Hindawi
Complexity
Volume 2018, Article ID 9104720, 23 pages
https://doi.org/10.1155/2018/9104720

WILEY | Hindawi

Research Article

Maximizing the Coverage of Roadmap Graph for Optimal Motion Planning

Jae-Han Park [1] **and Tae-Woong Yoon** [2]

[1] *Robotics R&D Group, Korea Institute of Industrial Technology (KITECH), Ansan, Republic of Korea*
[2] *School of Electrical Engineering, Korea University, Seoul, Republic of Korea*

Correspondence should be addressed to Tae-Woong Yoon; twy@korea.ac.kr

Received 26 April 2018; Revised 15 September 2018; Accepted 4 October 2018; Published 8 November 2018

Academic Editor: Zhiwei Gao

Automated motion-planning technologies for industrial robots are critical for their application to Industry 4.0. Various sampling-based methods have been studied to generate the collision-free motion of articulated industrial robots. Such sampling-based methods provide efficient solutions to complex planning problems, but their limitations hinder the attainment of optimal results. This paper considers a method to obtain the optimal results in the roadmap algorithm that is representative of the sampling-based method. We define the coverage of a graph as a performance index of its optimality as constructed by a sampling-based algorithm and propose an optimization algorithm that can maximize graph coverage in the configuration space. The proposed method was applied to the model of an industrial robot, and the results of the simulation confirm that the roadmap graph obtained by the proposed algorithm can generate results of satisfactory quality in path-finding tests under various conditions.

1. Introduction

Automatic motion planning for robots is an important technology for next-generation manufacturing systems, such as Industry 4.0. Sampling-based algorithms have been widely used in robot motion-planning problems because they are efficient at providing reasonable solutions [1–4]. Sampling-based algorithms determine feasible paths for the robot's motion using information from a graph that consists of randomly sampled nodes and connected edges in the given configuration space. Such randomized approaches have a strong advantage in terms of quickly providing solutions to complex problems, such as in a high-dimensional configuration space. However, sampling-based methods guarantee probabilistic completeness, and thus they require a large number of sampling nodes to find paths for the motion of the robot in arbitrary cases [2]. The *probabilistic roadmap (PRM)* [3] and *rapidly exploring random trees (RRT)* [5] are representative of such sampling-based motion-planning algorithms.

The PRM method is composed of learning and query phases. In the query phase, collision-free paths for the motion of the robot are obtained by using information from a graph known as the roadmap [2, 3]. The learning phase is the process of constructing a graph that represents the morphology of the given configuration space. In this phase, it is important to construct the graph appropriately to obtain the shortest path for the collision-free motion of the robot. Developments have recently been reported on the optimality of sampling-based algorithms [6–9]. For example, the PRM$*$ algorithm provides criteria for connecting edges that can guarantee asymptotic optimality in terms of the probability of finding path.

An important factor to consider when constructing a graph for optimal motion planning is the node sampling strategy used. Sampling-based algorithms were originally designed with a random sampling of uniform distribution [2, 10]. However, a few drawbacks have been discovered in these uniform sampling strategies, such as the narrow passage problem and failure to capture the true connectivity of spaces of arbitrary shapes [9–11]. Thus, various sampling schemes, such as the Voronoi diagram [11], Gaussian random sampling [12], the bridge test [13], and visibility PRM [14]

algorithms, have been proposed to overcome drawbacks in the uniform sampling method. Nevertheless, the effect of sampling strategies on optimal motion planning remains unclear. The node sampling strategy that fully captures the shape of the free configuration space remains an important problem in optimal motion planning [9, 10]. In the query phase of the roadmap method, the shortest path is obtained as the optimal path using graph information. Thus, constructing an appropriate graph is closely related to the problem of optimal motion planning.

Another area highly relevant to our work here is research on optimization techniques. In recent years, with growing interest in machine learning and Artificial Intelligence, techniques of distributed optimization that can handle large amounts of data and high-dimensional variables have emerged as an important issue. In [15, 16], distributed optimization techniques were considered for multi-agent systems in complex networks. Secure and resilient techniques [17–19] have also been developed to guarantee the stability of distributed algorithms in adversarial environments. As highly relevant to our work here, coverage maximization was studied using methods of distributed optimization in applications of sensor networks [20–22].

This paper proposes an algorithm to optimize the roadmap graph that can cover arbitrary morphologies of the free configuration space to maximize coverage. Our previous study [23] considered an adaptation algorithm for geometric graphs in an intuitive manner. However, in this study, we reanalyze that adaptive graph algorithm and present its results in terms of the maximization of graph coverage. We found that the optimization algorithm updates the positions of the nodes along the direction of expanding the region of covering until it reaches equilibrium, where expansion and contraction factors are balanced. The results of a simulation to test the proposed optimization algorithm show that it can construct a suitable graph to describe arbitrary configuration spaces, and the resulting graph can generate paths of shorter length for various test queries.

2. Problem Statement

The *configuration space* $\mathbf{Q} \subset \mathbf{R}^n$ is the space of all possible configurations of the system, where n is the number of degrees of freedom (DOF) of the system and $\mathbf{R}^n$ is n-dimensional Euclidean space [1]. In the formulation of the configuration space, the robot can be abstracted as a point $\mathbf{q} \in \mathbf{Q}$. *Free configuration* is defined as one where the robot $R(\mathbf{q})$ does not intersect with any obstacle in the workspace. The set of all free configurations is defined as the *free configuration space* $\mathbf{Q}_{free} \subset \mathbf{Q}$, and it can be determined by checking for collisions between obstacles and the robot in the workspace. Every workspace obstacle WO_i is mapped as an obstacle in the configuration space $QO_i = \{\mathbf{q} \in \mathbf{Q} \mid R(\mathbf{q}) \cap WO_i \neq \varphi\}$ [2]. Figure 1 shows the mapping relations between the workspace and the configuration space for a two-link articulated robot.

The problem of motion planning requires as solution a feasible path in the free configuration space for a specified initial motion and a goal. The *path planning problem* can then be defined as follows:

Definition 1 ((path planning problem) [1]). Given a set of collision-free configurations $\mathbf{Q}_{free} \subset \mathbf{Q}$, and initial and goal configurations $\mathbf{q}_{init}, \mathbf{q}_{goal} \in \mathbf{Q}_{free}$, find a continuous curve $\sigma \in \Sigma = \{t \mid t : [0,1] \longrightarrow \mathbf{Q}_{free}\}$, where $\sigma(0) = \mathbf{q}_{init}$ and $\sigma(1) = \mathbf{q}_{goal}$.

In this study, we employ the *roadmap* method for path planning. The roadmap represents the configuration space topologically by a network of one-dimensional (1D) curves. It is defined as follows.

Definition 2 ((roadmap) [2]). A union of 1D curves is a *roadmap* **RM** if, for all $\mathbf{q}_{init}$ and $\mathbf{q}_{goal}$ in $\mathbf{Q}_{free}$ that can be connected by a path, the following properties hold:

(1) ***Accessibility***: there exists a path from $\mathbf{q}_{init} \in \mathbf{Q}_{free}$ to some $\mathbf{q}'_{init} \in \mathbf{RM}$,

(2) ***Departability***: there exists a path from $\mathbf{q}'_{goal} \in \mathbf{RM}$ to $\mathbf{q}_{goal} \in \mathbf{Q}_{free}$,

(3) ***Connectivity***: there exists a path in **RM** between $\mathbf{q}'_{init}$ and $\mathbf{q}'_{goal}$.

The result of the roadmap method is a graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ consisting of a set of N nodes $\mathbf{V} = \{\mathbf{q}_i \in \mathbf{Q}_{free} \mid \forall i \in \{1, \cdots, N\}\}$ and a set of edges $\mathbf{E} = \{e_{ij} = (\mathbf{q}_i, \mathbf{q}_j) \mid \mathbf{q}_i, \mathbf{q}_j \in \mathbf{V}\}$. Edge e_{ij} is created if the line segment between nodes $\mathbf{q}_i$ and $\mathbf{q}_j$ persists in $\mathbf{Q}_{free}$, i.e., $t\mathbf{q}_i + (1-t)\mathbf{q}_j \in \mathbf{Q}_{free}$ $\forall t \in [0,1]$.

To obtain a feasible path for the robot's motion using information from the roadmap graph, the three properties in Definition 2 need to be met. Furthermore, for optimal planning, the best roadmap must be determined among a set of feasible roadmaps. We now employ roadmap coverage as a performance index to assess the optimality of the roadmap graph.

To describe roadmap coverage, the *covering region of node* $\mathbf{q}_i$ for a certain *neighbor radius* r_η, is defined as follows:

$$\mathbf{B}_i = \mathbf{B}\left(\mathbf{q}_i, r_\eta\right) = \left\{\mathbf{q} \in \mathbf{R}^n \mid \left\|\mathbf{q} - \mathbf{q}_i\right\|_2 \leq \frac{r_\eta}{2},\ \mathbf{q}_i \in \mathbf{V}\right\}. \quad (1)$$

Let $\mathscr{B} = \{\mathbf{B}_1, \mathbf{B}_2, \cdots, \mathbf{B}_N\}$ be the family of sets of $\mathbf{B}_i$; subsequently, the *roadmap coverage* for neighbor radius r_η is defined as the union of $\mathbf{B}_i$s as follows.

Definition 3 (roadmap coverage). *Roadmap coverage* **C** for roadmap **G** is the region of the union of the family of sets $\mathscr{B}$:

$$\mathbf{C}(\mathbf{G}) = \bigcup \mathscr{B} = \mathbf{B}_1 \bigcup \mathbf{B}_2 \bigcup \cdots \bigcup \mathbf{B}_N. \quad (2)$$

Figure 2 shows an example of a roadmap and its coverage for a configuration space.

Let $\mu(\cdot)$ be the Lebesgue measure; by the inclusion–exclusion principle [25, 26], the volume of roadmap coverage $\mu(\mathbf{C})$can be represented as follows:

$$\mu(\mathbf{C}(\mathbf{G})) = \mu\left(\bigcup \mathscr{B}\right) = \sum_{i=1}^{N} (-1)^{i-1} S_i, \quad (3)$$

where $S_i = \sum_{1 \leq j_1 < j_2 \cdots < j_i \leq N} \mu(\mathbf{B}_{j_1} \cap \mathbf{B}_{j_2} \cap \cdots \cap \mathbf{B}_{j_i})$.

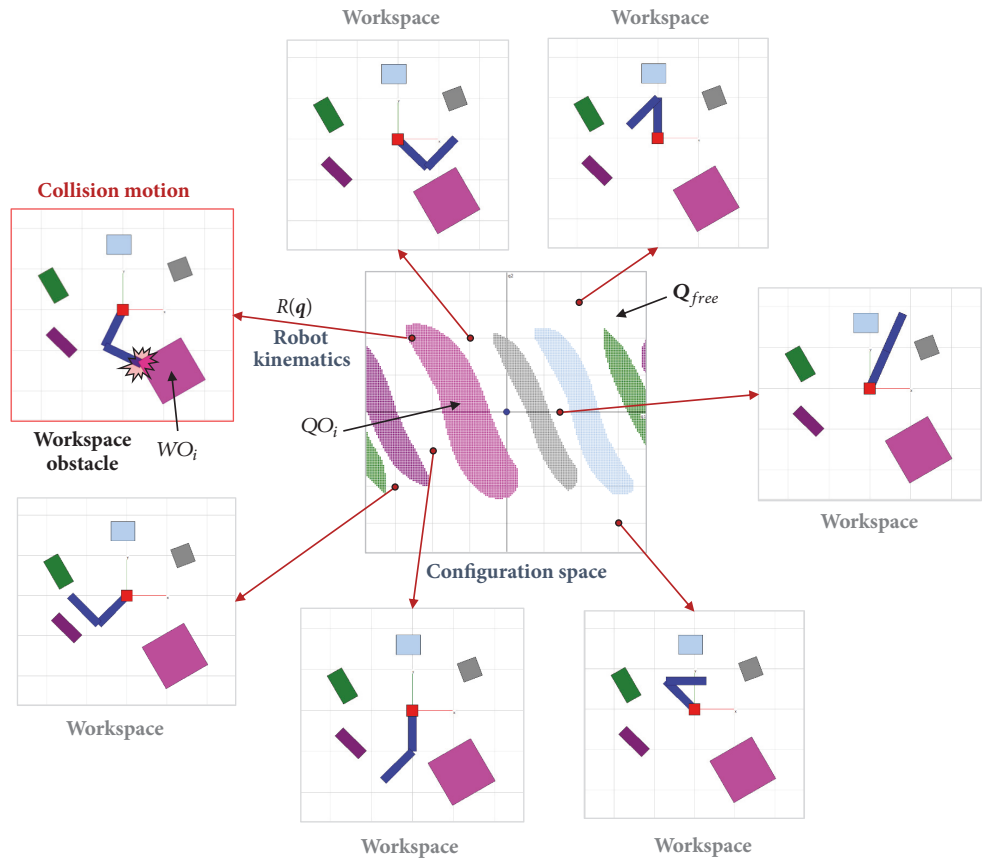


Figure 1: Mapping relations between the workspace and the configuration space.

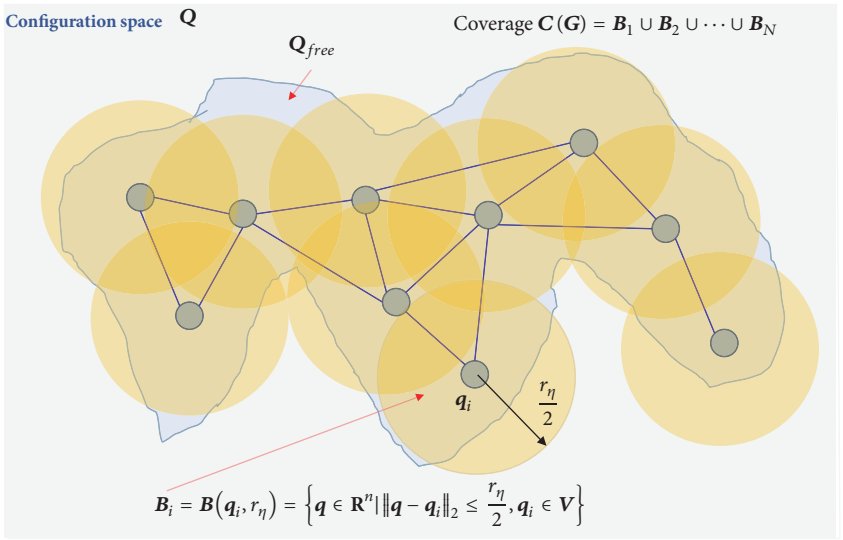


Figure 2: Example of a roadmap and its coverage.

According to the properties in Definition 2, the optimality of the roadmap can be considered to be the capability of finding paths for an arbitrarily set initial point and goal in the configuration space. To satisfy this requirement, roadmap coverage should cover the entire free configuration space. As a quantitative representation of this performance index, we define the *optimal roadmap* as the graph that maximizes the volume of roadmap coverage as follows.

Definition 4 (optimal roadmap). The *optimal roadmap* $\mathbf{G}^*$ is the graph that maximizes the volume of roadmap coverage within the finite free configuration space:

$$\begin{aligned} \mathbf{G}^* = \underset{\mathbf{V}=\{q_1,q_2,\cdots,q_N\}}{\text{maximize}} \quad & \mu\left(\mathbf{C}\left(\mathbf{G}\right)\right) \\ \text{subject to} \quad & \mathbf{q}_i \in \mathbf{Q}_{free}, \quad i = 1, \cdots, N. \end{aligned} \tag{4}$$

The volume of roadmap coverage is primarily related to the geometric structure of the graph. It is complicated to calculate, especially in case the configuration space is high dimensional and the roadmap consists of a large number of nodes. Thus, an alternative performance index that uses the upper or lower bound for the volume of roadmap coverage is needed to render the optimization problem solvable.

Let the sum of terms consisting of two or more intersections be $I(\mathscr{B}) = \sum_{i=2}^{N}(-1)^i S_i$. Therefore, (3) can be written as

$$\begin{aligned} \mu\left(\mathbf{C}\left(\mathbf{G}\right)\right) &= \mu\left(\bigcup \mathscr{B}\right) = \sum_{i=1}^{N}\left(-1\right)^{i-1} S_i \\ &= S_1 + \sum_{i=2}^{N}\left(-1\right)^{i-1} S_i = \sum_{i=1}^{N}\mu\left(\mathbf{B}_i\right) - I\left(\mathscr{B}\right), \end{aligned} \tag{5}$$

where $\sum_{i=1}^{N}\mu(\mathbf{B}_i) = N \cdot \mathbf{\Gamma}(n/2+1)^{-1}(\sqrt{\pi}\cdot r_\eta)^n$, and $\mathbf{\Gamma}(x)$ is the gamma function. The problem of designing the optimal roadmap can be transformed into a minimization problem as follows:

$$\begin{aligned} \mathbf{G}^* = \underset{\mathbf{V}=\{q_1,q_2,\cdots,q_N\}}{\text{minimize}} \quad & I\left(\mathscr{B}\right) \\ \text{subject to} \quad & \mathbf{q}_i \in \mathbf{Q}_{free}, \quad i = 1, \cdots, N. \end{aligned} \tag{6}$$

Moreover, the upper bound of $I(\mathscr{B})$, which is the sum of the intersection terms, can be expressed as follows [Section A.1]:

$$I\left(\mathscr{B}\right) \le \kappa \cdot S_2 = \kappa \cdot \left(\sum_{1\le i<j\le N}\mu\left(\mathbf{B}_i \cap \mathbf{B}_j\right)\right), \tag{7}$$

where $\kappa = 2/N(N-1)\cdot\sum_{i=2}^{N}\left(\coprod_{j=2}^{i}\binom{N}{j}\right)$ is a finite constant that depends on the number of nodes N.

This upper bound on $I(\mathscr{B})$ can be applied to the following *suboptimal roadmap* problem.

Definition 5 (suboptimal roadmap). The *suboptimal roadmap* $\widehat{\mathbf{G}}^*$ is the graph that minimizes the upper bound of $I(\mathscr{B})$ in (7) such that

$$\begin{aligned} \widehat{\mathbf{G}}^* = \underset{\mathbf{V}=\{q_1,q_2,\cdots,q_N\}}{\text{minimize}} \quad & \sum_{1\le i<j\le N}\mu\left(\mathbf{B}_i \cap \mathbf{B}_j\right) \\ \text{subject to} \quad & \mathbf{q}_i \in \mathbf{Q}_{free}, \quad i = 1, \cdots, N. \end{aligned} \tag{8}$$

3. Coverage Maximization Algorithm

Let $\Phi(\mathbf{q}_i, \mathbf{q}_j)$ be the volume of the region of intersection between nodes, $\mu(\mathbf{B}_i \cap \mathbf{B}_j)$. By replacing the constraints $\mathbf{q}_i \in \mathbf{Q}_{free}$ with the collision detection function $s(\mathbf{q}_i)$, the optimization problem in Definition 5 can be represented as follows:

$$\begin{aligned} \widehat{\mathbf{G}}^* = \underset{\mathbf{V}=\{q_1,q_2,\cdots,q_N\}}{\text{minimize}} \quad & \sum_{1\le i<j\le N}\Phi\left(\mathbf{q}_i, \mathbf{q}_j\right) \\ \text{subject to} \quad & s\left(\mathbf{q}_i\right) = 0, \quad i = 1, \cdots, N, \end{aligned} \tag{9}$$

where $s(\mathbf{q}_i)$ is the collision detection function defined as

$$s\left(\mathbf{q}_i\right) = \begin{cases} 1 & \text{(collision detected)} \\ 0 & \text{(otherwise)}. \end{cases} \tag{10}$$

The volume of the region of intersection between n-dimensional hyperspheres with respect to the neighbor radius r_η is

$$\Phi\left(\mathbf{q}_i, \mathbf{q}_j\right) = \begin{cases} 2\cdot\displaystyle\int_{d_{ij}/2}^{r_\eta/2}\frac{\sqrt{\pi}^{n-1}}{\mathbf{\Gamma}\left(1+(n-1)/2\right)}\sqrt{\left(r_\eta/2\right)^2 - t^2}^{\,n-1}\,dt & \left(0 \le d_{ij} < r_\eta\right) \\ 0 & \left(d_{ij} \ge r_\eta\right), \end{cases} \tag{11}$$

where $d_{ij} = \|\mathbf{q}_i - \mathbf{q}_j\|_2$. See [Section A.2] for details.

Using the *Lagrange multiplier* λ, the optimization problem in (9) can be transformed as follows [27]:

$$\widehat{\mathbf{G}}^* = \text{minimize} \quad \mathbf{H}, \tag{12}$$

where $\mathbf{H} = \sum_{1\le i<j\le N}\Phi(\mathbf{q}_i, \mathbf{q}_j) + \sum_{i=1}^{N}\lambda_i s(\mathbf{q}_i)$.

The solution of this minimization problem is a node set that makes the gradient of $\mathbf{H}$ to zero, i.e., $\nabla\mathbf{H} = \sum_{1\le i<j\le N}\nabla\Phi(\mathbf{q}_i, \mathbf{q}_j) + \sum_{i=1}^{N}\lambda_i\nabla s(\mathbf{q}_i) = 0$.

Let $\mathbf{x} = [\mathbf{q}_i] \in \mathbf{R}^{nN}$ be the stacked vector of N nodes. Using the steepest descent method [27], the algorithm to minimize (9) can be described as follows:

$$\begin{aligned} \mathbf{x}\left[k+1\right] &= \mathbf{x}\left[k\right] - \varepsilon\nabla\mathbf{H}_k, \\ \nabla\mathbf{H}_k &= \sum_{1\le i<j\le N}\nabla\Phi\left(\mathbf{q}_i\left[k\right], \mathbf{q}_j\left[k\right]\right) \\ &\quad + \sum_{i=1}^{N}\lambda_i\left[k\right]\nabla s\left(\mathbf{q}_i\left[k\right]\right), \end{aligned} \tag{13}$$

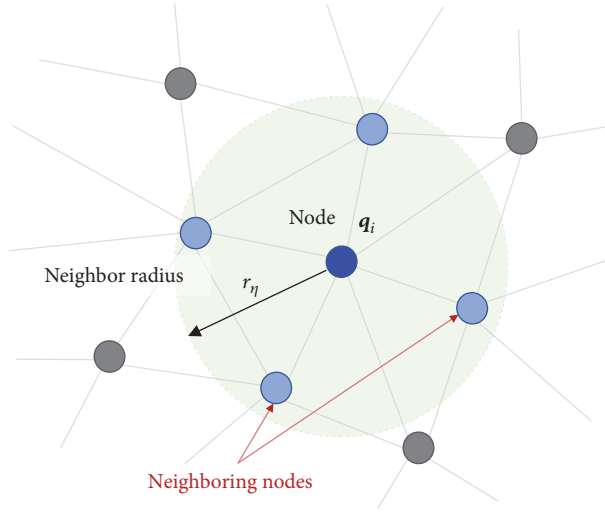


FIGURE 3: Nodes neighboring $\mathbf{q}_i$ within radius r_η [23].

where k is the iteration index, and $\varepsilon > 0$ is the step size of each iteration. By the penalty parameter $c[k]$, the Lagrange multiplier $\lambda_i[k]$ can be determined by the following updating formula [27]:

$$\lambda_i[k+1] = \lambda_i[k] + c[k]\, s(\mathbf{q}_i[k]), \tag{14}$$

where the penalty parameter is chosen to satisfy $c[k+1] \geq c[k]$.

If the distance $d_{ij} = \|\mathbf{q}_i - \mathbf{q}_j\|_2$ between nodes $\mathbf{q}_i$ and $\mathbf{q}_j$ is greater than the neighbor radius r_η, the volume of the region of intersection becomes zero, i.e., $\Phi(\mathbf{q}_i, \mathbf{q}_j) = 0$. Also $\nabla\Phi(\mathbf{q}_i, \mathbf{q}_j)$ for node $\mathbf{q}_i$ is valid only when $\mathbf{q}_j$ is located within r_η. Thereby, as shown in Figure 3, we define the *neighboring node* set located within the neighbor radius as

$$\boldsymbol{\eta}_i = \left\{\mathbf{q}_j \in \mathbf{V} \mid \left\|\mathbf{q}_j - \mathbf{q}_i\right\|_2 \leq r_\eta,\ j \neq i\right\}. \tag{15}$$

Furthermore, using the *decomposition method* [28–30], the process in (13) can be transformed into a decentralized form that is processed at each node as follows:

$$\begin{aligned}\mathbf{q}_i[k+1] = \mathbf{q}_i[k] &- \varepsilon \sum_{\mathbf{q}_j[k] \in \boldsymbol{\eta}_i} \nabla\Phi\left(\mathbf{q}_i[k], \mathbf{q}_j[k]\right) \\ &- \varepsilon\lambda_i[k]\, \nabla s(\mathbf{q}_i[k]).\end{aligned} \tag{16}$$

$\nabla s(\mathbf{q}_i)$ in (16) is, however, not available because $s(\mathbf{q}_i)$ is discontinuous. In such cases, the gradient of $s(\mathbf{q}_i)$ can be estimated stochastically through *response surface method* (RSM) [31–33]. To apply the RSM algorithm, we define the *sensing node* $\Delta\mathbf{q} \in \mathbf{R}^n$ as follows.

Definition 6 (sensing node). A *sensing node* $\Delta\mathbf{q}$ is a point evenly sampled by Poisson-disk sampling [34] on $\partial\mathbf{B}_{r_s} = \{\mathbf{q} \in \mathbf{R}^n \mid \|\mathbf{q}\|_2 = r_s\}$, which is the surface of an n-dimensional hypersphere of radius r_s. The sensing nodes detect the deformation of the configuration space around each node. The set of sensing nodes $\mathbf{S}$ can be expressed as

$$\begin{aligned}\mathbf{S} = \Big\{\Delta\mathbf{q}_l \in \partial\mathbf{B}_{r_s} \mid l = 1, \cdots, n_s,\ \left\|\Delta\mathbf{q}_i - \Delta\mathbf{q}_j\right\|_2 \\ > r_p \text{ for } \forall i, j \in \{1, \cdots, n_s\}\Big\},\end{aligned} \tag{17}$$

where r_p and n_s denote the sampling radius of the Poisson disk and the number of sensing nodes, respectively.

Figure 4(a) shows an example of sensing nodes sampled on the surface of a sphere, and Figure 4(b) shows a node and its sensing nodes in 3D configuration space.

Let $\mathbf{M} = [\Delta\mathbf{q}_l^T] \in \mathbf{R}^{n_s \times n}$ be a matrix composed of sensing nodes and $\mathbf{w}_i = [s(\mathbf{q}_i + \Delta\mathbf{q}_l)] \in \mathbf{R}^{n_s}$ be a stacked vector representing sensing information. If the sensing nodes are selected symmetrically such that $\sum_{l=1}^{n_s} \Delta\mathbf{q}_l = \mathbf{0}$, the estimated gradient $\nabla\hat{s}(\mathbf{q}_i)$ can be calculated by the RSM as follows:

$$\nabla s(\mathbf{q}_i) \approx \hat{\nabla} s(\mathbf{q}_i) = \nabla\hat{s}(\mathbf{q}_i) = \mathbf{M}^+ \mathbf{w}_i, \tag{18}$$

where $\mathbf{M}^+ = (\mathbf{M}^T\mathbf{M})^{-1}\mathbf{M}^T$ is the pseudoinverse matrix of $\mathbf{M}$ and $\mathbf{M}^T\mathbf{M}$ is a nonsingular matrix. See [Section A.3] for details.

$\Phi(\mathbf{q}_i, \mathbf{q}_j)$ represents the interaction between two nodes $\mathbf{q}_i$ and $\mathbf{q}_j$, and it is a function of the distance $d_{ij} = \|\mathbf{q}_i - \mathbf{q}_j\|_2$. The gradient of $\Phi(\mathbf{q}_i, \mathbf{q}_j)$ can be described as follows [35]:

$$\nabla\Phi\left(\mathbf{q}_i, \mathbf{q}_j\right) = \frac{\partial\Phi\left(d_{ij}\right)}{\partial d_{ij}} \cdot \frac{\mathbf{q}_i - \mathbf{q}_j}{d_{ij}}. \tag{19}$$

Then, it follows from (11) that

$$\frac{\partial\Phi\left(d_{ij}\right)}{\partial d} = \begin{cases} -2\dfrac{\sqrt{\pi/4}^{n-1}}{\Gamma\left(1 + (n-1)/2\right)}\sqrt{r_\eta^2 - d_{ij}^2}^{n-1} & \left(0 \leq d_{ij} < r_\eta\right) \\ 0 & \left(d_{ij} \geq r_\eta\right). \end{cases} \tag{20}$$

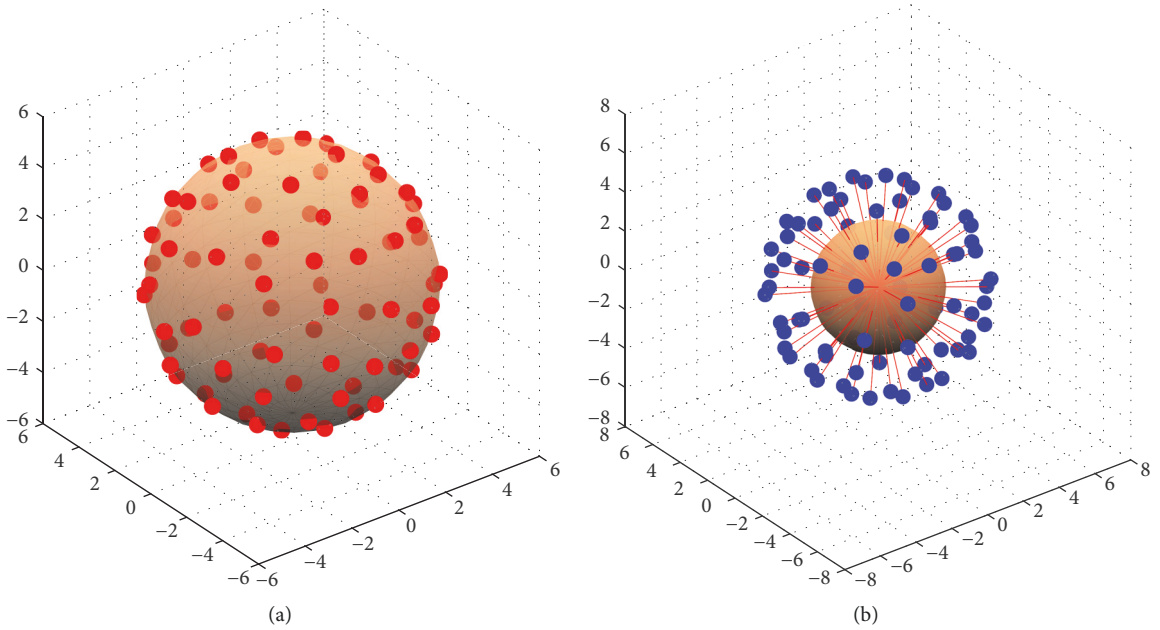


FIGURE 4: Example of sampled sensing nodes and a node in 3D configuration space. (a) Sensing nodes sampled on $\partial \mathbf{B}_{r_S}$ ($r_s = 5$). (b) A node and its sensing nodes.

It is noteworthy that $\partial\Phi(d_{ij})/\partial d \leq 0$. We now set a nonnegative value of $g(d_{ij})$ as

$$g\left(d_{ij}\right) = \begin{cases} -\dfrac{1}{d_{ij}} \cdot \dfrac{\partial \Phi\left(d_{ij}\right)}{\partial d_{ij}} & \left(0 < d_{ij} < r_\eta\right) \\ 0 & (\text{otherwise}) . \end{cases} \tag{21}$$

Subsequently, the distributed optimization algorithm in (16) can be reformulated as follows:

$$\begin{aligned} \mathbf{q}_i [k+1] = \mathbf{q}_i [k] & \\ + \varepsilon \sum_{\mathbf{q}_j[k] \in \boldsymbol{\eta}_i} & g\left(d_{ij}[k]\right)\left(\mathbf{q}_i[k] - \mathbf{q}_j[k]\right) \\ - \varepsilon \lambda_i [k] & \mathbf{M}^{+} \mathbf{w}_i [k] . \end{aligned} \tag{22}$$

In this algorithm, the major variable influencing the amount of computation needed is the number of nodes N. The main computational tasks of the optimization process are the construction of the Laplacian matrix and the computation of the sensing vector. In the original optimization process in (13), computational complexities according to the number of nodes N are estimated as quadratic $O(N^2)$ for the Laplacian matrix and linear $O(N)$ for the sensing vector, respectively. However, as this algorithm can be decomposed as shown in (16), if a many-core processor such as a General-Purpose computing on Graphics Processing Unit (GPGPU) is available, (22) can be computed at each node in a separate processor in parallel. Thus, if the optimization algorithm can be executed by using parallel computation, the computational complexity of the Laplacian matrix and the configuration of the sensing vector are linear $O(N)$ and constant $O(c)$, respectively.

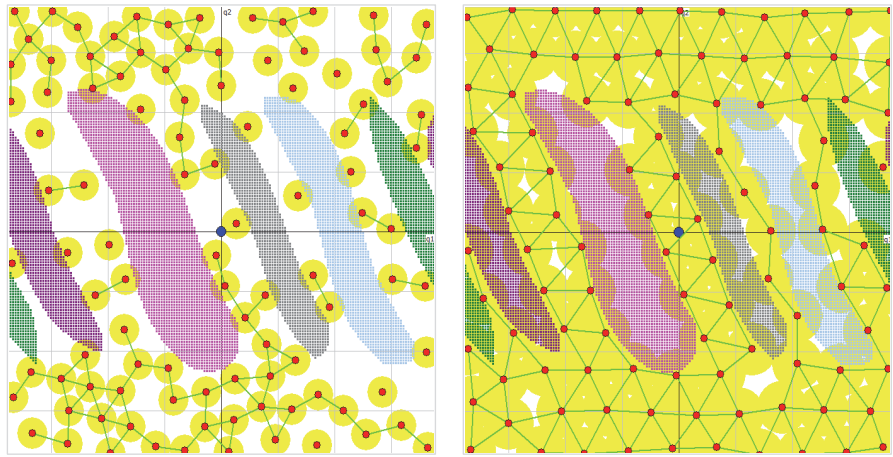
The algorithm in (22) is an iterative process that can converge to a suboptimal roadmap by detecting the boundary of the state of collision of the configuration space through the sensing vector $\mathbf{w}_i$. Even if the morphology of the configuration space changes, the structure of the roadmap graph can adapt to it accordingly because the optimization process is continuously performed using the sensing node. In the general optimization algorithm, the Lagrange multiplier $\lambda_i[k]$ is updated at every iteration. This is suitable in fixed configuration spaces, but it is difficult to apply to changeable spaces because if the space changes, the updated $\lambda_i[k]$ becomes useless and needs to be recalculated to obtain the initial conditions. We set the Lagrange multiplier to a constant value λ as an iteration gain. Further, we set the same value for each node such that $\lambda_1 = \lambda_2 = \cdots = \lambda_N = \lambda$. The results of a simulation show that the algorithm works well with a fixed λ even with a changing configuration space.

4. Equilibrium State Analysis

This section analyzes the states of equilibrium of the algorithm in (22). For the equilibrium analysis, the differential equation for the vector $\mathbf{x} = [\mathbf{q}_i] \in \mathbf{R}^{nN}$ should be considered.

Let the vector $\mathbf{w} = [\mathbf{w}_i] \in \mathbf{R}^{n_s N}$ be the stacked vector of sensing vector $\mathbf{w}_i$ at each node, and let the *Laplacian matrix* $\mathbf{L} = [L_{ij}] \in \mathbf{R}^{N \times N}$ for $g(d_{ij}) \geq 0$ be as follows:

$$\mathbf{L} = \left[L_{ij}\right], \quad L_{ij} = \begin{cases} -g\left(d_{ij}\right) & i \neq j \\ \displaystyle\sum_{k=1, i \neq k}^{N} g\left(d_{ik}\right) & i = j. \end{cases} \tag{23}$$



(a) (b)

Figure 5: Examples of states of equilibrium: The yellow region shows the coverage of the roadmap. (a) An ill-structured roadmap. (b) A properly constructed roadmap.

With the Lagrange multipliers λ_i set to $\lambda_1 = \lambda_2 = \cdots = \lambda_N = \lambda$, the decentralized differential equation shown in (22) can be integrated for $\mathbf{x} \in \mathbf{R}^{nN}$ as follows [Section A.4]:

$$\begin{aligned}\mathbf{x}[k+1] &= \mathbf{x}[k] + \varepsilon\left(\mathbf{L}[k] \otimes \mathbf{I}_n\right)\mathbf{x}[k] \\ &\quad - \varepsilon\lambda\left(\mathbf{I}_N \otimes \mathbf{M}^+\right)\mathbf{w}[k] \\ &= \left(\mathbf{I}_{nN} + \varepsilon\left(\mathbf{L}[k] \otimes \mathbf{I}_n\right)\right)\mathbf{x}[k] \\ &\quad - \varepsilon\lambda\left(\mathbf{I}_N \otimes \mathbf{M}^+\right)\mathbf{w}[k],\end{aligned} \tag{24}$$

where $\otimes$ is the *Kronecker product operator* and $\mathbf{I}_k \in R^{k\times k}$ is a k-dimensional identity matrix.

Let $\mathbf{x}^*$ be the whole-state variable in equilibrium, and let $\mathbf{w}^*$ and $\mathbf{L}^*$ be the entire sensing vector and the Laplacian matrix at that time, respectively. Subsequently, the differential equation at equilibrium is

$$\begin{aligned}\mathbf{x}^* &= \left(\mathbf{I}_{nN} + \varepsilon\left(\mathbf{L}^* \otimes \mathbf{I}_n\right)\right)\mathbf{x}^* - \varepsilon\lambda\left(\mathbf{I}_N \otimes \mathbf{M}^+\right)\mathbf{w}^* \\ &= \mathbf{x}^* + \varepsilon\left(\mathbf{L}^* \otimes \mathbf{I}_n\right)\mathbf{x}^* - \varepsilon\lambda\left(\mathbf{I}_N \otimes \mathbf{M}^+\right)\mathbf{w}^*.\end{aligned} \tag{25}$$

In (25), the condition that renders the whole-state vector $\mathbf{x}$ in equilibrium is

$$\varepsilon\left(\mathbf{L}^* \otimes \mathbf{I}_n\right)\mathbf{x}^* - \varepsilon\lambda\left(\mathbf{I}_N \otimes \mathbf{M}^+\right)\mathbf{w}^* = \mathbf{0}. \tag{26}$$

The cases of states of equilibrium that can meet such a condition can be summarized as follows.

Remark 7 (equilibrium conditions of suboptimal roadmap problem). The *conditions of equilibrium states* for the suboptimal roadmap algorithm are classified into three cases as follows:

Case 1 ($\mathbf{L}^* = \mathbf{0}, \mathbf{w}^* = \mathbf{0}$). All nodes and their corresponding sensing nodes are in $\mathbf{Q}_{free}$. Furthermore, no neighbor node exists because the neighbor radius r_η is small for $\mu(\mathbf{Q}_{free})$, such that $\mu(\mathbf{Q}_{free}) > N \cdot \mu(\mathbf{B}(\mathbf{q}_i, r_\eta))$. Figure 5(a) shows an example of this case.

Case 2 ($\mathbf{L}^* \neq \mathbf{0}, \mathbf{x}^* \in Null(\mathbf{L}^* \otimes \mathbf{I}_n), \mathbf{w}^* = \mathbf{0}$). In this case, $\mathbf{x}$ is a null-space element of the matrix $\mathbf{L}^* \otimes \mathbf{I}_n$, and all sensing nodes are in $\mathbf{Q}_{free}$. If the graph is formed by a single connected component, the null-space elements are $\mathbf{x} = [\mathbf{q}_i]$, where $\mathbf{q}_1 = \mathbf{q}_2 = \cdots = \mathbf{q}_N$ [36]. Such states are rare in dynamical processes. Thus, herein, we do not consider this type of equilibrium.

Case 3 ($\mathbf{L}^* \neq \mathbf{0}, \mathbf{x}^* \notin Null(\mathbf{L}^* \otimes \mathbf{I}_n), \mathbf{w}^* \neq \mathbf{0}$). The neighbor nodes and information regarding the sensing nodes exist. In this case, the equilibrium is a point, and this is the desirable solution to the suboptimal roadmap problem. Figure 5(b) shows an example of this case. In this state, the expanding factor between the neighbor nodes and the factor of sensing nodes that detect the boundary of collision are in balance as follows:

$$\left(\mathbf{L}^* \otimes \mathbf{I}_n\right)\mathbf{x}^* = \lambda\left(\mathbf{I}_N \otimes \mathbf{M}^+\right)\mathbf{w}^*. \tag{27}$$

The maximum rank of the Laplacian matrix is n–1 [37]. Although information concerning the Laplacian matrix $\mathbf{L}^*$ and sensing vector $\mathbf{w}^*$ in equilibrium state is given, the state vector $\mathbf{x}^*$ cannot be calculated using (27). However, if the graph is a single connected component such as $rank(\mathbf{L}^*) = n - 1$ [36, 37], the state vector $\mathbf{x}^*$ can be obtained with some additional constraints.

If a constraint such as $\sum_{i=1}^{N} \mathbf{q}_i = \mathbf{0}$ is added, the state vector in equilibrium can be calculated as follows:

$$\mathbf{x}^* = \left(\begin{bmatrix}\mathbf{L}^* \\ \frac{1}{N} \cdot \mathbf{1}_N^T\end{bmatrix} \otimes \mathbf{I}_n\right)^+ \begin{bmatrix}\lambda\left(\mathbf{I}_N \otimes \mathbf{M}^+\right)\mathbf{w}^* \\ \mathbf{0}\end{bmatrix}, \tag{28}$$

where the symbol "+" indicates the pseudoinverse matrix and $\mathbf{1}_N$ is an N-dimensional vector.

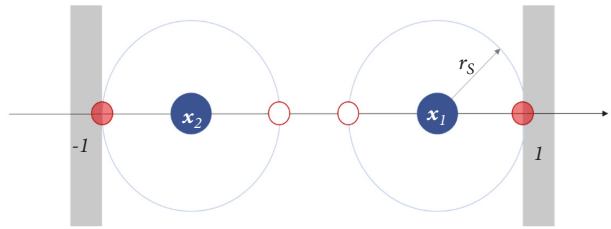


Figure 6: Structure of roadmap for Example 8.

We now verify this result through a simple example of a roadmap with two nodes in a 1D configuration space as shown in Figure 6.

Example 8 (two-node roadmap in 1D space). Obtain the equilibrium point $\boldsymbol{x}^* = \begin{bmatrix} x_1^* \\ x_2^* \end{bmatrix}$ for $\mathbf{Q}_{free} = \{x \in \mathbf{R} \mid -1 \leq x \leq 1\}$, $r_s = 0.6$, $\lambda = 1$, $\mathbf{L}^* = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$, and $\mathbf{w}^* = \begin{bmatrix} 1 & 0 & 0 & 1 \end{bmatrix}^T$.

The sensing node matrix is $\mathbf{M} = \begin{bmatrix} 0.6 \\ -0.6 \end{bmatrix}$ and its pseudo-matrix is $\mathbf{M}^+ = [0.8333 \quad -0.8333]$. The configuration space is symmetric to the origin. Thus, a constraint on the center of mass can be added, such as $(x_1 + x_2)/2 = 0$. Subsequently, the equilibrium point can be calculated using (28) as follows:

$$\begin{aligned} \mathbf{x}^* &= \begin{bmatrix} 1 & -1 \\ -1 & 1 \\ 1/2 & 1/2 \end{bmatrix}^+ \\ &\cdot \begin{bmatrix} \left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \otimes [0.8333 \quad -0.8333] \right) \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} 0.4167 \\ -0.4167 \end{bmatrix} \end{aligned} \tag{29}$$

Hence, the roadmap at equilibrium is as shown in Figure 7.

Figure 8 shows the phase portrait of the optimization process for various initial conditions. As shown in the phase portrait, the states of $\mathbf{x}$ converge for all initial conditions to $\mathbf{x}^* = [0.4167 \quad -0.4167]^T$ or $\mathbf{x}^* = [-0.4167 \quad 0.4167]^T$, as calculated in (29). In case of convergence to $\mathbf{x}^* = [-0.4167 \quad 0.4167]^T$, the sensing vector is $\mathbf{w}^* = [0 \quad 1 \quad 1 \quad 0]^T$.

Figure 9 shows the phase portrait of example 1 for $r_\eta = 0.6$, where the neighbor radius is small for the volume of the configuration space. In this case, the equilibrium condition of the optimization process is the first case of Remark 7. The state of equilibrium is not a point but a region here, as shown by the two symmetrical triangles in Figure 9.

5. Regulation of Internal Repulsion

From the differential equation in (24), the types of equilibrium can be summarized into three cases. We also confirmed that the equilibrium of Case 3, such that

$$\begin{aligned} &\mathbf{L}^* \neq \mathbf{0}, \\ &\mathbf{x}^* \notin Null\left(\mathbf{L}^* \otimes \mathbf{I}_n\right), \\ &\mathbf{w}^* \neq \mathbf{0}, \end{aligned} \tag{30}$$

is the desirable solution to the suboptimal roadmap problem. To construct the appropriate roadmap, the Laplacian matrix must not be a zero matrix. To investigate this issue, the *internal repulsion* is defined as the quantitative information of the Laplacian matrix.

Definition 9 (internal repulsion). The *internal repulsion* P is the sum of the repulsion factors between neighbor nodes in the graph. It can be defined as the *entry-wise 1-norm of the Laplacian matrix* as follows:

$$P = \frac{1}{2}\|\mathbf{L}\|_1 = \frac{1}{2}\|\mathrm{vec}\,(\mathbf{L})\|_1 = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\left|L_{ij}\right|. \tag{31}$$

In the Laplacian matrix, the diagonal element is the sum of the row elements except itself, such that $|L_{ii}| = \sum_{i=1, i\neq j}^{N} |L_{ij}|$. Considering (23), the internal repulsion can be represented as follows:

$$\begin{aligned} P &= \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\left|L_{ij}\right| = \frac{1}{2}\sum_{i=1}^{N} 2 \sum_{j=1, i\neq j}^{N} \left|g\left(d_{ij}\right)\right| \\ &= \sum_{i=1}^{N}\sum_{j=1, i\neq j}^{N} g\left(d_{ij}\right). \end{aligned} \tag{32}$$

Equilibrium in Case 1 can occur when the neighbor radius is small in comparison with the volume of the free configuration space, such that $\mu(\mathbf{Q}_{free}) > N \cdot \mu(\mathbf{B}(\mathbf{q}_i, r_\eta))$. Thus, whether the Laplacian matrix is a zero matrix is intimately related to the volume of the free configuration space $\mu(\mathbf{Q}_{free})$ and neighbor radius r_η. Therefore, to determine the properties of convergence of the algorithm, we observe the steady-state value of internal repulsion with respect to various neighbor radii.

FIGURE 7: State of equilibrium for Example 8.

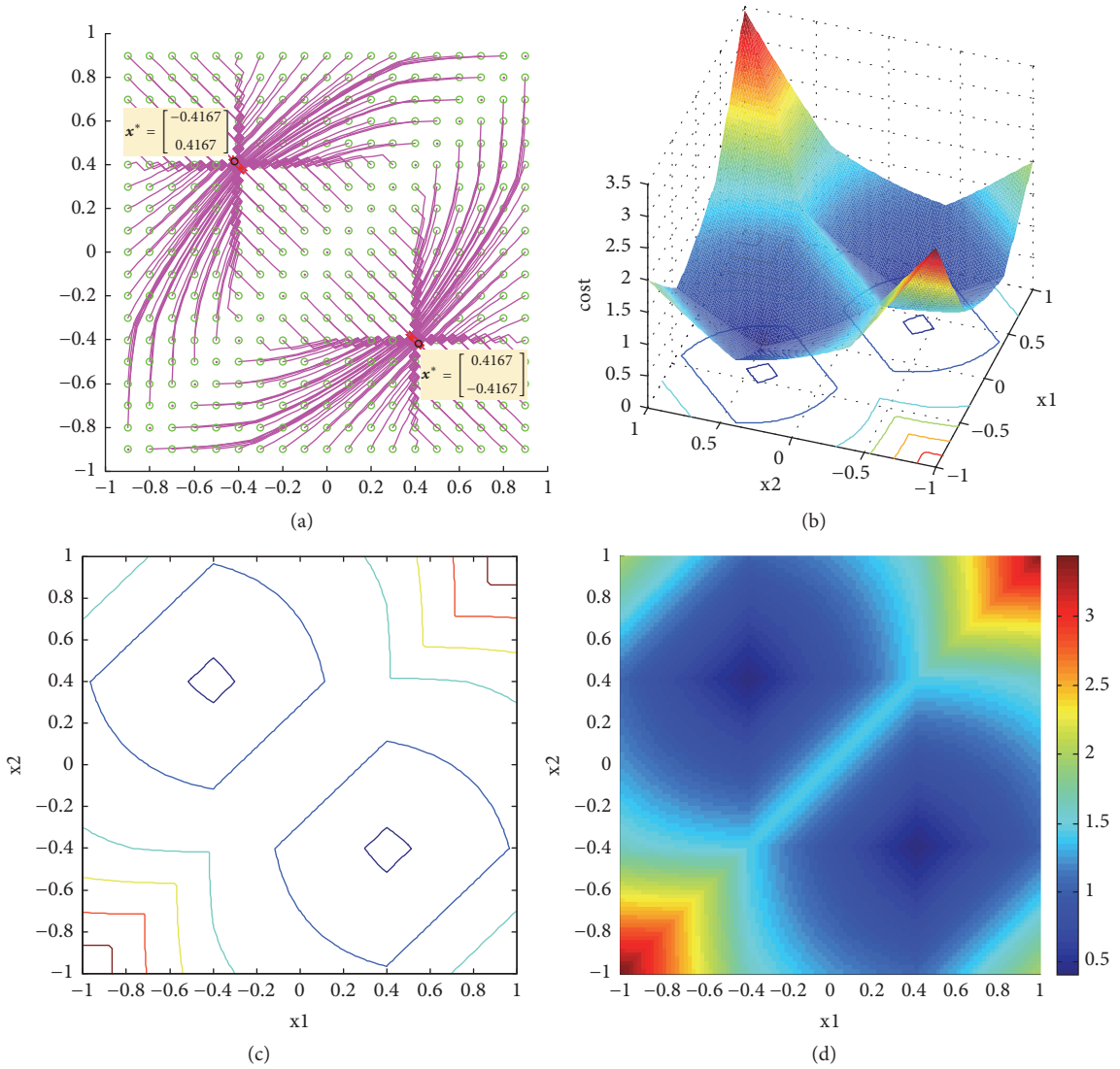


FIGURE 8: Phase portrait of the optimization process of Example 8. (a) State trajectories for each initial condition. (b) Cost graph of function $\boldsymbol{H}$. (c) Contour graph of the cost function. (d) Cost graph.

Figure 10 shows the maximum and minimum values of internal repulsion in the steady state for various neighbor radii. Figure 11 shows the structure of the roadmap in the steady state for some cases. In the internal repulsion graph, the maximum and minimum values are identical when the neighbor radius is smaller than 57. In this case, the internal repulsion and the nodes remain in a certain state, which means that the structure and the state of the roadmap converge to equilibrium. However, when the internal repulsion converges to zero, in the case of a neighbor radius smaller than 40, this means that the roadmap does not encompass the entire free configuration space, as shown in Figures 11(a) and 11(b). When the neighbor radius is 47, as shown in Figure 11(c), the structure of the roadmap encompasses the entire free configuration space well because the state of the roadmap converges with the appropriate internal repulsion, such as in the equilibrium of Case 3. When the neighbor radius is larger than 57, the maximum and minimum values are different. This means that the state of the nodes and the internal repulsion oscillate without converging. In this

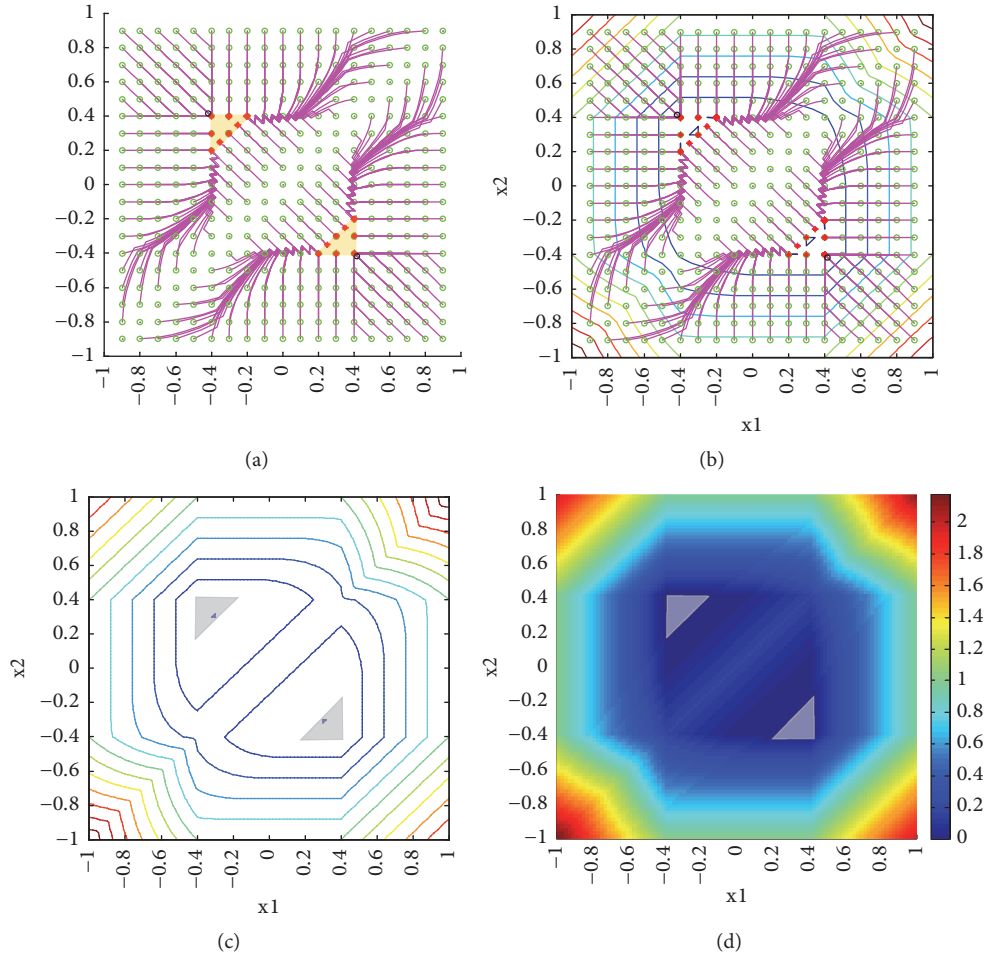


FIGURE 9: Phase portrait of the optimization process of Example 8 for $r_\eta = 0.6$. (a) State trajectories for each initial condition. (b) State trajectories and contour graph. (c) Contour graph. (d) Cost graph of function $\boldsymbol{H}$.

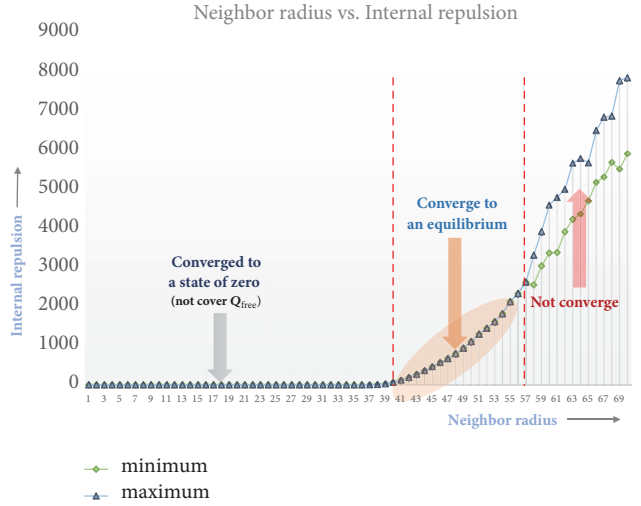
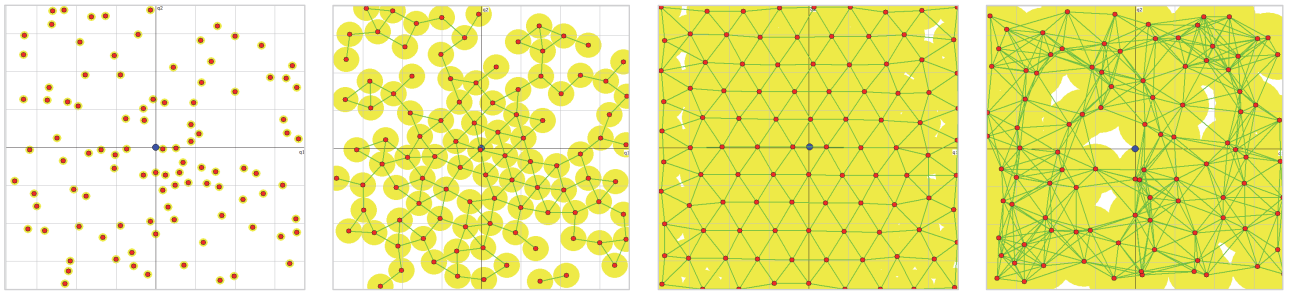


FIGURE 10: Graph of internal repulsion with respect to neighbor radius r_η.



(a) (b) (c) (d)

FIGURE 11: States of roadmap after 100 iterations. (a) $r_\eta = 10$ (no action); (b) $r_\eta = 30$ (ill-structured graph); (c) $r_\eta = 47$ (well-structured graph); (d) $r_\eta = 60$ (does not converge).

case, the algorithm cannot properly construct the roadmap because the state does not converge to equilibrium, as shown in Figure 11(d).

The deformation of the configuration space can induce changes in the volume of the free configuration space. When the volume of the free configuration space changes, the roadmap may no longer be constructed properly because the neighbor radius may be large or small relative to the changed volume of the free configuration space. Thus, the parameter of the neighbor radius should be adjusted with respect to the volume of the free configuration space. Generally, the volume of the free configuration space is difficult to calculate. However, the internal repulsion is highly related to it and is computable by executing the algorithm. With the regulation of internal repulsion by a desirable value, the roadmap can be constructed stably regardless of the volume of the free configuration space. To regulate internal repulsion, the controller that adjusts the neighbor radius should satisfy the following conditions.

Remark 10 (conditions for internal repulsion regulation). If the repulsion function and the regulation controller meet the following conditions for $\mathbf{L} \neq \mathbf{0}$, the *internal repulsion regulation* can be achieved. See [Section A.5] for proof.

(1) ***Repulsion function condition***:

$$\frac{\partial g\left(r_\eta, \rho_{ij}\right)}{\partial r_\eta} > 0 \quad \left(\exists \rho_{ij}, \rho_{ij} < r_\eta\right), \tag{33}$$

where r_η is added to the repulsion function as a variable because it is the control value in internal repulsion regulation, and ρ_{ij} is the expectation of neighbor distance, i.e., $\rho_{ij} = \mathbf{E}[d_{ij}]$.

(2) ***Regulation controller condition***:

$$r_\eta = G_{IRR} \int e_P dt + r_\eta\left(t_0\right), \quad \left(e_P = P_d - P_\rho\right), \tag{34}$$

where $G_{IRR} > 0$ is controller gain, P_d is the desired internal repulsion value, and P_ρ is the internal repulsion by the expectation of neighbor distance, i.e., $P_\rho = \sum_{i=1}^{N} \sum_{j=1, i\neq j}^{N} g(r_\eta, \mathbf{E}[d_{ij}])$.

In the suboptimal roadmap algorithm, the gradient of the repulsion function satisfies *Condition (1)* because it is calculated from (21) as follows:

$$\frac{\partial g}{\partial r_\eta} = \begin{cases} \dfrac{4}{d_{ij}} \dfrac{\sqrt{\pi/4}^{n-1}}{\Gamma\left(1 + (n-1)/2\right)} r_\eta \sqrt{r_\eta^2 - d_{ij}^2}^{n-3} & \left(0 < d_{ij} < r_\eta\right) \\ 0 & \text{otherwise.} \end{cases} \tag{35}$$

Thus, the internal repulsion can be regulated to the desired value using the controller of *Condition (2)*. Such a controller in (34) can be implemented as a discrete-time system as follows:

$$r_\eta[k] = G_{IRR} \cdot \sum_{l=k_0}^{k} \varepsilon \cdot \left(P_d - \sum_{i=1}^{N} \sum_{j=1, i\neq j}^{N} g\left(r_\eta[l], \overline{\rho}_{ij}[l]\right)\right) + r_\eta[k_0], \tag{36}$$

where ε is the sampling time or step size of the discrete-time system, and $\overline{\rho}_{ij}[k]$ is the estimation of ρ_{ij} that can be calculated as the average of d_{ij} for T_R intervals as $\overline{\rho}_{ij}[k] = (1/(T_R + 1)) \sum_{j=k-T_R}^{k} d_{ij}[j]$.

Figure 12 shows the efficacy of internal repulsion regulation in the case of a sudden change in free configuration space. At the initial condition, no obstacle exists and the free configuration space ratio $\mu(\mathbf{Q}_{free})/\mu(\mathbf{Q})$ is 100%. After the process reaches the state of equilibrium, some obstacles are added so that the volume of the free configuration space is reduced to 69.3%. The graphs of the internal repulsion for this situation are depicted in Figure 12 with respect to the time series. The green line represents the fixed neighbor radius and the blue line corresponds to the internal repulsion regulation. As shown in the graph, if no obstacles exist in the configuration space, both roadmaps converge to a similar structure. However, after the addition of obstacles into the configuration space, the roadmap with the fixed neighbor radius cannot converge and exhibits unstable oscillation, as depicted by the green line. However, the blue line graph shows

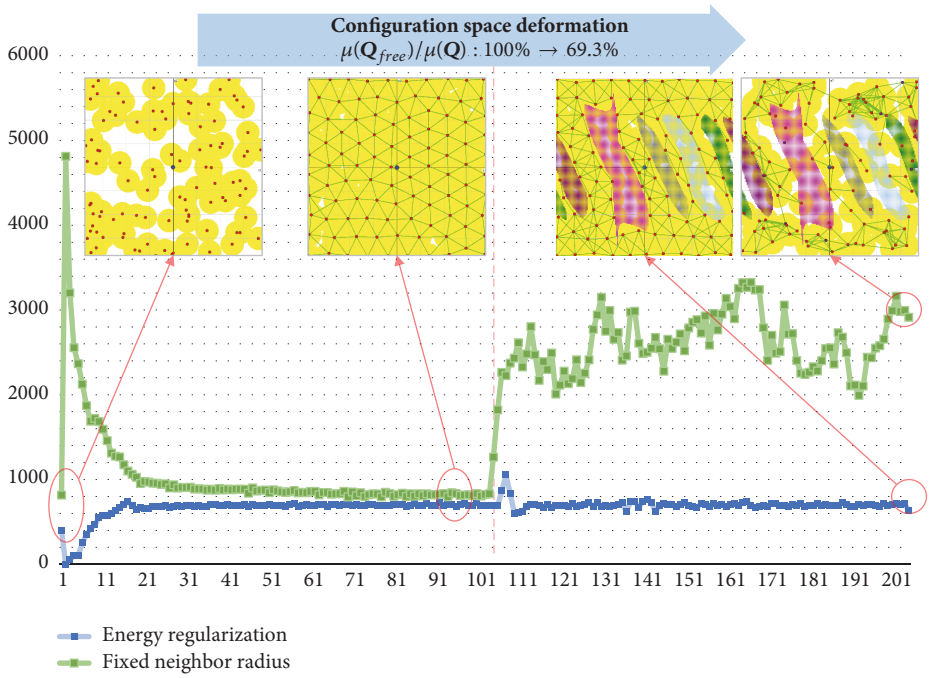


FIGURE 12: Graphs of internal repulsion for a change in the configuration space with a fixed neighbor radius ($r_\eta = 50$) and with internal repulsion regulation ($P_d = 700$).

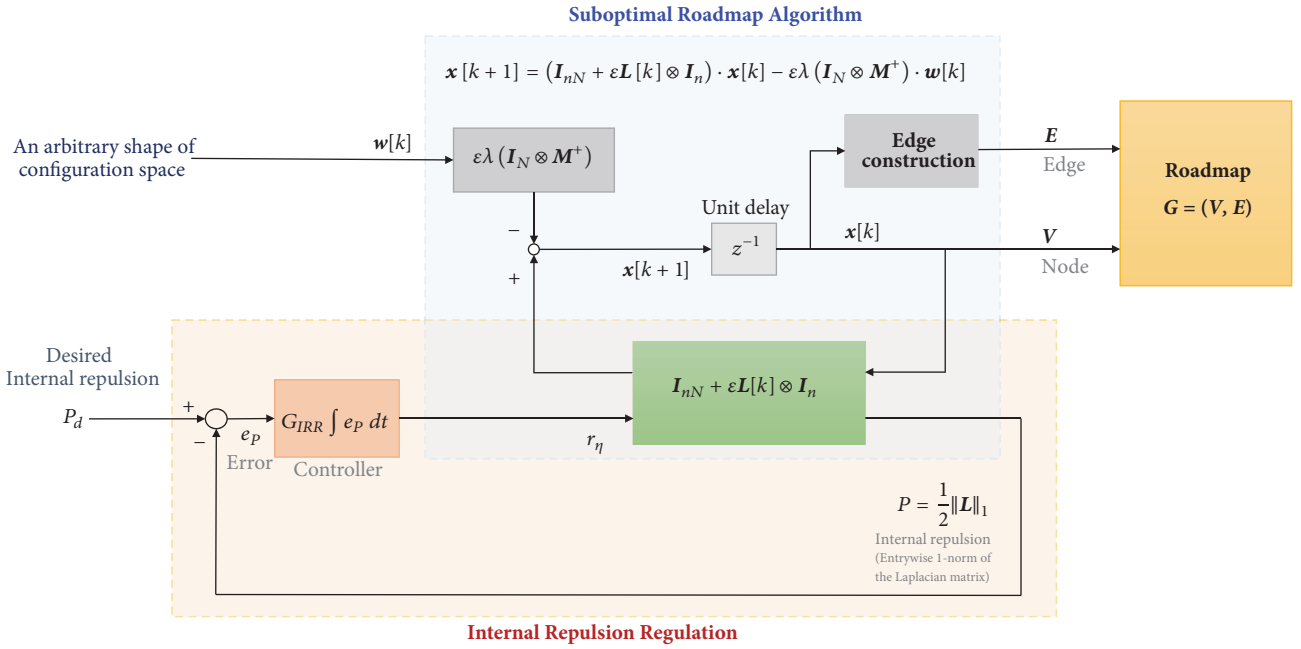


FIGURE 13: Overall structure of the suboptimal roadmap algorithm with internal repulsion regulation.

that the roadmap converges to equilibrium by regulating the internal repulsion uniformly regardless of the volume of the free configuration space.

Figure 13 shows an overall diagram of the optimization algorithm with internal repulsion regulation. This system features two types of differential equations. One is the differential equation used to maximize coverage that can construct an appropriate graph against deformations in the configuration space, and the other is that for internal repulsion regulation of the roadmap regardless of the volume of the free configuration space. With these differential equations, the suboptimal roadmap algorithm can construct the proper roadmap graph that covers arbitrary shapes of the free configuration space.

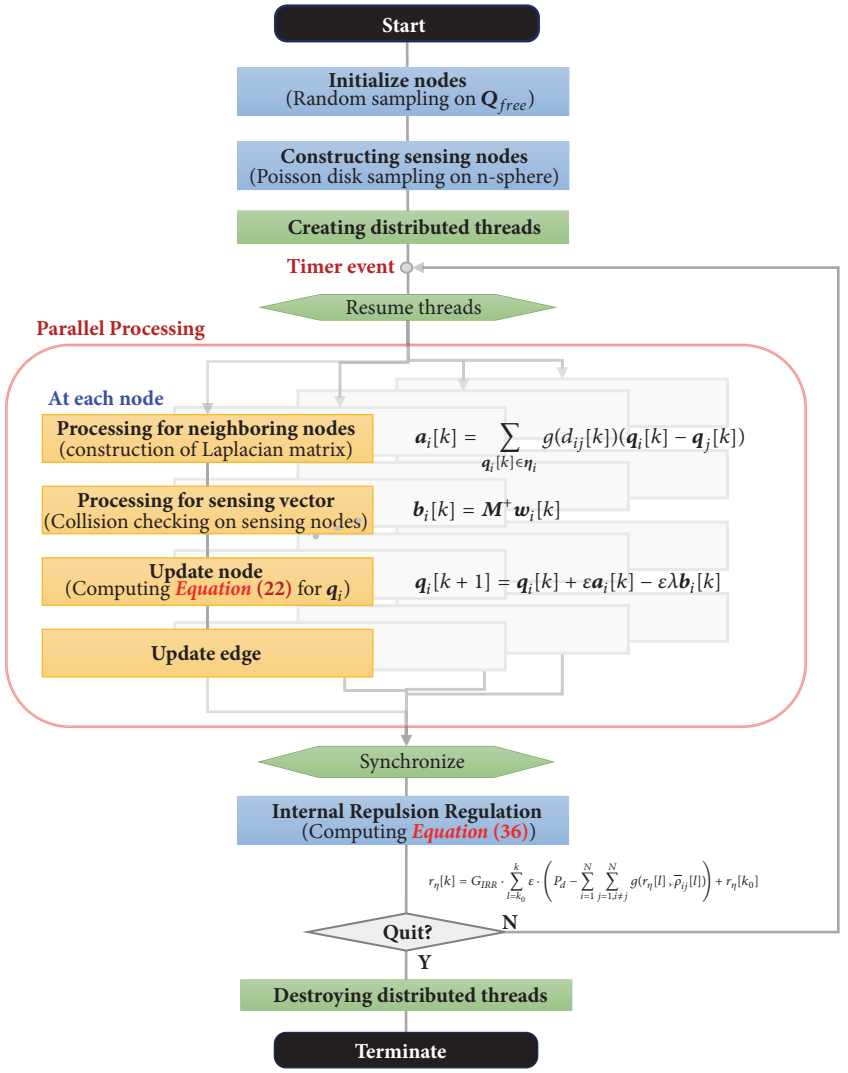


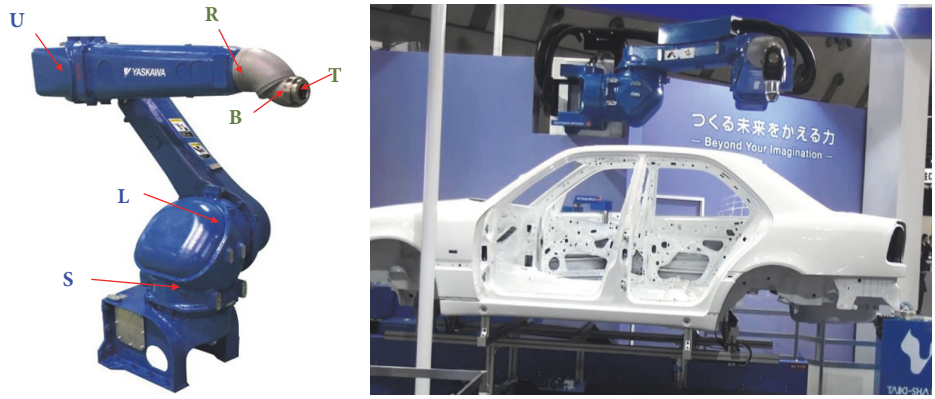
Figure 14: Flowchart for processing the suboptimal roadmap algorithm with internal repulsion regulation.

The suboptimal roadmap algorithm can be optimized by dividing its complex procedures using parallel computing techniques, such as multithreading or GPU processing. Thus, (22) is computed concurrently at each node. Figure 14 shows the flowchart for processing the suboptimal roadmap algorithm with internal repulsion regulation. The four procedures in the red border are performed in parallel at each node.

6. Case Study

To verify the effectiveness of the proposed method, we applied the suboptimal roadmap algorithm to a robot system in a car manufacturing assembly line. The target robot was the MPX3500, a painting robot system of the Yaskawa Electric Corporation (YEC) [24]. As shown in Figure 15, the MPX3500 robot had six axes; the S, L, and U axes rendered translation motions of the end effector. The other axes—R, B, and T—provided orientation motions. In conventional industrial robots, these axes are located around the end effector. In this application, only three axes—the S, L, and U axes—are considered as the orientation-related motions of the end effector do not affect motion related to collisions. Hence, the configuration space of the MPX3500 was a 3D space, and the constructed roadmap could be visualized with a geometrical representation.

The first step in applying the motion-planning algorithm is to create the collision-detection function $s(\mathbf{q})$. We implemented it using the *oriented bounding box* (OBB) method, which yields the best efficiency in terms of both accuracy and computation time [38]. The suboptimal roadmap algorithm was applied to plan the motion of the MPX3500 robot in the automobile manufacturing line. Figure 16 shows the simulation environment, where a model of a car and some obstacles were placed. To observe the performance of the constructed roadmap with respect to the number of nodes N, we varied N from 50 to 300 and set the step size and the Lagrange multiplier to $\varepsilon = 0.01$ and $\lambda = 25$, respectively. The number of sensing nodes and the radius of the sensing node were set to $n_s = 100$ and $r_s = 5$, respectively, and the desired internal repulsion was set to $P_d = 100$.



(a) (b)

Figure 15: YEC-MPX3500 painting robot system [24]. (a) MPX3500 robot. (b) Scene of its operation at an automobile painting line.

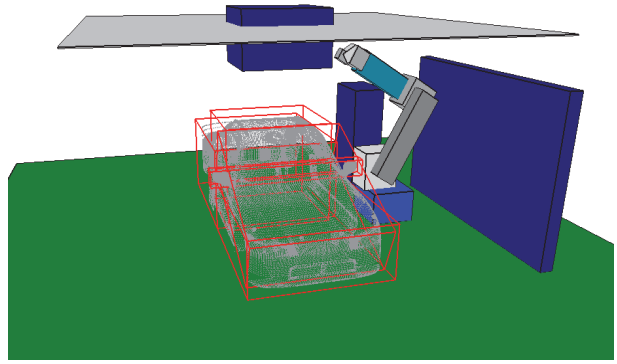


Figure 16: Workspace of an automobile manufacturing line. The car model and some obstacles are placed in the simulation environment.

Figure 17 shows the state of the roadmap for the iterations of the suboptimal roadmap algorithm for 150 nodes. Figure 17(a) displays the initial state of the roadmap, in which 150 nodes were scattered randomly in the configuration space. Figures 17(b) and 17(c) show the states of the roadmap after the first and the fifth iterations, respectively. These states are the initial steps of the execution of the algorithm. Figures 17(d), 17(e), and 17(f) show the states of the roadmap after the 25th, 50th, and 100th iterations, respectively. After 25 iterations, no considerable changes were observed in the state of the roadmap, which indicates that the iteration process might have arrived at a minimum state.

Figure 18 shows roadmaps of the suboptimal roadmap algorithm with respect to the number of nodes after 100 iterations. As shown in the figures, the appearances of the roadmap graphs are similar. However, the density of nodes is different with respect to the number of nodes. Although the quality of the results of planning depends on the density of nodes, near-optimal solutions were obtained even for a low density of nodes because the nodes were dispersed evenly.

Figures 19 and 20 show examples of the results of motion planning using the optimal roadmap, PRM∗, and RRT algorithms in the test environment. Figure 19(a) shows the initial motion and (b) shows the goal motion of the robot. (c) and (d) present the compared paths obtained by the three methods in the configuration space. Figure 20 shows the three trajectories of the robot's motion according to the planned path in Figure 19 in the workspace. As shown in the figures, the paths obtained by the suboptimal roadmap method were the shortest for the two workspaces; those obtained by the PRM∗ algorithm were the next shortest, and the RRT algorithm offered suboptimal results, such as paths involving detours, as confirmed in Figures 19(c) and 19(d).

Figure 21 shows comparison graphs of the results of motion planning using different algorithms—the suboptimal roadmap, PRM∗, and RRT—with respect to the number of nodes N. Using roadmaps with varying number of nodes, each calculation was performed 10 times to plan for 100 pairs of motions of arbitrary initial points and goals. For all roadmaps, the results of the suboptimal roadmap algorithm yielded the lowest value of accumulated length of the planned path. This indicates that the paths obtained using the proposed method were closest to the optimal results for the given motion-planning problem. Figure 22 shows the graph of the cumulative length of the paths obtained with the suboptimal roadmap and the PRM∗ algorithm with respect to the number of nodes. As shown in the figure, the graph of the suboptimal roadmap algorithm is always lower than that of the PRM∗. This indicates that better motion-planning results

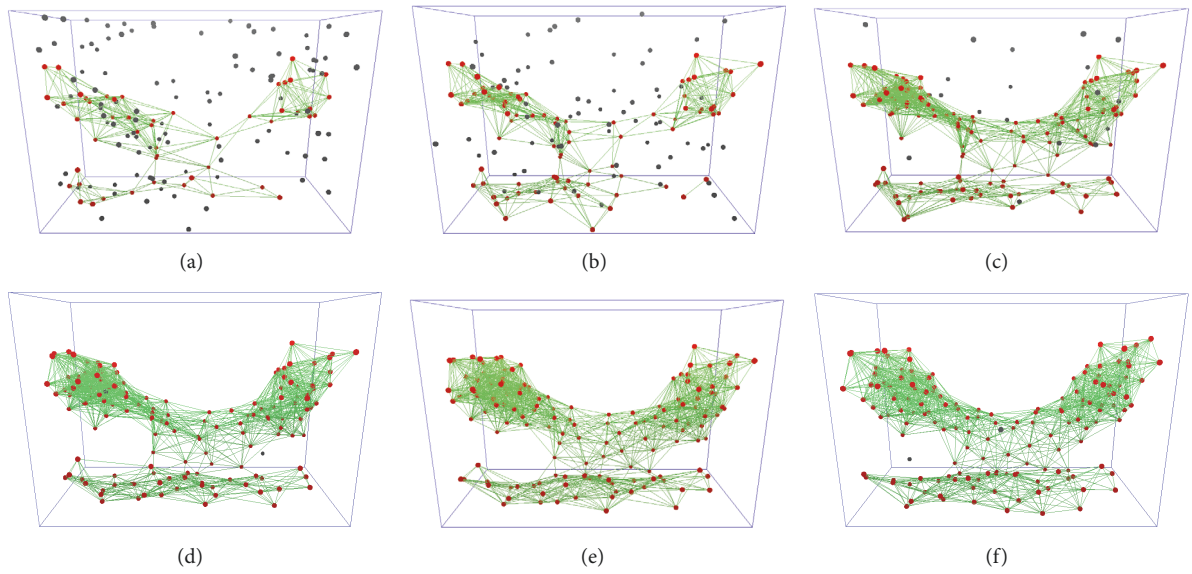


Figure 17: Operating procedures of the suboptimal roadmap algorithm in the configuration space (150 nodes). (a) Initial state. (b) 1st iteration. (c) 5th iteration. (d) 25th iteration. (e) 50th iteration. (f) 100th iteration.

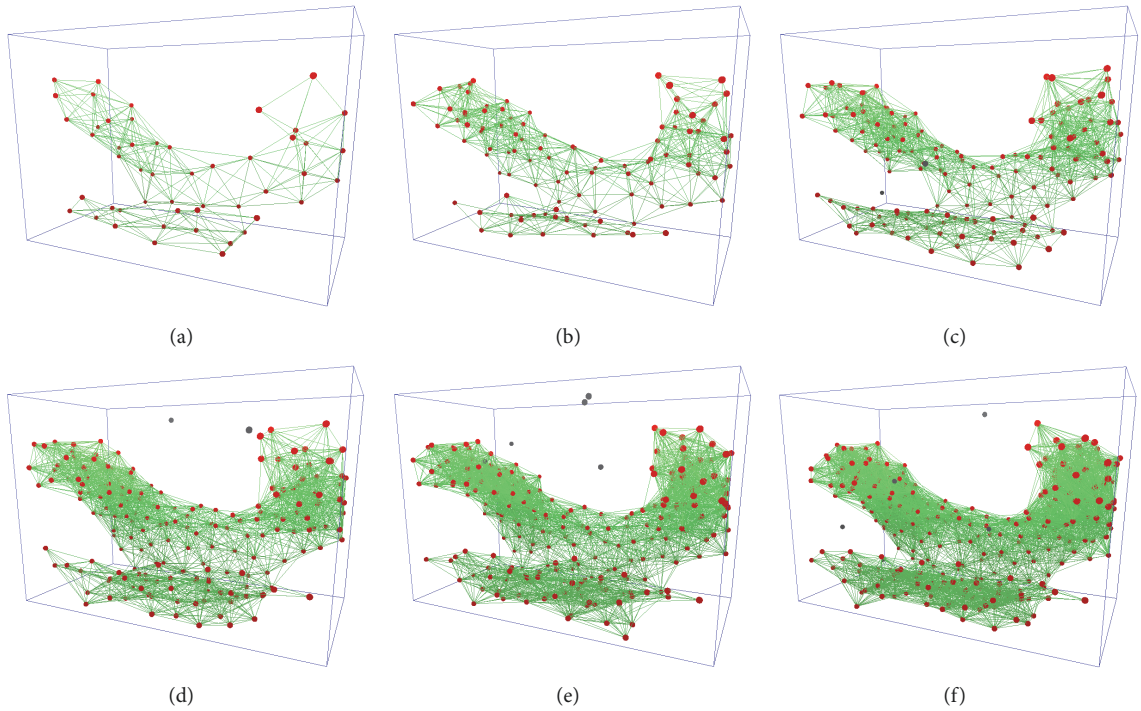


Figure 18: Roadmaps constructed by the suboptimal roadmap algorithm with respect to the number of nodes after 100 iterations. (a) $N = 50$; (b) $N = 100$; (c) $N = 150$; (d) $N = 200$; (e) $N = 250$; (f) $N = 300$.

can be obtained with the suboptimal roadmap algorithm with a small number of nodes. Moreover, the ratios of decrement of the suboptimal roadmap graphs were smaller than those of PRM$*$. This means that the motion-planning performance of the suboptimal roadmap algorithm was less sensitive to the number of nodes than current sampling-based methods.

7. Conclusion

This paper considered the problem of constructing a proper roadmap graph that fully encompasses the arbitrary morphologies of the free configuration space. To this end, the coverage of the graph was defined as a performance index on the optimality of a graph constructed by the sampling-based

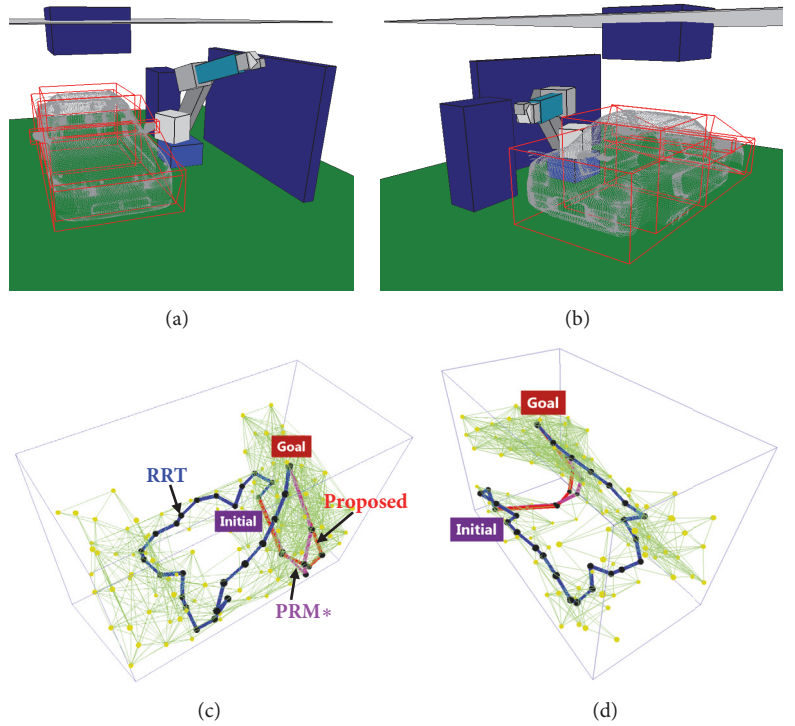


FIGURE 19: Comparison of the results of motion planning in the test environment. (a) Initial configuration. (b) Goal configuration. (c), (d) Motion paths obtained by the proposed method (red), PRM∗ (magenta), and RRT (blue) algorithms in the configuration space.

algorithm. We then proposed an optimization algorithm that can maximize graph coverage in the configuration space. Because of the computational complexity of coverage in practice, the proposed algorithm was designed using the concept of the suboptimal roadmap that minimizes the upper bound of the volume of intersection of the graph-covering region. Equilibrium conditions for the optimization algorithm were derived from the iteration equation. Among the results, we found the equilibrium condition that can construct an appropriate roadmap in an arbitrary configuration space. Furthermore, a method for regulating internal repulsion was proposed to utilize the suboptimal roadmap algorithm regardless of changes to the volume of the free configuration space. The convergence of the algorithm was found to be highly relevant to the volume of the free configuration space, accompanied by the number of nodes and the neighboring radius. Thus, to represent these relations quantitatively, internal repulsion was defined as the sum of repulsion factors between neighboring nodes in the graph. Further, we proposed an integrator-type controller that adjusts the neighboring radius to regulate internal repulsion according to the desired value. The feasibility of the proposed method was confirmed through a case study involving a robot. We applied the method to an industrial robot used in car manufacturing assembly lines. The results of the case study confirmed that the roadmap graph obtained by the proposed algorithm can produce more optimal results for paths of motion of the robot in the simulation environment.

Appendix

A.

A.1. Upper Bound of Intersection Volume $\boldsymbol{I}(\mathscr{B})$

Theorem A.1 (upper bound of intersection volume). *In a family of sets $\mathscr{B} = \{\mathbf{B}_1, \mathbf{B}_2, \cdots, \mathbf{B}_N\}$ composed of N sets, the upper bound of the intersection volume $I(\mathscr{B})$ can be represented by the multiplication of the constant κ and the sum of the intersection volume of two sets as follows:*

$$I(\mathscr{B}) \leq \kappa \cdot \left(\sum_{1 \leq i < j \leq N} \mu\left(\mathbf{B}_i \cap \mathbf{B}_j\right) \right), \tag{A.1}$$

where $\kappa = 2/N(N-1) \cdot \sum_{i=2}^{N} \left(\coprod_{j=2}^{i} \binom{N}{j} \right)$ is a finite constant related to number of sets N.

Proof. Let the sum of intersection volumes of i elements in $\mathscr{B}$ be

$$S_i = \sum_{1 \leq j_1 < j_2 \cdots < j_i \leq N} \mu\left(\mathbf{B}_{j_1} \cap \mathbf{B}_{j_2} \cap \cdots \cap \mathbf{B}_{j_i}\right); \tag{A.2}$$

subsequently, the intersection volume $I(\mathscr{B}) = \sum_{i=2}^{N} (-1)^i S_i$ meets the following condition:

$$I(\mathscr{B}) = \sum_{i=2}^{N} (-1)^i S_i \leq \sum_{i=2}^{N} S_i. \tag{A.3}$$

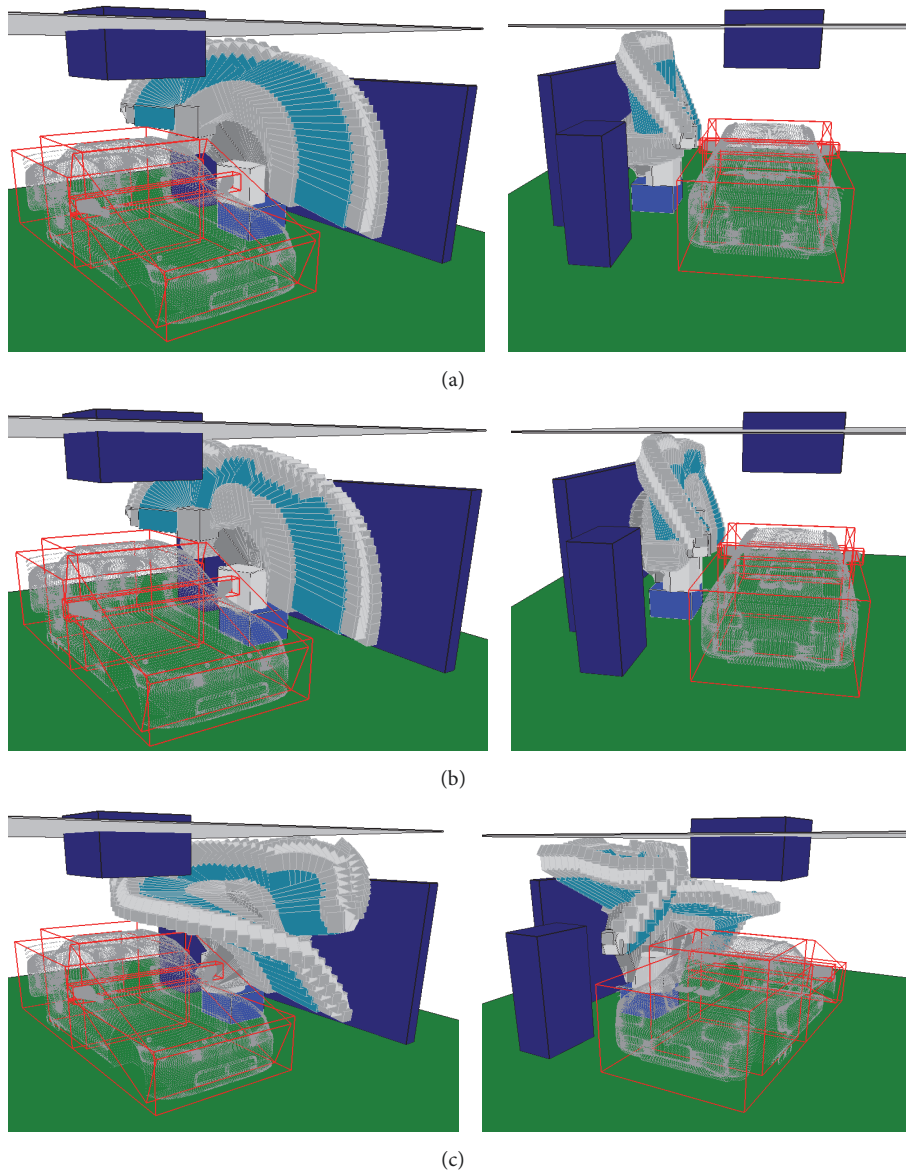


FIGURE 20: Trajectories of the robot's motion in the workspace with respect to the planning results of Figure 19. (a) Proposed method. (b) PRM∗. (c) RRT.

Because $S_{i+1} \leq \binom{N}{i+1} \cdot S_i$ by Lemma A.2, the following inequality is met for S_i:

$$S_i \leq \binom{N}{2}^{-1} \left(\coprod_{j=2}^{i} \binom{N}{j} \right) \cdot S_2, \quad (2 \leq i \leq N); \tag{A.4}$$

further, the following inequality is also satisfied for $\sum_{i=2}^{N} S_i$ ($\geq I(\mathscr{B})$):

$$\begin{aligned} \sum_{i=2}^{N} S_i &\leq \binom{N}{2}^{-1} \left(\coprod_{j=2}^{2} \binom{N}{j} \right) \cdot S_2 + \binom{N}{2}^{-1} \\ &\cdot \left(\coprod_{j=2}^{3} \binom{N}{j} \right) \cdot S_2 + \cdots + \binom{N}{2}^{-1} \left(\coprod_{j=2}^{N} \binom{N}{j} \right) \\ &\cdot S_2 = \binom{N}{2}^{-1} \\ &\cdot \left\{ \coprod_{j=2}^{2} \binom{N}{j} + \coprod_{j=2}^{3} \binom{N}{j} + \cdots + \coprod_{j=2}^{N} \binom{N}{j} \right\} \cdot S_2 \\ &= \frac{2}{N(N-1)} \\ &\cdot \sum_{i=2}^{N} \left(\coprod_{j=2}^{i} \binom{N}{j} \right) \cdot \left(\sum_{1 \leq i < j \leq N} \mu\left(\mathbf{B}_i \cap \mathbf{B}_j \right) \right). \end{aligned} \tag{A.5}$$

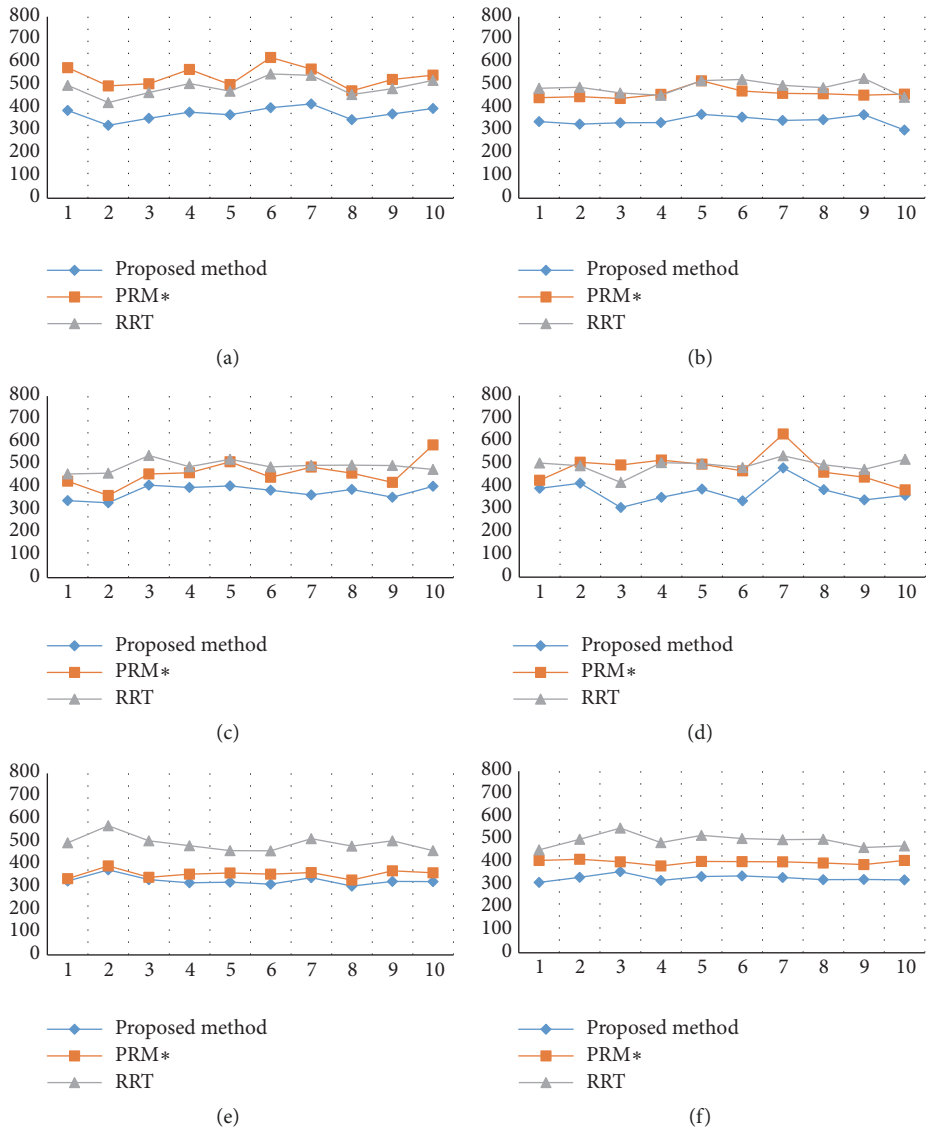


Figure 21: Results of comparison of path qualities obtained using the proposed method, PRM∗, and RRT algorithms with respect to the number of nodes. (a) $N = 50$; (b) $N = 100$; (c) $N = 150$; (d) $N = 200$; (e) $N = 250$; (f) $N = 300$.

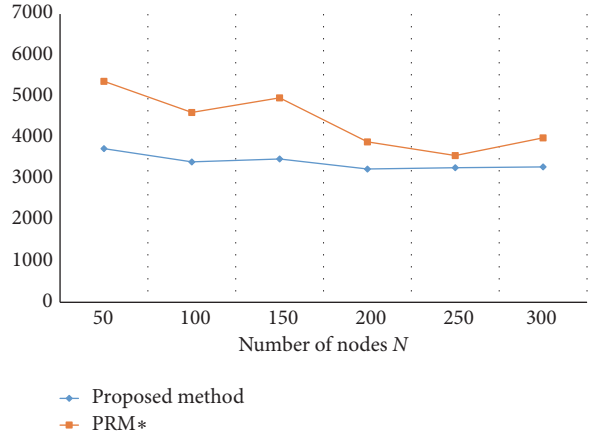


Figure 22: Results of evaluation of the total cumulative path of the proposed method and PRM∗ algorithm with respect to the number of nodes in the test environment.

Hence, the intersection volume $I(\mathscr{B})$ satisfies the inequality of (A.1). □

Lemma A.2. *S_{i+1} meets the following inequality for S_i:*

$$S_{i+1} \le \binom{N}{i+1} \cdot S_i. \tag{A.6}$$

Proof. Let the k-th intersection set in S_i be $\mathbf{C}_i(k) = \mathbf{B}_{j_1(k)} \cap \mathbf{B}_{j_2(k)} \cap \cdots \cap \mathbf{B}_{j_i(k)}$ and its volume be $\mu(\mathbf{C}_i(k))$; therefore, S_i can be expressed as follows:

$$\begin{aligned} S_i &= \sum_{1 \le j_1 < j_2 \cdots < j_i \le N} \mu\left(\mathbf{B}_{j_1} \cap \mathbf{B}_{j_2} \cap \cdots \cap \mathbf{B}_{j_i}\right) \\ &= \sum_{k=1}^{\binom{N}{i}} \mu\left(\mathbf{C}_i(k)\right). \end{aligned} \tag{A.7}$$

In S_{i+1}, as $\mathbf{C}_{i+1}(k) = \mathbf{C}_i(l_k) \cap \mathbf{B}_{j_{i+1}(k)}$ for $l_k \in \{1, \cdots, \binom{N}{i}\}$, $\mathbf{C}_{i+1}(k) \subseteq \mathbf{C}_i(l_k)$ can be represented as

$$\begin{aligned} &\mu\left(\mathbf{C}_{i+1}(k)\right) = \alpha_{i+1}(k) \cdot \mu\left(\mathbf{C}_i(l_k)\right) \\ &\qquad \text{for } 0 \le \alpha_{i+1}(k) \le 1. \end{aligned} \tag{A.8}$$

Thus, S_{i+1} can be expressed by the sum of $\binom{N}{i+1}$ terms of $\alpha_{i+1} \cdot \mu(\mathbf{C}_i)$ as follows:

$$S_{i+1} = \sum_{k=1}^{\binom{N}{i+1}} \mu\left(\mathbf{C}_{i+1}(k)\right) = \sum_{k=1}^{\binom{N}{i+1}} \alpha_{i+1}(k) \cdot \mu\left(\mathbf{C}_i(l_k)\right). \tag{A.9}$$

Let $\beta_{i+1}(l)$ be the sum of $\alpha_{i+1}(k)$ that is applied to the same $\mu(\mathbf{C}_i(l))$; therefore, S_{i+1} can be reformulated by the sum of $\binom{N}{i}$ terms of $\beta_{i+1} \cdot \mu(\mathbf{C}_i)$ as follows:

$$\begin{aligned} S_{i+1} &= \sum_{k=1}^{\binom{N}{i+1}} \alpha_{i+1}(k) \cdot \mu\left(\mathbf{C}_i(k)\right) \\ &= \sum_{l=1}^{\binom{N}{i}} \beta_{i+1}(l) \cdot \mu\left(\mathbf{C}_i(l)\right), \end{aligned} \tag{A.10}$$

where $\sum_{k=1}^{\binom{N}{i+1}} \alpha_{i+1}(k) = \sum_{k=1}^{\binom{N}{i}} \beta_{i+1}(k)$.

S_{i+1} can be represented by the inner product of two vectors $\boldsymbol{\beta}_{i+1} = [\beta_{i+1}(k)] \in \mathbf{R}^{\binom{N}{i}}$ and $\boldsymbol{\mu}_i = [\mu(\mathbf{C}_i(k))] \in \mathbf{R}^{\binom{N}{i}}$ as follows:

$$S_{i+1} = \sum_{l=1}^{\binom{N}{i}} \beta_{i+1}(l) \cdot \mu\left(\mathbf{C}_i(l)\right) = \boldsymbol{\beta}_{i+1}^T \cdot \boldsymbol{\mu}_i. \tag{A.11}$$

Because $\beta_{i+1}(k), \mu(\mathbf{C}_i(k)) \ge 0$, $S_{i+1} = \boldsymbol{\beta}_{i+1}^T \cdot \boldsymbol{\mu}_i \le \|\boldsymbol{\beta}_{i+1}\|_1 \cdot \|\boldsymbol{\mu}_i\|_1$ is satisfied by the Cauchy–Schwarz inequality. $\|\boldsymbol{\mu}_i\|_1 = \sum_{k=1}^{\binom{N}{i}} \mu(\mathbf{C}_i(k)) = S_i$ and

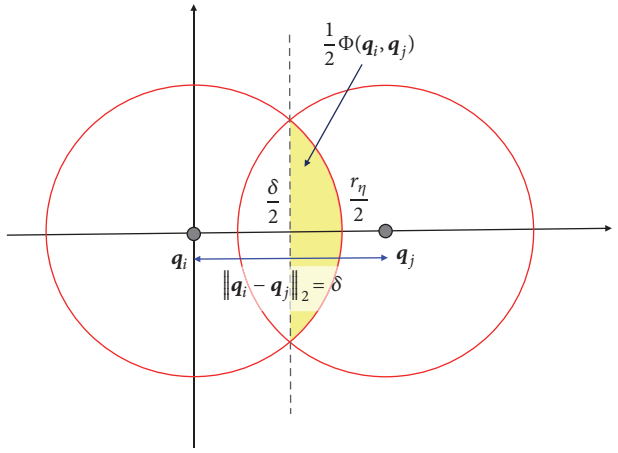


Figure 23: Two spheres and their region of intersection.

$$\|\boldsymbol{\beta}_{i+1}\|_1 = \sum_{k=1}^{\binom{N}{i}} \beta_{i+1}(k) = \sum_{k=1}^{\binom{N}{i+1}} \alpha_{i+1}(k) \le \binom{N}{i+1}, \quad (\because 0 \le \alpha_{i+1}(k) \le 1). \tag{A.12}$$

Hence, the inequality in (A.6) is satisfied as follows:

$$S_{i+1} = \boldsymbol{\beta}_{i+1}^T \cdot \boldsymbol{\mu}_i \le \|\boldsymbol{\beta}_{i+1}\|_1 \cdot \|\boldsymbol{\mu}_i\|_1 \le \binom{N}{i+1} \cdot S_i. \tag{A.13}$$

□

A.2. Volume of Region of Intersection between n-Dimensional Hyperspheres [39]. The volume of the n-dimensional hypersphere can be calculated by integrating the volume of an n–1-dimensional hypersphere along a certain axis. As shown in Figure 23 , the region of intersection between spheres can be calculated by integrating the volume of the n–1-dimensional sphere from $d_{ij}/2$ ($d_{ij} = \|\mathbf{q}_i - \mathbf{q}_j\|_2$) to $r_\eta/2$ along a certain axis. The volume of the n–1-dimensional hypersphere of radius r is

$$V_{n-1}(r) = \frac{\sqrt{\pi}^{n-1}}{\Gamma\left(1 + (n-1)/2\right)} r^{n-1}, \tag{A.14}$$

where $\Gamma(x)$ is the gamma function [40].

To obtain the volume of the n-dimensional hypersphere of radius $r_\eta/2$, the n–1 hypersphere is integrated for radius $r = \sqrt{(r_\eta/2)^2 - t^2}$ on $-r_\eta/2 \le t \le r_\eta/2$ [39]. Thus, the integral of the yellow region in Figure 23 can be calculated by integrating the volume of the n–1 sphere as follows:

$$\begin{aligned} &\int V_{n-1}(r)\, dr \\ &= \int_{d_{ij}/2}^{r_\eta/2} \frac{\sqrt{\pi}^{n-1}}{\Gamma\left(1 + (n-1)/2\right)} \sqrt{\left(\frac{r_\eta}{2}\right)^2 - t^2}^{n-1} \, dt. \end{aligned} \tag{A.15}$$

In Figure 23 , the yellow area calculated by the integration is the half-region of the area of intersection; thus, the volume of intersection is twice the calculated volume. Further, $d_{ij} \geq 0$ and $\Phi(\mathbf{q}_i, \mathbf{q}_j) = 0$ for $d_{ij} \geq r_\eta$; thus, the volume of the region of intersection between the n-dimensional hyperspheres can be summarized as follows:

$$\Phi\left(\mathbf{q}_i, \mathbf{q}_j\right) = \begin{cases} 2 \cdot \int_{d_{ij}/2}^{r_\eta/2} \frac{\sqrt{\pi}^{n-1}}{\Gamma\left(1 + (n-1)/2\right)} \sqrt{\left(\frac{r_\eta}{2}\right)^2 - t^2}^{n-1} \, dt & \left(0 \leq d_{ij} < r_\eta\right) \\ 0 & \left(d_{ij} \geq r_\eta\right) \end{cases} \tag{A.16}$$

A.3. Gradient Estimation of the Collision-Detection Function. Function $s(\mathbf{q})$ checks for collisions in the workspace for configuration $\mathbf{q} \in \mathbf{Q}$. As $s(\mathbf{q})$ is a discontinuous function, its gradient cannot be calculated. The gradient of $s(\mathbf{q})$ can be estimated stochastically using the *response surface method* (RSM) [31–33].

In the RSM, for $\mathbf{q}_i \in \mathbf{Q}$, $\boldsymbol{\varphi} \in \mathbf{R}^n$, $s(\mathbf{q})$ is assumed to be a linear model in the neighborhood of $\mathbf{q}_i$ as follows:

$$\widehat{s}(\mathbf{q}) = \varphi_0 + \boldsymbol{\varphi}^T\left(\mathbf{q} - \mathbf{q}_i\right). \tag{A.17}$$

The gradient of the function above, which is an estimated gradient of $s(\mathbf{q})$ at $\mathbf{q} = \mathbf{q}_i$ is

$$\widehat{\nabla} s(\mathbf{q})\Big|_{\mathbf{q}=\mathbf{q}_i} = \nabla \widehat{s}\left(\mathbf{q}_i\right) = \boldsymbol{\varphi}. \tag{A.18}$$

In the RSM algorithm, $\nabla \widehat{s}(\mathbf{q}_i)$ can be estimated using p design points $\mathbf{q}_m$ $(m = 1, \cdots, p)$ around node $\mathbf{q}_i$; here, the n_S $(> n)$ sensing nodes of $\mathbf{q}_i$ correspond to the design points, i.e., $\mathbf{q}_m = \mathbf{q}_i + \Delta\mathbf{q}_m$ $(m = 1, \cdots, n_S)$. By stacking the $s(\mathbf{q})$ information of n_S design points, (A.17) can be represented in matrix form as follows:

$$\begin{aligned} \begin{bmatrix} s\left(\mathbf{q}_i + \Delta\mathbf{q}_1\right) \\ s\left(\mathbf{q}_i + \Delta\mathbf{q}_2\right) \\ \vdots \\ s\left(\mathbf{q}_i + \Delta\mathbf{q}_{n_s}\right) \end{bmatrix} &= \begin{bmatrix} \varphi_0 + \boldsymbol{\varphi}^T\left(\mathbf{q}_i + \Delta\mathbf{q}_1 - \mathbf{q}_i\right) \\ \varphi_0 + \boldsymbol{\varphi}^T\left(\mathbf{q}_i + \Delta\mathbf{q}_2 - \mathbf{q}_i\right) \\ \vdots \\ \varphi_0 + \boldsymbol{\varphi}^T\left(\mathbf{q}_i + \Delta\mathbf{q}_{n_s} - \mathbf{q}_i\right) \end{bmatrix} \\ &= \begin{bmatrix} 1 & \Delta\mathbf{q}_1^T \\ 1 & \Delta\mathbf{q}_2^T \\ \vdots & \vdots \\ 1 & \Delta\mathbf{q}_{n_s}^T \end{bmatrix} \begin{bmatrix} \varphi_0 \\ \boldsymbol{\varphi} \end{bmatrix}. \end{aligned} \tag{A.19}$$

Using the sensing vector $\mathbf{w}_i = [s(\mathbf{q}_i + \Delta\mathbf{q}_m)] \in \mathbf{R}^{n_s}$ at node $\mathbf{q}_i$, the parameter $\widehat{\boldsymbol{\varphi}}_I = \begin{bmatrix} \widehat{\varphi}_0 \\ \widehat{\boldsymbol{\varphi}} \end{bmatrix} \in \mathbf{R}^{n+1}$ is estimated by the least-squares algorithm as follows [32]:

$$\widehat{\boldsymbol{\varphi}}_I = \left(\mathbf{M}_I^T \mathbf{M}_I\right)^{-1} \mathbf{M}_I^T \mathbf{w}_i, \tag{A.20}$$

where $\mathbf{M} = [\Delta\mathbf{q}_m^T] \in \mathbf{R}^{n_s \times n}$ and $\mathbf{M}_I = \begin{bmatrix} \mathbf{1}_{n_S} & \mathbf{M} \end{bmatrix} \in \mathbf{R}^{n_S \times (n+1)}$.

The matrix $\mathbf{M}_I^T \mathbf{M}_I$ is

$$\begin{aligned} \mathbf{M}_I^T \mathbf{M}_I &= \begin{bmatrix} 1 & \cdots & 1 \\ & \mathbf{M}^T & \end{bmatrix} \begin{bmatrix} 1 & \\ \vdots & \mathbf{M} \\ 1 & \end{bmatrix} \\ &= \begin{bmatrix} \sum_{m=1}^{n_S} 1 & \sum_{m=1}^{n_S} \Delta\mathbf{q}_m^T \\ \sum_{m=1}^{n_S} \Delta\mathbf{q}_m & \mathbf{M}^T\mathbf{M} \end{bmatrix}; \end{aligned} \tag{A.21}$$

if $\sum_{m=1}^{n_s} \Delta\mathbf{q}_m = \mathbf{0}$, then $\mathbf{M}_I^T \mathbf{M}_I = \begin{bmatrix} n_S & \mathbf{0}^T \\ \mathbf{0} & \mathbf{M}^T\mathbf{M} \end{bmatrix}$; therefore, (A.21) becomes

$$\begin{aligned} \widehat{\boldsymbol{\varphi}}_I = \begin{bmatrix} \widehat{\varphi}_0 \\ \widehat{\boldsymbol{\varphi}} \end{bmatrix} &= \begin{bmatrix} n_S & \mathbf{0}^T \\ \mathbf{0} & \mathbf{M}^T\mathbf{M} \end{bmatrix}^{-1} \begin{bmatrix} 1 & \cdots & 1 \\ & \mathbf{M}^T & \end{bmatrix} \mathbf{w}_i \\ &= \begin{bmatrix} n_S^{-1} \cdot \sum_{m=1}^{n_S} s\left(\mathbf{q}_i + \Delta\mathbf{q}_m\right) \\ \left(\mathbf{M}^T\mathbf{M}\right)^{-1} \mathbf{M}^T \mathbf{w}_i \end{bmatrix}. \end{aligned} \tag{A.22}$$

Hence, the estimated gradient of $s(\mathbf{q})$ is

$$\widehat{\boldsymbol{\varphi}} = \left(\mathbf{M}^T\mathbf{M}\right)^{-1} \mathbf{M}^T \mathbf{w}_i. \tag{A.23}$$

A.4. Derivation of the Differential Equation for the Entire Optimization Variable $\mathbf{x} \in \mathbf{R}^{nN}$. Let the Lagrange multipliers λ_i be $\lambda_1 = \lambda_2 = \cdots = \lambda_N = \lambda$; then, the decentralized optimization algorithm can be formulated as follows:

$$\begin{aligned} \mathbf{q}_i[k+1] = \mathbf{q}_i[k] &+ \varepsilon \sum_{\mathbf{q}_j[k] \in \boldsymbol{\eta}_i} g\left(d_{ij}[k]\right)\left(\mathbf{q}_i[k] - \mathbf{q}_j[k]\right) \\ &- \varepsilon\lambda \mathbf{M}^+ \mathbf{w}_i[k], \end{aligned} \tag{A.24}$$

where d_{ij} is the distance between nodes $\mathbf{q}_i$, $\mathbf{q}_j$; $\boldsymbol{\eta}_i$ is the set of neighbors of $\mathbf{q}_i$; and $g(d_{ij})$ is a function defined in (21).

Because $g(d_{ij}) = 0$ for $\mathbf{q}_j \notin \boldsymbol{\eta}_i$, (A.24) becomes

$$\begin{aligned}\mathbf{q}_i[k+1] &= \mathbf{q}_i[k] \\ &+ \varepsilon \sum_{j=1,i\neq j}^{N} g\left(d_{ij}[k]\right)\left(\mathbf{q}_i[k] - \mathbf{q}_j[k]\right) \\ &- \varepsilon\lambda \mathbf{M}^{+}\mathbf{w}_i[k] = \mathbf{q}_i[k] \\ &+ \varepsilon\left(\left(\sum_{j=1,i\neq j}^{N} g\left(d_{ij}[k]\right)\right)\mathbf{q}_i[k]\right. \\ &\left. - \sum_{j=1,i\neq j}^{N} g\left(d_{ij}[k]\right)\mathbf{q}_j[k]\right) - \varepsilon\lambda \mathbf{M}^{+}\mathbf{w}_i[k].\end{aligned} \tag{A.25}$$

Let the row vector $\mathbf{L}_i$ be

$$\mathbf{L}_i = \left[L_{ij}\right] \in \mathbf{R}^{1\times N}, \quad L_{ij} = \begin{cases} -g\left(d_{ij}\right) & i \neq j \\ \sum_{k=1,i\neq k}^{N} g\left(d_{ik}\right) & i = j, \end{cases} \tag{A.26}$$

(A.25) can be represented for $\mathbf{x} = [\mathbf{q}_i] \in \mathbf{R}^{nN}$ as follows:

$$\begin{aligned}\mathbf{q}_i[k+1] &= \mathbf{q}_i[k] + \varepsilon\left(\mathbf{L}_i[k] \otimes \mathbf{I}_n\right)\mathbf{x}[k] \\ &- \varepsilon\lambda \mathbf{M}^{+}\mathbf{w}_i[k],\end{aligned} \tag{A.27}$$

where $\otimes$ denotes the Kronecker product operator.

The equations of iteration performed at every node can be integrated such that

$$\begin{aligned}\begin{bmatrix}\mathbf{q}_1[k+1]\\ \mathbf{q}_2[k+1]\\ \vdots \\ \mathbf{q}_N[k+1]\end{bmatrix} &= \begin{bmatrix}\mathbf{q}_1[k]\\ \mathbf{q}_2[k]\\ \vdots \\ \mathbf{q}_N[k]\end{bmatrix} \\ &+ \varepsilon\left(\begin{bmatrix}\mathbf{L}_1[k]\\ \mathbf{L}_2[k]\\ \vdots \\ \mathbf{L}_N[k]\end{bmatrix} \otimes \mathbf{I}_n\right)\mathbf{x}[k] \\ &- \varepsilon\lambda\left(\mathbf{I}_N \otimes \mathbf{M}^{+}\right)\begin{bmatrix}\mathbf{w}_1[k]\\ \mathbf{w}_2[k]\\ \vdots \\ \mathbf{w}_N[k]\end{bmatrix}\end{aligned} \tag{A.28}$$

Let $\mathbf{w} = [\mathbf{w}_i] \in \mathbf{R}^{n_s N}$ be the stacked vector of N sensing vectors $\mathbf{w}_i$; then, using the Laplacian matrix $\mathbf{L} = [\mathbf{L}_i] \in \mathbf{R}^{N\times N}$, (A.28) can be reformulated for the entire variable $\mathbf{x} \in \mathbf{R}^{nN}$ as follows:

$$\begin{aligned}\mathbf{x}[k+1] &= \mathbf{x}[k] + \varepsilon\left(\mathbf{L}[k] \otimes \mathbf{I}_n\right)\mathbf{x}[k] \\ &- \varepsilon\lambda\left(\mathbf{I}_N \otimes \mathbf{M}^{+}\right)\mathbf{w}[k] \\ &= \left(I_{nN} + \varepsilon\mathbf{L}[k] \otimes \mathbf{I}_n\right)\mathbf{x}[k] \\ &- \varepsilon\lambda\left(\mathbf{I}_N \otimes \mathbf{M}^{+}\right)\mathbf{w}[k].\end{aligned} \tag{A.29}$$

A.5. Proof of Convergence For Internal Repulsion Regulation. This section proves Remark 10.

Proof. Because the neighbor radius r_η is added to the repulsion function as a variable, the internal repulsion is

$$P = \sum_{i=1}^{N}\sum_{j=1,i\neq j}^{N} g\left(r_\eta, d_{ij}\right). \tag{A.30}$$

The neighbor distance d_{ij} can be considered a random variable with expectation $\mathbf{E}[d_{ij}] = \rho_{ij}$; thus, we define P_ρ as the internal repulsion by expectation of neighbor distance as follows:

$$P_\rho = \sum_{i=1}^{N}\sum_{j=1,i\neq j}^{N} g\left(r_\eta, \mathbf{E}\left[d_{ij}\right]\right) = \sum_{i=1}^{N}\sum_{j=1,i\neq j}^{N} g\left(r_\eta, \rho_{ij}\right). \tag{A.31}$$

Let e_P be the tracking error of P_ρ for the desired value P_d:

$$e_P = P_d - P_\rho = P_d - \sum_{i=1}^{N}\sum_{j=1,i\neq j}^{N} g\left(r_\eta, \rho_{ij}\right). \tag{A.32}$$

Using the quadratic form of e_P, we define the Lyapunov candidate function Vas follows:

$$V = \frac{1}{2}\left(P_d - P_\rho\right)^2 = \frac{1}{2}{e_P}^2. \tag{A.33}$$

The derivative of V is

$$\begin{aligned}\frac{dV}{dt} &= \left(P_d - P_\rho\right)\cdot\left(-\frac{d}{dt}\left(\sum_{i=1}^{N}\sum_{j=1,i\neq j}^{N} g\left(r_\eta, \rho_{ij}\right)\right)\right) \\ &= -e_P \cdot \left(\sum_{i=1}^{N}\sum_{j=1,i\neq j}^{N}\frac{\partial g}{\partial r_\eta}\right)\cdot\frac{dr_\eta}{dt}.\end{aligned} \tag{A.34}$$

We set the derivative of r_η to $G_{IRR}e_P$ for a positive gain G_{IRR}, and the differentiation of the repulsion function to always be greater than zero for $0 < \rho_{ij} < r_\eta$:

$$\begin{aligned}\frac{dr_\eta}{dt} &= G_{IRR}e_P, \\ \frac{\partial g\left(r_\eta, \rho_{ij}\right)}{\partial r_\eta} &> 0 \quad \left(\exists \rho_{ij}, \rho_{ij} < r_\eta\right).\end{aligned} \tag{A.35}$$

Therefore, the derivative of V is always smaller than zero for $\mathbf{L} \neq \mathbf{0}$:

$$\frac{dV}{dt} = -G_{IRR} e_P^2 \cdot \left(\sum_{i=1}^{N} \sum_{j=1, i \neq j}^{N} \frac{\partial g}{\partial r_\eta} \right) < 0. \quad \text{(A.36)}$$

If the conditions of (A.35) are satisfied, V converges to zero. Thus, the tracking error for internal repulsion by the expectation of neighbor distance goes to zero:

$$e_P = P_d - P_\rho \longrightarrow 0 \quad \text{as } t \longrightarrow \infty. \quad \text{(A.37)}$$

□

Data Availability

The simulation software and data files used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors have no conflicts of interest to declare regarding the publication of this paper.

Acknowledgments

This research was financially supported by the Ministry of Trade, Industry, and Energy (MOTIE) and the Korea Evaluation Institute of Industrial Technology (KEIT) through the Industrial Strategic Technology Development Program (10080636).

References

[1] J. C. Latombe, *Robot Motion Planning*, Kluwer Academic Publishers, 1991.

[2] H. Cheset, K. M. Lynch, S. Hutchinson et al., *Principles of Robot Motion: Theory, Algorithms, and Implementations*, The MIT Press, Cambridge, UK, 2005.

[3] L. E. Kavaraki, P. Svestka, J. C. Latmobe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration space," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[4] J. Barraquand, L. Kavraki, J.-C. Latombe, R. Motwani, T.-Y. Li, and P. Raghavan, "A random sampling scheme for path planning," *International Journal of Robotics Research*, vol. 16, no. 6, pp. 759–774, 1997.

[5] J. J. Kuffner and S. M. LaValle, "RRT-Connect: An Efficient Approach to Single-Query Path Planning," in *Proceedings of the IEEE ICRA*, vol. 2, pp. 995–1001, 2000.

[6] D. Hsu, J.-C. Latombe, and H. Kurniawati, "On the probabilistic foundations of probabilistic roadmap planning," *International Journal of Robotics Research*, vol. 25, no. 7, pp. 627–643, 2006.

[7] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[8] J. D. Marble and K. E. Bekris, "Asymptotically near-optimal planning with probabilistic roadmap spanners," *IEEE Transactions on Robotics*, vol. 29, no. 2, pp. 432–444, 2013.

[9] D. Balkcom, A. Kannan, Y.-H. Lyu, W. Wang, and Y. Zhang, "Metric Cells: Towards Complete Search for Optimal Trajectories," in *Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2015*, pp. 4941–4948, Germany, October 2015.

[10] M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: a review," *IEEE Access*, vol. 2, pp. 56–77, 2014.

[11] Z. Sun, D. Hsu, T. Jiang, H. Kurniawati, and J. H. Reif, "Narrow passage sampling for probabilistic roadmap planning," *IEEE Transactions on Robotics*, vol. 21, no. 6, pp. 1105–1115, 2005.

[12] V. Boor, M. H. Overmars, and A. F. van der Stappen, "The Gaussian sampling strategy for probabilistic roadmap planners," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1018–1023, 1999.

[13] S. A. Wilmarth, N. M. Amato, and P. F. Stiller, "MAPRM: A probabilistic roadmap planner with sampling on the medial axis of the space," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1024–1031, 1999.

[14] T. Siméon, J.-P. Laumond, and C. Nissoux, "Visibility-based probabilistic roadmaps for motion planning," *Advanced Robotics*, vol. 14, no. 6, pp. 477–493, 2000.

[15] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1520–1533, 2004.

[16] J. Yan and H. Yu, "Distributed Optimization of Multiagent Systems in Directed Networks with Time-Varying Delay," *Journal of Control Science and Engineering*, vol. 2017, Article ID 7937916, 9 pages, 2017.

[17] S. Weerakkody, X. Liu, S. H. Son, and B. Sinopoli, "A graph-theoretic characterization of perfect attackability for secure design of distributed control systems," *IEEE Transactions on Control of Network Systems*, vol. 4, no. 1, pp. 60–70, 2017.

[18] R. Zhang and Q. Zhu, "Secure and resilient distributed machine learning under adversarial environments," in *Proceedings of the 18th International Conference on Information Fusion*, pp. 644–651, 2015.

[19] S. Sundaram and B. Gharesifard, "Distributed Optimization Under Adversarial Nodes," *IEEE Transactions on Automatic Control*, 2018.

[20] J. Cortés and F. Bullo, "Coordination and geometric optimization via distributed dynamical systems," *SIAM Journal on Control and Optimization*, vol. 44, no. 5, pp. 1543–1574, 2005.

[21] A. Kwok and S. Martinez, "Unicycle coverage control via hybrid modeling," *Institute of Electrical and Electronics Engineers Transactions on Automatic Control*, vol. 55, no. 2, pp. 528–532, 2010.

[22] H. Mahboubi and A. G. Aghdam, "Distributed deployment algorithms for coverage improvement in a network of wireless mobile sensors: relocation by virtual force," *IEEE Transactions on Control of Network Systems*, vol. 4, no. 4, pp. 736–748, 2017.

[23] J.-H. Park, J.-H. Bae, and M.-H. Baeg, "Adaptation algorithm of geometric graphs for robot motion planning in dynamic environments," *Mathematical Problems in Engineering*, vol. 2016, Article ID 3973467, 19 pages, 2016.

[24] https://www.yaskawa.eu.com/en/products/robotic/motoman-robots/productdetail/product/mpx3500/.

[25] K. Ferland, *Discrete Mathematics*, CENGAGE Learning, Belmont, NY, USA, 2008.

[26] T. Tao, *An Introduction to Measure Theory*, American Mathematical Society, 2011.

[27] D. P. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods*, Athena Scientific, Massachusetts, MASS, USA, 1996.

[28] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*, Athena Scientific, Massachusetts, MASS, USA, 2014.

[29] C. Langbort, L. Xiao, R. D'Andrea, and S. Boyd, "A decomposition approach to distributed analysis of networked systems," in *Proceedings of the 2004 43rd IEEE Conference on Decision and Control (CDC)*, pp. 3980–3985, Bahamas, Caribbean, December 2004.

[30] J. N. Tsitsiklis, D. P. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *IEEE Transactions on Automatic Control*, vol. 31, no. 9, pp. 803–812, 1986.

[31] P. Sadegh, "Constrained optimization via stochastic approximation with a simultaneous perturbation gradient approximation," *Automatica*, vol. 33, no. 5, pp. 889–892, 1997.

[32] K. Marti, *Stochastic Optimization Methods*, Springer, Berlin Heidelberg, 3rd edition, 2008.

[33] R. H. Myers, D. C. Montgomery, and C. M. Anderson-Cook, *Response Surface Methodology: Process and Product in Optimization Using Designed Experiments*, John Wiley & Sons, 3rd edition, 2009.

[34] M. S. Ebeida, S. A. Mitchell, A. Patney, A. A. Davidson, and J. D. Owens, "A simple algorithm for maximal poisson-disk sampling in high dimensions," *Computer Graphics Forum*, vol. 31, no. 2, pp. 785–794, 2012.

[35] E. A. Jackson, *Equilibrium Statistical Mechanics*, Prentice-Hall, New Jersey, NJ, USA, 1968.

[36] L. Kocarev, *Consensus and Synchronization in Complex Networks*, Springer, Berlin Heidelberg, 2013.

[37] A. Barrat, M. Barthélemy, and A. Vespignani, *Dynamical Processes on Complex Networks*, Cambridge University Press, Cambridge, UK, 2008.

[38] C. Ericson, *Real-Time Collision Detection*, CRC Press, New York, NY, USA, 2004.

[39] W. F. Basener, *Topology and Its Applications*, John Wiley & Sons, Hoboken, NJ, USA, 2006.

[40] X. Wang, "Volumes of Generalized Unit Balls," *Mathematics Magazine*, vol. 78, no. 5, pp. 390–395, 2005.