Path optimization with limited sensing ability

Sung Ha Kang[a,1], Seong Jun Kim[a], Haomin Zhou[a,2]

[a]*School of Mathematics, Georgia Institute of Technology, Atlanta, GA 30332, USA*

Abstract

We propose a computational strategy to find the optimal path for a mobile sensor with limited coverage to traverse a cluttered region. The goal is to find the shortest feasible path to achieve the complete scan of the environment. We pose the problem in the level set framework, and first consider a related question of placing multiple stationary sensors to obtain the full surveillance of the environment. By connecting the stationary locations using the nearest neighbor strategy, we form the initial guess for the path planning problem of the mobile sensor. Then the path is optimized by reducing its length, via solving a system of ordinary differential equations (ODEs), while maintaining the complete scan of the environment. Furthermore, we use intermittent diffusion, which converts the ODEs into stochastic differential equations (SDEs), to find the global optimal solution. To improve the computation efficiency, we introduce two techniques, one to remove redundant connecting points to reduce the dimension of the system, and the other to break the entangled path so the solution can escape the local traps. Numerical examples are shown to illustrate the effectiveness of the proposed method.

1. Introduction

In this paper, we study the problem of planning a path for a mobile sensor with limited coverage to traverse an environment filled with obstacles. The goal is to achieve complete coverage of the region with the shortest possible travel distance. This problem is motivated by a number of real world applications, such as using heartbeat sensor to detect hidden lives in rescue mission, searching or patrolling a confined region with complicated terrain by mobile radar or sonar devices [19, 41]. Sensors used in those applications have limited coverage range, meaning an object cannot be detected if its distance to the sensor is larger than $r > 0$, even if there may not be any obstacle blocking the object from the sensor. This problem is very similar to the well-known *watchman route* problem [6], which shares the same goal except observers in the watchman route problem have unlimited coverage range. The observer can see an object as long as it is not blocked by obstacles, regardless the distance between them.

Email addresses: kang@math.gatech.edu (Sung Ha Kang), skim396@math.gatech.edu (Seong Jun Kim), hmzhou@math.gatech.edu (Haomin Zhou)

[1]Kang is supported by Simons Foundation grant 282311.

[2]Zhou is partially supported by NSF Awards DMS–1042998, DMS–1419027, and ONR Award N000141310408.

The watchman route problem, and the more general path planning problems have been studied intensively in the past few decades, especially in the robotics literature. Many different methods have been developed depending on what types of obstacles are considered in the problems. For polygonal obstacles, path planning problems can be re-casted as optimization problems on graphs, therefore combinatoric and graph-based approaches, such as the Dijkstra's algorithm [10], Floyd's algorithm [12], and Johnson's algorithm [16], are well studied and broadly used. Additional algorithms and associated evaluations along this line of research may be found in [5, 9, 15, 20, 23, 25, 26, 38], including a survey [24] that provides many more references therein. It is worth noting that some path planning problems have been found to be NP-hard [3, 6, 28], such as the famous traveling sale person problem [33, 35], indicating that one often needs to face some extraordinary computation challenges in solving those problems.

However, the combinatoric strategy may not be directly applicable to problems with non-polygonal obstacles. To account for obstacles with piecewise smooth boundaries, many different strategies have been proposed, including the commonly used level set based methods [4, 21, 22, 39]. In this case, the environment is described by a level set function with positive values outside of the obstacles and negative values inside. The zero level curves are the boundaries of obstacles. Feasible traveling paths can only go through the regions with non-negative level set values.

In literature, the watchman route problem is often associated with the *art gallery* problem, another classical problem that aims at placing some stationary observers to maximize the surveillance area [1] in a cluttered region. Like the watchman route problem, the art gallery problem and its variations have drawn considerable attention in recent decades [2, 8, 11, 13, 14, 17, 34, 36]. Readers are referred to a book [29] and survey papers [37, 40] for more references.

Among various versions of watchman route and art gallery problems, the one studied in [28], called watchman route under limited visibility or d-sweeper problem, aims at the same goal of this paper, but with different set ups and strategies. In [28], the polygonal obstacles are studied, and the method is essentially graph based. While we consider obstacles with piecewise smooth boundaries and employ level set based computational method.

Despite of the existence of extensive literature on path planning, the problem that we are interested in remains challenging. The difficulties often come from the following three aspects: (1) The obstacles have arbitrary shapes, which prevent us from using more efficient graph based strategies. (2) Finding a feasible path with (even locally) shortest length is often costly because it is an infinite dimensional problem. Finding the global optimal path, if possible, is usually computational intractable. (3) Achieving the complete coverage of the environment is a constraint that is hard to be satisfied in general, and it counter-plays with the shortest length requirement. This constraint becomes harder to enforce for sensors having limited coverage range, which makes the problem more complex.

Unlike the existing methods, in this paper, we tackle the problem using the level set formulation together with the intermittent diffusion, a stochastic differential equation (SDE) based global optimization strategy [42]. The level set framework gives us the freedom to consider various shapes of obstacles and a way to incorporate the finite coverage range into the definition of the coverage level set function. The total coverage area of a certain path can be easily calculated by integrating each sensor's coverage level set function along the

path. The intermittent diffusion is then used to find the optima, the shortest path, in the set of all feasible paths which gives the complete coverage of the environment.

To find the the shortest path, our proposed method consists of three main steps: the initialization, the gradient flow to find local optimal solutions, and the intermittent diffusion to find the global solution. For the initialization, we first solve the art gallery problem with limited sensor coverage, i.e., we apply a greedy strategy to find the locations of stationary sensors for the full surveillance. Then we connect the locations, called connecting points, to their nearest neighbors by the shortest path algorithm presented in [7]. In this way, we obtain a feasible path that guarantees the complete coverage. The gradient flow is deduced by shrinking the length of a path on the complete coverage manifold, which is defined as the collection of all feasible paths attaining complete coverage of the environment. Algorithmically, the gradient flow moves the connecting points by using a system of ordinary differential equations (ODEs). A locally shortest path is obtained when the gradient flow reaches its steady state. However, the complete coverage manifold is highly non-convex, there are many local optimal solutions. To find the global optimal path, we employ the intermittent diffusion method, by adding random perturbations to help the path moving out of local traps and have a chance to find the global optimal solution. Furthermore, we introduce two techniques in the algorithm: redundant point removal and a disentanglement strategy. These simple ideas reduce the dimension of the ODEs or SDEs, and improve the computation efficiency in our numerical experiments.

The layout of the paper is as follows. In Section 2, the new level set formulation of limited coverage range is introduced. We use it to solve the associated art gallery problem and find the optimal locations for the stationary sensors. The resulting locations are then used for the initialization of the path optimization. In Section 3, we propose the main algorithm to optimize the path length while achieving complete coverage of the environment. Section 4 provides several numerical results, and the new techniques of disentanglement and redundant points removal. Finally, we give a conclusion in Section 5.

2. Optimal locations for sensors with limited coverage ranges

In this section, we start by introducing the level set formulation for the coverage function of sensors with limited coverage range. Then we propose a strategy to place multiple stationary sensors to achieve the full surveillance of the environment. Finally, we connect the found locations to their nearest neighbors, which forms the initial path attaining the complete coverage of the environment.

2.1. New level set formulation with limited coverage

In the level set framework, the environment is described by a level set function with positive values outside of the obstacles, negative values inside, and the zero level curves representing boundaries of obstacles. This level set setting is exploited to solve visibility related problems in [4, 39] where light rays from a vantage point (e.g., observer or sensor) travel in straight lines and are obliterated upon contact with the surface of an obstacle. A point is covered by the observer (or sensor) placed at a given vantage point, if the line segment between the point and the vantage point does not intersect any of the obstructions. Under this point of view, in prior studies including [21, 22], it is assumed that the observer

has an infinite coverage range. Here, we extend this formulation to account for sensors with limited coverage range.

To set up the problem, we denote computational domain as D, a compact subset of $\mathbb{R}^2$. Let Ω be the collection of obstacles, i.e. a closed set containing finite number of connected components comprising one or multiple given obstacles in D. The level set function $\psi(x)$ represents the environment: inside the obstacles Ω is negative, and outside is positive. In our algorithms, we define $\psi(x)$ as the typical signed distance function from the boundaries of Ω. A sensor located at a point x in $D\backslash\Omega$, has the line-of-sights as straight line segments originated from x and ended at y with either $|x-y| = r$ and $y \in D\backslash\Omega$, or $|x-y| \leq r$ and y is a point on the boundary of the obstacles. The coverage region of the sensor is the union of all line-of-sights emanating from the sensor.

Now, we define the sensor coverage as a level set function ϕ given by

$$\phi(y; x, r) = \min_{z \in \mathcal{L}(x,y)} \{\psi(z), r - |z - x|\} \tag{2.1}$$

where $\mathcal{L}(x, y)$ is the line segment connecting x and y. This way the function $\phi(\cdot; x, r)$ represents the coverage in a bounded domain D by

$$\begin{cases} \phi(\cdot; x, r) > 0, & \text{covered regions,} \\ \phi(\cdot; x, r) < 0, & \text{non-covered regions.} \end{cases} \tag{2.2}$$

The following lemma shows that $\phi(y; x, r)$ is Lipschitz continuous in x.

Lemma 1. *$\phi(y; x, r)$ is Lipschitz continuous in x. i.e., there exists K such that*

$$|\phi(y; x, r) - \phi(y; \tilde{x}, r)| \leq K\,|x - \tilde{x}|$$

for $x, \tilde{x}, y \in D$.

Proof. We refer readers to [4] for the proof. □

With the coverage level set function $\phi(y; x, r)$ we can calculate the coverage area of the sensor at x by

$$V(x; r) = \int_D H(\phi(y; x, r))dy, \tag{2.3}$$

where H denotes the one-dimensional Heaviside function.

To compute the coverage function $\phi(y; x, r)$, we present an simple strategy in Algorithm 1, which mostly follows the PDE based strategy presented in [39]. The main idea is to use the method of characteristics for the nonlinear first order PDEs, and we add a modification to handle the finite coverage range. We use a normalized viewing vector $\vec{v}_x(y) = \frac{y-x}{|y-x|}$ which connects an initial point x with a terminal point y.
We solve (2.4) using the upwind finite differencing scheme. See [30, 39] for details. Note the computational workload in each iteration is dominated by the second and third steps. However, these steps are only needed for computing ϕ inside the sensor's coverage range.

Algorithm 1 PDE based algorithm for computing the sensor coverage ϕ

Input: level set function ψ, a sensor $x \in D\backslash\Omega$, and a radius of coverage r.

1. Set $\phi(y) = \psi(y)$ for $y \in D$.
2. If $y \in B_r(x)$, solve for ϕ:

$$\nabla_y \phi(y; x, r) \cdot \vec{v}_x(y) = 0, \tag{2.4}$$

subject to the boundary condition $\phi = \psi$. Otherwise, set $\phi(y; x, r) = 0$.

3. Update $\phi = \min(\phi, \psi, r - |y - x|)$.

2.2. Maximum coverage for a single stationary sensor

Based on the above setting, we consider a problem of maximizing the coverage of a singe sensor. More precisely, here the goal is to move the sensor to the optimal location that maximizes the coverage area V in (2.3). We use the gradient ascent method to achieve this goal, namely we follow the gradient flow to move the location x:

$$\partial_t x = \nabla_x V(x), \tag{2.5}$$

where ∇_x is the gradient operator with respect to x.

The gradient operator can be approximated by many finite difference schemes, for example, we use the central difference scheme in this paper, which leads to

$$\partial_t x = D_0^h V(x), \tag{2.6}$$

where h is the spatial step size. Algorithm 2 describes the steps to compute the gradient ascent flow using this central difference scheme in space with forward Euler scheme in time. In the algorithm, $\{e_1, e_2\}$ is a standard basis of $\mathbb{R}^2$.

Algorithm 2 Maximum coverage of a single stationary sensor

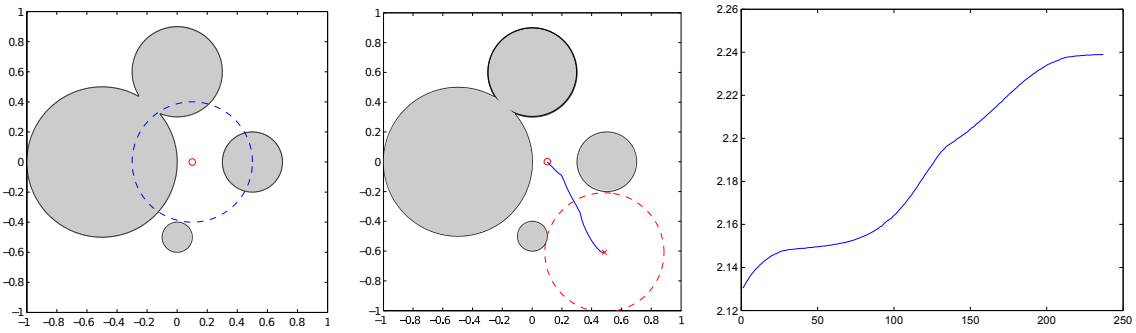
Input: single sensor x, radius of coverage r, spatial step size h, and temporal step size k.

Compute: For a step size h, obtain the 4 neighborhood values of the coverage level set functions $\phi(\cdot; x \pm he_i, r)$ over a grid in $D \cap B_{x \pm he_i}(r)$ for each $i = 1, 2$ using Algorithm 1 for ϕ.

Repeat until convergence

1. Evaluate $V(x \pm he_i; r)$ and use central differencing for $D_0^h V(x)$.
2. Use Euler's method to update x by $x + kD_0^h V(x)$.

In Figure 1 we demonstrate the trajectories of the gradient ascent flow generated by Algorithm 2. The computational domain is $D = [-1, 1] \times [-1, 1]$ and the obstacles are the shaded four disks of various radii. A sensor x_1 is initially located at $(0.1, 0)$, and the blue dotted circle represents the sensor coverage with a radius of coverage $r = 0.4$. Figure 1(b) shows the result of Algorithm 2, where the sensor moves away from the obstacle to maximize the coverage area. Figure 1(c) shows the corresponding coverage area with respect to the gradient ascent flow. Notice that in the final location, the red dotted circle describing the sensor coverage is tangent to two obstacles, and the sensor attains the maximum coverage. This is the globally optimal result. It is worth mentioning that for sensors with unlimited



(a) (b) (c)

Figure 1: A single sensor coverage optimization. The obstacles are the shaded four disks of various radii. The initial location of a sensor in (a) is moved from 'o' to 'x' in image (b) by following the blue gradient flow path in (b). The dotted circle represents the sensor coverage. (c) The value of coverage area with respect to the gradient ascent flow. This value is increasing until the maximum coverage is achieved.

coverage range [4], the sensor prefers to run away towards infinity and the algorithm stops the computation when it hits the boundary of D.

We also note that if there is no obstacle inside $B_r(x)$, then V already attains its maximum, so the gradient ascent stops immediately. This is unlike the results where the infinite coverage range is assumed as [4].

2.3. Maximum coverage for multiple stationary sensors

We generalize the previous problem to multiple sensors. Let $\{x_1, \cdots, x_m\}$ denote the location of m sensors and let $\{r_1, \cdots, r_m\}$ be the individual coverage range respectively, i.e., each sensor may have a different coverage range. The coverage of multiple sensors is the union of the coverages of all sensors. Similar to (2.1), we define the coverage level set function with respect to multiple sensors by

$$\phi(y; x_1, \cdots, x_m, r_1, \cdots, r_m) = \max_{i=1,\cdots,m} \phi(y; x_i, r_i). \tag{2.7}$$

In addition, the coverage area of multiple sensors is given by

$$V(x_1, \cdots, x_m) = \int_\Omega H(\phi(y; x_1, \cdots, x_m, r_1, \cdots, r_m))dx. \tag{2.8}$$

Then the solution $\{x_1, \cdots, x_m\}$ maximizing (2.8) will be the desired optimal locations for the sensors. Algorithm 3 presents a greedy strategy for the optimal locations. The sensors are ordered sequentially, and only one is moved, by a gradient flow given by (2.9), at a time. The new location of each sensor depends only on the locations of sensors placed before it.

Figure 2 shows an example where two sensors are placed very close to each other initially. The algorithm successfully found two new locations, where their coverages do not intersect each other, and the combined coverage area is maximized.

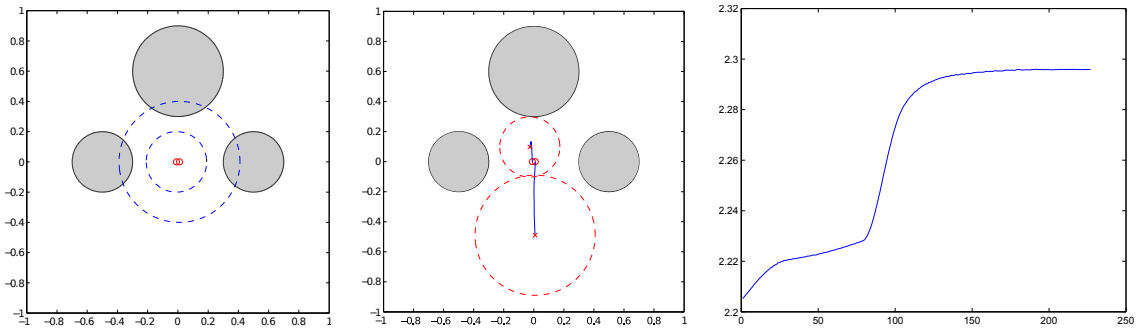
Algorithm 3 Maximum coverage for multiple stationary sensors

Input: multiple sensors $\{x_1, \cdots, x_m\}$, radii of coverage $\{r_1, \cdots, r_m\}$, spatial step size h, and temporal step size k.

Iterate: For each $i = 1, \cdots, m$, solve

$$\partial_t x_i = \nabla_{x_i} V(x_1, \cdots, x_m) \tag{2.9}$$

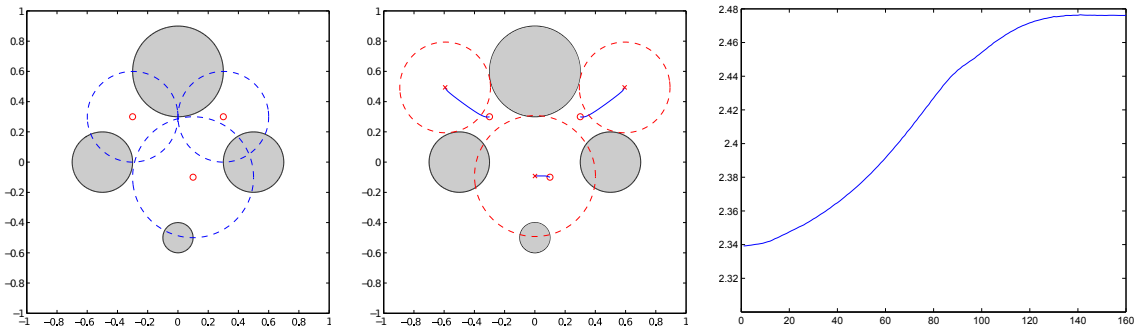
using Algorithm 2 where ∇_{x_i} denotes the gradient in the argument x_i.



(a) (b) (c)

Figure 2: Two sensors with different coverage ranges converge to the final locations. Two sensors originate at 'o' in (a) and end at 'x' in (b). Two curves in (b) shows the gradient flow paths for the sensors. The graph (c) shows the combined coverage area plotted with respect to time in the gradient flow.

In Figure 3, three sensor locations converge to the final positions. The graph in Figure 3 (c) shows that the coverage area increases and converges. However, we want to point out that the middle sensor in (b) still intersects with the obstacles, because this is a local, not global, optimal location. In contrast, the global optimal location is at either the left or the right bottom area of the computation domain.



(a) (b) (c)

Figure 3: Three sensors in (a) follow the gradient flow paths (blue curves in (b)) and converge to the final locations indicated by 'x' in (b) respectively. One sensor has a greater coverage range than the others. The coverage area in (c) is an increasing function and shows convergence. However, the final locations in (b) are a local maximum not a global one.

2.4. Full surveillance and path initialization

Up to now, we considered maximizing the coverage area when a certain number of stationary sensors with different coverage radii being given. Next, we extend this idea to establish a full surveillance of the environment by adding more sensors one by one. For simplicity, we consider all sensors having the same coverage radius r, and seek to place a small number (minimal if possible) of sensors that achieve the full surveillance. Then, we connect the optimal locations to construct a path for a mobile sensor. Since the mobile sensor visits every found location, it also attains the complete coverage of the environment. We present the method in Algorithm 4.

Algorithm 4 Full surveillance and path initialization

Input: initial sensor x_1, radius of coverage r, spatial step size h, and temporal step size k.

1. Evaluate $V(x_1; r)$ and $V_{D\backslash\Omega}$ which is the area of $D\backslash\Omega$. Set $i = 1$.
2. Repeat until $V(x_1, \cdots, x_i; r) = V_{D\backslash\Omega}$ for the full surveillance:
   When $V(x_1, \cdots, x_i; r) < V_{D\backslash\Omega}$, set $i = i + 1$ and
   add another sensor x_i into the set of sensors.
   Apply Algorithm 3 to $\{x_1, \cdots, x_i\}$ to find the optimal locations.
3. Path construction: Choose any starting sensor location x_1 and connect the closest location one by one, by using the shortest path algorithm [7].

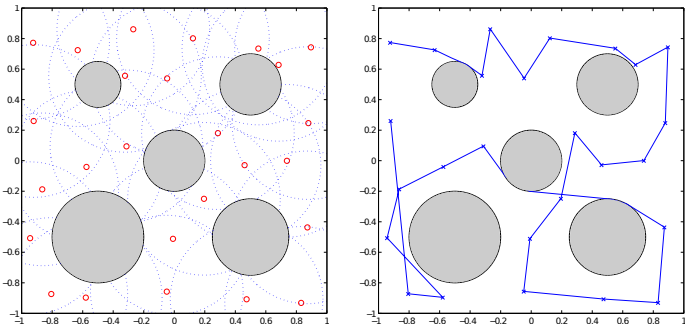
In Algorithm 4, we add sensors one by one, until the combined coverage area of multiple sensors equals to the area of the environment excluding the obstacles. Figure 4 shows an example of Algorithm 4. Image (a) illustrates all sensor locations for the full surveillance, and image (b) shows a constructed path connecting these locations. To be more precise, given the sensor locations $\{x_1, \cdots, x_m\}$, we denote $\Gamma(x_1, \cdots, x_m)$ the path which starts at x_1 and finishes at x_m, such that x_i and its nearest neighbor x_{i+1} are connected by the shortest path l_i for $1 \leq i \leq m-1$. As we shift our focus from stationary sensor placement to path planning for a mobile sensor, the locations x_i, $i = 1, \cdots, m$, are used as connecting points to construct the path. For this reason, we call them connecting points in the rest of the paper.

The length of the path Γ is defined by the sum

$$L(\Gamma(x_1, \cdots, x_m)) = \sum_{i=1}^{m-1} L(l_i). \tag{2.10}$$

Each l_i and its length in Euclidean metric are efficiently computed by the shortest path algorithm called evolving junction on obstacle boundaries (E-JOB) in [7]. Although this path provides the complete coverage of the environment, its length is far from being optimal. We use this path as the initial guess for the algorithm presented in the following sections.

Remark 1. Note that we are not optimizing the number of sensor locations for the full surveillance, i.e., a minimum number of sensor locations is not claimed. However, the smaller number of sensor locations is, the less computational effort it will require to compute the path length. Finding the minimal sensor locations itself is a challenging problem, and some related work can be found in [6, 28, 29], and [27].



(a) (b)

Figure 4: (a) Full surveillance using multiple sensors with a fixed coverage range of $r = 0.5$. (b) An path connecting the 26 sensor locations computed in (a). The Euclidean length of this path is 10.587. A mobile sensor achieves the complete coverage of the environment following this path. However, this path clearly is not optimal.

Remark 2. In step 3 of Algorithm 4, it is clear that the path can be constructed in many different ways. Finding the globally minimized path is related to the symmetric Traveling Salesman Problem [31, 32]. This problem is known to be NP-hard. Therefore, in this paper, we choose an alternative approach which does not require much effort. We pick any simple path as an initialization, and then find the global minimum via intermittent diffusion, which will be discussed next.

3. Path optimization with limited sensing ability (POLSA)

In previous sections, we presented algorithms to construct an initial path using the level set framework. In this section, we propose the path optimization algorithm to achieve the complete coverage of the environment. We assume the environment is given via level set settings. Starting from an initial path described in Section 2.3, we optimize the path by applying the following steps:

1. Define the manifold of complete coverage, and introduce a projection to map the sensor locations onto the manifold. (See Section 3.1.)
2. Use the gradient flow to minimize the path length, and then add random perturbations to push the path out of the traps of local optima. This step is essential to find a global minimizer. (See Section 3.2.)
3. Implement two techniques to speed up the computation. One is to remove redundant connecting points in the path, which leads to the dimension reduction in the gradient flow. The other is a disentanglement strategy to break some traps of local optimal solutions. (See Section 4.1.)

In the following, we explain each step in detail.

3.1. Complete coverage manifold and a local projection method

We define the complete coverage manifold as

$$\mathcal{M} = \{x_1, \cdots, x_m \in D\backslash\Omega \mid V(\Gamma(x_1, \cdots, x_m)) - V_{D\backslash\Omega} = 0\}, \tag{3.1}$$

where $V_{D\setminus\Omega}$ is the area of the environment excluding the obstacles. $\Gamma(x_1, \cdots, x_m)$ is a feasible path determined by connecting $\{x_1, \cdots, x_m\}$ as described in the previous section. The coverage area of the path is computed by the accumulated coverage as the sensor moves along the path. It is computationally expensive to evaluate $V(\Gamma)$ exactly. We handle this by discretely sampling the path $\Gamma(x_1, \cdots, x_m)$ using the locations $\{z_1, \cdots, z_M\}$ on Γ with $M \gg m$. For example, in our numerical experiments, we choose certain number of points along each path l_i, $i = 1, \cdots, m$, and then take the union. The coverage area of the path can then be approximated by

$$V(\Gamma(x_1, \cdots, x_m)) \approx \int_\Omega H(\phi(y; z_1, \cdots, z_M, r))dy. \tag{3.2}$$

Let us assume that a given set of connecting points $\{\tilde{x}_1, \cdots, \tilde{x}_m\}$, e.g., the one obtained by the initialization in Section 2.4, is on the complete coverage manifold $\mathcal{M}$. That is, the path associated with these points achieves the complete coverage of the environment. We consider a situation in which these points are moved to nearby, but different, locations. Then, it is not generally guaranteed that the perturbed points still remain on $\mathcal{M}$. To enforce that the new path achieves the complete coverage, we need to project the perturbed points back onto $\mathcal{M}$ before the path construction.

We define the projection to $\mathcal{M}$ as the solution $\{x_1 \cdots, x_m\}$ of the following constrained minimization problem:

$$\min \sum_{i=1}^{m} ||x_i - \tilde{x}_i||^2 \quad \text{subject to} \quad g(x_1, \cdots, x_m) = 0, \tag{3.3}$$

where $||\cdot||$ is the Euclidean norm, and the constraint is given by

$$g(x_1, \cdots, x_m) = V(\Gamma(x_1, \cdots, x_m)) - V_{D\setminus\Omega}. \tag{3.4}$$

In words, we search for the closest connecting points x_i to $\tilde{x}_i$ that provide complete coverage of the environment. This optimization problem can be solved by converting it to the following unconstrained optimization problem with a Lagrange multiplier λ,

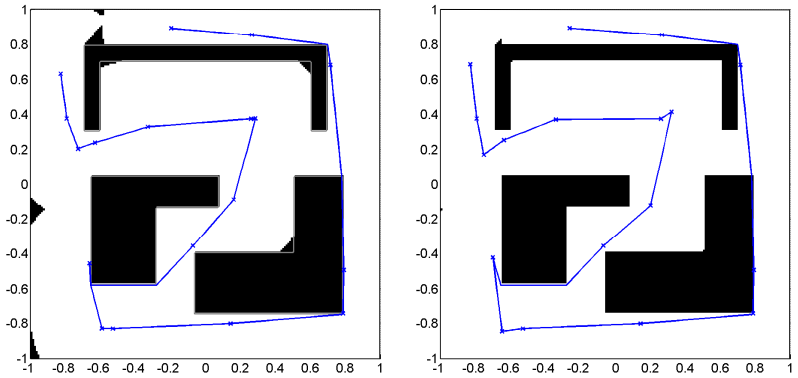
$$\mathcal{L}(x_1, \cdots, x_m, \lambda) = \sum_{i=1}^{m} \frac{||x_i - \tilde{x}_i||^2}{2} - g(x_1, \cdots, x_m)\lambda. \tag{3.5}$$

The necessary condition for (3.3) becomes $\partial\mathcal{L}/\partial x_i = 0$ for $i = 1, \cdots, m$, and $\partial\mathcal{L}/\partial\lambda = 0$. This leads to a system of equations for x_i, $i = 1, \cdots, m$,

$$\begin{aligned} x_i &= \tilde{x}_i + \partial_{x_i} g(\tilde{x}_1, \cdots, \tilde{x}_m)\lambda, \\ 0 &= g(x_1, \cdots, x_m). \end{aligned} \tag{3.6}$$

Note that we replaced $(x_1, \cdots, x_m)$ with $(\tilde{x}_1, \cdots, \tilde{x}_m)$ in the argument of $\partial_{x_i} g$ to speed up the numerical evaluations. Then (3.6) can be efficiently solved by Newton's iteration given as,

$$\begin{aligned} \Delta\lambda^{(i)} &= -\left(\partial_{x_i} g(\tilde{x}_1, \cdots, \tilde{x}_m)^2\right)^{-1} g(\tilde{x}_1, \cdots, \tilde{x}_i + \partial_{x_i} g(\tilde{x}_1, \cdots, \tilde{x}_m)\lambda^{(i)}, \tilde{x}_m), \\ \lambda^{(i+1)} &= \lambda^{(i)} + \Delta\lambda^{(i)}. \end{aligned}$$



(a) A path before the projection. 98.137% of the environment is covered by a moving sensor.

(b) A path after the projection. 99.997% of the environment is covered by a moving sensor.

Figure 5: Example of the projection method onto the complete coverage manifold $\mathcal{M}$.

Figure 5 shows an example for this step. The obstacles are three concave polygons of various sizes. Image (a) shows a path before the projection step. Notice that there are dark shaded regions outside the polygons, and they are the non-covered regions by the mobile sensor. Image (b) shows the path after the projection step, which achieved 99.997% of the total are, which is viewed as achieving the complete coverage due to the numerical error.

3.2. POLSA

The optimal path problem we want to solve can be formulated as optimizing the path length on the complete coverage manifold $\mathcal{M}$:

$$\min_{\Gamma(x_1,\cdots,x_m)\in\mathcal{M}} L(\Gamma(x_1,\cdots,x_m)). \tag{3.7}$$

To solve it, a typical numerical method is the gradient descent flow,

$$\frac{d}{dt}x_i(t) = -\nabla_{x_i} L\left(\Gamma(x_1,\cdots,x_m)\right), \tag{3.8}$$

followed by a projection step onto $\mathcal{M}$ as explained in the previous section. When the solution reaches a steady state, it is a local minimum. In general, there may be many such local minima. To find a global optimal solution, we use the intermittent diffusion which was proposed in [42]. It adds random perturbations intermittently to the gradient flow (3.8) which leads to the following SDE,

$$dx_i(t,w) = -\nabla_{x_i} L\left(\Gamma(x_1,\cdots,x_m)\right) dt + \sigma(t) dW(t), \tag{3.9}$$

where $W(t)$ is the Brownian motion in $\mathbb{R}^{2m}$, and $\sigma(t)$ is a piecewise constant function alternating between zero and a random positive constant. The perturbation is added so that the path can escape from the trap of a local minimizer and approach other ones.

Now, we are ready to present the algorithm, POLSA, to compute the global optimal path achieving the complete coverage of the environment.

Path optimization with limited sensing ability (POLSA)
Input: initial sensor x_1, radius of coverage r, spatial step size h, and temporal step size k.

1. Initialization of the path: use Algorithm 4.
2. (Optional) Disentanglement of path for a fast convergence (details in Section 4.1).
3. Set $j = 1$ and iterate N times to obtain N different paths.
   (a) Set α as the scale for diffusion strength, and γ the scale for diffusion time. Let $\sigma := \alpha d$, and $T := \gamma t$ where two positive random numbers d, t within $[0, 1]$ by uniform distribution.
   (b) Set the optimal path $\Gamma_{opt} = \Gamma^{(j)}(x_1, \cdots, x_m)$.
   (c) Taking Γ_{opt} as the initial condition, compute the stochastic equation for $t \in [0, T]$, $i = 1, \cdots, m$,
   $$dx_i(t, w) = -\nabla_{x_i} L\left(\Gamma(x_1, \cdots, x_m)\right) dt + \sigma dW(t),$$
   and record the final state.
   (d) (Optional) Disentanglement of path for a fast convergence (details in Section 4.1).
   (e) Compute the solution for the following system until a convergence criterion is satisfied,
   $$\frac{d}{dt} x_i(t) = -\nabla_{x_i} L\left(\Gamma(x_1, \cdots, x_m)\right).$$
   (f) (Optional) removal of redundant locations (details in Section 4.1).
   (g) Project the final state of step 3(e) onto the manifold $\mathcal{M}$ to obtain $\Gamma^{(j+1)}(x_1, \cdots, x_m) \in \mathcal{M}$. If $L(\Gamma^{(j+1)}) < L(\Gamma_{opt})$, set $\Gamma_{opt} = \Gamma^{(j+1)}$.
   (h) Repeat with $j = j + 1$.

The equation (3.8) is solved by the forward difference scheme with respect to the time and the central difference scheme with respect to the space. To solve the SDE (3.9), we used the Euler-Maruyama method [18]. For the convergence criterion in step 3(e), we check if the Euclidean distance between two successive iterates is less than a prescribed tolerance $\epsilon > 0$ in our experiments. This algorithm produces N (possibly different) local optimal paths $\Gamma^{(1)}, \cdots, \Gamma^{(N)}$. By the theory of intermittent diffusion, for a large enough N, we obtain Γ_{opt} as an approximation to the globally shortest path [42]. We note that in POLSA, there are optional steps in 2, 3(d) and 3(f), which will be discussed later.

4. Numerical implementations and improvements

In this section, we present various numerical experiments showing the robustness with respect to the initial guesses, different environment, and effects of different values for the radius r. In addition, we introduce two strategies to improve the efficiency of the numerical algorithm.

Initialization: The first example in Figure 6 shows the performance of our algorithm by using different initializations. Three different initial paths, depicted in the left column, have the same initial sensor locations, and attain the complete coverage of the environment.

The results are collected from $N = 200$ iterations. Noticed that the resulting path length converges to a similar value (with an error depending on the spatial step size h), and the configurations of the resulting paths are similar.

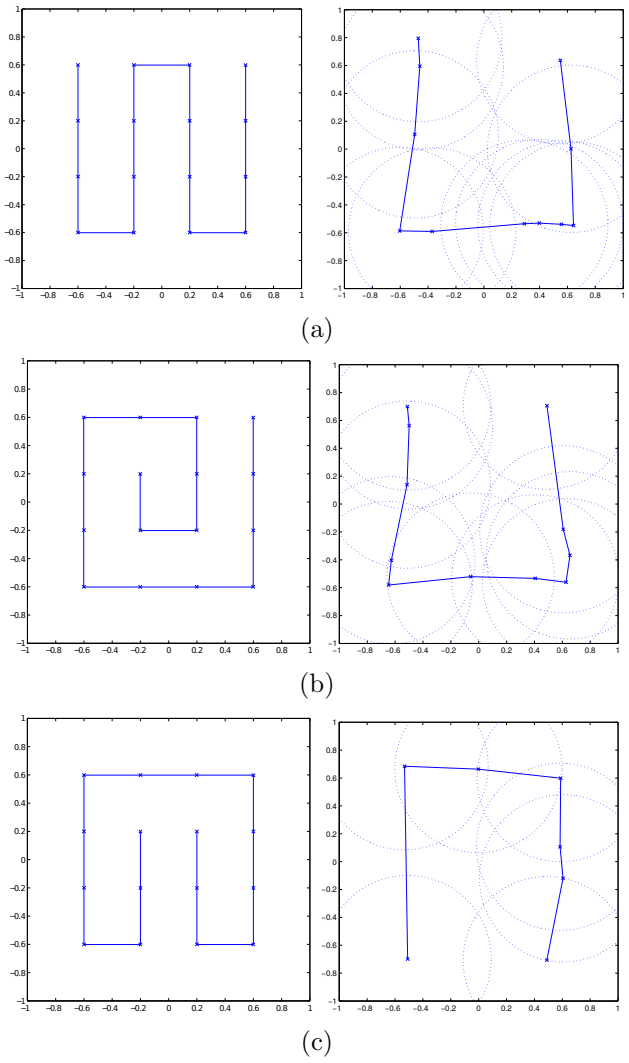
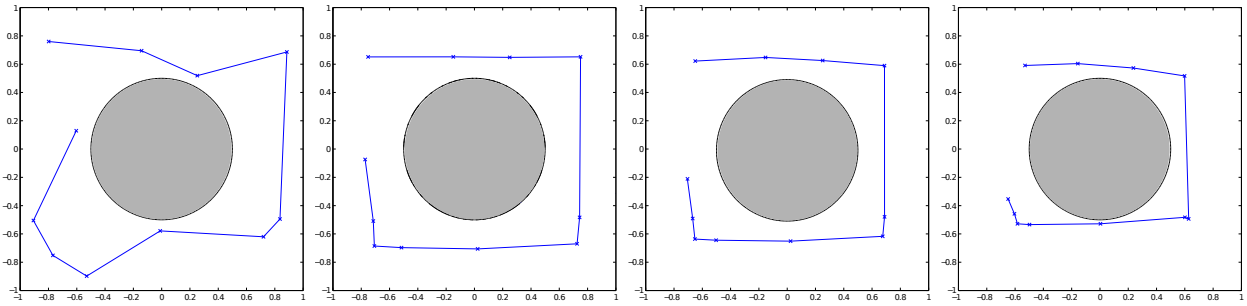


Figure 6: The first column of images shows three different initial paths, and the second column of images shows the resulting paths. These all have the same initial length of 6 with the same 16 connecting points, but with different path configurations. For the experiments, $r = 0.6$ and $h = 0.01$ are used. The lengths of the shortest paths are (a) 3.833, (b) 3.856, and (c) 3.826. They converge around 3.8 with the similar configuration.

Different values for the radius: Figure 7 shows the effect of different coverage radius. From the same initial path given in (a), three different results are computed using different values of r. Notice that as r increases the path gets closer to the obstacle, and gives a shorter path length.

Closed path: The proposed algorithm can also construct a closed path, i.e., a path which returns to its starting point while achieving the complete coverage of the environment. For the closed path, its length is computed by including the final segment returning to the



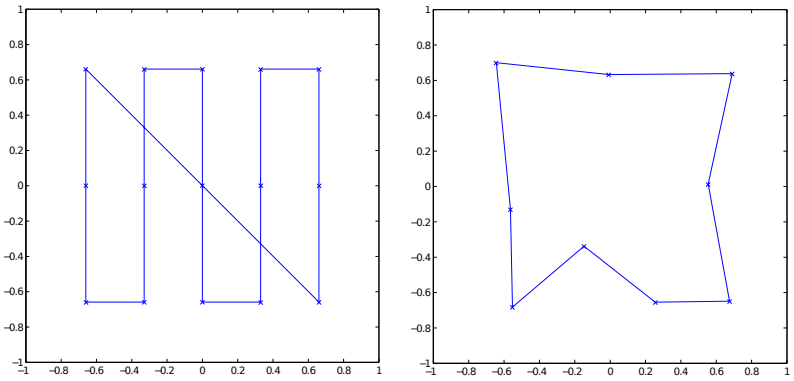
(a) Initial path. (b) $r = 0.45$. (c) $r = 0.53$. (d) $r = 0.65$.

Figure 7: Using different radii of coverage r. (a) The length of the initial path is 5.707. As r increases, the coverage of a sensor becomes broader. This is reflected on the resulting path lengths (b) 4.875, (c) 4.338, and (d) 3.547.

starting location. This is a relatively easy extension in our setting: In (2.10), let l_m be the shortest path connecting the ending location x_m and the starting location x_1. Then, the length of the path Γ returning to the start location becomes

$$L(\Gamma(x_1, \cdots, x_m)) = \sum_{i=1}^{m} L(l_i).$$

In Figure 8, we illustrate an example of the shortest path returning to the starting location. The result is collected from $N = 100$, $r = 0.5$, and $h = 0.01$. Compared with the complicated initial path showing in the left picture, the resulting path is much shorter, without self-intersections.



(a) Initial path. (b) The final path.

Figure 8: (a) The Initial path was constructed as a closed route. (b) The resulting path is obtained using the parameters $r = 0.5$, $h = 0.01$ and $N = 100$. The length of the path changed from 9.787 to 5.500.

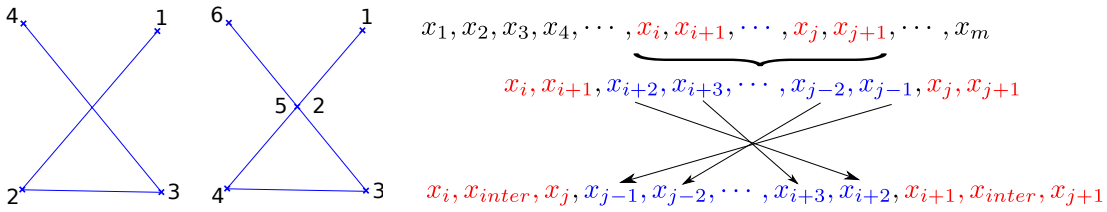
4.1. Two strategies to improve the efficiency

In our numerical experiments, we made two observations: (1) the path may self-intersect in the initialization and during the gradient flows; and (2) all of the connecting points move together, and some points can get cluttered. In this section, we discuss two strategies to improve the efficiency of the POLSA algorithm.

First, we introduce a simple method to **disentangle** a path. When a path forms a self-intersection, it is always possible, by the triangle inequality, to reduce the length of the path by disentangling it. The main idea is, after finding the location of self-crossing, to insert two new points at the crossing and reverse the order of the sensor locations within the tangled region. Figure 9 depicts this strategy with a simple example. In (a), the path on the left tangles with itself. We insert two connecting points at the intersection and reverse the order of the connecting points to form the new list shown on the right of Figure 9(a). After this step, two added points 2 and 5 can split and move away from each other in the gradient flow because it reduces the path length. In Figure 9 (b), we illustrate how the strategy works in general situations, and the steps are described in Algorithm 5.

Algorithm 5 Disentanglement of the path

1. Given the path $\Gamma(x_1, \cdots, x_m)$, identify the location of the crossing, and denote the vertices of the lines to be $\overline{x_i x_{i+1}}$ and $\overline{x_j x_{j+1}}$.
2. Let x_{inter} be the intersection of two lines $\overline{x_i x_{i+1}}$ and $\overline{x_j x_{j+1}}$, and insert x_{inter} between x_i and x_{i+1} as well as between x_j and x_{j+1}.
3. Reorder the points to $x_i, x_{inter}, x_j, x_{j-1}, x_{j-2}, \cdots, x_{i+2}, x_{i+1}, x_{inter}$ and x_j.
4. Repeat 1-3 until no crossing exists.



(a) Simple version. (b) General version.

Figure 9: A strategy for disentangling the path.

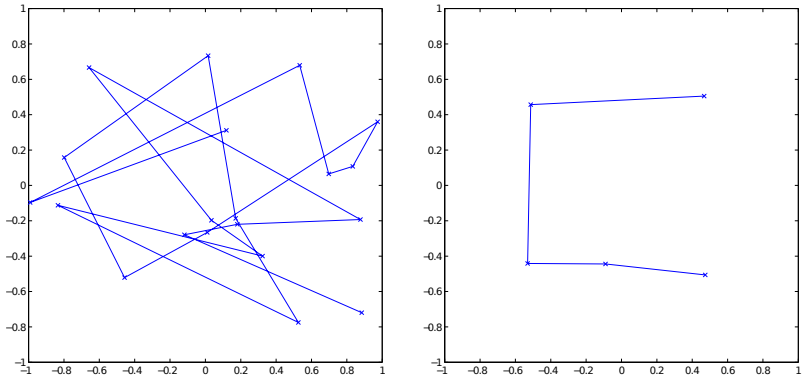
We remark that even without this step of disentanglement, the intermittent diffusion will disentangle the path as well, however at a very slow pace. By introducing this automated step of disentanglement, the convergence of our algorithm can be dramatically improved.

Next, we **remove redundant** connecting points if some are clustered around each other. The idea is to retain at least one connecting point from the cluster without any loss of coverage area. This strategy thus reduces the dimensions in (3.8) and (3.9), and helps us to improve the computation speed in the simulation while keeping the complete coverage. The details of the implementation are given in Algorithm 6.

Algorithm 6 Removing redundant locations

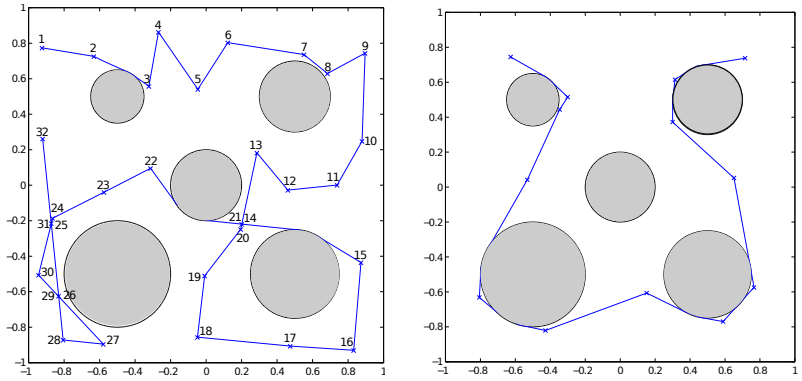
1. For $i = 1, 2, \cdots, m$, get rid of the sensor x_{i+1} in the sensor locations $\{x_1, \cdots, x_m\}$ if the following inequality holds:
$$||x_i - x_{i+1}|| \leq \epsilon$$
where $\epsilon = \mathcal{O}(h)$, the size of spatial discretization.

Figure 10 shows the results of the improved full algorithm with $N = 100$. We note that the number of connecting points reduces from 20 to 5, and the complicated path is untangled. Moreover, Figure 11 illustrates the final result of POLSA applied to the environment given in Figure 4. Image (a) shows the initial path configuration after disentanglement. The number of connecting points increases from 26 to 32 because intersections are counted twice. Concurrently, the points are reordered for after the disentanglement. Image (b) shows the resulting optimal path. The path length is reduced to 5.322, and the number of connecting points is reduced to 13.



(a) Initial path. (b) Optimal path.

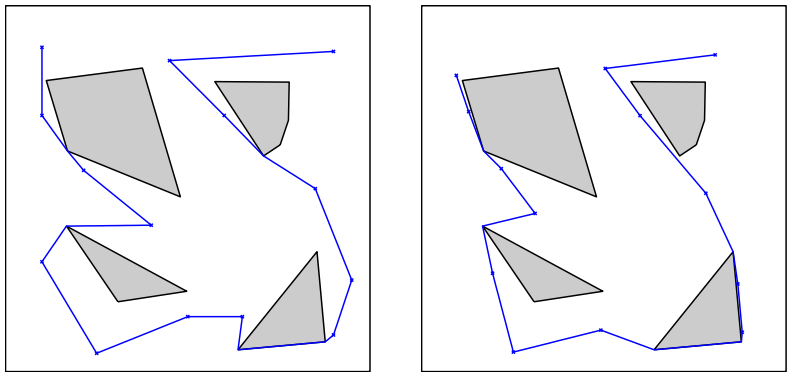
Figure 10: With two added efficient strategies, the number of connecting points changes from 20 to 5, and the path is simplified. The path length reduced from 17.0404 to 2.8895 using $r = 0.75$, $h = 0.01$ and $N = 200$.



(a) (b)

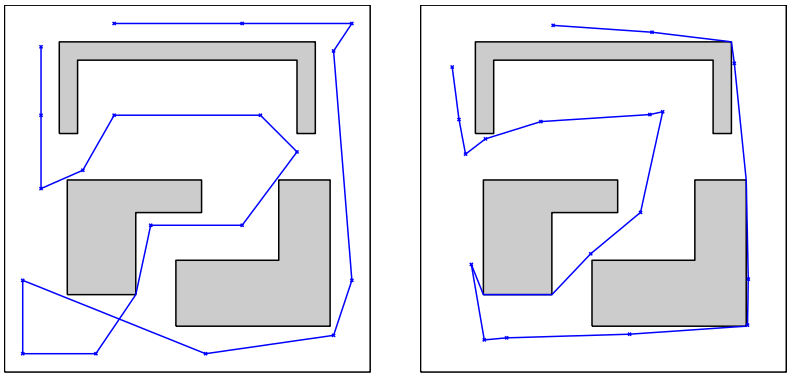
Figure 11: The POLSA result of Figure 4. (a) Initial path configuration after disentanglement. The number of connecting points increases from 26 to 32, and the sensors are reordered in the path for the efficient disentanglement. (b) The resulting optimal path. The length reduced from 10.587 to 5.322, and the number of connecting points is reduced to 13. We set the radius of coverage $r = 0.5$, and the spatial step size $h = 0.01$.

Remark 3. In Figure 11 (b), several points adjacent to the obstacles are not located on the boundary. This is a numerical aspect. If we choose either more points on the path or smaller spatial discretization h, the points will be placed closer to the boundary.



(a) Initial path. (b) Optimal path.

Figure 12: (a) Initial path is given with $r = 0.5$, and $h = 0.01$. 14 stationary sensor locations are initially found for the full surveillance of the environment. (b) The resulting path length reduces from 6.899 to 5.511. $N = 30$.



(a) Initial path. (b) Optimal path.

Figure 13: (a) Initial path is given with $r = 0.4$, $h = 0.01$. (b) The number of connecting points are changed from 19 to 21 by disentanglement, and then to 18 by removing redundant sensors. The length of the path reduces from 9.785 to 7.770. $N = 30$.

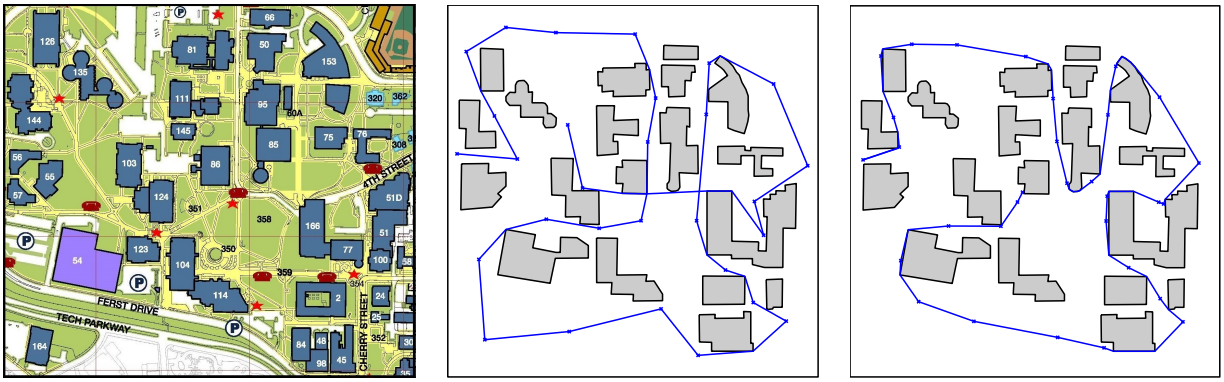
4.2. Applications

In this section, we apply our algorithm in more complicated realistic environments. Figure 12 (a) shows the convex polygonal obstacles with an initial path. Figure 12 (b) describes the shortest path of a mobile sensor scanning the full environment. The result is obtained from $N = 30$.

In Figure 13, the room-like environment with concave polygonal obstacles is considered. We notice the path has a sharp turn around one corner of the lower left obstacle, but then quickly backtracks. The length of the path is doubled in segment, but this is needed to obtain the complete coverage of the environment. This feature is thanks to the projection method which enforces to scan the complete environment.

A more realistic application of our path optimization using the Georgia Tech campus map is presented in Figure 14. A campus map [3] is represented by a level set function with

[3] http://map.gtalumni.org/campusmap.pdf



(a) (b) (c)

Figure 14: (a) Campus map of Georgia Tech. (b) An initial path with the complete coverage. We used parameters $r = 0.5$, $h = 0.01$, and $N = 10$. (b) The resulting optimal path. The path length is reduced from 10.666 to 8.100. And the final path achieves 99.641% coverage of the environment.

buildings as obstacles. Some buildings have complicated shapes. The initial and optimal paths are depicted in Figure 14 (b) and (c) respectively. The final path achieves 99.412% coverage of the environment, with the error within the numerical discretization error in the level set computation. The number of connecting points changes from 29 to 33 after the disentanglement, and then to 31 by removing redundant points.

5. Conclusion

We proposed a novel method for the path optimization for sensors with limited coverage range in a known environment. We first presented the level set formulation that enables us to solve the art gallery problem with limited coverage range and find the optimal placements for stationary sensors. While the resulting locations are used for the initialization of the path, we employed the intermittent diffusion and gradient descent to optimize the length of the path iteratively. During this procedure, the complete coverage of the environment is ensured by projections onto the complete coverage manifold. From the results of several examples, we verified the effectiveness of our algorithm. In principle, our approach can be generalized to the path planning problem in multi(> 2) dimensions, though the implementation will be substantially more involved. This will be one of the future developments to pursue. The disentanglement may be a useful strategy in many other path optimization problems.

References

[1] A. Aggarwal. *The Art Gallery Theorem: Its Variations, Applications and Algorithmic Aspects*. PhD thesis, 1984. AAI8501615.

[2] H. Ando, Y. Oasa, I Suzuki, and M. Yamashita. Distributed memoryless point convergence algorithm for mobile robots with limited visibility. *Robotics and Automation, IEEE Transactions on*, 15(5):818–828, Oct 1999.

[3] S. Arora. Polynomial time approximation schemes for Euclidean TSP and other geometric problems. In *37th Annual Symposium on Foundations of Computer Science (Burlington, VT, 1996)*, pages 2–11. IEEE Comput. Soc. Press, Los Alamitos, CA, 1996.

[4] L.-T. Cheng and Y.-H. Tsai. Visibility optimization using variational approaches. *Commun. Math. Sci.*, 3(3):425–451, 2005.

[5] B. V. Cherkassky, A. V. Goldberg, and T. Radzik. Shortest paths algorithms: theory and experimental evaluation. *Math. Programming*, 73(2, Ser. A):129–174, 1996.

[6] W.-P. Chin and S. Ntafos. Optimum watchman routes. *Information Processing Letters*, 28(1):39 – 44, 1988.

[7] S.-N. Chow, J. Lu, and H.-M. Zhou. Finding the shortest path by evolving junctions on obstacle boundaries (e-job): An initial value ode's approach. *Applied and Computational Harmonic Analysis*, 35(1):165 – 176, 2013.

[8] R.T. Collins, A.J. Lipton, H. Fujiyoshi, and T. Kanade. Algorithms for cooperative multisensor surveillance. *Proceedings of the IEEE*, 89(10):1456–1477, Oct 2001.

[9] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition, 2001.

[10] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numer. Math.*, 1:269–271, 1959.

[11] U. M. Erdem and S. Sclaroff. Optimal placement of cameras in floorplans to satisfy task requirements and cost constraints. In *In Proc. of OMNIVIS Workshop*, 2004.

[12] R. W. Floyd. Algorithm 97: Shortest path. *Commun. ACM*, 5(6):344–345, June 1962.

[13] G. L. Foresti, C. S. Regazzoni, and P. K. Varshney. *Multisensor Surveillance Systems: The Fusion Perspective*. Springer, 2003.

[14] R. J. Fowler, M. S. Paterson, and S. L. Tanimoto. Optimal packing and covering in the plane are np-complete. *Information Processing Letters*, 12(3):133 – 137, 1981.

[15] J. Hershberger and S. Suri. Efficient computation of euclidean shortest paths in the plane. In *Foundations of Computer Science, 1993. Proceedings., 34th Annual Symposium on*, pages 508–517, Nov 1993.

[16] D. B. Johnson. Efficient algorithms for shortest paths in sparse networks. *J. ACM*, 24(1):1–13, January 1977.

[17] G. D. Kazazakis and A. A. Argyros. Fast positioning of limited-visibility guards for the inspection of 2d workspaces. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 3, pages 2843–2848 vol.3, 2002.

[18] P. E. Kloeden and E. Platen. *Numerical solution of stochastic differential equations*, volume 23 of *Applications of Mathematics (New York)*. Springer-Verlag, Berlin, 1992.

[19] B. Kuipers and Y.-T. Byun. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Robotics and Autonomous Systems*, 8(1-2):47 – 63, 1991. Special Issue Toward Learning Robots.

[20] Y. Landa, D. Galkowski, Y.R. Huang, A. Joshi, C. Lee, K.K. Leung, G. Malla, J. Treanor, V. Voroninski, A.L. Bertozzi, and Y.-H. R. Tsai. Robotic path planning and visibility with limited sensor data. In *American Control Conference, 2007. ACC '07*, pages 5425–5430, July 2007.

[21] Y. Landa and R. Tsai. Visibility of point clouds and exploratory path planning in unknown environments. *Commun. Math. Sci.*, 6(4):881–913, 2008.

[22] Y. Landa, R. Tsai, and L.T. Cheng. Visibility of point clouds and mapping of unknown environments. In *Springer Notes in Computational Science and Engineering*, pages 1014–1025, 2006.

[23] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Norwell, MA, USA, 1991.

[24] S. M. LaValle. *Planning algorithms*. Cambridge University Press, Cambridge, 2006.

[25] J. S. B. Mitchell. Shortest paths and networks. In *Handbook of discrete and computational geometry*, CRC Press Ser. Discrete Math. Appl., pages 445–466. CRC, Boca Raton, FL, 1997.

[26] J. S. B. Mitchell. Geometric shortest paths and network optimization. In *Handbook of computational geometry*, pages 633–701. North-Holland, Amsterdam, 2000.

[27] A. T. Murray, K. Kim, J. W. Davis, R. Machiraju, and R. Parent. Coverage optimization to support security monitoring. *Computers, Environment and Urban Systems*, 31(2):133 – 147, 2007.

[28] S. Ntafos. Watchman routes under limited visibility. *Comput. Geom.*, 1(3):149–170, 1992.

[29] J. O'Rourke. *Art gallery theorems and algorithms*. International Series of Monographs on Computer Science. The Clarendon Press, Oxford University Press, New York, 1987.

[30] S. Osher and R. Fedkiw. *Level set methods and dynamic implicit surfaces*, volume 153 of *Applied Mathematical Sciences*. Springer-Verlag, New York, 2003.

[31] M. Padberg and G. Rinaldi. Optimization of a 532-city symmetric traveling salesman problem by branch and cut. *Operations Research Letters*, 6(1):1 – 7, 1987.

[32] M. Padberg and G. Rinaldi. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Review*, 33(1):60–100, 1991.

[33] C. H. Papadimitriou. The Euclidean traveling salesman problem is NP-complete. *Theoret. Comput. Sci.*, 4(3):237–244, 1977.

[34] I. Pavlidis, V. Morellas, P. Tsiamyrtzis, and S. Harp. Urban surveillance systems: from the laboratory to the commercial world. *Proceedings of the IEEE*, 89(10):1478–1497, Oct 2001.

[35] G. Reinelt. *The Traveling Salesman: Computational Solutions for TSP Applications*. Springer-Verlag, Berlin, Heidelberg, 1994.

[36] C. A. Rogers. *Packing and covering*. Cambridge Tracts in Mathematics and Mathematical Physics, No. 54. Cambridge University Press, New York, 1964.

[37] T.C. Shermer. Recent results in art galleries. *Proceedings of the IEEE*, 80(9):1384–1399, Sep 1992.

[38] B. Tovar, R. Murrieta-Cid, and S. M. LaValle. Distance-optimal navigation in an unknown environment without sensing distances. *Trans. Rob.*, 23(3):506–518, 2007.

[39] Y.-H. R. Tsai, L.-T. Cheng, S. Osher, P. Burchard, and G. Sapiro. Visibility and its dynamics in a PDE based implicit framework. *J. Comput. Phys.*, 199(1):260–290, 2004.

[40] J. Urrutia. Art gallery and illumination problems. In *Handbook of Computational Geometry*, pages 973–1027. North-Holland, 2000.

[41] B. Yamauchi. A frontier-based approach for autonomous exploration. In *Computational Intelligence in Robotics and Automation, 1997. CIRA'97., Proceedings., 1997 IEEE International Symposium on*, pages 146–151, Jul 1997.

[42] H.-M. Zhou, S.-N. Chow, and T.-S. Yang. Global optimizations by intermittent diffusion. In *Chaos, CNN, Memristors and Beyond*, chapter 37, pages 466–479. Mar 2013.