the eCDRO-SLAM and CDRO-SLAM will be given in Section 4.3. The map estimation and the tracking performance of moving nodes were also investigated.

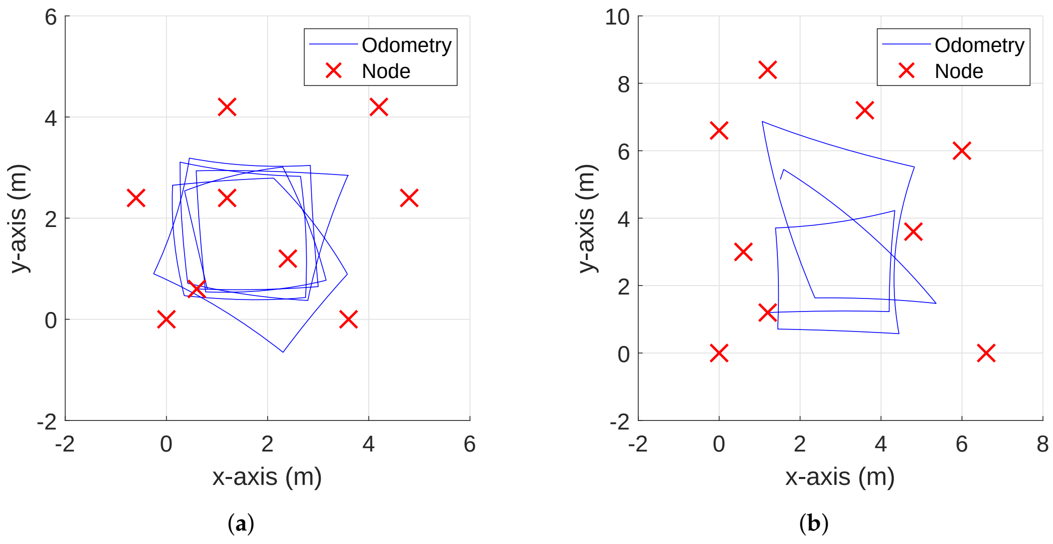
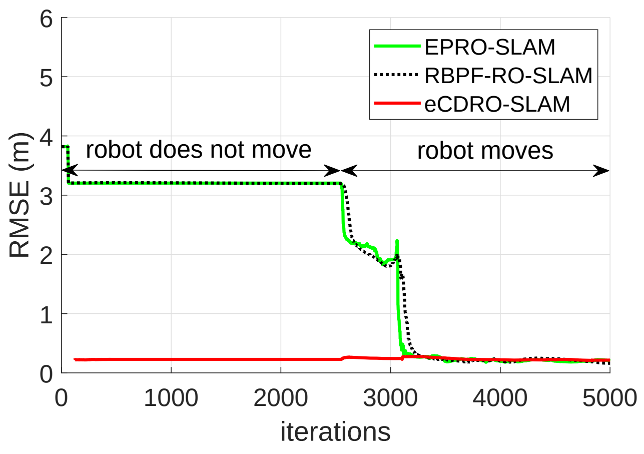


Figure 7. Deployment of nodes and odometry for (**a**) Experiment 1 and (**b**) Experiment 2.

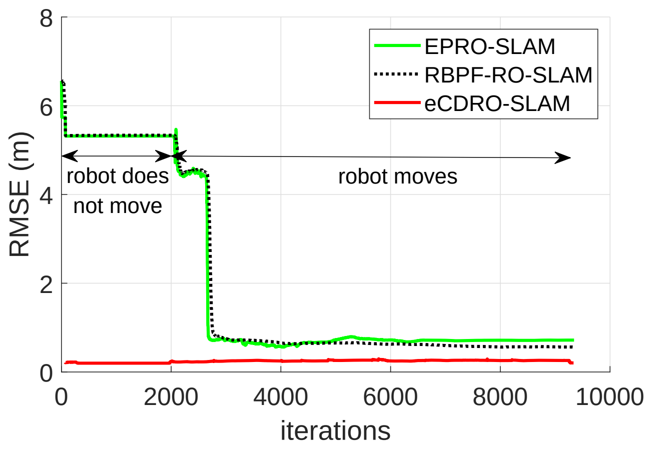
4.1. Map Estimation Performance

The map estimation performance for the two experiments is illustrated in Figure 8, and the following results can be observed:

- The conventional algorithms exhibited a slower convergence rate and higher initial map estimation error than the proposed eCDRO-SLAM, since their convergence depends on both the movement of the robot and the rate of measurements. On the other hand, the convergence of the proposed eCDRO-SLAM depends only on the rate of measurements and is accelerated by employing the inter-node measurements, as discussed before.
- As shown in Figure 8, the proposed eCDRO-SLAM estimates the map even without the movement of a robot; on the other hand, the conventional algorithms can only estimate the map when a robot moves.
- During the refinement of Experiment 1, all algorithms achieved a similar RMSE of about 0.2 m. On the other hand, for Experiment 2, the RBPF-RO-SLAM algorithm (0.56 m) yielded a higher RMSE than the EPRO-SLAM algorithm (0.72 m), as expected. Furthermore, the proposed eCDRO-SLAM algorithm (0.21 m) obtained a better performance than the non-cooperative RBPF-RO-SLAM algorithm. The difference between the two experiments comes from the odometry error. The performance of the RBPF-RO-SLAM (also the EPRO-SLAM) suffered from not only the measurement error but also the odometry error, especially in the large area; on the other hand, the proposed eCDRO-SLAM was affected only by the measurement error, which led to a similar map estimation result for both experiments, regardless of the difference in size.



(**a**)



(**b**)

Figure 8. Map estimation performance obtained by the proposed eCDRO-SLAM and the conventional RO-SLAM algorithms [6,7] for (**a**) Experiment 1 and (**b**) Experiment 2.

4.2. Node Tracking Performance

The node tracking performance of the proposed eCDRO-SLAM was verified with Experiment 1. Each node moves continually in a square shape one-by-one. The tracking performance was measured after the initialization stage, and its result is illustrated in Figure 9. The average localization error was about 0.3688 m in the movement stage, somewhat larger than that of the initial stage; however, a reasonable tracking performance was achieved using the range sensor only. The tracking performance of the proposed eCDRO-SLAM can be further improved if a node moves in a stop-and-go way or if some sensors such as the inertial measurement unit (IMU) are fused. On the other hand, the most conventional RO-SLAM algorithms (including EPRO-SLAM [6] and RBPF-RO-SLAM [7]) cannot be applied to dynamic environments where nodes can have motion.

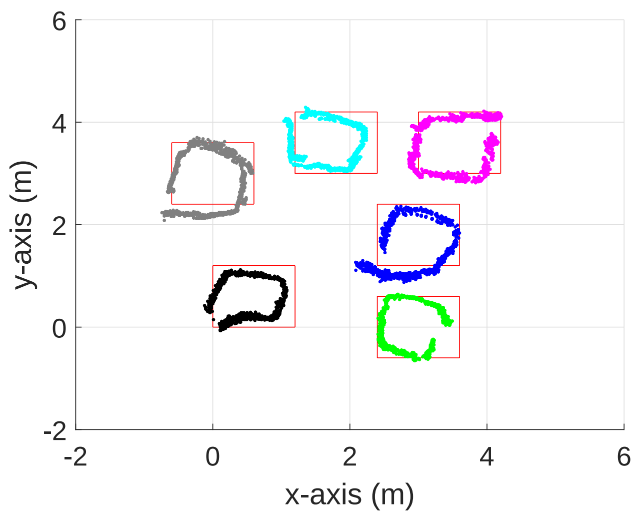


Figure 9. Node tracking performance of the eCDRO-SLAM for Experiment 1.

4.3. Computational Complexity

In Table 2, the computational complexity of the proposed algorithm and the conventional algorithms is summarized in terms of the average CPU time per sample for updating the weight w_{ij}^k and for executing the complete algorithm. The CPU time was measured through implementation in a MATLAB simulation with an Intel Core i7 processor of 3.3 GHz and 8 GB memory.

Table 2. Required CPU time for Experiment 1.

Algorithms	CPU Time (ms)	
	Weight	**Total**
EPRO-SLAM [6]	0.032	1.70
RBPF-RO-SLAM [7]	0.415	13.3
CDRO-SLAM	0.170	22.7
eCDRO-SLAM	0.101	9.20

The required CPU time for computing the weight shown in Table 2 has a similar tendency to that of the efficiency analysis shown in Section 3.4. When considering the total CPU time, it is assumed that all algorithms spend most of the CPU time in updating the weight and merging SoG variables (or particles) to simplify the analysis. From the total CPU time results, we can see the following:

- The total computational burden ratio of the proposed eCDRO-SLAM over the CDRO-SLAM ($9.20/22.7 \approx 0.405$) is smaller than the expected ratio of the weight ($0.101/0.170 \approx 0.594$), since the eCDRO-SLAM is efficiently implemented for both updating and merging SoG processes. As indicated in Section 3.4, the efficiency of the eCDRO-SLAM is about half that of the CDRO-SLAM.
- The eCDRO-SLAM requires $9.20 - 0.101 \times (9 \times 8) = 1.93$ ms to merge SoG variables between all inter-nodes, and the EPRO-SLAM needs $1.70 - 0.032 = 1.67$ ms to merge only one SoG distribution. This efficiency of the eCDRO-SLAM is obtained due to the fast convergence of the weight and because small enough weights are dropped in the merging process. Furthermore, the eCDRO-SLAM is computationally efficient in terms of achieving a similar ambiguity level, since the EPRO-SLAM may require $M = 9$ iterations (see Section 3.4) and, in this case, it needs 1.7 ms $\times$ 9 = 15.3 ms in total.

- Despite the cooperative approach, the proposed eCDRO-SLAM was executed using a smaller total CPU time than the non-cooperative RBPF-RO-SLAM. Note that the RBPF-RO-SLAM spends most of the CPU time for transforming particles into a Gaussian filter.

5. Conclusions

In this paper, an efficient cooperative dynamic RO-SLAM (eCDRO-SLAM) based on the sum of Gaussian (SoG) filter integrating all inter-node measurements for the localization was proposed. A new framework to update the SoG variables using the inter-node measurements leads to faster map estimation with less error than conventional RO-SLAM algorithms using the direct measurements only. Also, several efficient implementation techniques such as the weight symmetry of the SoG distributions were introduced to alleviate the additional computational burden induced when calculating the SoG variables between inter-nodes.

Furthermore, the proposed eCDRO-SLAM was designed to track the movement of some nodes by resetting and updating the SoG variables. With the dynamic property and computational efficiency, its application can be extended to various dynamic environments that are full of movable objects such as unstable disaster sites, unstructured fields, or temporal deployment of robots/devices.

As a future work, a sensor fusion approach integrating the IMU, odometry, and range sensor will be considered to improve the localization and node tracking performance. Furthermore, we will apply the proposed algorithm to various RO-SLAM applications such as robot-assisted search and rescue application and ambient intelligence environments.

Author Contributions: J.-H.K. conducted the experiments and analyzed the data and wrote the manuscript; D.K. provided suggestions and guidance for the work and reviewed and edited the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Korea Institute of Science and Technology (KIST) Institutional Program (2E29460/2E30280).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Lee, T.J.; Kim, C.H.; Cho, D.I.D. A monocular vision sensor-based efficient SLAM method for indoor service robots. *IEEE Trans. Ind. Electron.* **2019**, *66*, 318–328. [CrossRef]
2. Yu, C.; Liu, Z.; Liu, X.; Xie, F.; Yang, Y.; Wei, Q.; Fei, Q. DS-SLAM: A semantic visual SLAM towards dynamic environments. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Madrid, Spain, 1–5 October 2018; pp. 1168–1174.
3. Bjrne, E.; Johansen, T.; Brekke, E. Redesign and analysis of globally asymptotically stable bearing only SLAM. In Proceedings of the IEEE International Conference on Multimedia and Expo, Hong Kong, China, 10–14 July 2017; pp. 1–8.
4. Menegatti, E.; Zanella, A.; Zilli, S.; Zorzi, F.; Pagello, E. Range-only SLAM with a mobile robot and a wireless sensor networks. In Proceedings of the IEEE International Conference on Robotics and Automation, Kobe, Japan, 12–17 May 2009; pp. 8–14.
5. Kantor, G.; Singh, S. Preliminary results in range-only localization and mapping. In Proceedings of the IEEE International Conference on Robotics and Automation, Washington, DC, USA, 11–15 May 2002; pp. 1818–1823.
6. Blanco, J.L.; Fernandez-Madrigal, J.A.; Gonzalez, J. Efficient probabilistic range-only SLAM. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Nice, France, 22–26 September 2008; pp. 1017–1022.
7. Blanco, J.L.; Gonzalez, J.; Fernandez-Madrigal, J.A. A pure probabilistic approach to range-only SLAM. In Proceedings of the IEEE International Conference on Robotics and Automation, Pasadena, CA, USA, 19–23 May 2008; pp. 1436–1441.

8. Djugash, J.; Singh, S.; Kantor, G.; Zhang, W. Range-only SLAM for robots operating cooperatively with sensor networks. In Proceedings of the IEEE International Conference on Robotics and Automation, Orlando, FL, USA, 15–19 May 2006; pp. 2078–2084.
9. Torres-Gonzalez, A.; de Dios, J.M.; Ollero, A. Efficient robot-sensor network distributed SEIF range-only SLAM. In Proceedings of the IEEE International Conference on Robotics and Automation, Hong Kong, China, 31 May–5 June 2014; pp. 1319–1326.
10. Wang, C.; Zhang, H.; Nguyen, T.M.; Xie, L. Ultra-wideband aided fast localization and mapping system. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Vancouver, BC, Canada, 24–28 September 2017; pp. 1602–1609.
11. Urzua, S.; Munguia, R.; Grau, A. Monocular SLAM system for MAVs aided with altitude and range measurements: A GPS-free approach. *J. Intell. Robot. Syst.* **2019**, *94*, 203–217. [CrossRef]
12. Torres-Gonzalez, A.; de Dios, J.M.; Ollero, A. An adaptive scheme for robot localization and mapping with dynamically configurable inter-beacon range measurements. *Sensors* **2014**, *14*, 7684–7710. [CrossRef] [PubMed]
13. Torres-Gonzalez, A.; de Dios, J.M.; Ollero, A. Range-only SLAM for robot-sensor network cooperation. *Auton. Robot.* **2018**, *42*, 649–663. [CrossRef]
14. Kim, J.; Kim, D. Cooperative range-only SLAM based on sum of Gaussian filter in dynamic environment. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Macau, China, 4–8 November 2019; pp. 2139–2144.
15. Vallicrosa, G.; Ridao, P.; Ribas, D.; Palomer, A. Active range-only beacon localization for AUV homing. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, 14–18 September 2014; pp. 2286–2291.
16. Youssef, A.; Agrawala, A. Accurate anchor-free node localization in wireless sensor networks. In Proceedings of the IEEE International Performance, Computing, and Communications Conference, Phoenix, AZ, USA, 7–9 April 2005; pp. 465–470.
17. Kwok, N.; Dissanayake, G.; Ha, Q. Bearing-only SLAM using a SPRT-based Gaussian sum filter. In Proceedings of the IEEE International Conference on Robotics and Automation, Barcelona, Spain, 18–22 April 2005; pp. 1109–1114.
18. Kwok, N.; Ha, Q.; Huang, S.; Dissanayake, G.; Fang, G. Mobile robot localization and mapping using a Gaussian sum filter. *Int. J. Control Autom. Syst.* **2007**, *5*, 251–268.
19. Moratuwage, D.; Vo, B.N.; Wang, D. Collaborative multi-vehicle SLAM with moving object tracking. In Proceedings of the IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, 6–10 May 2013; pp. 5702–5708.
20. Torres-Gonzalez, A.; de Dios, J.M.; Ollero, A. Integrating internode measurements in sum of Gaussians range only SLAM. In *ROBOT2013: First Iberian Robotic Conference*; Springer: Cham, Switzerland, 2014; pp. 473–487.
21. Pozyx Labs, Pozyx Positioning System. Available online: https://www.pozyx.io/ (accessed on 27 April 2020).