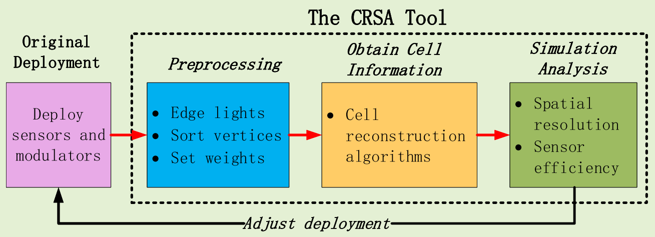
A Cell Reconstruction Tool to Deploy Binary Pyroelectric Sensor Arrays

Longxiang Luo, Yang Xiao, *Senior Member, IEEE*, Wei Liang, *Senior Member, IEEE*, and Meng Zheng, *Member, IEEE*

***Abstract*—Due to the advantages of being low cost and low power consumption, pyroelectric infrared sensors are widely employed in many applications, such as target location and tracking. In such applications, reference structure which is used to modulate views of binary sensors can help improve spatial resolutions by segmenting monitoring spaces into many cells which are identified by states (called signatures). A spatial resolution determines localization or tracking accuracy, which depends on the sizes, the shapes, and the signatures of the cells, and is drastically impacted by deployment of sensors and reference structure. However, in order to obtain a better deployment formation and a better spatial resolution, researchers have to spend lots of time to deploy and measure the deployment results of cells (such as their locations, sizes, shapes, and signatures), i.e., a huge amount of measurements which cost time and money in practice. Hence, in this paper, we propose a tool to visually and efficiently represent and analyze the deployment formation and spatial resolution. Generally, the location of a cell is represented by a list of vertices scattered in a cartesian coordinate system and can help researchers get its size, shape, and signature. A cell reconstruction algorithm is proposed to reconstruct cells generated by the deployment formation in which lists of vertices are generated by regarding cells as shortest path rings. We further provide some theoretical analysis and computational complexity analysis of the proposed algorithm. We conduct experiments using our tool to study sensor efficiency and spatial resolution easily and effectively.**



***Index Terms*—Pyroelectric infrared (PIR) sensor array, reference structure, cell and signature, shortest path ring.**

Manuscript received October 2, 2019; accepted October 20, 2019. Date of publication October 29, 2019; date of current version January 24, 2020. This work was supported in part by the National Key Research and Development Program of China under Grant 2017YFE0101300, in part by the National Natural Science Foundation of China under Grant 61673371 and Grant 71661147005, in part by the International Partnership Program of Chinese Academy of Sciences under Grant 173321KYSB20180020, and in part by the Liaoning Provincial Natural Science Foundation of China under Grant 2019-YQ-09. The associate editor coordinating the review of this article and approving it for publication was Dr. Cheng-Ta Chiang. *(Corresponding authors: Yang Xiao; Wei Liang.)*

L. Luo is with the Key Laboratory of Networked Control Systems, Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang 110016, China, with the State Key Laboratory of Robotics, Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang 110016, China, with the Institutes for Robotics and Intelligent Manufacturing, Chinese Academy of Sciences, Shenyang 110169, China, and also with the University of Chinese Academy of Sciences, Beijing 100049, China (e-mail: luolongxiang@sia.cn).

Y. Xiao is with the Department of Computer Science, The University of Alabama, Tuscaloosa, AL 35487-0290 USA (e-mail: yangxiao@ieee.org).

W. Liang and M. Zheng are with the Key Laboratory of Networked Control System, Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang 110016, China, with the State Key Laboratory of Robotics, Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang 110016, China, and also with the Institutes for Robotics and Intelligent Manufacturing, Chinese Academy of Sciences, Shenyang 110169, China (e-mail: weiliang@sia.cn; zhengmeng_6@sia.cn).

Digital Object Identifier 10.1109/JSEN.2019.2950018

I. Introduction

SPACE monitoring is one of the most typical applications of wireless sensor networks, including object tracking, habitat monitoring, atypical behavior detection, etc [1]. Pyroelectric infrared (PIR) sensors are increasingly attracting attentions in these fields because of their low cost, low power consumption, good privacy protection, and robustness to interference from non-human infrared sources. A PIR sensor works by detecting changes of infrared radiation. When a warm object, such as a human, enters into the monitoring space of the PIR sensor, the warm object triggers a small signal fluctuation with infrared lights that it emits. This signal fluctuation, which is amplified and conditioned in the signal conditioning circuit, makes the PIR output 1; otherwise, the PIR outputs 0 [2]. However, due to the fact that binary passive sensors output either 0 or 1, PIR sensors can neither distinguish absolute locations nor identify the number of persons in a Field of Interest (FOI) [3]. It also cannot provide enough information to illustrate accurate trajectories of moving objects [4]. Reference Structure Tomography (RST) is applied to enhance sensor spatial awareness by segmenting a monitoring space into different cells. The term ‘reference structure’ is first proposed in computational imaging systems [5], [6]. An RST based system consists of more than one 3D reference structures (called modulators), placed in between the monitoring space and sensors [5], [6]. Reference structures or modulators are opaque masks made of plastic materials. They segment the monitoring space in front of them into regions (called cells) that are either visible or invisible to a sensor. Fusing the data of different sensors, researchers can obtain the motion features (such as location and trajectory) and judge behavioral states of an object rather than only judge the occurrence of the object in the monitoring space [7]. Hence, RST is said to extend the application scenarios of PIR sensor arrays to low-level spatiotemporal properties (namely: presence, count, location, and tracking) and behavioral properties (namely: pose, gait, and behavior) by modulating the sensor field of view, and creating overlapping detection cells [9].

However, RST also makes the deployment of PIR sensors much more complex. Not only do researchers have to study the number, locations, and shapes of modulators, but also they have to study the amounts, shapes, and signatures of cells generated by modulators. Furthermore, researchers have to put the sensors in places and then measure the locations, sizes, shapes, and signatures of cells. These take a lot of measurements, but may not be able to achieve a desired deployment effect. If a deployment formation (e.g., the number of sensors, the number, locations, and shapes of modulators, etc.,) is changed, people need to re-measure those. This wastes a lot of time and money. We believe that an automatical and intelligent tool (proposed in this paper) which can automatically calculate the deployment results and present the results simply and intuitively is necessary. Such a tool which works correctly and efficiently can save a lot of time, manual measurement cost, and unnecessary spending of purchasing sensors.

This paper proposes a tool to help researchers coping with the measurement problems induced by modulators. Researchers can use it to acquire deployment information without deploying sensors and reference structure in practice. For instance, they can get the vertex coordinates and signature of each cell through the tool in a computer. With the tool, researchers can adjust their deploy plans (such as, adjusting locations of sensors or modulators, changing the number of sensors, or changing the number of modulators in reference structure) and achieve better results easily and effectively. The contributions of this paper can be summarized as follows:

1) We propose and implement a tool to automatically obtain useful information, such as signatures and vertices of cells, based on the input parameters such as locations of sensors, locations of modulators, the number of sensors, and the number of modulators in reference structure, etc. A cell reconstruction algorithm is proposed to reconstruct cells generated by the deployment formation in which lists of vertices are generated by regarding cells as shortest path rings. In the proposed tool, cells are represented by cycles of vertices and constructing a vertex cycle is the same as a shortest path ring problem.

2) We further provide some theoretical analysis and computational complexity analysis of the proposed algorithm. We prove that an upper bound of computational complexity of the algorithm is $O(m^6 log_2 m)$, under the condition that each sensor has at least one modulator to modulate its view, i.e. $m \geq n$, where m and n denote the numbers of modulators and sensors, respectively.

3) We define two novel metrics as follows. Let d and u denote the discrimination degree of cells and the utilization ratio of signatures, respectively, where the discrimination degree is defined as the ratio of the number of non-repetitive signatures to the total number of cells and is used to appraise the spatial resolution; and the utilization ratio is defined as the ratio of the number of non-repetitive signatures to the maximum number of signatures in theory and is used to appraise the sensor efficiency. Such two metrics can help researchers to analyze and improve the spatial resolution and sensor efficiency of a deployment formation. Commonly, spatial resolution is considered prior to sensor efficiency since in order to get a better spatial resolution, researchers may sacrifice the sensor efficiency (i.e. may allow u to have a relative smaller value). Hence, we expect the values of d and u to be as large as possible. Experiment results show that the tool can be used to analyze spatial resolution easily and efficiently. To our knowledge, there are no metrics in the literature for these kinds of sensor arrays before this paper and most of the published papers about sensor arrays are for applications.

4) However, in our experiments using our tool, we count the actually utilized signatures to be $|N| = n^2 - n + 2$ $(n > 3)$, and find that seeking the most number of cells is not an appropriate way to segment the monitoring space. When achieving the maximum number of cells, the d value is around 0.5 in our experiments, and this means almost half of the cells are indistinguishable, where if some cells have the same signature, we call them "indistinguishable". Hence, some new algorithms should be designed to make the value of d as large as possible under the condition $d \leq 1$. Besides, the sensor efficiency approaches to 0 as the number of sensors increases. Hence, the number of sensors should be as small as possible to attain a good sensor efficiency. However, the dilemma is that the more sensors there are, the better the spatial resolution. Hence, a good way to segment monitoring space is to design new algorithms satisfying the following three conditions: 1) it generates as many cells as possible; 2) each cell has a unique signature; 3) under conditions 1) and 2), it improves sensor efficiency as much as possible.

The rest of this paper is organized as follows. Section II presents related works of this paper. Section III gives the problem definition. Section IV presents the deployment formation and preprocess and the cell reconstruction of the tool in detail. Section V provides theoretical and computation complexity analysis. Section VI provides experimental results. Finally, we conclude this paper in Section VII.

II. Related Works

Related works of RST with PIR sensors are summarized as two categories: 1) applications on human motion (tracking and recognition) [10], [11] and 2) fundamental theory research on RST and monitoring space segmentation. We focus on fundamental theory research on RST in this paper.

Preliminary RST fundamental theory studies include multi-dimensional imaging, data-efficiency sensing, direct estimation of object size, and principles and approaches in the design and usage of RST in object analysis. The authors in [5], [12] explore transformation matrixes between an object space and states of sensor arrays with RST in imaging systems to measure a small object in an object space. In [12], the authors implement Hardmard matrix (constructed by states of sensors) as transformation over a range of angular perspectives to demonstrate the feasibility of using RST to monitor object space. In [5], the authors utilize multi-angular view two-dimensional imaging generated by occlusion of RST to reconstruct a 3D target object in images. Hence, their studies can be regarded as the foundation of object activity recognition.

The authors in [13] define sensor efficiency formulation based on the number of distinguishable source states and the number of measurements required to estimate a source state and deign a two-dimensional RST with coded apertures in a pyroelectric motion tracking system to detect source motion in one of the 15 cells uniformly distributed over a $1.6m \times 1.6m$ domain using 4 pyroelectric detectors. The authors in [14] show how to predict the size of an uniform-brightness object and how to achieve complex shape recognition via determining the higher moments of the statistics of the RST measurement; they also provide a probability model between RST and the measurement states, and obtain the n^{th} order statistics of measurements and demonstrate the dependance of moments of measurement states and their variance on an object area for RST based sensors. The authors in [6] explore the potential of RST for the purpose of accelerating data acquisition and simplifying temporal correction, and present a geometric model of RST, referring to a mathematic model form of computational imaging systems, to use the characteristic functions of cells as basis functions to limit the amount of deployed sensors; they also discuss the effects of different cases of dimensions of object space and measurement space on the geometric model. The authors in [15] study a simple model consisting of geometric radiation filed propagation and opacity-based field modulation; they illustrate the physical constraints on realizable mapping by bounding the number of distinct signatures that can be realized for a particular RST geometry; they also prove a almost tight bound on the number of distinct signatures and illustrate the limits on the complexity of signature fields induced by RST and binary sensors.

The authors in [16] explore fundamental performance limits of tracking a target in a two-dimensional field of binary sensors and design algorithms that attain those limits; they propose a spatial resolution upper bound of binary sensors and use a piecewise linear approximation to estimate the true path of the target within the spatial resolution; moreover, the relative error in velocity estimation is constructed to solve the problem of preferring a path that uses a small number of segments as opposed to the one that uses a large number of segments. The authors in [17] study the optimal deployment of binary sensors by partitioning sensors into two sub-regions: omni-directional and directional static binary sensors; they compute the upper bound on the number of unique cells, which are cells having common edges with FOI; they also propose an algorithm to deploy sensors within FOI in which the number of unique cells is asymptotically equivalent to the upper bound.

The authors in [18], [19] explore that most signatures can be achieve with RST; they show that the maximum number of signatures in a sensor network with n binary sensors is 2^n and that it can be achieved by n modulators under some assumptions; furthermore, they prove that the maximum number value is subject to the number of modulators [18]; besides, they explore a series of bad sensor system deployments to be avoided, where the number of signatures are relatively small or minimum. They also study the size of cells and present a sensor system deployment where the size of cells can be controlled, no matter how small it is [19]. They do a survey of RST research history from the computational imaging system (in a mathematical way) to some representative applications and also give some fundamental theory bounds on monitoring space segmentation.

There are many benefits of using sensor arrays rather than using cameras. First, the senors are cheap and light-weight (such as \$1 each) [11]. Second, the sensors are stealth. Third, the system with the sensors can automatically record the locations, movements, and tracks of living objects, whereas cameras need very complex pattern recognition programs to identity and track living objects and still are not very accurate without human intervention. In other words, even though cameras can record the videos, it is very difficult to understand what are going on in the videos without human intervention [18].

In this paper, we attempt to eliminate the gap between theoretical studies and applications of RST by presenting results of theoretical research visually using our tool. The proposed tool can help researchers in acquiring the vertex sets of cells, shapes of cells, and signatures easily and efficiently since the number, structure, and distribution of cells are of profound interest to the design and data analysis of a sensor system. Based on our knowledge, there isn't either such an existing tool or a paper in the literature to explain how to build such a tool.

III. Problem Definition

A. Definitions and Assumptions

We focus on the problem of FOI segmentation using an arbitrary number of sensors and modulators to investigate cells of FOI and their signatures. Modulators, used to modulate the view of sensors, work as reference structure. Assume that all of the modulators are the same size and sensors have the same device parameters. FOI is a four edges polygon and these four edges of the polygon called as boundary lines of FOI. Applications of the FOI include monitoring a museum room, monitoring a government room, etc. Our proposed methods are not limited by the shape of the FOI.

As illustrated in Fig. 1, a sensor deployment region is an annular region between the two circles with two radiuses R_{si} and R_{so} and a modulator deployment region is an annular region between the two circles with two radiuses R_{mo} and R_{mi}, where 'i' in 'mi' and 'si' means the inside circle and 'o' in 'mo' and 'so' means the outside circle. Sensors and modulators are scattered inside two limited annular sensor and modulator regions, respectively, surrounding an FOI which is a field or region monitored by sensor and modulator arrays, where warm objects appear and disappear. In the sensor

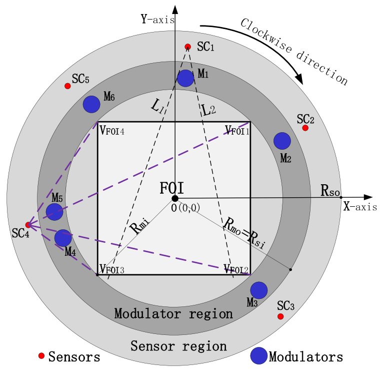


Fig. 1. Sensor and modulator deployments.

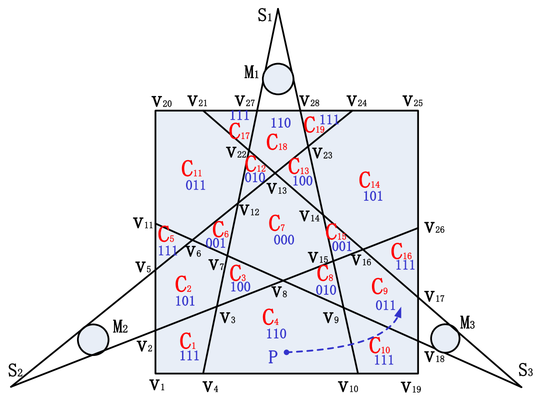


Fig. 2. Segment FOI with 3S and 3M (S: Sensor; M: Modulator).

(or modulator) region, locations of sensors (or modulators) can be adjusted to achieve required deployment results. *Edge light* is an infrared light which cuts across an FOI, is emitted by a warm object, and is tangent to a modulator. Since the distance between a sensor and its modulator is very small, the infrared lights emitted by a warm object in the subregion between two edge lights (for instance, the subregion between l_1 and l_2 shown in Fig. 1) cannot be received by the sensor SC_1. On this perspective, edge lights can be taken as bounds of the FOV of a sensor. If a warm object locates in an FOI, it emits infrared rays to every direction. Two of the infrared rays (not for the object) are tangent to a modulator and sensed by a PIR sensor behind the modulator as illustrated in Fig. 1. The straight trajectories of these two rays are edge lights. Edge lights intersect with each other and intersect with boundary lines of FOI generating *vertices*. A boundary lines of an FOI generate *vertices* too. An *edge* is defined as two vertices, which are located on the same edge light or boundary line of FOI, and no vertices located between them. Such two vertices are two *endvertices* of the edge. Consequently, an edge light may contain more than one edges. Two vertices are *neighbors*, if they are two endvertices of an edge. Two distinct edges are *adjacent* if they have an endvertex in common. In this paper, vertex and endvertex will be used indiscriminately.

A subregion of an FOI without any edge light cutting across it is defined as a *cell*. As shown in Fig. 2, the pentagon where a point P locates is a cell, where *point* is used to describe a certain location inside a cell. Similar to *cycle* in graph theory [20], a cell might be signified by its cyclic sequence of vertices. There are two methods to represent a cell: 1) using its *edge set*, e.g., $C = \{e_1, e_2, e_3, e_4\}$; 2) using its *vertex cycle*, e.g., $C = v_1v_2v_3v_4v_1$. Hence, the problem of obtaining cells in an FOI is to acquire their vertices. Our proposed cell reconstruction method to acquire vertices of each cell. Based on this method, the tool is proposed.

Signature is used to code and identify cells [18], [19]. For any point P in an FOI in Fig. 2, if it is visible to a sensor, the sensor will mark as 1; otherwise, the sensor will mark 0. The value of point P marked by sensor s_i is denoted as $f_i(P)$. We have $f_i(P) = 1$ if P is visible to s_i and $f_i(P) = 0$ if P is invisible to s_i. Base on [18], [19], we have a formal definition of signature as follows.

Definition 1: The *signature* of a cell is a binary sequence denotes as $f(P)$, which is a concatenation of n digits of $f_i(P)$, $i = 1 \cdots n$ as follows:

$$f(P) = f_n(P)f_{n-1}(P)\cdots f_1(P), \tag{1}$$

where $f_i(P)$ is the value of the i-th digit, and n is the number of sensors deployed around the FOI.

As illustrated in Fig. 2, point P in the FOI is visible to sensors s_2 and s_3, but invisible to sensor s_1. Consequently, we have $f_3(P) = 1$, $f_2(P) = 1$, and $f_1(P) = 0$. From equation (1), the signature of P is $f(P) = 110$. We also observe that the FOI is segmented into 19 cells as indicated as $C_1, C_2, \ldots, C_{19}$ in Fig. 2 and $|C| = 19$. In Fig. 2, the cells form a set of 8 signatures denoted as $N = \{000, 001, 010, 100, 011, 101, 110, 111\}$ and $|N| = 8$. Each cell has a signature $S_i = f(P) \in N$, $S = \{S_1, S_2 \ldots S_{19}\}$, and $|S| = 19$. Since there are more cells than signatures, different cells may have the same signature.

B. Problem Statement

Spatial resolution and sensor efficiency are two predominant problems in a surveillance system with binary sensor arrays. Spatial resolution determines the accuracy of object tracking and localization. Sensor efficiency is used to avoid waste of sensors. In this paper, we attempt to design a tool to help researchers visually and efficiently observe experiment results and study spatial resolution and sensor efficiency. The proposed tool is to acquire C and S with the following 3 major questions.

Problem 1: how to acquire the data generated by a deployment formation is our first problem. Generally, the data contains cell vertices, cell signatures, cell shapes, and cell sizes. The predominant data are cell vertices, with which we can get cell shapes and cell sizes easily. Moreover, cell signatures are another important problem, which not only need cell vertices, but also need the data of deployment formation. Hence, this paper focus on getting cell vertices and cell signatures.

Problem 2: Since the maximum number of cells is $2n^2 + 1$ (under the condition that each sensor is modulated by only one modulator) [17], how to get all the cell vertices of each cell quickly and correctly is our second problem. Since reconstructing cells in the FOI is a combinatorial problem and as the increment of n and m, the problem becomes much more complex, we aim to completing the cell reconstruction work as fast as possible in polynomial time. In order to solve

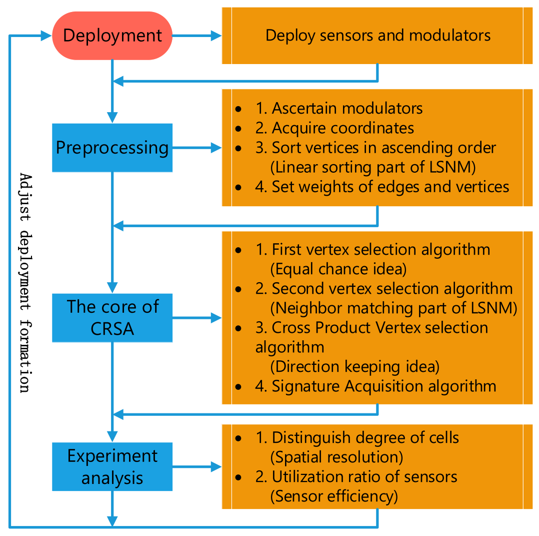

Deployment
Deploy sensors and modulators
Preprocessing
1. Ascertain modulators
2. Acquire coordinates
3. Sort vertices in ascending order (Linear sorting part of LSNM)
4. Set weights of edges and vertices
The core of CRSA
1. First vertex selection algorithm (Equal chance idea)
2. Second vertex selection algorithm (Neighbor matching part of LSNM)
3. Cross Product Vertex selection algorithm (Direction keeping idea)
4. Signature Acquisition algorithm
Experiment analysis
1. Distinguish degree of cells (Spatial resolution)
2. Utilization ratio of sensors (Sensor efficiency)
Adjust deployment formation

Fig. 3. The function structure of utilizing the CRSA Tool.

this problem, the computation complexity of the designed algorithm should be achieved with its lower bound.

Problem 3: We aim to studying spatial resolution and sensor efficiency of deployment formation. Then, how to measure the deployment results and guide researchers to get better deployment formation is our third problem. Since deployment formation using RST segments a monitoring space into many different cells in shapes and sizes, conventional spatial resolution and sensor efficiency calculation formulations are unapplicable. New metrics to judge or calculate spatial resolution and sensor efficiency should be designed.

IV. The Tool

Deployment formation is to design numbers and locations of sensors and modulators based on a monitoring purpose (intrusion detection, localization, tracking, etc.). When a deployment formation is designed, a cell reconstruction and signature acquisition (CRSA) tool is used to analyze and adjust deployment formation with experiments. As illustrated in Fig. 3 [1], the CRSA tool consists of four parts: deployment, preprocessing, the core of CRSA, and experiment analysis.

A. Step 1: Deployment

Deployment is to deploy sensors and modulators at some certain places. Sometimes, shapes of modulators are designed differently [7], [8], [10], [11]. Let n, m, and R_m denote the number of sensors, the number of modulators, and the radius of modulators, respectively. Sensors and modulators are assumed to be points and disks, respectively, and are placed on the n equal division points of circles with radiuses R_{si} and R_{mi}, respectively, as illustrated in Fig. 1. Coordinates of sensors and modulators are stored in sets $SC = \{SC_1, SC_2 \ldots SC_n\}$ and $M = \{M_1, M_2 \ldots M_m\}$, respectively.

[1]Note that some of the details of the figure will be explained later.

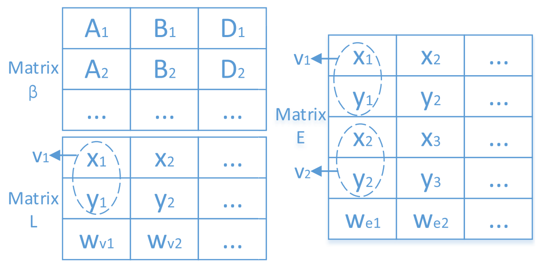


Fig. 4. Data Structures of matrixes: β, E, and L.

B. Step 2: Preprocessing

The coordinators of sensors SC and modulators M are used with the radiuses of modulators to generate edge lights. Edge lights cut across each other segmenting FOI into many cells. The numbers of sensors and modulators may be different, i.e. $m \neq n$, e.g., as shown in Fig. 1, there are two modulators used to modulate the sensing view of sensor SC_4. Connecting sensor SC_4 with four vertices of FOI with dashed lines, several triangles are constructed. If a modulator locates inside one of such triangles, the modulator can modulate the sensing view of sensor SC_4. For instance in Fig. 1, sensor SC_4 is modulated by modulators M_4 and M_5.

An edge light is a tangent line of a modulator sensed by its responding sensor and can be represented as $Ax+By+D = 0$, where either $D = 1$ or $D = 0 \wedge B = 1$ after we normalize the equation for D value to be either 0 or 1. Let l denote the number of edge lights. Let β (shown in Fig. 4) denote a matrix of $3 \times l$ for storing coefficients (i.e. A, B, D) of edge lights so that each row indicates one edge light. For any line $i = 1, \ldots, l$, $\beta(1, i)$, $\beta(2, i)$, and $\beta(3, i)$ are the line's coefficients, A, B, and D, respectively. Therefore, we call each row of β as a line (in fact, coefficients of a line). Rows in β are built in an order as follows:

In order to sort edge lights, the order of sensors should be defined first. We start from the point $(0, R_{so})$ and go along the sensor region in the clockwise direction. When finding the first sensor, we denote the sensor as SC_1, and then the second as SC_2, the third as SC_3, etc. The order of edge lights are defined similar to the order of sensors. Assume that each edge light can only cut across the sensor region once. Each sensor may have many edge lights. We call the edge light belonging to sensor SC_1 as the first edge light, if we can order all the other edge lights of SC_1 along the sensor region in clockwise direction from it. Taking this edge light as the first edge light, order all the other edge lights of all the sensors in the clockwise direction (shown in Fig. 1). We can have a sequence of edge lights.

In order to make our algorithm search vertices easily, we need to sort the vertices in order. Sorting the intersected vertices in FOI once for all is complex since the vertices are scattered in a two-dimension coordination system. However, all of the vertices are intersections of edge lights or boundary lines of the FOI. It will be much easier to sort the vertices that are on the same edge light or boundary line. Accordingly, sort the vertices of each edge light. Take the vertices of the same edge light as a group. Each time we only sort the group by the ascending order and store in matrix $L = [L_1; L_2; \ldots; L_l]$ (shown in Fig. 4), where $0 < l \leq 2mn$ is the number of edge lights. Note that l and L are different variables. Normally, $l \leq 2m$, except the case that they are deployed with a high density that leads to one modulator modulating more than one sensor. A sparse matrix $E = [E_1; E_2; \ldots; E_l]$ (shown in Fig. 4) is utilized to store edges of the edge lights in L where an edge is a line segment of a edge light constructed by two adjacent vertices. Moreover, any element in E stores all the edges of the same edge light with the same index in L. For instance, E_1 stores all the edges of edge light L_1, E_2 stores the edges of $L_2, \ldots$, and E_l stores the edges of L_l. In addition, L and E also store weights (defined later) of the vertices and edges, respectively.

Definition 2: We call the process of finding all the vertices and edges belonging to a cell as *cell reconstruction*.

Before reconstructing cells, the weight of every vertex and every edge should be set. The weight of an edge is set by how many cells it belongs to.

Definition 3: The *weight* of an edge or vertex is the total number of times that it is used to reconstruct cells. The weight of an edge is defined as

$$w_e = \begin{cases} 1 & e \ is \ on \ a \ boundary \ line \ of \ FOI \\ 2 & e \ is \ on \ an \ edgelight \ across \ FOI \end{cases} \quad (2)$$

Any two cells C_1 and C_2 have at most one edge in common:

$$|C_1 \cap C_2| = \begin{cases} 1 & C_1 \ is \ adjacent \ to \ C_2 \\ 0 & C_1 \ is \ not \ adjacent \ to \ C_2 \end{cases} \quad (3)$$

where $C_1 \cap C_2$ and $|C_1 \cap C_2|$ denote the set of common edges and the number of elements of the intersection set, respectively.

Since a cell can be represented by a cycle set of vertices, we can imagine the cell reconstruction process as an ant crawling along the vertices of the cell in the clockwise direction and coming back to its start vertex. Denote a cell $C_1 = v_1v_2v_3v_4v_1$ in FOI, as illustrated in Fig. 2. When constructing C_1, suppose that the ant starts from vertex v_1. Then vertex v_1 is called as the *first vertex*. The ant may crawl to one of the neighbors of v_1 (i.e., vertex v_2 or v_4). At this moment, we call vertex v_2 and v_4 are the *candidates* of its next destination. If it arrives at v_2, we call v_2 as the *second vertex*. If the ant is on its way to v_2, but not arrives at v_2, we say that the ant *leaves* from v_1 and is about to *arrive* at v_2. The ant is supposed to crawl with a cycle in the clockwise direction and finally come back to vertex v_1. When the ant arrives at a vertex and leaves from the vertex, the vertex is used twice. Hence, the weight of a vertex in FOI is defined as

$$w_v = 2c = \sum_{e \in \Gamma} w_e \quad (4)$$

where c is the number of cells containing the vertex, Γ denotes the set of edges containing this vertex. The proof of the above equation is given later.

Finally, the weights of vertices and edges are stored with their coordinates in L and E, respectively. Hence, we can easily judge the weight when selecting a vertex or an edge.

C. Predominant of Cell Reconstruction

Since we have defined the weight of vertices and edges, the problem is how to ensure that the selected vertices and edges can construct a cell. During the vertices selecting process, the following two constraints should be satisfied:

Constraint 1: Throughout a cell reconstruction process, the direction to select vertices and edges should keep the same as clockwise;

Constraint 2: The current vertex and the candidates to be selected from are neighbors.

Accordingly, the CRSA tool is proposed to reconstruct cells in FOI and acquire their signatures. The CRSA core contains four algorithms: FirstVertexSelectionAlgorithm, SecondVertexSelectionAlgorithm, CrossProductVertexSelectionAlgorithm, and SignatureAcquisitionAlgorithm.

Algorithm 1 Cell Construction Algorithms

```
Input: $SC$, $M$, $R_m$, $V_{FOI}$
Output: $S$, $C$
 $C = \varnothing$; $F = \varnothing$; $S = \varnothing$; $i = 1$;
 Firstly calculate $L$ and $E$ with $SC$, $M$, $R_m$ and $V_{FOI}$;
while do
    $C_i = \varnothing$;
    Select the first vertex $v_f$ of cell $C_i$ with
    FirstVertexSelectionAlgorithm;
    if no $v_f$ selected then
        break;
    Use $v_f$ to select the second vertex $v_s$ of cell $C_i$ with
    SecondVertexSelectionAlgorithm;
    Select the other vertex $v_t$ of cell $C_i$ with
    CrossProuductVertexSelectionAlgorithm;
    Calculate the signature $S_i$ of cell $C_i$ with
    SignatureAcquisitionAlgorithm;
    $C = C \cup C_i$, % $C$ is a set of vertex cycles;
    $S = S \cup S_i$; $i = i + 1$;
 return $S$, $C$;
```

During the cell reconstruction process (shown in Algorithm 1), FirstVertexSelectionAlgorithm (shown in Algorithm 2) and SecondVertexSelectionAlgorithm (shown in Algorithm 3) are used to select the first vertex and second vertex of the cell (here a cell is taken as a cycle of vertices). Note that in Algorithm 1, the output F is initially for usage in later algorithms and % means comments. CrossProductVertexSelectionAlgorithm (shown in Algorithm 4) is used to select the other vertices until the cycle of selected vertices come back to the first vertex. SignatureAcquisitionAlgorithm (shown in Algorithm 5) is used to get the signatures of cells in FOI.

The weight computation rules are given as follows:

- If a vertex is selected, its weight will minus 2.
- If an edge is selected, its weight will minus 1.
- If a vertex or an edge is abandoned, its weight keeps the same.

If the weight of a vertex or an edge is greater than zero (i.e. $w > 0$), we say that the vertex or edge is available. Otherwise, we say that the vertex or edge is unavailable.

When selecting a vertex, we need to judge whether the vertex could be used to construct a cell. If the vertex could be used to construct a cell, it is selected to do the reconstruction work. Otherwise, abandon the vertex and select another vertex from its candidates. Now, the rules to abandon a vertex and select another one are given as follows:

Vertex Changing Rules (VCR): During the process of cell reconstruction, three cases might happen when selecting a vertex v, as follows:

a) The present vertex v is unavailable.

b) The vertex v is available, but the edge which joints two endvertices v and v_{pre} is unavailable, where the vertex v_{pre} denotes the selected vertex before vertex v.

c) The vertex v and edge $v_{pre}v$ are both available, but we cannot select a vertex to keep the direction of selected vertices as being clockwise.

If the selected vertex v satisfies one of the cases of the VCR, we have to abandon vertex v and select another one from its candidates.

Note that every two neighboring vertices must be two ends of an edge. For instance, the first vertex and the second vertex should be two ends of the same edge. When both of the two ends of an edge are selected, its weight will minus 1.

D. Cell Reconstruction and Signature Acquisition

Now, the FirstVertexSelectionAlgorithm (shown in Algorithm 2) is described in detail. The inputs of the algorithm are the coordinates vertices and edges, i.e., L and E. The outputs are the selected first vertex v_f, the ordered set F, and the cell set C. Since each vertex has at most one chance to be the first vertex, vertices are stored in an ordered set F to be orderly selected as the first vertex. For instance, the first vertex v_1 of the first cell is selected from the four vertices of FOI, might as well the V_{FOI3} (as shown in Fig. 1). Then, it is used to reconstruct the first cell and store the vertices into an ordered set F. Moreover, each element in F is unique. For instance, $C_1 = v_1v_2v_3v_4v_1$, $F = \{v_1v_2v_3v_4\}$. $|F|$ is used to denote the number of elements in F. After that, v_2, v_3, v_4 in F are seriatim selected as the first vertex to reconstruct cells C_2, C_3, C_4, respectively. In fact, in CRSA algorithms, each time only one cell is reconstructed. In this instance, we list three cells to explain the first vertex selection clearly. As illustrated in Fig. 2, $C_2 = v_2v_5v_6v_7v_3v_2$, $C_3 = v_3v_7v_8v_3$, and $C_4 = v_4v_3v_8v_9v_{10}v_4$, then we have $F = \{v_1v_2v_3v_4v_5v_6v_7v_8v_9v_{10}\}$. If vertex v_2 is selected as the first vertex to reconstruct the cell C_2, these vertices of C_2 are stored into F except the repeated vertices. Then, the next vertex v_3 is selected as the first vertex to reconstruct the new cell C_3 and vertices v_7 and v_8 are stored into F. Note that when selecting a vertex, we should check whether the vertex satisfies the VCR or not. Repeat the cell reconstruction work until the weights of edges and vertices achieve zero.

Algorithm 2 FirstVertexSelectionAlgorithm

```
Input: $L$, $E$, $F$, $i$, $C_i$
Output: $v_f$, $F$, $C_i$
$j = 1$;
if $i = 1$ then
    Select $v_f \in V_{FOI}$ to reconstruct cell $C_1$;
    $w_{v_f} = w_{v_f} - 2$;
    $C_i = v_f$; $F = \{v_f\}$;
else
    $F = F \cup C_i$;
    % $F$ is an ordered set without repeated elements;
    while $j \le |F|$ do
        if $v_f$ satisfies the VCR then
            $j = j + 1$;
        else
            $w_{v_j} = w_{v_j} - 2$;
            break;
return $v_f$, $F$, $C_i$;
```

Algorithm 3 SecondVertexSelectionAlgorithm

```
Input: $L$, $E$, $v_f$
Output: $v_s$
Select the neighbors of $v_f$ from $L$;
Store the selected neighbors into $V_{candidates}$;
Sort $V_{candidates}$ by weights in ascending order;
$k = 1$; $v_k \in V_{candidates}$;
while do
    $v_s = v_k$;
    if $w_{v_k} > 0$ and $w_{v_f v_s} > 0$ then
        apply CrossProductVertexSelectionAlgorithm to
        obtain $\alpha$;
        if $\alpha=1$ then
            $k = k + 1$;
        else
            $w_{v_f v_s} = w_{v_f v_s} - 1$;
            $w_{v_s} = w_{v_s} - 2$;
            Store $w_{v_s}$ into $L$ and store $w_{v_f v_s}$ into $E$;
            break;
    else
        $k = k + 1$;
return $v_s$;
```

Now, we explain the SecondVertexSelectionAlgorithm (shown in Algorithm 3) in detail, which carries out the matching part of linearly-sorting and-neighbor-matching (LSNM). The inputs are the selected first vertex v_f, the coordinates of vertices and edges: L and E. The algorithm outputs the selected second vertex v_s.

Since it is easy to get the first vertex v_f from the FirstVertexSelectionAlgorithm, neighbors of v_f are the candidates of the second vertex v_s. Then, candidates are sorted in the ascending order by their weights. They are serially selected one by one to be a v_s until finding the vertex which does not satisfy the VCR. In this part, the process contains two steps: a) whenever a vertex is selected as v_s from the candidates, its availability should be judged first; b) since edge E_{fs} joints v_f and v_s, its availability will be judged.

If the edge E_{fs} satisfies the VCR. The selected candidate will be changed by another candidate vertex. Otherwise, the candidate is selected as the second vertex v_s. Then, v_f and v_s are plugged into the CrossProductVertexSelectionAlgorithm to select the third vertex v_t. If there is no third vertex (no vertex satisfying constraint 1, i.e., $\alpha = 0$ in Algorithm 3) to be selected, then the v_s will be abandoned and another candidate of v_s will be selected as the new v_s to repeat the above steps in the SecondVertexSelectionAlgorithm.

Algorithm 4 CrossProductVertexSelectionAlgorithm

```
Input: $C_I$, $E$, $L$, $i$, $v_f$, $v_s$
Output: $C_i$, $\alpha$
 Set the direction as clockwise, i.e., $CD > 0$;
 $C_i = [v_f, v_s]$; $v_{cp1} = v_f$; $v_{cp2} = v_s$; $\alpha = 1$;
while do
    Select the neighbors of $v_{cp2}$ from $L$;
    Store the selected neighbors into $V_{candidates}$;
    Set $c$ as the number of candidates in $V_{candidates}$;
    $k = 1$; $v_k \in V_{candidates}$; $v_{cp3} = v_k$;
    $CP_{candiates} = \emptyset$; $IP_{candiates} = \emptyset$;
    $count_{CP} = 0$; $count_{IP} = 0$;
    if $v_k$ is equal to $v_f$ then
        $\alpha = 0$; return $C_i$, $\alpha$;
    while $k \le c$ do
        if $w_{v_{cp3}} > 0$ and $w_{v_{cp2}v_{cp3}} > 0$ then
            $CP_{candiates} = [CP_{candiates}, v_k]$;
            $count_{CP} = count_{CP} + 1$;
        $k = k + 1$;
    while $count_{CP} > 0$ do
        if $CP\left(v_{cp1}, v_{cp2}, v_{cp3}\right) > 0$ then
            $IP_{candiates} = [IP_{candiates}, v_k]$;
            $count_{IP} = count_{IP} + 1$; $\alpha = 0$;
        $count_{CP} = count_{CP} - 1$;
    if $count_{IP} > 1$ then
        Calculate $\theta = \angle v_{cp1}v_{cp2}v_{cp3}$;
        Select the vertex $v_k$ with the smallest $\theta$ as $v_{cp3}$;
    if $k = c$ and $\alpha = 1$ then
        break;
    $w_{v_{cp3}} = w_{v_{cp3}} - 2$; $w_{v_{cp2}v_{cp3}} = w_{v_{cp2}v_{cp3}} - 1$;
    Store $w_{v_{cp3}}$ into $L$ and store $w_{v_{cp2}v_{cp3}}$ into $E$;
    $C_i = [C_i, v_{cp3}]$;
    Assign the last two vertices in $C_i$ to $v_{cp1}$ and $v_{cp2}$;
    %Loop until come back to the vertex $v_f$.
```

Now, the CrossProductVertexSelectionAlgorithm (shown in 4) is explained with four steps, and not only carries out the neighbor matching part of the LSNM, but also introduces a direction keeping strategy. The inputs are v_f, v_s, L, and E. It outputs the vertices cycle of cell C_i and the value of α to judge if changing the second vertex or not.

Firstly, the direction of vertices selection is set as being clockwise. In fact, there is no difference to set the direction as being clockwise or counterclockwise, but to be consistent, we need to follow one direction throughout the reconstruction process of a cell.

Secondly, find all the neighbors of the second vertex v_s as the candidates of the third vertex v_t. Supposing that $v_f\left(x_f, y_f\right)$, $v_s\left(x_s, y_s\right)$ are the first vertex and the second vertex, respectively, candidates of the third vertex are $V_{candidates} = \{v_{t1}\left(x_{t1}, y_{t1}\right), v_{t2}\left(x_{t2}, y_{t2}\right), \ldots v_{th}\left(x_{th}, y_{th}\right)\}$, where h denotes the number of candidates.

Thirdly, select a vertex from the candidates as the third vertex v_t and judge the vertex v_t and edge $v_s v_t$ by the VCR. If none of them satisfies the VCR, the direction from v_f, v_s, to v_t should be judged whether it is clockwise or not. Generally speaking, the cross product of vectors is an easy and efficient method to judge the direction. Calculate vectors $v_s v_f$ and $v_s v_{tr}$ with $v_s v_f = (x_f - x_s, y_f - y_s)$, $v_s v_{tr} = (x_{tr} - x_s, y_{tr} - y_s)$, where $r = 1, 2 \ldots h$. $CP = v_s v_f \times v_s v_{tr} = [(x_f - x_s)(y_{tr} - y_s) - (x_{tr} - x_s)(y_f - y_s)]\overrightarrow{J}\,|CP| = |v_s v_f| \cdot |v_s v_{tr}| \cdot \sin\theta$, where $\theta \in [0, \pi]$ is the included angle between $v_f v_s$ and $v_s v_{tr}$. Then we have :

$$CP \begin{cases} > 0, \ v_f v_s v_{tr} \ is\ clockswise \\ = 0, \ v_f v_s v_{tr} \ is\ in\ a\ line \\ < 0, \ v_f v_s v_{tr} \ is\ counterclockswise \end{cases} \quad (5)$$

If more than one vertex v_{tr} satisfies the constraint 1 and makes $v_f v_s v_{tr}$ clockwise, the vertex with the smallest value of θ is selected as the third vertex to satisfy constraint 2. Then, the inner product of $v_f v_s, v_s v_{tr}$ should be calculated. $IP = v_s v_f \cdot v_s v_{tr} = \left(x_f - x_s\right)\left(y_{tr} - y_s\right) + \left(x_{tr} - x_s\right)\left(y_f - y_s\right) |IP| = |v_s v_f| \cdot |v_s v_{tr}| \cdot \cos\theta$.

From the value of IP, we can find the value range of θ:

$$IP \begin{cases} > 0, \ \theta \in [\pi/2, \pi) \\ = 0, \ \ \theta = \pi/2 \\ < 0, \ \theta \in [0, \pi/2) \end{cases} \quad (6)$$

The value of CP and IP will help us find the smallest θ.

Finally, repeat the second and third steps until coming back to the first vertex. Then, a cell is reconstructed.

E. The Algorithm to Calculate Signatures

In Algorithm 5 to calculate signatures, inputs are the vertices cycle of a cell C_i, the coordinates of modulators M, and the line equation matrix of edge lights β. It outputs the signature of a cell S_i.

As depicted formerly, the points in the very same cell have the same signature. The signature of a cell can be obtained through judging the visibility of one point in the cell. Since the coordinates of a cell C_i are calculated by above three algorithms, it is easy to calculate a point p inside the cell by lemma 4 in the next section. Then, the signature of C_i can be calculated by judging whether p is in the shadow region of the modulator of sensors. Since the shadow region is between the two edge lights of the modulator, we can plug the coordinates of p and the modulator into the line equation of the two edge lights, respectively. Then multiply the results to judge the visibility of C_i to the sensor. If the product is positive, p is visible to the sensor. Otherwise, p is invisible to the sensor

The signature of a cell C_i consists of n binary bits. Each time one bit of the signature is calculated, and all the edge

Algorithm 5 SignatureAcquisitionAlgorithm

```
Input: $C_i$, $\beta$, $M$
Output: $S_i$
$S_i = []$;
for $j = 1:1: length(C_i)$ do
    for $k = 1:1: length(\beta)$ do
        Use Lemma 4 to find a point $p$ in cell $C_i$;
        Find the modulator $M_k$ which is tangent with $\beta_{k1}$ and $\beta_{k2}$;
        $H_p = \beta_{k1}(p) \times \beta_{k2}(p)$;
        $H_M = \beta_{k1}(M_k) \times \beta_{k2}(M_k)$;
        if $H_p \times H_M > 0$ then
            $S_i = [S_i, 1]$;
        else
            $S_i = [S_i, 0]$;
return $S_i$;
```

lights of the same sensor should be calculated to determine one bit of the signature. Because a sensor might be modulated by more than one modulators, and the results of different pairs of edge lights might be different. For instance, both "1" and "0" signature bits are calculated. In this case, the "0" signature bit will be chosen. The cell is still invisible to the sensor, due to it is shadowed by one modulator of the sensor. Until all the "n" bits of a signature is calculated, the signature of the cell C_i is finished. Note that when calculating one bit of the signature, edge lights should be chosen first. When edge lights are chosen, their corresponding sensor and modulator can be determined because the edge lights of the same sensor are store in the same row of β.

Now, the signature calculation algorithm of a cell is explained as follows. There are six steps. Firstly, calculate a point p in cell C_i with lemma 4. Secondly, select the edge lights (β_{k1} and β_{k2}), and the modulator (M_k) of a sensor. Thirdly, plug the coordinates of p into β_{k1} and β_{k2}, and calculate $H_p = \beta_{k1}(p) \cdot \beta_{k2}(p)$, where $\beta_{k1}(p) = A_{k1}x_p + B_{k1}y_p + D_{k1}$, $\beta_{k2}(p) = A_{k2}x_p + B_{k2}y_p + D_{k2}$ and H_p denotes the multiplication result of $\beta_{k1}(p)$ and $\beta_{k2}(p)$. Fourthly, plug the circle-center coordinates of M_k into β_{k1} and β_{k2}, and calculate $H_M = \beta_{k1}(M_k) \cdot \beta_{k2}(M_k)$, where $\beta_{k1}(M_k) = A_{k1}x_{M_k} + B_{k1}y_{M_k} + D_{k1}$, $\beta_{k2}(M_k) = A_{k2}x_{M_k} + B_{k2}y_{M_k} + D_{k2}$ and H_M denotes the multiplication result of $\beta_{k1}(M_k)$ and $\beta_{k2}(M_k)$. Fifthly, multiply H_p and H_M and if the result is larger than zero, the signature is "0", otherwise "1". Finally, repeat the steps from step 2 to step 5 until all the modulators of the same sensor are calculated. Use the logical operation "and" (i.e. $\bigcap$) to the results and get the final value of this bit in the signature.

$$H_p \cdot H_M \begin{cases} < 0 & a\ visible\ cell\ with\ siganture\ 1 \\ > 0 & an\ invisible\ cell\ with\ signature\ 0 \end{cases} \tag{7}$$

V. Theoretical Analysis

A. Cell Reconstruction

This part provides theoretical analysis including convexity of cells, selection rule of vertices, and a signature acquisition rule.

Lemma 1: Any edge light cuts across a cell in FOI segmenting the cell into two convex cells.

Proof: Suppose that there is a cell C in FOI. An edge light h cuts across C intersecting with two different edges at two points. Then, h segments C into two cells C_r and C_l. It is easy to see that C_r is fully contained in one side of the closed half-spaces determined by h. So is to C_l.

Now, we prove that all the cells in FOI are convex. We might as well suppose that the edges lights segment FOI one by one. The first edge light segments FOI into two cells and they are convex. The second edge light might segment one or both of the two cells. If there is one, they become three convex cells. If there are two, they become four convex cells. It is no difference how many convex cells segmented by edge lights, but every edge light cutting across a cell will not change its convexity and the generated cells will keep convex. This scenario happens repeatedly, until every edge light has segmented FOI. Then, it is easy to find that all of the cells formed FOI are convex. Hence, Lemma 1 holds. ■

From the proof of Lemma 1 we can get Lemma 2 as follows:

Lemma 2: All of the cells in FOI are convex.

Lemmas 1 and2 are simple but very important. In the algorithm of cell reconstruction, a rule (one of the weight computation rules introduced before) is given to limit the direction of vertices selection when constructing a cell. Lemmas 1 and 2 can guarantee the algorithm works as our expectation.

As illustrated in Fig. 2, suppose that a concave polygon $V_2V_5V_{12}V_7V_8V_2$ is a cell, where V_6 and V_3 is neglected since they are on the line of V_5V_{12} and V_2V_8, respectively. Let us use CRSA to construct the cell. We start from vertex V_2 in the clockwise direction. CRSA can select V_5, V_{12}, and V_7 easily. However, the fifth vertex V_8 could not be selected. Because $V_{12}V_7V_8$ are not in the clockwise direction. This case happens whenever the CRSA is used to construct a concave cell. Hence, Lemma 2 is necessarily to be used to ensure that all the cells in FOI are convex. Moreover, the convexity of a cell also restrict Lemma 4, which will be discussed later.

Next, the proof of equation 4 is given in detail.

Proof: Two edge lights intersect with each other at vertex υ and segment FOI into four cells. There are four cells containing vertex υ. This means that the weight of vertex υ is 8. If there are k edge lights intersect with each other at vertex υ, they will segment FOI into $2k$ cells [19]. The weight of vertex υ is $2k$.

Edges on edge lights are belonging to two adjacent cells. This means that each of them will be used twice to construct all the cells that are containing them. However, when vertex υ is at the boundary lines of FOI, there exist two cases:

Case I: vertex υ is the intersection of two boundary lines. Vertex υ is at one corner of FOI. It is used twice to construct one cell, i.e., there are two edges at vertex υ, and each will be used once to construct the same cell.

Case II: vertex υ is the intersection of one edge light and one boundary line. Vertex υ is on the boundary line (except the corner) of FOI and will be used four times to construct two adjacent cells. The edge which is a part of edge light is used twice; the two edges which are parts of the boundary line will be used once. We can conclude that the weight of one vertex equals to the double number of cells that it belongs to, and equals to the sum of weights of edges containing it. Hence, equation 4 holds. ■

Theorem 1: If there exists a vertex v which is available, we can construct a cell with v as the first vertex.

Proof: From equation 4, we know that the weight w_v of vertex v is an even number. In definition 5, when constructing a cell, leaving and arriving vertex v will lead to $w_v - 2$. Then w_v will still be an even number. Hence, if a vertex is available, it means that its weight is an even number no less than 2, i.e., $w_v \geq 2$.

Now, we prove that a vertex must belong to at least two edges. Suppose that vertex v only belong to edge L. Then the weight w_e of L should have $w_e = w_v \geq 2$. It is easy to know $w_e \leq 2$. Hence, if $w_e = w_v > 2$, vertex v cannot only belong to an edge L. If $w_v = w_e = 2$, there should be two cells containing edge L and vertex v since in a plane one cell can only contains an edge once. Since a cell is a cycle set of vertices, the weight of vertex v should be $w_v = 4$ while it is belonging to two cells. This is contradict to $w_v = w_e = 2$. Hence, vertex v could not only belong to an edge L.

Vertex v joints two edges and these two edges still have other two vertices denoted as v_1 and v_2 available. From v_1 and v_2, we can prove that there should be other vertices available until we construct a cell. Hence, Theorem 1 holds. ■

Theorem 1 provides the selection rule of the first vertex to construct a cell. In fact, Theorem 1 also implicates that no matter which vertex is selected as the first vertex, it has no effect on the cell reconstruction result. In other words, each vertex of a cell can be selected as the first vertex to reconstruct it.

Lemma 3: If there exists a vertex v that is available, there must exist at least two neighbors of v available, which can be two vertices of an unreconstructed cell.

Lemma 4: If $v_a(x_a, y_a)$, $v_b(x_b, y_b)$, and $v_c(x_c, y_c)$ are three vertices of the same cell, point $v_d(x_d, y_d)$ must be inside the same cell and have the same signature with any other points inside the cell, where

$$\begin{cases} \mathrm{x}_d = \alpha \mathrm{x}_a + \beta \mathrm{x}_b + \gamma \mathrm{x}_c \\ \mathrm{y}_d = \alpha \mathrm{y}_a + \beta \mathrm{y}_b + \gamma \mathrm{y}_c \end{cases} \tag{8}$$

where $0 < \alpha, \beta, \gamma < 1$ and $\alpha + \beta + \gamma = 1$.

Proof: We might as well set the cell as a convex set of points as $C \subseteq R^2$ and points $v_a, v_b, v_c \in C$. From the definition of convex set in [21], we can easily get $l_1 = \lambda v_a + (1 - \lambda) v_b \subseteq C, 0 < \lambda < 1$. However, if v_a, v_b are two ends of an edge of one cell in FOI, they may belong to several cells. Hence, the points at the line segment $v_a v_b$ have several different signatures.

Let $l_2 = \mu v_b + (1 - \mu) v_c \subseteq C, 0 < \mu < 1$. Plus l_1 to l_2 and use 2 to divide the sum: $T_1 = \alpha v_a + \beta v_b + \gamma v_c \subseteq C$, where $\alpha = 0.5\lambda, \beta = 0.5(1 - \lambda + \mu), \gamma = 0.5(1 - \mu)$. Consequently, we have $0 < \alpha, \beta, \gamma < 1$, and $\alpha + \beta + \gamma = 1$. v_d is a point inside triangle T_1. However, T_1 is only belong to the cell who owns vertices v_a, v_b, v_c at the same time. From Lemma 3, the cell who owns v_a, v_b, v_c is unique. Hence, Lemma 4 holds. ■

Lemma 4 is only applicable to convex cells. For a concave cell, Lemma 4 might yield a point outside the cell.

Conjecture 1: we have

$$|N| = [(\max |C|) - 2n - 1)]/2 + 2 = n^2 - n + 2, \tag{9}$$

where $|N|$ and $|C|$ are the numbers of different signatures and cells, there are n sensors and n modulators, and each modulator matches one sensor.

We can verify the above conjecture via experiments later, but we cannot prove it mathematically.

B. Complexity Analysis

Building the tool is not simple so that we need to propose better algorithms for the tool. Furthermore, we need to analyze the proposed algorithms via complexity analysis to see the effectiveness of the algorithms as follows.

Theorem 2: The lower bound of computational complexity of CRSA is $T(n) = O(m^6 log_2 m)$ under the condition $m \geq n$, where n and m are the number of sensors and modulators, respectively.

Proof: There are six parts in the analysis. 1) for calculation complexity of getting coordinates of vertices, line equations of edge lights should be calculated. There are at most $2m$ edge lights and each of them with a constant computation time. Then we have the computation time is $O(m)$. After that, we calculate the intersections (i.e., the vertices) of them. Hence, the edge lights of the same sensor intersect at the location of the sensor. Actually, the upper bound number of edge lights of a sensor is a constant, which is limited by the spatial resolution and sensor efficiency. The computation time to calculate coordinates of vertices is $O(mn)$. We have $T_1 = O(m) + O(mn)$.

2) The ordered storage of sensor and modulator locations, coefficients of line equations of the edge lights, coordinates of vertices will cost lots of computation time. In our experiments, the edge lights are stored into the same row of a matrix E_i of the same sensor i, where the row number is corresponding to the sensor number. The computation time is $O(mn)$. As to the vertex storage, we should order the edge lights and store vertices on the edge light into the same column of the matrix L, where the row number is corresponding to the edge light number. The computation time is $O(mn)$. We have the storage computation time is $T_2 = 2O(mn)$.

3) Now consider the sorting time of vertices. According to our LSNM strategy, vertices of the same edge lights should be sorted in an ascending order. We use the bubble sort, whose computation time is $O(n^2)$. Sometimes, a modulator may modulate more than one sensor and generate more than $2m$ edge lights. In this case, the density of sensors is relatively higher. However, our research focuses on the lower density sensors and higher sense radius scenarios. We assume that the number of edge lights is at most $2m$. The computation time for sorting is $T_3 = O(2mn^2)$.

4) Next we consider the search time of vertices in cell reconstruction. Suppose that each edge light intersects with all the other edge lights in FOI, except the edge lights which intersect with it at the location of its responding sensor.

Suppose that no three edge lights intersect on the same vertex. There are at most $2m(2m-1+2)$ vertices (each edge light intersect with the boundary of FOI, generating two vertices) in FOI. For a vertex v on edge light E_{li}, the computation time to search its location from the $2m+1$ vertices on E_{li} is at most $O(2m(2m+1)log_2(2m+1))$. However, we have to search all the $2m$ edge lights to determine which edge lights it is on. From [17], we know that the number of vertices is directly related the number of cells. That is to say, we can calculate the number of vertices with the number of cells and both of them will achieve their maximum at the same time. During the cell reconstruction process, the number of cells determines the cyclic times cell reconstruction. Hence, we have to estimate the number of cells. Commonly, we deploy more modulators than sensors to improve the efficiency of sensors, i.e., $m \geq n$. Under the consideration of attaining a better spatial resolution with the same number of modulators, the amounts of vertices should be as much as possible, i.e., $m = n$. From [17], a simplified form of the formulation to calculate the number of cells can be given as: $c = 2m^2+1$. However, during the cell reconstruction process, one vertex might be selected at most 4 times from the candidates of the present location of the cell. Then the computation time is $T_4 = O(4m^2(2m^2+1)(2m+1)^2\ log_2(2m+1))$.

5) There are three kinds of storage operation during the cell reconstruction process in CRSA, which are storage of vertices to be selected as the first, storage of the candidates to be selected, and storage of the selected vertices in a cell. The computation time of them are $O(2m(2m+1))$, $O(8m(2m+1))$, $O(2m(2m+1)(2m^2+1))$. Then, the computation time of storage in cell reconstruction process is $T_5 = O(10m(2m+1)+O(2m(2m+1)(2m^2+1)))$.

6) Finally, the computation time to calculate signatures contains two aspects: the computation time to calculate a signature and the cyclic time to calculate all the signatures. As illustrated in the previous section, there are five steps to calculate a signature and the computation time is $O(2mn)$. The computation time to cyclic is $O(2m^2+1)$. Thus, we have $T_6 = O(2mn(2m^2+1))$.

Now, we can give the whole computation time of our CRSA method is

$$T(n) = T_1 + T_2 + T_3 + T_4 + T_5 + T_6 \tag{10}$$

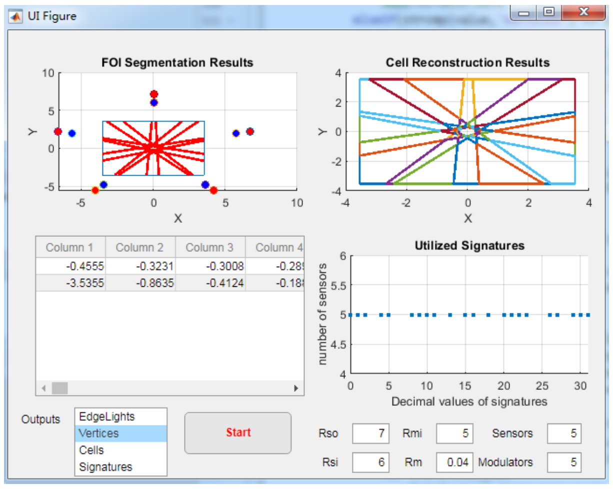
Under the condition that $m \geq n$, the simplified representation of $T(n)$ is: $T(n) = O(m^6 log_2 m)$. Hence, theorem 2 holds. ∎

VI. Deployment Performance Studies Using the Proposal Tool

In this section, we use the proposed tool with an implementation to study sensor-modulator deployment performance to attempt to make some scientific discoveries behind the deployments.

A. Experiment Settings

A deployed sensor is assumed to have an directional view covering FOI. FOI is a square region with a circumscribed circle whose radius is R_{oi}. Since the modulators are used to limit the view range of the sensors, they are deployed closer to the circumscribed circle of FOI. Consequently, sensors are assumed to be deployed in an annular region with radius R_{si} and R_{so}, while modulators are deployed in an annular with radius R_{mi} and R_{mo}. As shown in Fig. 1, it is easy to know that $R_{so} > R_{si} = R_{mo} > R_{mi}$ and a modulator is assumed to be in a round shape with radius R_o. The default geometric relationship in our experiments is $R_{so} = 2R_{mi}$, $R_{si} = R_{mo} = 1.5R_{mi}$, and $R_m = 0.01R_{mi}$, where R_m is the radius of a modulator. The sizes of sensors and modulators are not proportional so that the cells might appear to be tilted with different shapes. FOI is inside the inner circle of the modulator annular region. Their positions are based on a coordinate system with the FOI centering as the origin. The data precision is 1.0e-5 with $R_{mi} = 4$. The experiments are aiming to acquire the vertices of the cells to obtain the signatures of all cells. Experiments of $n = 1, \ldots, 14$ are designed to analyze the utilization ratio of signatures and the discrimination degree of cells based on the results CRSA. Note that our tool can test any number of sensors and modulators although we show results of n up to 14.



Column 1	Column 2	Column 3	Column 4
-0.4555	-0.3231	-0.3008	-0.28
-3.5355	-0.8635	-0.4124	-0.18

Rso	7	Rmi	5	Sensors	5
Rsi	6	Rm	0.04	Modulators	5

Fig. 5. The Results of the tool.

TABLE I
Store Formats of Data

Edge lights	$\beta_1 = [A_1, B_1, C_1]$	$\ldots$	$\beta_6 = [A_6, B_6, C_6]$
Vertices	$v_1 = [x_1, y_1]$	$\ldots$	$v_{28} = [x_{28}, y_{28}]$
Cells	$C_1 = v_1 \ldots v_4$	$\ldots$	$C_{19} = v_{23}v_{24}v_{28}$
Signatures	$S_1 = 111$	$\ldots$	$S_{19} = 111$

In Fig.5, each subfigure illustrates its output results with its title. There are three figure outputs: cell segmentation results, cell reconstruction results, and utilized signatures to code cells. Moreover, the store formats of data in Fig.5 are shown in Table I. Specially, the x-coordinate and y-coordinate of a vertex in the table of Vertices and Cells are stored in two adjacent rows, respectively. For instance, for vertex v_1 and v_2, x_1, x_2 stores in row 1 and y_1, y_2 stores in row 2. Parameters of edge lights and signatures are stored as follows: each row stores all the parameters of an edge light (for instance,

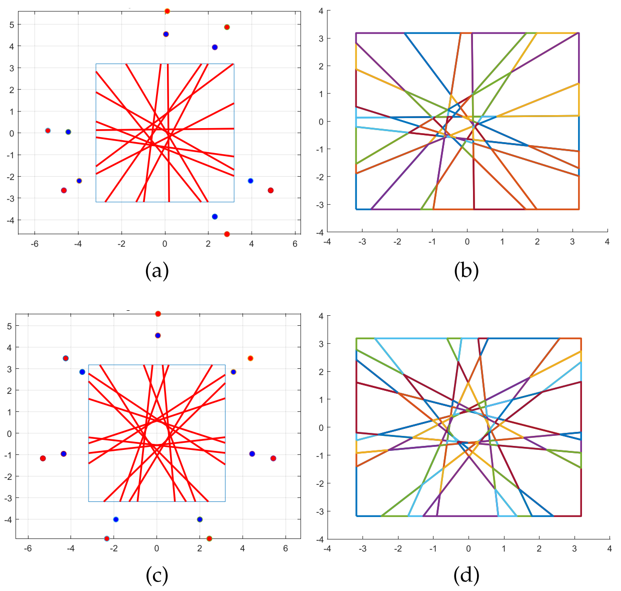


Fig. 6. a) 6S6M FOI segmentation; b) 6S6M cell reconstruction; c) 7S7M FOI segmentation; d) 7S7M reconstruction.

A_1, B_1, C_1 of the edge light β_1 are stored in row 1 and A_2, B_2, C_2 of the edge light β_2 are stored in row 2). So is the signatures.

B. Searching the Optimal Deployment Strategy

In this part, we utilize the proposed CRSA tool to apply several optimal deployment strategies for different deployment scenarios. Firstly, sensors and modulators are deployed in pairs (one modulator corresponding one modulator). Secondly, the optimal deployment strategy is defined as the maximum number of cells, i.e., utilizing a number of sensor and modulator pairs to segment FOI into the maximum number of cells. Thirdly, the radius of modulators can be adjusted. Actually, we can achieve the same result by adjusting the distance between sensor and its corresponding modulator. In our experiments, the distances are immobilized so we adjust the radius of modulators. We conduct experiments with $n = m = \{4, 5, 6, 7\}$ sensor and modulator pairs, and find they can segment FOI into at most 33, 51, 73 and 99 cells, respectively. These results are in accordance with the formula $\max|C| = 2n^2 + 1$ in [17], where $\max|C|$ is the maximum number of cells, n is the number of sensor and modulator pairs. This formula can only be used to one sensor and one modulator pair to calculate the maximum number of cells that FOI is segmented.

Fig. 6 shows the experiment results in two parts, where the notation nSmM means n sensors and m modulators: 1) the FOI segmentation results which illustrate the deployment locations of sensors and modulators and give the segments cells of FOI; 2) the cell reconstruction results where different cells differ in colors. We observe that the reconstructed cells match well the original cells. Cells are constructed and plotted in an ordered sequence. Due to the intersection of cells, some edges of a cell may have different colors with other edges. These experiment results show that our proposed method can help researchers searching the optimal deployment strategy.

C. Scientific Discoveries Behind the Deployments

Definition 4: Utilization ratio of signatures in a deployment strategy is defined as: $u = |N|/2^n$, where $|N|$ is the number of different signatures and n is the number of sensors.

There always exist some cells with the same signature. This leads to a confusion of which cell the signature represents. Hence, a definition of discrimination degree is introduced as follows.

Definition 5: Discrimination degree of cells in a deployment strategy is defined as: $d = |N|/|C|$, where $|N|$ and $|C|$ are the numbers of different signatures and cells, respectively.

Definition 6: Ideal ratio of cells and signatures is defined as $ideal = (\max|C|)/2^n$, i.e., using the theoretical maximum number of cells to be divided by the theoretical maximum number of signatures.

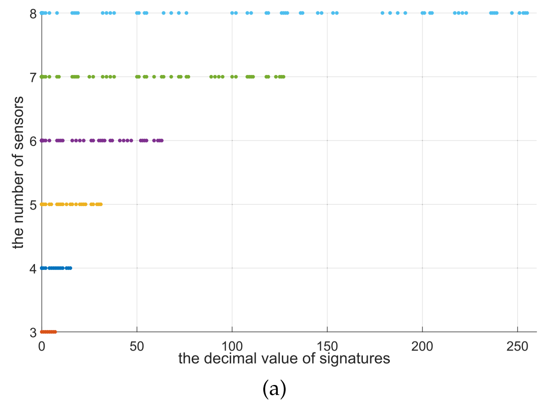
If $ideal \leq 1$, theoretically speaking, there exist enough signatures to encode all the cells, each with a different signature. However, such a theoretical scenario may not be achievable. Since $\max|C| = 2n^2 + 1$, $ideal > 1$ might holds. The purpose of the ideal ratio is to guide researchers to design deployment formations with as many cells and signatures as possible. It represents a ratio between the maximum number of cells and the ideal number of signatures (i.e., 2^n, which may not be achievable with one-on-one modulator-sensor pairs). However, it provides an upper bound of sensor efficiency and spatial resolution. Since the ideal ratio specifies the maximum number of cells over the maximum number of signatures, the spatial resolution is the best with the maximum number of cells and the sensor efficiency is the best with the maximum number of signatures.

Since different cells may have the same signature in a 2-dimension plane. We calculate the times that each signature used to code cells.

Definition 7: Utilization time of a signature is defined as the time that the signature is used to code cells

Since signatures are represented with a binary states, we can treat a signature as a binary number which can be transferred into a decimal number. With transferring the binary signatures into decimal numbers, we can draw and observe them easily. In Fig. 7, the decimal number of signatures are represented with dots with the same colors and the vacancies between dots denote the unutilized signatures. The black dots of 3 sensors are continuous closely since all the signatures are utilized to code cells of FOI. As the number of sensors increases, both the number of utilized signatures and the number of vacant signatures increase too. Signatures are utilized to code and distinguish different cells. The increment of the number of signatures also means that the number of cells is increased.

We draw Table II, Fig. 8, and Fig. 9 to answer the questions listed in an early section. Table II is obtained by using our CRSA tool to deploy 1-14 sensors and modulators to generate the maximum number of cells. Then, we calculate the number of utilization times of signatures and list them in the table.



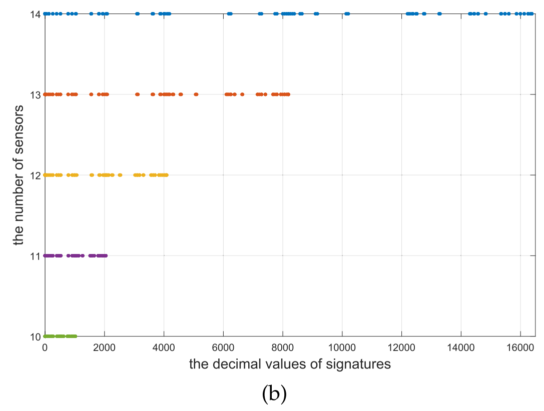


Fig. 7. The decimal represent of signatures.

TABLE II
THE DETAILED UTILIZATION TIMES OF SIGNATURES TABLE

times	0	1	2	4	6	8	10	12	...	28	$\|N\|$
1SM	0	1	1	0	0	0	0	0	...	0	2
2SM	0	1	2	1	0	0	0	0	...	0	4
3SM	0	1	6	0	1	0	0	0	...	0	8
4SM	0	1	10	0	0	1	0	0	...	0	16
5SM	10	1	20	0	0	0	1	0	...	0	22
6SM	22	1	30	0	0	0	0	1	...	0	32
7SM	84	1	42	0	0	0	0	0	...	0	44
8SM	198	1	56	0	0	0	0	0	...	0	58
9SM	438	1	72	0	0	0	0	0	...	0	74
10SM	932	1	86	0	0	0	0	0	...	0	88
11SM	1936	1	110	0	0	0	0	0	...	0	112
12SM	3952	1	134	0	0	0	0	0	...	0	136
13SM	8034	1	156	0	0	0	0	0	...	0	158
14SM	16200	1	182	0	0	0	0	0	...	1	184

The first row represents the number of times that the signatures are utilized to code cells, i.e., many cells in FOI have the same signature, and the utilized time of the latter is increased by 2. The first column represents the number of sensors and modulators that are used to be deployed, e.g., $2SM$ means two sensors and two modulators and each sensor being modulated by one modulator. The other rows mean that the numbers of signatures used, e.g., the number 6 in row 4 column 4 means that there are 6 different signatures, each being used twice to code cells when 3 sensors and 3 modulators are deployed. Based on the table, we have the following observations: 1) from the second column, we observe that as the number of sensors increases, the number of unutilized signatures are increasing very fast; 2) from the third column, we observe that there is always a signature which is used only once: in fact all the bits of the signature are '0', i.e., "00…0"; 3) from the fourth column, we observe that there are many signatures that are utilized twice, i.e., there are many signatures that code two cells; besides, as the number of sensors increases, the number of utilized signatures increases, but not faster than the second column; 4) from the other columns except the last, we observe that no matter how many sensors that we deploy, there is always one cell that is utilized more than twice while the number of sensors are no less than 2; in fact all the bits of the signature are '1', i.e., "11…1"; 5) from the last column, we observe that the number of utilized signatures increases as the number of sensors increases; however, the number of the utilized signatures is fewer than the number of unutilized signatures that is increasing dramatically as illustrated in the second column; in other words, there is a significant number of unutilized signatures as the number of sensors is large; 6) we observe that the number of the utilized signatures as $|N| = [(\max|C|) - 2n - 1)]/2 + 2 = n^2 - n + 2$ as indicated in Conjecture 1 (equation 9) since there are $2n$ cells having the same signature as $11\ldots1$ (all bits "1"), one cell having a signature as $00\ldots0$ (all bits "0"), and the remain signatures are used twice; 7) almost all the actual signatures are utilized twice or more; cells that have the same signatures might lead to wrong localizations of targets; this problem should be solved by designing the combination of RST (for instance, the RSTs in [10]).

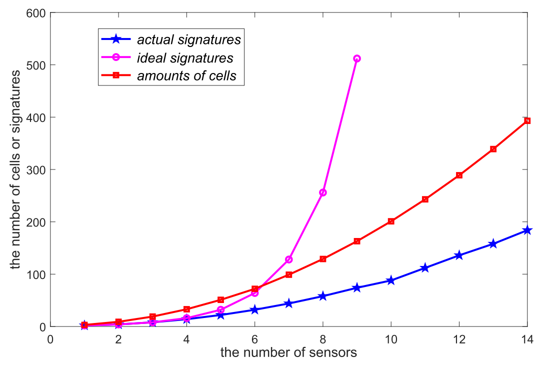


Fig. 8. The relationship between cells and signatures.

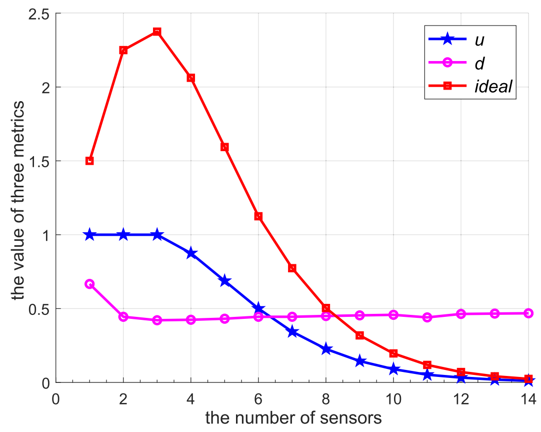


Fig. 9. The decimal represent of signatures.

In Fig. 8, we adopt 1-14 sensors, where each sensor is modulated by one modulator. Fig. 8 shows the relationship of the number of cells/signatures vs. n (i.e., the number of sensors) with three curves: 1) the pink curve with circle marks as the ideal number of signatures using the function 2^n, 2) the red curve with square marks as the maximum number of cells generated by our CRSA tool, and 3) the blue line with star marks as the number of utilized signatures generated by our CRSA tool. From Fig. 8, we have the following observations: 1) as n increases, all three curves increase; this means that we can achieve a better spatial resolution by deploying more sensors; 2) the number of cells is always greater than the number of utilized signatures; this means that there are always many cells with the same signatures; 3) when $n \leq 4$, the number of ideal signatures overlaps with the number of actual signatures; this means that when $n \leq 4$, we can achieve the best sensor efficiency, i.e., 1. 4) when $n > 4$, the number of ideal signatures are more than utilized signatures and the gap between them continues to widen as n increases; this means that when $n > 4$, the more sensors that we deploy, there are more unutilized signatures, and therefore the sensor efficiency are becoming worse; 5) when $n > 6$, the number of ideal signatures are more than the number of cells; that means that when $n > 6$, we can never utilize all the signatures with each sensor modulated by one modulator; moreover, as n increases, the gap between the number of cells and the number of ideal signatures continues to widen; this is one of the reasons that the sensor efficiency becomes worse.

Fig. 9 shows the three metrics: the utilization ratio of signatures (u), the discrimination degree of cells (d), and the ideal ratio of signatures ($ideal$) with the pink line with circle marks, the blue line with star marks, and the red line with square marks, respectively. From Fig. 9, we have the following observations: 1) as n increases, $ideal$ increases first and then decreases approaching to zero; from the definition of $ideal = (\max|C|)/2^n$, we can think that both $\max|C|$) and 2^n are increasing functions of n and when $ideal$ increases/deacreases, the speed of increasing the number of cells is larger/smaller than the speed of increasing the number of ideal signatures; since the function 2^n is exponentially increasing while $\max|C|$ is increasing but not exponentially, $ideal$ finally declines approaching to zero; 2) when $n \leq 6$, we have $ideal > 1$ and this indicates that the number of cells is larger than the number of ideal signatures; 3) as n increases, u keeps 1 first and then declines to zero; $u = 1$ means that all the signatures are utilized to code cells and the sensor efficiency is 1; as n increases, the sensor efficiency becomes worse and approaches to zero; 4) as n increases, d decreases below 0.5 and tends to be stable around 0.5; this means that there are almost half of the cells having the same signatures and the spatial resolution is far more enough to $d = 1$; the best discrimination degree of cells (d) achieves when we deploy one sensor.

Since the purpose of using RST is to achieve a better localization accuracy, we not only want a larger d, but also more cells. Hence, the metric, the discrimination degree of cells, should be considered firstly. Under the condition that achieves an expected discrimination degree of cell, we expect the ratio of utilized signatures as larger as possible. Besides, the number of cells is smaller than the number of ideal signatures when $n > 4$. The spatial resolution depends on the number of cells in FOI. This means that the more cells we have, the better the spatial resolution might be. In our experiments, the maximum number of cells are attained. The value of u can be regard as the spatial resolution when n is chosen. Hence, the discrimination degree of cells in a deployment formation is very important to improve spatial resolution. In Fig.9, it is easy to observe that the discrimination degree of cells (namely, u) is around 0.5, due to the fact that almost half of the cells have the same signatures as shown in Table II. This means that more than half of the cells are undistinguishable. Actually, it is no sense to seek the maximum cells with so many undistinguishable cells.

The another metric that the utilization of signatures (namely, d) is utilized to measure the sensor efficiency of a deployment formation. Commonly, researchers take the measure that enhance the sensor efficiency to achieve a better spatial resolution (for instance, using more sensors). However, there still another way to improve spatial resolution without increasing the number of sensors, i.e., using more modulators. In this way, the number of cells also increases and the modulators should be designed specially (for instance, [10], [22]). Hence, utilizing specially designed RST can also improve sensor efficiency.

VII. Conclusions and Future Work

In this paper, we proposed a tool to visually and efficiently represent and analyze the deployment formation and spatial resolution. A cell reconstruction algorithm was proposed to reconstruct cells generated by the deployment formation in which lists of vertices are generated by regarding cells as shortest path rings. We further provided some theoretical analysis and computational complexity analysis of the proposed algorithm.

We utilized our tool to conduct some experiments and the tool can help us to find the maximum number of cells. We defined three metrics: the utilized ratio of signatures, and the discrimination degree of cells, and the ideal ratio of signatures. From our experiments, we verified the conjecture between utilized signatures and cells, $|N| = n^2 - n + 2$, when one sensor is modulated by one modulator. From the experiments, we learnt that as the number of sensors increases, the spatial resolution becomes better and the number of cells increases while the discrimination degree of cells tend to 0.5. At the meantime, we learnt that the sensor efficiency keeps declining to zero as the number of sensors increases. Hence balance between the spatial resolution and sensor efficiency can help researchers choose better parameters, such as the number of sensors, etc.

However, there are still much work to do. For instance, there are many variables in our studies, such as the number of sensors, the number of modulators, the coordinates of sensors and modulators, the size of modulators, etc. In our future work, we aims to utilizing our tool to explore the relationships among such parameters and exploring the metric of RST design and deployment. Besides, the shapes and the areas of cells have a great effect on spatial resolution. It is easy to find that the shape of some cells might be too small or too big. We expect cells to have a small variance in an area. How to deploy the proper number of sensors and modulators to achieve a small variance in areas among cells is still under exploring.

Our current studies focus on a software tool for sensor deployments. As future work, we will conduct physical experiments on sensor deployments, target counting, and localization.

The space accuracy, positioning accuracy, and positioning rate mainly depend on the deployment algorithm. Future work also includes designing new deployment formations to achieve the expected accuracy. New deployment formations will be designed to obtain the expected space accuracy, positioning accuracy, and positioning rate.

In another ongoing paper of ours, we work on studying positioning and tracking with multiple people in the sensing region. In our work, we study a problem called 'the invisible target' problem. When multiple targets move in the monitoring space, using one instant sensing result cannot distinguish them. We are solving the problem by designing appropriate detection algorithms by utilizing traces of moving objects for a time period.

Our tool in this paper can be used indoors and outdoors. However, as we mentioned earlier, we need additional detection algorithms built on top of this tool to study multiple people and effects of the number of people (many or few).

References

[1] L. Yuan, Q. Liu, M. Lu, S. Zhou, C. Zhu, and J. Yin, "A highly efficient human activity classification method using mobile data from wearable sensors," *Int. J. Sensor Netw.*, vol. 25, no. 2, pp. 86–92, 2017.

[2] W. Zhou, F. Li, D. Li, X. Liu, and B. Li, "A human body positioning system with pyroelectric infrared sensor," *Int. J. Sensor Netw.*, vol. 21, no. 2, pp. 108–115, Jan. 2016.

[3] A. A. Salah, B. Lepri, F. Pianesi, and A. S. Pentland, "Human behavior understanding for inducing behavioral change: Application perspectives," in *Human Behavior Understanding*. Berlin, Germany: Springer: 2011, pp. 1–15.

[4] Y. Ivanov, A. Sorokin, C. Wren, and I. Kaur, "Tracking people in mixed modality systems," *Proc. SPIE*, vol. 6508, Jan. 2007, Art. no. 65080L.

[5] P. Potuluri, M. Xu, and D. J. Brady, "Imaging with random 3D reference structures," *Opt. Express*, vol. 11, no. 18, pp. 2134–2141, Sep. 2003.

[6] D. J. Brady, N. P. Pitsianis, and X. Sun, "Reference structure tomography," *J. Opt. Soc. Amer. A, Opt. Image Sci.*, vol. 21, no. 7, pp. 16–18, Jul. 2004.

[7] Q. Hao, D. J. Brady, B. D. Guenther, J. B. Burchett, M. Shankar, and S. Feller, "Human tracking with wireless distributed pyroelectric sensors," *IEEE Sensors J.*, vol. 6, no. 6, pp. 1683–1696, Dec. 2006.

[8] A. Crivello, F. Mavilia, P. Barsocchi, E. Ferro, and F. Palumbo, "Detecting occupancy and social interaction via energy and environmental monitoring," *Int. J. Sensor Netw.*, vol. 27, no. 1, pp. 61–69, May 2018.

[9] R. A. Nadi and M. G. H. A. Zamil, "A profile based data segmentation for in-home activity recognition," *Int. J. Sensor Netw.*, vol. 29, no. 1, pp. 28–37, 2019.

[10] B. Yang, J. Luo, and Q. Liu, "A novel low-cost and small-size human tracking system with pyroelectric infrared sensor mesh network," *Infr. Phys. Technol.*, vol. 63, pp. 147–156, Mar. 2014.

[11] Q. Hao, F. Hu, and Y. Xiao, "Multiple human tracking and identification with wireless distributed pyroelectric sensor systems," *IEEE Syst. J.*, vol. 3, no. 4, pp. 428–439, Dec. 2009.

[12] P. Potuluri, U. Gopinathan, J. R. Adleman, and D. J. Brady, "Lensless sensor system using a reference structure," *Opt. Express*, vol. 11, no. 8, pp. 965–974, Nov. 2003.

[13] U. Gopinathan, D. J. Brady, and N. P. Pitsianis, "Coded apertures for efficient pyroelectric motion tracking," *Opt. Express*, vol. 11, no. 18, pp. 2142–2152, 2003.

[14] A. Sinha and D. J. Brady, "Size and shape recognition using measurement statistics and random 3D reference structures," *Opt. Express*, vol. 11, no. 20, pp. 2606–2618, Sep. 2003.

[15] P. K. Agarwal, D. Brady, and J. Matoušek, "Segmenting object space by geometric reference structures," *ACM Trans. Sensor Netw.*, vol. 2, no. 4, pp. 455–465, Nov. 2006.

[16] N. Shrivastava, R. M. U. Madhow, and S. Suri, "Target tracking with binary proximity sensors: Fundamental limits, minimal descriptions, and algorithms," in *Proc. 4th Int. Conf. Embedded Netw. Sensor Syst.*, Oct. 2006, pp. 251–264.

[17] P. Asadzadeh, L. Kulik, E. Tanin, and A. Wirth, "On optimal arrangements of binary sensors," in *Spatial Information Theory* (Lecture Notes in Computer Science), vol. 6899. Berlin, Germany: Springer, 2011, pp. 168–187. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-23196-4_10

[18] M. Peng and Y. Xiao, "Signature maximization in designing wireless binary pyroelectric sensors," in *Proc. IEEE Global Telecommun. Conf.*, Dec. 2010, pp. 1–5.

[19] M. Peng, Y. Xiao, N. Li, and X. Liang, "Monitoring space segmentation in deploying sensor arrays," *IEEE Sensors J.*, vol. 14, no. 1, pp. 197–209, Jan. 2014.

[20] H. Yang, H. Xu, and K. Tang, "WiHumo: A real-time lightweight indoor human motion detection," *Int. J. Sensor Netw.*, vol. 24, no. 2, pp. 110–117, 2017.

[21] J. Matousek, *Lectures on Discrete Geometry*. New York, NY, USA: Springer-Verlag, 2002, ch. 1.

[22] L. Zhao, W.-Z. Song, L. Shi, and X. Ye, "Decentralised seismic tomography computing in cyber-physical sensor systems," *Cyber-Phys. Syst.*, vol. 1, nos. 2–4, pp. 91–112, 2015, doi: 10.1080/23335777.2015.1062049.

Longxiang Luo received the B.E. degree from Gaungxi University, China, in 2012. He is currently pursuing the Ph.D. degree with the Shenyang Institute of Automation, Chinese Academy of Sciences, China. His research interests include mainly in space segmentation and coding.

Yang Xiao (M'98–SM'04) is currently a Professor with the Department of Computer Science, The University of Alabama, Tuscaloosa, AL, USA. He has published over 200 journal articles and over 200 conference papers. His current research interests include cyber-physical systems, the Internet of Things, security, wired/wireless networks, smart grid, and telemedicine. Dr. Xiao was a Voting Member of IEEE 802.11 Working Group from 2001 to 2004, involving IEEE 802.11 (WIFI) standardization work. He is a Fellow of IET. He currently serves as the Editor-in-Chief for *Cyber-Physical Systems* (Journal). He had (s) been an Editorial Board or Associate Editor for 20 international journals. He served (s) as a guest editor for over 20 times for different international journals.

Wei Liang (M'11–SM'17) received the Ph.D. degree in mechatronic engineering from the Shenyang Institute of Automation, Chinese Academy of Sciences in 2002. As a Primary Participant/a Project Leader, she developed the WIA-PA and WIA-FA standards for industrial wireless networks, which are specified by IEC 62601 and IEC 62948, respectively. She is currently a Professor with the Shenyang Institute of Automation, Chinese Academy of Sciences. Her research interests include industrial wireless sensor networks and wireless body area networks. She was a recipient of the International Electrotechnical Commission 1906 Award in 2015 as a Distinguished Expert of industrial wireless network technology and standard.

Meng Zheng (M'14) received the B.S. degree in information and computing science and the M.S. degree in operational research and cybernetics from Northeastern University, Shenyang, China, in 2005 and 2008, respectively, and the Ph.D. degree in mechatronic engineering from the Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang, in 2012. From 2010 to 2012, he was a Visiting Student with the Fraunhofer Institute for Telecommunication, Heinrich Hertz Institute, Berlin, Germany. He is currently a Professor with the Shenyang Institute of Automation, Chinese Academy of Sciences. His research interests include wireless ad hoc and sensor networks, cognitive radio networks, and security in smart grids.