Solving Optimal Camera Placement Problems in IoT Using LH-RPSO

XIAOHUI WANG, HAO ZHANG, AND HAORAN GU
School of Aeronautics, Beihang University, Beijing 100191, China
Corresponding author: Xiaohui Wang (xhwang@buaa.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 41804165, and in part by the Key Laboratory of Spacecraft Design Optimization and Dynamic Simulation Technologies, Ministry of Education.

ABSTRACT

With the increasing need for public security and intelligent life and the development of Internet of Things (IoT), the structure and application of vision sensor network are becoming more and more complex. It is no longer a system with simple static monitoring, but a complex system that can be used for intelligent processing, such as target localization, identification, tracking and so on. In order to accomplish various tasks efficiently, it is important to determine the deployment plan of camera network in advance. Many researches discretize the optimal camera placement problem into a binary integer programming (BIP) problem, which is NP-hard, and put forward some approximate solutions including greedy heuristics, semi-definite programming, simulated annealing, etc. In practice, however, camera parameters include both continuous values (location and orientation) and discrete values (camera type). To get a much more accurate result, we do not discretize the continuous camera parameters any more, on the contrary, we handle the continuous values in continuous domain directly. Meanwhile, a Latin Hypercube based Resampling Particle Swarm Optimization (LH-RPSO) algorithm is proposed to effectively solve the problem. To validate the proposed algorithm, we compared it with standard Particle Swarm Optimization (PSO) and Resampling Particle Swarm Optimization (RPSO). Simulation results for an outdoor planar regions illustrated the efficiency of the proposed algorithm.

INDEX TERMS

Camera placement, Internet of Things, Latin hypercube sampling, resampling particle swarm optimization, smart sensor networks, visibility.

I. INTRODUCTION

Internet of Things (IoT) is the extension and expansion of Internet technology. The IoT connects objects and realizes information sharing by using technologies such as recognition, perception, and communication. The development and application of the IoT has brought great convenience to people’s daily life. Undoubtedly, it has become a technical with great development potential. Many technologies in the IoT rely on sensor networks because they are the hardware foundation for sensing, acquiring, processing, and transmitting information. The performance of the sensor network largely determines the execution of subsequent tasks. Therefore, sensor placement is a vital design issue, and scholars have done a lot of researches on sensor network [1], [2].

This paper focuses on the deployment of visual sensor networks for computer vision-related tasks, i.e., the optimal camera placement problems, which have been studied for decades. Camera placement is a very important issue in computer vision and IoT. Camera network is a necessary facility for all complex tasks, and its performance directly affects the execution of subsequent tasks. For example, regional monitoring requires a high coverage camera network, target recognition and tracking require a high resolution and multiple coverage camera network, etc. Therefore, the optimal camera placement problem is an important and practical topic. Current research in optimal camera placement has focused on two main directions: formulating the problem to address specific user requirements, and developing an effective optimization strategy. Most researches discretize the optimal camera placement problem into a BIP problem, which is NP-hard, and put forward some approximate solutions including greedy heuristics, semi-definite programming, simulated annealing, etc., [3]–[6]. In practice, however, camera parameters include both continuous values, such as location and orientation, and discrete values, such as camera type. It would
be more accurate if we handle the continuous values in continuous domain directly, rather than discretize them. Some researchers argue that the continuous-based formulations are not suitable for large-scale problems because of the dramatically increased complexity when practical considerations are incorporated. But we believe this situation will be improved with increased computing power and new approaches [7], [8].

The optimal camera placement is essentially a complex optimization problem, and efficient algorithm is needed to deal with it. Particle swarm optimization (PSO), a kind of swarm intelligence algorithm inspired by the social behaviors of bird flocks when searching for food [9], has been proved to be efficient and robust in solving complex optimization problems. However, the PSO algorithm is not without defects. Its biggest disadvantage is that it is easy to prematurely converge and fall into the local optimal solution. Most of the existing improvements to the algorithm are designed to overcome this shortcoming. Besides, as a kind of swarm intelligence optimization algorithm, the PSO needs a great deal of computation time, so improving efficiency is also a key issue. In order to improve the algorithm, we proposed a novel variant of PSO, named LH-RPSO, which combines Latin hypercube sampling and resampling particle swarm optimization.

In this paper, we defined the optimal camera placement problem, which includes camera model, environment model, visibility algorithm and optimization model. The camera model contains geometric information about the camera’s coverage area. The environment model presents a mathematical representation of the target region. The visibility algorithm is used to determine whether a position is covered by the cameras. And the optimization model is formulations of the optimization problem corresponding to the optimal camera placement problem. Then, we introduced the flow of the LH-RPSO algorithm in detail and analyzed the key technologies. Finally, we selected a real-world campus as the target area to deploy the camera network. The process consisted of two steps. In the first step, we find a solution with a lowest cost, satisfying the coverage constraint. And in the second step, we further improve the coverage of the network based on the solution of the previous step. Experimental results of both steps have shown that the LH-RPSO performs better than the PSO and the RPSO, which also fully demonstrated that the LH-RPSO can be used in practical large-scale camera placement problems.

The contributions of this study are as follows:

- We introduced the camera model, environment model, and a visibility judgment method based on a line drawing algorithm, and then gave the two most important formulation of camera network placement.
- In order to improve the performance of the RPSO, we combined Latin hypercube sampling and proposed the LH-RPSO algorithm.
- To test the performance of the algorithm, we selected a real-world campus as the target area to deploy the camera network. Experimental results have shown that the LH-RPSO has higher performance than the PSO and the RPSO.

The rest of this paper is organized as follows. First, we give a brief literature review in Section II. Then, Section III defines the optimal camera placement problems. In Section IV, we show the details of the LH-RPSO algorithm. Extensive experimental results are presented in Section V to illustrate the performances. Finally, we conclude this paper, and point out several potential directions for the future works in Section VI.

II. RELATED WORK

Sensor networks have a wide range of applications in IoT, for examples, the structural health monitoring [1] and multimedia big data communications [2]. And the vision sensor network is the hardware foundation of computer vision related tasks, such as target localization, identification, tracking [10] and so on. The optimal camera placement problems have been studied for decades. Actually, the earliest related work can be traced back to the Art Gallery Problem (AGP) in the field of computational geometry [11]. The AGP focus on how to theoretically place the cameras in an art gallery so as to maximize the visual coverage of the valuable assets. Due to the widespread and diverse deployment of camera networks, the focus of optimal camera placement problem has gradually shifted from theoretical analysis to practical application, and the models is becoming more complex with realistic assumptions. Junbin Liu etc. summarized the research progress of optimal camera placement post 2000 in [12], analyzed the general steps for the conventional camera placement approaches, and present that current research in automated camera placement has focused on two main directions: addressing specific user requirements, or developing an effective optimization strategy.

Most camera placement researches focuses on the formulation of the problem. Erdem and Sclaroff proposed a general formulation of the problem [13], with the goal of determining the optimal positions and number of cameras for a region to be observed, satisfying a set of task-specific constraints and minimizing a given cost function. Horster and Lienhart did some similar work [14], but the difference is that they used points to represent space instead of polygons. These points with different importance have different weights. Zhao and Cheung proposed a sophisticated probabilistic model to capture the uncertainty of object orientation and mutual occlusion [15]. Other similar works can be found in [16]–[19], etc. Most of the above papers consider the optimal camera placement problem in discrete domain entirely. In order to reduce the complexity of the problem, the parameters including location and orientation are quantized. In other words, the camera positions and orientations are restricted to a set of specific points and angles. There are two main methods to select candidate camera parameters from continuous space: sampling [13] and decision from the expert system [20]. It’s worth noting that the selection of candidate cameras has a direct effect on the solution.
In these formulations, the objective functions and constraints are all linear expressions of the binary decision variable, which indicates whether a candidate is chosen. This approach to formulating the problem complies with the format required for Binary Integer Programming (BIP) [12]. In contrast, some researchers formulate the problem in continuous domain. Bodor et al. considered aggregated motion observability [21], and developed a general analytical formulation of the observation problem in terms of motion statistics of a scene and resolution of observed actions. Mittal and Davis [22] probabilistically estimated dynamic occlusion, introducing a constraint: visibility in the presence of random occluding objects, which had not been addressed earlier. Liu et al. [23] proposed a general statistical formulation of the optimal selection of camera configurations, with a Trans-Dimensional Simulated Annealing algorithm to effectively solve it. All the objectives and constraints mentioned in above papers are formulated in a nonlinear form. Some researchers argue that the continuous-based formulations are not suitable for large-scale problems because of the dramatically increased complexity when practical considerations are incorporated. But we believe this situation will be improved with increased computing power and new approaches. In fact, the continuous-space model is more accurate, while the discrete version can effectively reduce the complexity of the problem.

Another major issue of optimal camera placement is the design of optimization algorithms which are used to solve the formulated problems. The optimal camera placement problem is NP-hard and often tackled by using BIP methods in most papers. Unfortunately, it is impractical to obtain an exact solution to the BIP problem with reasonable-size, as a result, many approximation techniques have been proposed. For example, the greedy approach [3], the greedy heuristics [4], and the semidefinite programming relaxations (SDP) [5]. Zhao et al compared the accuracy, efficiency, and scalability of a wide variety of approximate algorithms in solving the BIP optimal camera placement problems [6]. Although the above algorithms work well, they can’t handle nonlinear objectives and constraints. In these circumstances, some metaheuristic optimization algorithms appeared. Genetic Algorithms (GA), as one of the most popular metaheuristic optimization algorithm, has been effectively used in many camera placement problems in [7], [8], etc. A GA mimics the process of natural evolution through three operations: selection, crossover, and mutation. Besides, Junbin Liu solved the problem by Simulated Annealing (SA), which simulates the annealing of metal, and its variants [23], Chrysostomou and Gasteratos handle it with variants of the Artificial Bee Colony (ABC) algorithms inspired by the foraging behavior of bees [24]. In particular, PSO has widely used in this domain. Inspired by the social behaviors of bird flocks when searching for food, Kennedy and Eberhart proposed the PSO algorithm [9]. On the basis of observing the behavior of animal cluster, the PSO makes use of the information sharing of individuals in the group to generate the evolution process from disorder to order in the search space, so as to obtain the optimal solution. The convergence of the PSO has been proved: when the number of iterations tends to infinity, the probability of finding the optimal solution is 1. To get the optimal camera placement, Morsly et al. proposed a BPSO-Inspired Probability (PSO-IP) algorithm to extend the standard binary PSO by probabilistically updating the value of the velocity of a particle according to the information sharing mechanism of PSO [25]. Xu et al. also proposed three variations of the PSO to handle the constraints, especially the moving distance limitation, in camera network [26]. The standard PSO has a shortcoming of premature convergence. To fix this drawback, we introduced the resampling technique, which has been widely used in Particle Filtering (PF), and proposed a novel variation of the PSO, named Resampling Particle Swarm Optimization (RPSO), in our previous work. Indeed, the RPSO algorithm has been successfully applied to the coverage control of sensor networks [27]–[29], and the virtual resource allocation in cloud computing [30].

III. PROBLEM DEFINITION

There are three subproblems need to be identified before defining the camera placement problem: camera model, environment model, and visibility algorithm. So in this section, we introduce the camera model used in this paper at first. We consider three types of cameras, and we focus on 2D-planar regions, thus the field of view of camera is represented by a sector or circle. Then, the environment is discretized into many square grids, which are represented as pixels in the image. On this basis, the visibility algorithm based on the Bresenham algorithm is proposed to determine whether a grid (pixel) is covered by the camera network and then calculate the coverage of the whole camera network. Finally, the mathematical formulations of the two kinds of problems that we are most concerned about are given.

A. CAMERA MODEL

Camera network, in general, is required to cover a region of interest, it is necessary to provide the camera coverage model first. Scholars have done a lot of research on camera model. In order to facilitate the understanding of the subsequent contents, we briefly describe the camera model commonly used in optimal camera placement problem in this section. There are three crucial parameters associated with the camera coverage model:

- **Field of View (FoV)**. The FoV of a camera is usually a rectangular pyramid region in which objects can be projected onto the image plane. The apex of the pyramid is located at the optical center of the camera. Besides, the horizontal FoV angle and the vertical FoV angle, are used to describe the size of the pyramid. In this paper, we focus on the planar regions for reasonable simplification, so just consider the horizontal FoV angle, which can be calculated as \( \alpha = 2\tan^{-1}(\frac{w}{f}) \), where \( w \) is the width of the image plane and \( f \) is the focal length.
- **Depth of Field (DoF)**. The DoF of a camera is a range of distances within which objects can be clearly imaged
on the image plane. The aperture, lens and distance from the subject are important factors affecting the DoF. The DoF if one of the factors limiting the viewing distance, at which an object is visible. For most surveillance cameras, the DoF can be adjusted by changing the focal length, so it is not the bottleneck of viewing distance.

- **Pixel Resolution (PR).** When an object is imaged in an image plane, it will occupy several pixels. The PR of a camera is defined as the ratio between the number of pixels and the object’s real size. For many computer vision tasks, there are minimum resolution requirements, which directly limits the camera’s viewing distance, such as face recognition. And the maximum viewing distance can be calculated as 
  \[ r = \frac{f \times h_o}{n_p \times h_p}, \]
  where \( f \) is the focal length, \( h_o \) is real size of an abject, \( n_p \) is the number of pixels imaged from the object, and \( h_p \) is the real size of one pixel.

In addition, other factors may also affect the camera coverage model, such as perspective distortion, occlusion, and camera’s pose.

In surveillance system, three types of camera are usually applied.

- **Static Perspective Camera** with a fixed position and orientation.
- **Pan-Tilt-Zoom (PTZ) Camera,** which is a extend of Static Perspective Camera with adjustable orientation.
- **Omnidirectional Camera** with a \( 2\pi \) horizontal FoV angle.

The field layout of camera can be represented as a circle, isosceles triangle, sector, or trapezoid [31]. In this study, we use a sector to represent the coverage of Static Perspective Camera and PTZ Camera, and a circle for Omnidirectional Camera, shown in Fig. 1. Camera parameters can be divided into two categories, one is the geometric parameters including 2D-position \((x, y)\) and orientation \(\theta\) (The orientation of the omnidirectional camera is fixed at 0), the other is intrinsic parameters, including horizontal FoV angle \(\alpha\) (For an omnidirectional camera, \(\alpha \equiv 2\pi\)) and maximum viewing distance \(r\).

Intrinsic parameters (i.e. \(\alpha \) and \(r\)) can be determined by the camera type \(t\), therefore, the camera coverage model can be illustrated by a vector \(c = (x, y, \theta, t)\), where \(x, y, \theta\) and \(t\) are continuous while \(t\) is discrete.

### B. ENVIRONMENT MODEL

For camera network, an environmental model is necessary because it is the basis for subsequent analysis. A complete environmental model should include the following information: the shape of the region, the shape and position of obstacles in the region, the characteristics of the objects to be observed, the importance of different subregions, communication interference and so on. However, if all factors are taken into account, the environmental model will be very complex and the amount of calculation will be huge. In order to reduce the complexity of the problem, the researchers often simplify the environment model to some extent [12], [32]. This article is no exception. First, we establish the environment model in the 2D floor-plan, which is vastly simpler than the full 3D space, without loss of generality. Secondly, we discretize the target region into the same square grid, which is convenient for visibility estimation and coverage calculation. In addition, we consider the occlusion of fixed obstacles to the line of sight in the target area, but do not consider the movement of obstacles and mutual occlusion of the observed objects. Meanwhile, we assume that the importance of each position is the same.

### C. VISIBILITY ALGORITHM

Given a 2D region with obstacles, the goal of visibility estimation is to find a subregion containing all visible points from a given camera position, e.g., Fig. 2. In the discrete environment model, the target region can be represented by a 2D Boolean matrix \(Re\). Each element of \(Re\) corresponds to a grid, \(Re[i][j] = 0\) means that the grid at \(ith\) row and \(jth\) column needs to be monitored, instead, \(Re[i][j] = 1\) means that this grid represents an obstacle. In the same way, visibility can also be represented by a 2D Boolean matrix \(V\). If the grid at \(ith\) row and \(jth\) column is invisible, \(V[i][j] = 0\), otherwise, \(V[i][j] = 1\). So the visibility estimation problem can be defined as: Given a \(Re\) and a camera \(c = (x, y, \theta, t)\), then determine the visibility matrix \(V\) for this camera.

A grid is considered visible if it meets the following two conditions: 1) the grid falls within the camera’s coverage area, a sector for PTZ camera and circle for Omnidirectional camera, mentioned in section III-A; 2) the grid is unobstructed.
Algorithm 1 Visibility Algorithm

1: procedure GetVisibilityMatrix(Re, c)
2: \hspace{1em} Initialize the visibility matrix V with the same size of Re. Let v = 0, \forall v \in V.
3: \hspace{1em} for all re \in Re do
4: \hspace{2em} if re.value = 0 & re.visited = 0 then
5: \hspace{3em} if Inequalities in (1) and (2) hold then
6: \hspace{4em} let vm = 1
7: \hspace{4em} let x0 = c.x, y0 = c.y
8: \hspace{4em} let x1 = re.x, y1 = re.y
9: \hspace{4em} let dx = abs(x1 - x0), dy = abs(y1 - y0)
10: \hspace{4em} let sx = x0 < x1 \hspace{1em} ? \hspace{1em} 1 : -1
11: \hspace{4em} let sy = y0 < y1 \hspace{1em} ? \hspace{1em} 1 : -1
12: \hspace{4em} let e1 = (dx > dy \hspace{1em} ? \hspace{1em} dx : -dx)/2
13: \hspace{4em} loop
14: \hspace{5em} if Re[y0][x0].value = 0 then
15: \hspace{6em} if vm = 1 then
16: \hspace{7em} V[y0][x0] = 1
17: \hspace{6em} end if
18: \hspace{5em} else
19: \hspace{6em} vm = 0
20: \hspace{5em} end if
21: \hspace{4em} Re[y0][x0].visited  = 1
22: \hspace{4em} if x0 = x1 & y0 = y1 then
23: \hspace{5em} break
24: \hspace{4em} end if
25: \hspace{4em} let e2 = e1
26: \hspace{4em} if e2 > -dx then
27: \hspace{5em} e1 = e1 + dy, x0 = x0 + sx
28: \hspace{5em} end if
29: \hspace{4em} if e2 > -dx then
30: \hspace{5em} e1 = e1 + dx, y0 = y0 + sy
31: \hspace{5em} end if
32: \hspace{4em} end loop
33: \hspace{4em} end if
34: \hspace{4em} re.visited = 1
35: \hspace{4em} end if
36: \hspace{1em} end for
37: \hspace{1em} return V
38: end procedure

The first condition is met if
\[
\sqrt{(x_g - x)^2 + (y_g - y)^2} \leq r \tag{1}
\]
and
\[
\frac{(x_g - x)\cos(\theta) + (y_g - y)\sin(\theta)}{\sqrt{(x_g - x)^2 + (y_g - y)^2}} \geq \cos(\alpha/2) \tag{2}
\]
where \((x_g, y_g)\) is the grid position, \((x, y)\) is the camera position, \(\theta\) is the camera orientation, \(\alpha\) is the horizontal FoV angle, \(r\) is the maximum viewing distance. The Bresenham Algorithm is used for the second condition. As shown in Fig. 3, draw a line from the camera position (green grid) to the target position (yellow grid), the passing grids are visible (blue grids) before encountering obstacles (black grids), once crossing an obstacle, the following grids are all invisible (red grids). Finally, we present the visibility algorithm as Algorithm 1.

Define a camera network as a set of cameras \(C = \{c_i, i = 1, \ldots, M\}\), where \(M\) is the number of cameras in the network. The visible area of each camera can be calculated by the Visibility Algorithm, corresponding to a set of visibility matrices \(V = \{V_i, i = 1, \ldots, M\}\). Then, the coverage of the network is:
\[
P_c = \frac{\sum_{i=1}^{N_1} \sum_{j=1}^{N_2} (1 - \prod_{k=1}^{M} (1 - V_k[i][j]))}{\sum_{i=1}^{N_1} \sum_{j=1}^{N_2} (1 - Re[i][j])} \tag{3}
\]
where, \(N_1\) and \(N_2\) is the number of rows and columns of \(Re\) (or \(V_i\)), respectively.

D. FORMULATION
In this paper, we focus on the task of effectively monitoring a given region, and the most concerned indicator is the coverage and cost of the camera network, so we define two optimization problems.
Problem 1: Minimum-cost Problem: given a floor-plan \( R \) and the required coverage \( P_{\text{min}} \), find a camera set \( C \) minimizing the cost of the camera network subjecting to the coverage constraint:

\[
\arg \min_C G(C) \quad \text{s.t.} \quad F(C, R) \geq P_{\text{min}} \tag{4}
\]

where \( F(\bullet) \) is the coverage function, \( G(\bullet) \) is the cost function.

Problem 2: Maximum-coverage Problem: given a floor-plan \( R \) and the number of cameras \( M \), find the parameters of all cameras \( C = [c_i|i = 1, \ldots, M, c_i = (x_i, y_i, \theta_i, t_i)] \) to maximum the coverage of the camera network \( P_C \):

\[
\arg \max_C F(C, R) \tag{5}
\]

where \( F(\bullet) \) is the coverage function.

IV. LATIN HYPERCUBE RESAMPLING PARTICLE SWARM OPTIMIZATION

The PSO algorithm is a kind of swarm intelligence algorithm, which imitates the foraging process of a bird flock. Specifically, the search space of optimization problem is regarded as the activity space of birds, and particles are equivalent to birds. Particles move in the search space and share information with each other, so as to guide all of them move toward the optimal region. The core of the PSO is to determine the velocity of each particle:

\[
v_i(t + 1) = w v_i(t) + c_1 r_1(p_i - x_i(t)) + c_2 r_2(p_g - x_i(t)) \tag{6}
\]

where \( x_i(t) \) is the position of the \( i \)th particle at time \( t \), \( v_i(t) \) is the velocity of the \( i \)th particle at time \( t \), \( p_i \) is the optimal position of which the \( i \)th particle have got, \( p_g \) is the optimal position of the whole population, \( r_1 \) and \( r_2 \) are random factor in \((0, 1)\), \( c_1 \) and \( c_2 \) are constants which indicate the effect of \( p_i \) and \( p_g \) on velocity respectively, and \( w \) is inertia weight. With the velocity update formula, we can obtain the new position by (7).

\[
x_i(t + 1) = x_i(t) + v_i(t + 1) \tag{7}
\]

The flow of the PSO is presented in Algorithm 2. In summary, the PSO algorithm repeats two steps: finding the best position from the particle swarm, and moving the particle under the guidance of the optimal position.

A. RESAMPLING PARTICLE SWARM OPTIMIZATION

The PSO algorithm has a potential defect of premature, many researchers have proposed improved versions of the PSO to address this flaw. Wang et al also proposed a Resampling-PSO (RPSO) algorithm by combining with the resampling technique in particle filter [27]. In particle filter, the mean value is mostly determined by the particles with high weights, it means that the particles with extremely low weights almost have no effect to the solving process, but with a lot of time wasted. It is similar in the PSO algorithm. To fix this, a resampling operation is introduced as follow:

Algorithm 2 PSO Algorithm

1. Set a weight for each particle by (8) and (9).

\[
q_i = \frac{1}{\sqrt{2\pi \sigma}} \exp \left( - \frac{(F(x_i) - b_{x_i})^2}{2\sigma} \right) \tag{8}
\]

\[
Q_i = \frac{q_i}{\sum_{i=1}^{N} q_i} \tag{9}
\]

where \( Q_i \) is the \( i \)th particle’s normalized weight, \( F(\bullet) \) is the fitness function, \( b_{x_i} \) is the current global optimal value, \( \sigma \) is the variance of \([F(x_i) - b_{x_i}, i = 1, \ldots, N] \).

2. Give a threshold value \( q_t \in (0, 1) \). For each particle, if \( Q_i < q_t \), then update the position and velocity by (10) and (11) respectively.

\[
x_i(t) = \bar{x}_i(t) \tag{10}
\]

\[
v_i(t) = \frac{T + t}{2T} \bar{v}_i(t) + \frac{T - t}{2T} v_i(t) \tag{11}
\]

where, \( \bar{x}_i(t) \) and \( \bar{v}_i(t) \) are generate randomly.

The main function of the resampling operation is to eliminate particles that perform poorly in the population when the particles are brought together and replace them with new ones. This has two advantages: first, new particles are generated and the diversity of the population is increased, which is conducive to wider area search; second, the particles with low fitness are eliminated, thus improving the efficiency. The flow of the RPSO is similar to the PSO as shown in Algorithm 3. The only different is that there is a resampling operation before updating the position and velocity vectors in the RPSO. Comparing with the PSO, the advantages of the RPSO are mainly reflected in the following aspects: improving the premature defects, avoiding the waste of computing resources, and improving the efficiency to a certain extent [28]. And the RPSO algorithm has been successfully and efficiently applied to the coverage control of sensor networks, and the virtual resource allocation in cloud computing.

B. LATIN HYPERCUBE SAMPLING IN RPSO

In the process of initializing and resampling, the positions and velocities are generated randomly. Generally, the simple random sampling is used in the RPSO, but now, we prefer to replace it with Latin hypercube sampling (LHS).
**Algorithm 3 RPSO Algorithm**

1. Set algorithm parameters: the size of the swarm \( N \), the maximum iterations \( T \), the inertia weight \( w \), and the accelerated factors \( c_1, c_2 \). Give a criterion \( \xi \) to determine whether resampling is required. Initialize the particles’ position vectors \( x_1, \cdots, x_N \) with uniform random variables. Initialize the particles’ velocity vectors \( v_1, \cdots, v_N \) with uniform random variables. Set \( t := 0 \).

2. If the condition \( \xi \) if met, perform the resampling operation.

3. Find the optimal position of each particle \( p_1, \cdots, p_N \). Find the optimal position of the whole population \( p_g \). Update the velocity vector of each particle by (6), then update the position vector of each particle by (7). Let \( t := t + 1 \).

4. If \( t > T \), return current \( p_g \) as the final solution. Otherwise, go to Step 2.

*The criterion \( \xi \) could be a variance of the distances between all the positions and their center, for example.*

The LHS is proposed by McKay et al. in 1979 [33], and it is perhaps the most widely used random sampling method for Monte Carlo-based uncertainty quantification and reliability analysis, employed in nearly every field of computational science, engineering, and mathematics [34]. Actually, the LHS can be regarded as a kind of stratified sampling, which makes the sample structure more close to the real data and has a higher estimated accuracy compared with the simple random sampling. In statistical sampling, a Latin square matrix contains only one sample in per row or per column. The Latin hypercube is the extension of the Latin square matrix in multi-dimensions in which each hyperplane contains at most one sample. Assume \( S \) is \( N \)-dimensional sample space, and \( s_i \) is the \( i \)th dimension of \( S \). Take \( M \) samples from \( S \) using the LHS method, the steps are shown in Algorithm 4.

In this paper, inspired by resampling and LHS, we proposed the LH-RPSO algorithm, the pseudocode of LH-RPSO is shown in Algorithm 5. We believe that the resampling and LHS can help the PSO performs better, the main reasons are as follow: first, the resampling operation is helpful to maintain the diversity of the population and reduce the probability of the algorithm falling into local optimum; second, resampling operation eliminates lagging particles and saves computational resources; finally, the initial population obtained by LHS is more uniform and representative.

**Algorithm 4 Latin Hypercube Sampling**

1. **procedure** LHS(\( S, M \))
2. 
3.  
4. 
5.  
6. 
7. 
8. 
9. 
10. 
11. 
12. 
13. 
14. 
15. 
16. 
17. 
18. 
19. 

**Algorithm 5 LH-RPSO algorithm**

1. Set algorithm parameters: the size of the swarm \( N \), the maximum iterations \( T \), the inertia weight \( w \), and the accelerated factors \( c_1, c_2 \). Give a criterion \( \xi \) to determine whether resampling is required. Initialize the particles’ position vectors \( (x_1, \cdots, x_N) = \text{LHS}(S, N) \). Initialize the particles’ velocity vectors \( (v_1, \cdots, v_N) = \text{LHS}(S, N) \). Set \( t := 0 \).

2. If the condition \( \xi \) if met, perform the resampling operation.

3. Find the optimal position of each particle \( p_1, \cdots, p_N \). Find the optimal position of the whole population \( p_g \). Update the velocity vector of each particle by (6), then update the position vector of each particle by (7). Let \( t := t + 1 \).

4. If \( t > T \), return current \( p_g \) as the final solution. Otherwise, go to Step 2.

*The criterion \( \xi \) could be a variance of the distances between all the positions and their center, for example.*

In this section, in order to verify the effectiveness of the LH-RPSO algorithm in the domain of camera placement problem, we did a simulation with a real-world map of a campus (Fig. 4). This image is 782 pixels width and 732 pixels height in which one pixel corresponds to a 0.455m \( \times \) 0.455m grid, and the area required to be monitored is marked in red lines. Next, we preprocessed the above image, including rotation, clipping and digitization as shown in Fig. 5. The new image is 681 pixels width and 591 pixels height, corresponding to an area of about 83322m\(^2\). And in this picture, the black area represents obstacles such as buildings, on the contrary, the white area represents the region required to be monitored. Besides, this simulation selected four types of cameras, and the detailed camera intrinsic parameters were listed in Tab. 1.

All the experiments were carried out in MATLAB R2016a and run on a server with 2.2 GHz Intel Core i7-8750H CPU and 16 GB RAM.
First, we focus on the Problem 1, the Minimum-cost Problem. That is, we want to achieve the required coverage at the lowest cost. As can be seen from Fig.4 and Fig.5, this is a large area with a complex topology. Thus, it is a complicated optimization problem with a large number of design variables, and complex objective function and constrains. In order to obtain optimization results within acceptable time consumption, we simplified the problem by sampling. And in particular, we sampled a set of alternative cameras in the design space, and then selected a subset from them as the optimal solution to the above problem. In this case we preselected 200 cameras using Latin hypercube sampling, and the total coverage reached 99.27%, the overlay is shown in Fig.6.

Then the optimization problem is described as: Select a subset from the 200 cameras to minimize the total price, meeting the constraint that the coverage is greater than or equal to 80%.

\[
\text{argmin} \sum_{i=1}^{M} e_i \phi_i \\
\text{s.t.} \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} (1 - \prod_{k=i}^{M} (1 - e_i V_k[i][j])) \geq 0.8 \quad (12)
\]

where, \( e_i = 1 \) means the \( i \)th camera is selected, on the contrary, \( e_i = 0 \) means not. Besides, \( \phi_i \) is the price of the \( i \)th camera. Other than that, other symbols have the same meaning as above. Then the PSO, RPSO and LH-RPSO are used to solve this optimization problem. For all the three algorithm, we set the parameters as: \( N = 30, T = 200 \), where \( N \) is the population size and \( T \) is the maximum number of iterations. We repeated it 10 times. The experiment results are recorded in Tab.2. From the table, we can see that the averages of the three algorithms(PSO, RPSO, and LH-RPSO) are $697.5, $687 and $685 respectively. According the results, the LH-RPSO performs better than the PSO and the RPSO. And compared with the PSO, the LH-RPSO reduces the cost by 1.79%, but the advantages of LH-RPSO are not significant compared with the ROSO, only 0.29%. In addition, according to the table, the LH-RPSO performs best in the experiments 2, 3, 4, 6, 7 and 9. In other words, the LH-RPSO performs best in more than half of the experiments. It means that the LH-RPSO is more stable and has a higher probability of obtaining the optimal solution compared with the general PSO and the RPSO. Fig.7 shows the final overlays of one
of the experiments. In general, the cameras distribution in the solution of the LH-RPSO is more uniform. Thus, the above results demonstrate that the LH-RPSO is effective in solving the camera placement problem and has better performance than PSO and RPSO.

### B. RESULTS TO THE MAXIMUM-COVERAGE PROBLEM

By solving the Problem 1, we had obtained a solution with a minimum cost and a coverage of not less than 80%. Through observation, we found that it is possible to further improve the above solution. In other words, we may get higher coverage by adjusting the position and posture of each camera, with keeping the number of cameras and the inherent properties of each camera the same. This is the Maximum-coverage Problem mentioned in the Section III-D. And here, we formulated it as (13).

\[
\text{arg max}_C \frac{\sum_{i=1}^{N_1} \sum_{j=1}^{N_2} (1 - \prod_{k=1}^{M} (1 - V_k[i][j]))}{\sum_{i=1}^{N_1} \sum_{j=1}^{N_2} (1 - Re[i][j])}
\]

\[C = \{c_i | i = 1, \ldots, M, c_i = (x_i, y_i, \theta_i, t_i)\}\]

where all the symbols have the same meaning as before.

Before solving the Problem 2, we selected a solution of Problem 1 as the initial conditions. In this solution, 16 cameras are needed (Tab.3), the coverage is 80.41% (Fig.7c), and the cost is $645. Then the PSO, RPSO and LH-RPSO are used to solve this optimization problem. For all the three algorithms, we set the parameters as: \(N = 30, T = 100\). We repeated it 10 times, the increments of coverage are shown in Tab.4.

According to the results in this table, the averages increments of coverage of the three algorithms (PSO, RPSO, and LH-RPSO) are 7.78%, 7.86% and 8.35% respectively. And the result of the LH-RPSO is 7.32% higher than that of the PSO, and is 6.23% higher than that of the RPSO. Besides, the LH-RPSO performs best in 7 experiments of the total 10 experiments. It illustrates that the LH-RPSO is more robust and efficient than the PSO and the RPSO. In the end, Fig.8 shows a new final overlay of the camera network. Compared with the Fig.7c, the coverage has increased significantly.

The above two examples can fully prove the effectiveness of LH-RPSO algorithm for the camera placement problems.
VI. CONCLUSION AND FUTURE WORK
The camera placement problem has very important significance in IoT and computer vision tasks. An efficient network layout scheme can better support the subsequent tasks such as recognition and tracking. The contributions of this study are as follows:

- In this paper, we introduced the camera model, environment model, and a visibility judgment method based on a line drawing algorithm, and then gave the two most important formulation of camera network deployment: the Minimum-cost Problem and the Maximum-coverage problem.

- The RPSO is an optimization algorithm that combines resampling technology in particle filtering and the PSO. The RPSO can solve the above optimization problems and has been proved to have better performance than the standard PSO, in order to improve its performance again, we introduced Latin hypercube sampling and proposed the LH-RPSO algorithm.

- To test the performance of the algorithm, we selected a real-world campus as the target area to deploy the camera network. The process consisted of two steps. In the first step, we solved the Minimum-cost problem and had found the solution with a minimum cost satisfying the coverage constraint. In the second step, we solved the Maximum-coverage problem. That is, on the basis of the solution of the first step, we kept the number and inherent properties of cameras unchanged, and found the layout plan with higher coverage by adjusting the camera position and posture. Experimental results of both steps have shown that the LH-RPSO has higher performance than the PSO and the RPSO, which also fully demonstrated that the LH-RPSO can be used in practical large-scale camera placement problems.

In the future work, we will focus on two areas. The first one is to use more realistic and complex mathematical models which can be used in IoT. Second, we will improve the optimization algorithm unceasingly, not only the accuracy of the algorithm, but also the speed. Besides, the influence of each key parameter on the algorithm performance will be analyzed, so as to provide the parameter configuration strategy for different problems.

REFERENCES
[1] M. Z. A. Bhuiyan, G. Wang, J. Cao, and J. Wu, “Sensor placement with multiple objectives for structural health monitoring,” ACM Trans. Sensor Netw., vol. 10, no. 4, p. 68, Jun. 2014.
[2] S. H. Islam, P. Vijayakumar, M. Z. A. Bhuiyan, R. Amin, M. R. Varun, and B. Balusamy, “A provably secure three-factor session initiation protocol for multimedia big data communications,” IEEE Internet Things J., vol. 5, no. 5, pp. 3408–3418, Oct. 2018.
[3] M. Al Hasan, K. K. Ramachandran, and J. E. Mitchell, “Optimal placement of stereo sensors,” Optim. Lett., vol. 2, no. 1, pp. 99–111, Jan. 2008.
[4] J. Zhao, S.-C. Cheung, and T. Nguyen, “Optimal visual sensor network configuration,” in Multi-Camera Networks. Salt Lake City, UT, USA: Academic, 2009, ch. 6, pp. 139–141.
[5] J. Zhao, “Camera planning and fusion in a heterogeneous camera network,” Ph.D. dissertation, College Eng., Univ. Kentucky, Lexington, KY, USA, 2011.
[30] X. Wang, H. Gu, and Y. Yue, “The optimization of virtual resource allocation in cloud computing based on RBPSO,” *Concurrency Comput., Pract. Exper.*, to be published. doi: 10.1002/cpe.5113.

[31] A. A. Altahir, V. S. Asirvadam, N. H. Hamid, P. Sebastian, N. Saad, R. Ibrahim, and S. C. Dass, “Modeling multicamera coverage for placement optimization,” *IEEE Sensors Lett.*, vol. 1, no. 6, Dec. 2017, Art. no. 5500604.

[32] X. Yang, H. Li, T. Huang, X. Zhai, F. Wang, and C. Wang, “Computer-aided optimization of surveillance cameras placement on construction sites,” *Comput.-Aided Civil Infrastruct. Eng.*, vol. 33, no. 12, pp. 1110–1126, Dec. 2018.

[33] M. D. McKay, R. J. Beckman, and W. J. Conover, “Comparison of three methods for selecting values of input variables in the analysis of output from a computer code,” *Technometrics*, vol. 21, no. 2, pp. 239–245, 1979.

[34] M. D. Shields and J. Zhang, “The generalization of Latin hypercube sampling,” *Rel. Eng. Syst. Saf.*, vol. 148, pp. 96–108, Apr. 2016.

**XIAOHUI WANG** received the B.S. and Ph.D. degrees from Beihang University, Beijing, in 2001 and 2009, respectively. She is currently an Assistant Professor with the Department of Spacecraft Design, School of Astronautics, Beijing University of Aeronautics and Astronautics. Her research interests include optimization theory and methods, and complex system optimization.

**HAO ZHANG** received the B.S. degree from Beihang University, Beijing, in 2016, where he is currently pursuing the master’s degree with the School of Astronautics. His research interests include optimization theory and mathematical methods.

**HAORAN GU** received the B.S. degree from Beihang University, Beijing, in 2017, where he is currently pursuing the master’s degree with the School of Astronautics. His research interests include optimization algorithms and methods in the field of wireless sensor networks, as well as complex system optimization.

***