EFFICIENT OBJECT LOCALIZATION AND TRACKING WITH DISCRETE SPATIAL MAPPING

by

Yunhui Zheng

Department of Electrical and Computer Engineering
Duke University

Date:_____

Approved:

_____
Dr. David J. Brady, Supervisor

_____
Dr. Krishnendu Chakrabarty

_____
Dr. Qing H. Liu

_____
Dr. Xiaobai Sun

_____
Dr. Steven Cummer

Dissertation submitted in partial fulfillment of the
requirements for the degree of Doctor of Philosophy
in the Department of Electrical and Computer Engineering
in the Graduate School of
Duke University

2005

ABSTRACT

EFFICIENT OBJECT LOCALIZATION AND TRACKING WITH DISCRETE SPATIAL MAPPING

by

Yunhui Zheng

Department of Electrical and Computer Engineering
Duke University

Date: _____

Approved:

_____
Dr. David J. Brady, Supervisor

_____
Dr. Krishnendu Chakrabarty

_____
Dr. Qing H. Liu

_____
Dr. Xiaobai Sun

_____
Dr. Steven Cummer

An abstract of a dissertation submitted in partial fulfillment of
the requirements for the degree of Doctor of Philosophy
in the Department of Electrical and Computer Engineering
in the Graduate School of
Duke University

2005

Abstract

Optical sensors systems implement mappings between object and sensor spaces. Traditional systems are based on continuous models, in which both the objects and sensors are assumed to be in a continuous volume and the optical field propagation is also continuous. Since physical sensors are distributed discretely in the 3D physical space, in practice an approximation must be made in traditional systems to fit the measurement data into the continuous model. This thesis explores an alternative approach to studying the optical sensor system design based on a completely discrete model so that not only are the measurements on a finite point set but the reconstructed object is described with a finite point set instead of a continuous function.

The thesis gives general discussion about the theoretical background of optical sensor systems and spatial mappings. Both the continuous and discrete models are discussed. A taxonomy is given to classify the design method for the discrete models. We summarize several approaches on how to design the optical sensor arrays with discrete models. Two design examples are presented in the following chapters. In Chapters 2 and 3, the design, implementation and experimental demonstration of fiber webs are presented in detail as examples for volume sampling. We introduce the concept of K-neighbor-code and show that when K=2, the 2D gray code is best for localizing an extended object. We first apply the superimposed code to the optical sensor array design for localizing multiple objects simultaneously. Specifically, we

use the genetic algorithm to find the most efficient UD_2 code to localize up to 2 persons in 64 positions. We use low cost fiber web to realize these efficient coding designs and the result shows that fiber webs work well. In Chapters 4 and 5, we propose the boundary sensor arrays for object tracking. An analysis of both 1D and 2D spaces is given based on the graph theory. We introduce the concept of boundary sensing based on detected temporal domain sequences and prove a series of theorems as the foundation of boundary sensing. We propose the data structure for boundary sensing and show that the simulation results confirm the theoretical analysis. The result shows that in the 1D domain we find the optimal solution and in the 2D domain we prove the random deployment scheme is a workable deployment scheme when the number of sensors and the length of the sensor sequence are large enough. In Chapter 6, we generalize the boundary sensor model to the statistic partial boundary sensor model. Since the boundary sampling is a broad area, we will leave the complete solution as an open problem.

Acknowledgements

It has been very hard for me to write this part of the thesis more than any other documents I have written, for the fear of missing any one who has supported me all the way towards this accomplishment. Still, I want to try my best to write down these comments and hope that they can convey my gratitude to those kind people.

I want to start with my advisor, David J. Brady. From UIUC to DUKE, he has been the best advisor and friend a student can ever have. His creative ideas and advice guided me through the maze of scientific research. His kindness made our group a big joyful family. His guidance has made the past four years a happy life journey for me.

I thank the other members of my committee: Krishnendu Chakrabarty, Qing H. Liu, Xiaobai Sun and Steven Cummer for their valuable time and advice.

I want to thank Pankaj K. Agarwal, for his insightful comments on the boundary sensors. I also want to thank Mark A. Neifeld from the University of Arizona, who offers a lot of helpful comments and suggestions on both technical contents and professional writing of this thesis.

The Duke Integrated Sensing and Processing group (DISP), the research group I have been sharing my happiness and sadness with, is like a ship to me. Although the crew members change when time passes, the wonderful friendship and research atmosphere are always there. I want to thank every one in this group for every favor they have done for me, especially Bob Guenther, who helped a lot on the first

fiber paper; Nikos Pitsianis, who contributed to the generic search algorithms for superimposed code; Mike Sullivan and Steve Feller, who gave a lot of advice on the experimental setup. I will not try to list all of them in fear of missing some of them. I am proud of being a member of DISP and I will never forget the moments I have spent with my fantastic collegues.

I want to thank my parents and my sister. Their support is always there no matter where I am and what I am. I owe them everything. I can only wish I could make up for a portion of their love during my lifetime.

Finally, this is for my wife, Jirun Zhang. It is the old Chinese saying that you don't want to say thanks when you owe someone too much. That is exactly true here. I just want to say that when the life journey continues, we will always be there, TOGETHER.

Contents

List of Figures

List of Tables

Chapter 1

Spatial Mappings for Optical Sensors

1.1 Introduction

Optical sensing systems are based on spatial mappings. From cameras to spectrometers, from lenses to gratings, many of the optical sensor systems use spatial mappings to map the desired information about the object into a set of points in our physical space. The data points are then captured by spatially distributed sensor arrays and used to reconstructed the original object based on the mapping algorithm. A sensor array is a set of sensors placed at different locations to spatially sample a propagating wave field, which may be electromagnetic, acoustic, seismic, etc [37]. This thesis focuses on the optical field. The challenge here is to determine the optimum set up for a sensor array to meet a specific set of requirements.

There are two ways to design an optical sensing system by spatial mapping: the continuous way and the discrete way. Traditional sensor system designs are based on continuous mappings. Sensor models generally assume the field is measurable at every point within a continuous physical volume. Such volume includes lines and surfaces. However, real world sensors are located at discrete physical points, such as the discrete pixel arrays in optical cameras. The measurements from the sensor

array are actually discrete samplings. A traditional sensor system is designed in this way because the radiation field is usually modulated by continuous volume (such as a lens), which keeps the geometry and topology of the object space. The advantage of such a mapping is its natural relationship with the physical space. Quite a number of examples of continuous mapping exist, such as Fourier systems [19, 39], computerized tomographic imaging [38], computed-tomography imager [21, 24], coded aperture [13, 14], and interferometric imaging systems [33, 48–50].

A geometric sensor system, on the other hand, develops the mapping in a completely discrete way. It assumes geometric field propagation and is aware that the measurements are taken at discrete points. Correspondingly, the information about the object is also reconstructable on a finite number of points. For example, for a discrete imaging system, the image of the object is described by a 2D or 3D discrete array instead of a continuous 2D or 3D function. The number of discrete measurements or reconstructible object states are limited by system resolution or precision. The object space and measurement space are segmented into discrete cells. Each cell is associated with a unique object state or measurement state. The mapping is then defined between each pair of segmented cells in object space and measurement vectors in measurement space. The next step is to realize this mapping with spatial mapping so that the information can be captured by a spatially distributed sensor array.

A number of researchers have been working on such discrete transformation based on geometric spatial mapping. Examples include Hadamard transform imager and imaging spectrometer [36], in which the Hadamard matrix is employed to improve system signal to noise ratio; feature specific imaging [56], in which the Karhuren-Loeve matrix is employed to improve imaging system fidelity; sensor placement problem [15], in which an identifying coding matrix is employed

to determine the best sensor deployment scheme; the multimodal multiplex spectrometer [66], in which a random transformation matrix is employed to efficiently convert source spectrum curve into spatial intensity map; reference structure tomography(RST) [10], in which a detailed discussion is given about how to design a new tomography system based on modulation of transformation matrix T with a reference structure.

The motive of this thesis is to further develop discrete spatial mapping method and apply it to optical sensor system design, particularly for object localization and tracking. To our knowledge, this kind of design has not been discussed systematically before. We will start from general discussion about the spatial mapping method, the difference between the continuous and discrete model, and then give a taxonomy of discrete mapping models. After that, two examples about how to use the discrete spatial mapping method to design and optimize a sensor array for object localization and tracking are discussed in detail. The result shows that the discrete spatial mapping model is a useful and effective tool for optical sensor systems design.

1.2 Spatial Mapping Method

An optical sensing system may be described by a state-space model. Such model typically include the following components:

Object space: The 3D physical space that contains the objects being studied.

The objects in the object space generate a radiating field that carries the information about the parameters of the objects.

Object configuration space: The space spanned by the object state

vector, whose components are the values of independent parameters of the object.

Radiation space: The 3D physical space in which the radiation field propagates.

The radiation space can be a free space or contain some optical elements to modulate the radiation field.

Sensor/measurement space: The space spanned by the measurement vector, whose components are the measurements from each individual sensor.

In our problem, we have unknown object(s) in the object space, the optical field carrying the information about the object propagates through the physical space and is captured by the optical sensors. The goal of our system is to estimate the object states from the detected sensor states. A general model of the mapping between the object space and the measurement space is:

$$\vec{M}(\vec{r}_m) = \mathbf{T}\{\vec{S}(\vec{r}_s)\} \tag{1.1}$$

Where $\vec{r}_m$ and $\vec{r}_s$ are parameter vectors in the measurement and object spaces, $\vec{M}$ and $\vec{S}$ are measurement vector and object vectors, respectively. $\mathbf{T}$ is an operator that performs transformation between the object and sensor space, or in other words, $\mathbf{T}$ is a coding function that transforms the desired information into measurable elements.

The parameter vector $\vec{r}_s$ indexes the object space. $\vec{r}_s$ may be any vector whose components include position, frequency, time... etc, depending on the desired information. The object configuration space spanned by $\vec{r}_s$ contains the information about the object that we are interested in and this information is indexed with $\vec{r}_s$.

The parameter vector $\vec{r}_m$ for a sensor array indicates how the measurement varies from one sensor to another. The measurement of a particular sensor corresponds to the measurement for a specific valule of $\vec{r}_m$. Similar to the object configuration space, the measurement can change with space, time, frequency, etc. For example, each sensor can have its own spectrum filter to directly measure a different spectrum component. However, for a typical sensor array, different sensor nodes are always distributed in different locations within the 3D physical space. The space spanned by $\vec{r}_m$ can be reduced to the physical space spanned by $\mathbf{r}_m$. We will focus our discussion on how to design optical systems to map the information distributed in the object configuration space into sensor points distributed in the physical space by the method of spatial mapping.

A conventional optical spectrometer is a typical example of a measurement system design based on spatial mapping. For a spectrometer, we want to know the radiation intensity distribution as a function of wavelength. Instead of measuring the radiation intensity at each wavelength directly, a grating is put before the radiating object so that the radiation energy as a function of wavelength is also now function of radiating angles, as shown in Fig.1.1. After the modulation by the grating, the spectrum information can be easily collected by a distributed sensor array.

An optical imaging system is another example. In a conventional camera, the lens performs the spatial transformation so that the sensor array (CCD/ CMOS array) can capture the image at the focal plane, as shown in Fig.1.2.

A third example is the localization and tracking of the moving objects, which will be the primary focus of this thesis in the following chapters. In this case, each object, as a rigid body, has a fixed intensity profile. For example, the shape of the human body does not change much when walking or running. The intensity profile

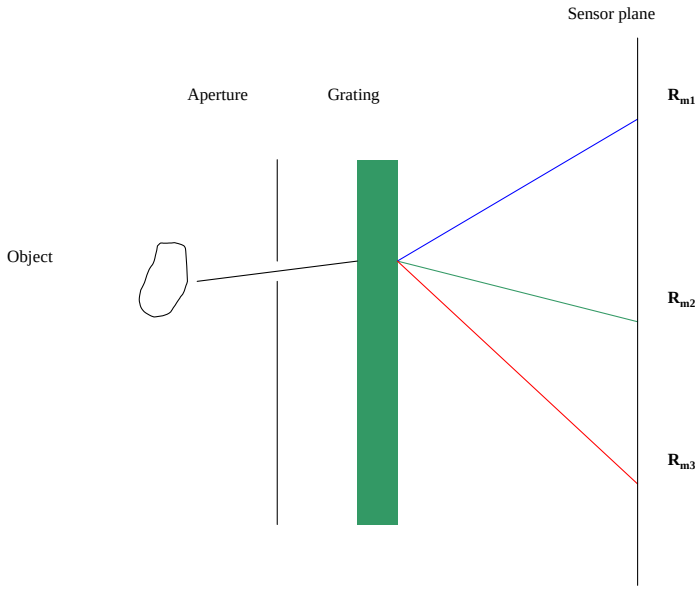

Figure 1.1: A typical optical spectrometer based on spatial mapping design.

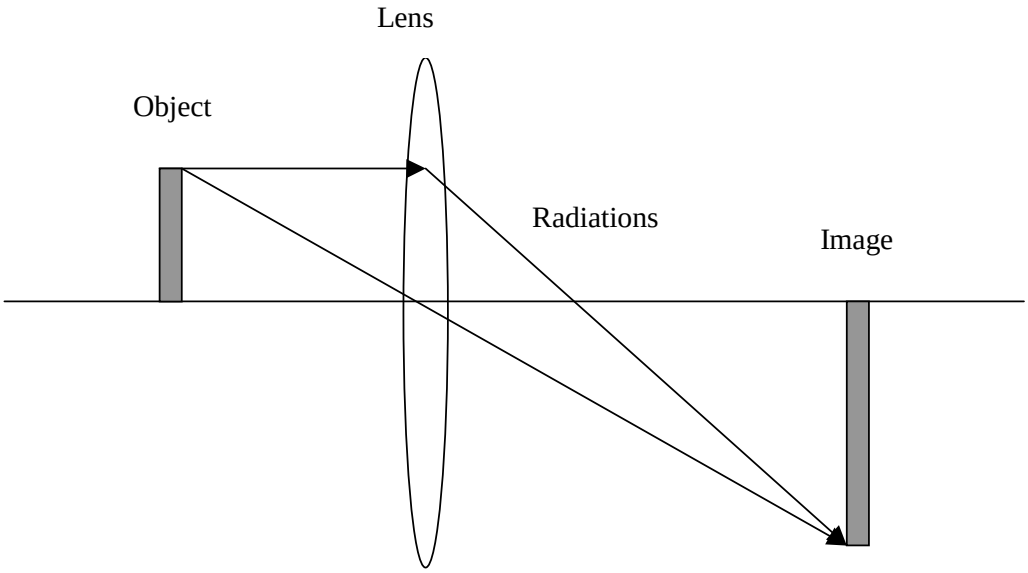

Figure 1.2: A confocal system based camera. The transform is performed by lens.

as a whole changes with the location, orientation, tilt and identity of this object. If a proper spatial mapping is established to capture the variation of the intensity profile, the corresponding parameter can thus be identified. A configuration for tracking a human with distributed sensor array is shown in Fig.1.3. Similarly to previously discussed examples, we need a spatially distributed sensor array to capture the signals. A key difference between a tracking system and an imaging system is that in a tracking system we do not need to capture the detail of the intensity profile, but only the way it changes as a rigid body. This difference can help to significantly reduce the complexity of tracking systems.

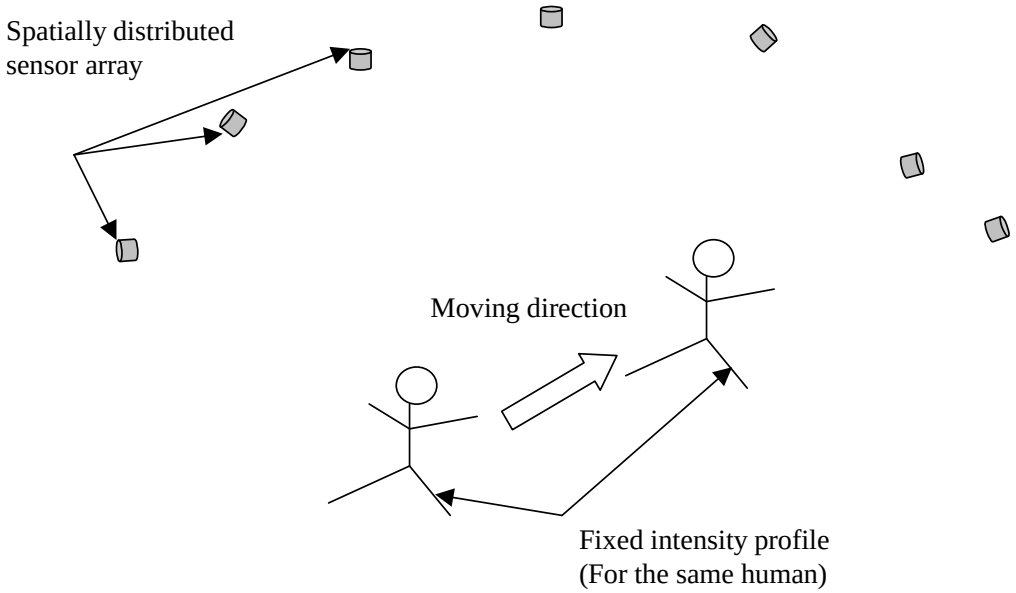

Figure 1.3: A tracking system tracks the change of intensity profile such as translation, rotate, tilt, etc.

Since one can only have a limited number of sensors and the measurement by each sensor has to be digitalized, the object state, either continuous or discrete, has

to be mapped to a discrete array of measurement data. This kind of continuous to discrete mapping, as will be discussed later, is the key to measurement space coding. A number of researcher have touched on this topic; among them is Harrison Barrett [6], who gave a rigorous discussion about continuous to continuous, continuous to discrete and discrete to discrete mappings. In this thesis, we will focus on the discrete to discrete mapping and discuss it in detail with several design examples.

1.3 Continuous Model vs. Discrete Model

In confocal systems such as regular cameras, the measurement space is isomorphic to the object space because the transformation from $\vec{r}_s$ to $\vec{r}_m$ is a one to one mapping when we try to image a particular plane in the object space. The focal plane is scaled reproduction of the object plane. The film or pixel array on the focal plane takes discrete sampling to finish the continuous to discrete conversion. The continuous mapping part can be expressed as,

$$\vec{m}(\vec{r},t) = \int \delta(\vec{r},\vec{r}^{'},t,t')s(\vec{r}^{'},t)d\vec{r}^{'}dt \tag{1.2}$$

Where $\vec{m}(\vec{r},t)$ describes the measurement, $s(\vec{r}^{'},t)$ describes the object and $\delta(\vec{r},\vec{r}^{'},t,t')$ is the sampling function. This kind of isomorphic mapping can also happen in the temporal domain. The scanning systems are typical examples of such. A traditional scanning system usually uses a probe (either physical or virtual, such as beam steering) to measure the information of the object with a particular value of $\vec{r}_s$ at a time. The object can be reconstructed by varying the scanning point and fusing all the measurements within a time frame by interpolation. Examples include laser scanning imaging systems [35], scanning spectrometers [4, 11]

and hyperspectral imaging systems [26].

In transforming mapping systems, the measurement space is a continuous projection transform of the object space. Such systems can be described by substituting the delta function in Eqn (1.2) with a more complicated continuous kernel function, as shown in Eqn (1.3). This generalization further indicates that many inverse problems fall into this category since many of them also have a separable propagation kernel.

$$\vec{m}(\vec{r},t) = \int K(\vec{r},\vec{r}',t,t')s(\vec{r}',t)d\vec{r}'dt \tag{1.3}$$

The transforming systems are close to the geometric system which will be introduce later and can be regarded as a group of predecessors. We notice in all the systems mentioned before, the discrete sampling of the sensor merely performs the continuous to discrete conversion and does not take the full advantage of discrete mappings. The object information, the field propagation and distribution are all descrbed by continuous functions. This continuous distribution is then measured with discrete sampling. The data from the discrete sampling is used to interpolate and estimate a continuous field distribution again. Finally this estimation is employed to reconstruct the original continuous object. The geometric sensor performs the whole process in a completely discrete way, from the object description to the field distribution. Since a discrete mapping and inversion model usually involves computation of matrices and Boolean algebra for a binary sensor (sensors that only have on or off states), one can take advantages of many nice properties belonging to this kind of discrete math. In this case Eqn(1.3) becomes:

$$M = TS \tag{1.4}$$

With the kernel substituted by transformation matrix T, both the object and

measurement are discretized to form Eqn(1.4). The transformation matrix T is a key to our system design which describes the spatial mapping. The advantage of using matrix T lies in the freedom of fine tuning each of its elements and its dimension to improve system performance by using certain unconventional optical element or sensor deployment strategies. Such addtional freedom is usually unavailable when a system is designed with the conventional continuous model. For example, it is well known that Hadamard matrix can be employed to improve the signal to noise ratio. It has been applied to various kinds of optical systems based on discrete mapping design [36]. The Hadamard matrix can be physically implemented by optical elements such as masks. However, as for the continuous transform kernel $K(\vec{r}, \vec{r}', t, t')$ there is no exact counterpart which can easily modulate the field propagation to conviniently and directly get this kind of SNR improvement. As another example, the group testing code which will be discussed in detail in Chapter 3, can help optical systems achieve efficient sensing by having less measurments than the number of unknowns. It employs the binary code family which can be physically implemented by certain sensor deployment schemes. This kind of efficient sensing, again, is difficult to be realized under the continuous model with the continous kernel $K(\vec{r}, \vec{r}', t, t')$.

In the next sections we will further discuss the geometric model in discrete sampling with the focus on the application of object localization and tracking.

1.4 Geometric Sampling of the Radiation Field

The geometric model assumes that radiation field propagates from the object space to the measurement space is characterized by a simple ray propagation model rather than the wave propagation model. The geometric model makes the following as-

sumptions:

1. Each point in the object is described by a positive and real intensity field.
2. The field propagates along a straight ray in the free space.
3. A point in the measurement space measures the sum of the intensity of all the rays incident on it.
4. The propagation field may be modulated by optical components.

In the geometric model, Eqn (1.1) can be reduced to

$$m(\mathbf{r_m}) = \int v(\mathbf{r_m}, \vec{r}_s) s(\vec{r}_s) \tag{1.5}$$

where the kernel $v(\mathbf{r_m}, \vec{r}_s)$ has exact physical meaning. It describes the visibility between the pair of points. Note again that although $\mathbf{r_m}$ is a point in 3D physical space, $\vec{r}_s$ can include time, frequency or any other desired parameter about the object. The detected value by the sensors under the geometric model is assumed to be a real and positive number when $\vec{r}_s$ is "visible" from $\mathbf{r_m}$. Specifically, in a binary sensor network, its value is "1" when the pair of points are visible to each other and "0" when not. Under this model, Eqn (1.5) becomes

$$m(\mathbf{r_m}) = \bigcup_{\vec{r}_s : v(\mathbf{r_m}, \vec{r}_s)=1} s(\vec{r}_s) \tag{1.6}$$

The visibility function can be modulated by any transparent/opaque components including regular optical components such as lenses, prisms, phase plates and gratings. These components can be combined with the optical sensor to form a compound sensor shaping a unique field of view, which is called the receiver pattern

of a single detector. It is described by the kernel function $v(\mathbf{r_m}, \vec{r}_s)$ when we fix the parameter $\mathbf{r_m}$. It determines the way we sample and group those individual points in the object configuration space. Different sampling methods can lead to completely different system design strategies. The receiver pattern is generalized from the term polar pattern in the literature of acoustics, which describes the microphone's sensitivity to signal from different directions. Currently we summarize several ways to do the geometric sampling of the radation field.

Volume sampling: The optical sensor's receiver pattern is one or several continuous volumes (or regions) in the object space. Inside those individual regions, each point has exactly the same visibility relationship with the sensor points. In such a sensor system, each point in the object is sampled by at least one of the sensors since the sensor system is designed to cover the whole object space.

Volume sampling is a popular sampling method because it is a natural way for optical sensing. Many of the continuous mapping systems can be regarded as volume sampling systems with the volume equal to an ideal point. Each point in the object space will have different responses in the sensors, while in geometric sensor system we discretized the space so the ideal points become individual volumes within which all the points share the same response. We will list two examples in the next two paragraphs.

In the sensor placement problem, the sampling volume of each sensor is a ball as shown in Fig. 1.4 which is one of the popular samples volume among optical sensors. Other common patterns include the fan and cone beam pattern. A target will be detected if it falls within the receiver pattern. The challenge is to design a sensor placement scheme so that the minimum number of sensors are used to localize targets.

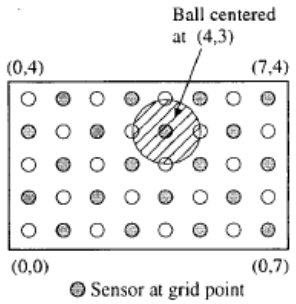

Figure 1.4: Volume sampling used in sensor placement problem with ball center pattern [15]

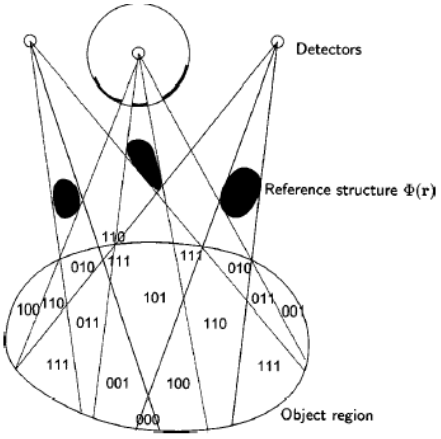

Figure 1.5: Volume sampling used in reference structure tomography. Each binary code in the object region indicates the region's visibility to each sensor in the sensor array.

In RST, some generalizations of the common sample volume of optical sensors are introduced by employing the reference structure. As shown in Fig. 1.5, the natural cone beam pattern of each sensor becomes several discrete fan zones. The intersection of this fan zone leads to the irregular sample volumes. The challenge of the system design is to determine the size and distribution of these reference structures and modulate sample volumes so that a specific sensing goal can be achieved.

Boundary sampling: The optical sensor's receiver pattern is the boundary between two or more continuous volumes. It can be either a surface or a line. In such a system, only a limited set of points in the object space are sampled by the sensors. The boundary sampling can be regarded as the spatial derivative of the volume sampling. For example, in Fig. 1.4 and Fig. 1.5, imagine that the sensors now are only sensitive to the surface of the ball and the edges between the polygon volumes. Thus, if there are any movements from volume to volume, the boundary sensors can capture it. This is extremely useful for object tracking because a moving object will tend to cut a series of boundaries if the boundaries are carefully deployed. By sampling a limited set of points, it is further possible to save system resources and reduce cost. The examples of boundary sampling can be found in a few pieces of literatures [22, 68]. Similarly to volume sampling, the boundary can be created in the 3D physical space or the object configuration space. For example, in chapter 3 we will discuss its application in object tracking and give an example of how to optimize system design based on boundary sampling. One can create a series of boundaries in the object space. When an object moves around, it will trigger a number of boundary sensors in sequence. Based on the detected temporal domain sequence and a look-up table, we will be able to localize this object.

Point sampling: The optical sensor's receiver pattern is a point right at the position of the sensor. This kind of sensor is usually applied to the situation when we are interested in the field at a specific point and it is impossible or inaccurate to infer the field at that specific point by measuring the field at other locations.

Differential sampling: The optical sensor's spatial domain receiver pattern can be either volumes, boundaries, or points, but it is only activated when object state has a non-zero time derivative. In a scenario when the object state only changes occasionally, the sensors do not sample at every time step. The differential sampling measures the time derivative while the boundary sampling measures the spatial derivative in the object configuration space. It is highlighted because time is different from other coordinates (x,y,z) and very few optical sensors are sensitive to the time derivative. The pyroelectric sensor is one example of such a kind of sensor. The application of differential sampling by pyroelectric sensor has been demonstrated recently by our lab [27].

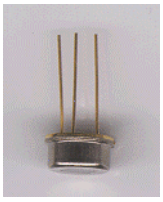

Figure 1.6: A pyroelectric sensor capable of detecting the temporal derivative of the infrared signals.

1.5 Object Analysis: Three Classes of Identification Problem

The previous section discusses how the information about the object is collected. In order to pick a sampling method for an optimized system design, one must

also be aware of how much is to be known about the object. The argument is , if we only need to estimate or identify part of the object information set, why should we capture the whole set first to extract the part of information in which we are interested? Why can't we design our system so that it is targeted at the desired information subset directly? After one figures out what kind of subset of information is desired from the object, the corresponding measurement method should be designed to optimize measurement efficiency. Based on this, we classify the information identification problems into three categories:

Static Identification: The object state is identified immediately based on the current measurement result only.

Dynamic Identification: The object state is identified based on a series of successive measurement results of fixed length. If for some reason a measurement is lost or mistaken, a new series of measurements must be initiated to guarantee correct identification result.

Path Identification: Given the initial and final states of the object, its intermediate states are determined by a series of consecutive measurement results of various lengths. For example, when tracking humans, we know his initial and final position, as well as a series of signals related to his intermediate positions. With this information in hand, it is possible to reconstruct his path between the initial and final positions with even less requirement for system resource.

The relationship between these there classes of problems is illustrated by Fig.1.5.

In general, if a sensor arrangement scheme satisfies a static identification problem, it will automatically satisfy a dynamic identification problem. The similar relationship exists between dynamic identification and path identification. These relationships indicate the system requirement is sequentially reduced from static identification to path identification.

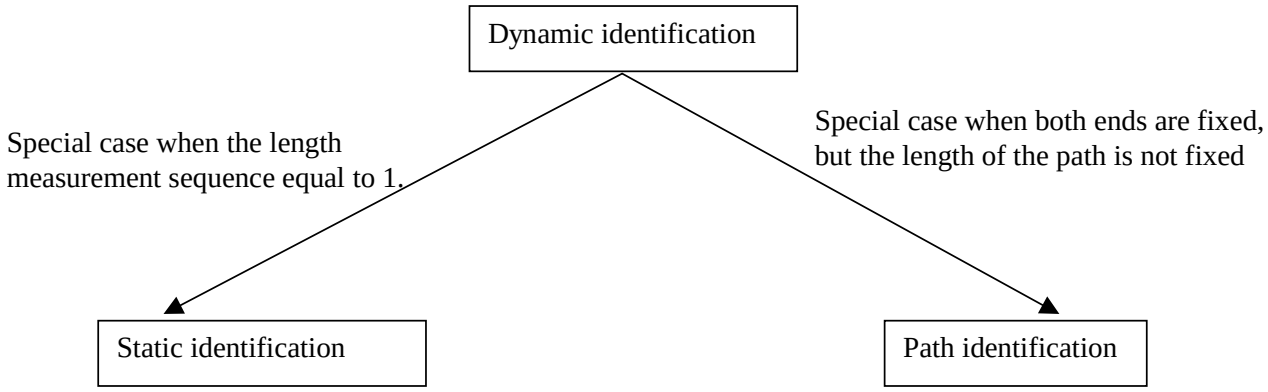

Figure 1.7: The relationship between three classes of identification problems.

In the static identification problem, the measurement results at the current time step associated with each object state should be distinct from each other. In the dynamic identification, however, this uniqueness is not required. Instead, we get a sequence of consecutive measurements. This measurement sequence is then needed to be uniquely mapped to an object state sequence. The current object state is extracted from the object state sequence. Path identification problems, as a special class of dynamic identification problems, lock the initial and final object states so that the total number of possible state sequences is further reduced.

Also, in the static identification problems, the sensor system monitors the object configuration space constantly. This strategy is useful when high resolution and sensitivity are required. However, when the object space is vast and high resolution is not imperative, sensing each point is neither necessary nor efficient. In a typical case in which the object space is segmented into a number of regions, one only wants to know which region the object is currently in, but not the exact location within that region. One only needs to monitor the boundaries between regions with sensor networks to detect a series of boundary-crossing events. Thus the object can be localized by the dynamic identification technique mentioned above.

The purpose of introducing these three classes of identification problem is to

further introduce the concept of efficient sensing. Efficient sensing involves using the least number of measurements to reconstruct desired information. Efficient sensing can be further generalized to include compressive sensing in which one has less than one measurement per object state. Efficient sensing can be realized by employing an optimized coding scheme for a particular class of identification problem. It can also be realized by selecting the most appropriate and efficient identification scheme. For example, if one can tolerate sacrificing real time error correction capability occasionally, a dynamic identification scheme is more efficient than a static identification scheme since fewer measurements are required.

1.6 Hybrid Sampling and Identification

The combination of various kinds of sampling and identification problem yield interesting results for the geometric sensor system. The different sampling methods indicate physically how to sense efficiently by changing ways of measurement. The different identification scheme indicates how to sense efficiently by picking up a reasonable ensemble of object information. In particular, we will focus on volume sampling and boundary sampling because they will be used in our object tracking applications and discussed in detail in the following chapters.

1. Static identification with volume sampling method: Since a volume sampling method samples every point in the object space, it is natural to use it in a static identification system. Because the volume sampling method constantly monitors every point of the object space, a measurement error can be corrected at the next time step. It is the most robust system (correct the measurement error within the least number of time steps) among all the combinations.

2. Static identification with boundary sampling method: The nature of boundary sampling method makes the static identification system inappropriate unless the boundaries themselves are the region in which we're interested.

3. Static identification with differential sampling: Since the differential sampling method samples the object space only when the radiation field changes, its measurement sequences appear as a number of non-zero measurements among null measurements. When a null measurement is picked up as current measurement data, one needs to look backwards in memory to find the most recent distinct measurement. Thus the measurements are actually compressed, as compared to static volume sampling, to save system bandwidth at the price of just a little more memory to store the measurement result in the previous time steps.

4. Dynamic identification with volume sampling: As mentioned before for dynamic sampling, only the measurement sequence needs to be globally unique to guarantee the detection of a unique object state sequence at a predefined resolution. Particularly for binary sensors, we also note that an interesting feature in volume sampling is that the signature for the adjacent region varies by exactly one bit, as shown in Fig.1.5. Thus the whole object space is effectively coded with generalized 2D gray code. This enforces an constraint when we design the unique signature sequence. We can take advantage of this interesting feature to design the boundary sampling dynamic identification system discussed below.

5. Dynamic identification with boundary sampling: The boundary sampling method is more efficient for the dynamic identification system in that only desired movement (boundary crossing) is tracked. The design of the bound-

ary sampling dynamic identification system is a dual for volume sampling dynamic identification system. In a system with m boundary sensors, if we encode each segmented region with m bit binary signature, then if the kth boundary sensor is triggered, it indicates the object travels to a new region whose signature is flipped in the kth bit.

6. Dynamic identification with differential sampling: The basic form of differential sampling acts like dynamic volume sampling. One can use the same coding algorithm as in volume sampling for dynamic identification. The major differences between them are still the power consumption and first order data compression (Recall that dynamic sampling measures the first order spatial derivatives). The differential sampling is always in a power saving mode because it only generates signals when object states change. Differential sampling can also be modulated so that two sensors' field of view almost overlaps with each other to firm a compound boundary. In such systems, the differential sampling method actually acts like the boundary sampling method. This idea is used for a commercially available two-element pyroelectric motion sensor.

7. Point sampling with various identifications: Since for point sampling we are only interested in the object parameters at the sensor point, (except static identification), there is no much sense in doing dynamic identification or path identification with such a sampling method.

8. Path identification with various sampling method: From the previous discussion we know that path identification is a special case of dynamic sampling with both ends fixed. Thus the discussion about dynamic sampling applies to path identification as well.

1.7 Geometric Sensor Coding Design

From the discussion in previous sections, we notice that the design of geometric sensor networks basically involve how we design the mapping from object configuration space to measurement space. The ideal mapping should be not only unique but also efficient. By unique we mean that one should be able to identify the object state once a complete measurement state is available. By efficient we mean the ratio of used measurement states to the total number of available measurement states should be as large as possible. Besides, there are other requirements from a practical application, such as error correction/detection, space/time limit, and also some special properties related to a particular kind of sensor.

A unique mapping usually is not difficult to achieve when the system is redundant. Here the redundancy means the number of available measurement states is much more than the number of object states. The situation is similar to the design of a physical hash function to hash a few object states into a great number of measurement states. However there are a number of motives to reduce the number of available measurement states. These motives include reducing the number of sensors, reducing computation complexity and reducing the requirement on system bandwidth. To accomplish this, we need to incorporate every piece of prior knowledge of the object into the system design.

One example of the prior information is when the possible object states are limited to a subspace of the whole space. For example, for object tracking, sometimes we know that only a few objects exist in the physical space. Thus, the possible object states are significantly reduced since we know that object states corresponding to more objects are forbidden. Another example is when a set of object states share some common parameters, one only needs to identify any one of the states

to measure this parameter. A third example is when we are only interested in a number of object states. Other states may occur, but they can simply be ignored. For instance, one may only want to to find if a set of particular points in a image is illuminated of not. All these reductions of object states indicate efficient sensing designs. We will show several efficient fiber web designs in chapters 2 and 3. The fiber web also shows an example of the combination of volume sampling and static identification.

On the other hand, we note that indirect measurement of the object states points out another direction to improve sensing efficiency. For example, for localization purposes, we do not need to measure the object position directly, instead, we can measure its velocity or path. These alternatives of measurement provide another degree of freedom for sensor network design. We will illustrate them in the discussion of boundary sensor array in chapters 4 and 5. The boundary sensor array also serves as the example of the combination of dynamic identification and boundary sampling.

Chapter 2

A Coded Fiber Floor Sensor System

2.1 Introduction

Human tracking has been an active research area for years. Video sensor networks for human tracking have been extensively studied [12, 43, 65]. Such systems have suffered from high computational and energy costs. However, a number of other choices exist, such as floor sensor arrays [28, 34, 51] and infrared sensor arrays [3, 27, 41]. Recently, our group demonstrated a coded pyroelectric sensor array for modest resolution and low computational cost sensing. In this chapter, we describe similar codes for fiber optic tracking networks. A code (here we focus on binary codes), is a number array which is used to guide the deployment of the sensor array or modulate the sensor's receiver pattern. A coding design helps to reduce data redundancy inherent in video sensor networks. Such a system is a low cost, low data transmission substitute for video cameras.

The physical coding is carried out by multiplex sampling and sensor receiver pattern modulation. The sensor receiver pattern describes a sensor's sensitivity to the points in object space. A multiplexed sensor measures multiple regions in the object space. By multiplexing, one is able to physically encode the object space to

achieve efficient mapping between an object state and a sensor state. Multiplexed sampling can be realized in several ways. Volume sampling is the most natural since many optical sensors are sensitive to a volume. An example is reference structure tomography [10], where the sampling volume is modulated by optical components such as lenses and masks. Boundary sampling is a second multiplexed sampling example where each sensor measures the object parameters on the boundaries between volumes. We here employ a third approach, line sampling, where samples are taken along a curve. The flexibility of a curve provides additional freedom in multiplex sampling design.

A line sensor is a curve wandering through the object space. The sensor is triggered if any object touches this curve. In this paper, we use plastic fiber as a line sensor. A variety of fiber strain sensor [42, 64] designs exist, and we have selected what we believe to be the lowest-cost technique. A pressure applied to a fiber causes microbending, which in turn leads to a reduced transmission through the fiber. A series of microbending based fiber sensors have been built to sense vibration, pressure and other environmental effects [8, 55, 61]. In our work we configure the fiber as a binary sensor by setting an appropriate detection threshold. When a group of sensors are deployed in the object space, the presence of an object will trigger response in one or more fibers. A binary sensor state vector is created with each element of the vector corresponding to the response of a single fiber. The relationship between object position and the corresponding binary sensor state vector can be defined in a look-up table. During reconstruction, the look-up table serves to localize the object with the binary measurement vector. A properly designed codebook containing the lookup table for the object space can lead to efficient and robust sensing.

2.2 General System Model

To demonstrate the coding concept, a line sensor web is deployed in this 2D object space as indicated in Fig.2.1(a). The binary response function of ith line sensor is described by the equation

$$h_i(x,y) = \begin{cases} 1, c_i(x,y) = 0 \\ 0, c_i(x,y) \neq 0 \end{cases} \tag{2.1}$$

where $c_i(x,y) = 0$ describes the curve of ith fiber. The function $c_i(x,y) = 0$ when (x,y) is on the curve, while $c_i(x,y) \neq 0$ when (x,y) is not on the curve. In Fig.2.2(a), the equations for fiber curves are $x - x_i = 0$ and $y - y_i = 0$, where x_i and y_i are constants. The object is assumed to be a rigid body with finite area and has a degree of freedom in both translation and rotation. The binary pressure field produced by the object is described by

$$S(x,y,x_0,y_0,\theta_0) = \begin{cases} 1, (x,y)\text{if within the object} \\ 0, \text{otherwise} \end{cases} \tag{2.2}$$

where (x_0, y_0) denotes the center of mass and θ_0 denotes the rotation.

When the response field of a particular line sensor as described in eqn (2.1) overlaps with object pressure field described in eqn (2.2), an output signal '1' will be generated. In a multi-sensor scenario, the system response is a binary vector, which will be called the sensor signature vector. If the object is a point object at

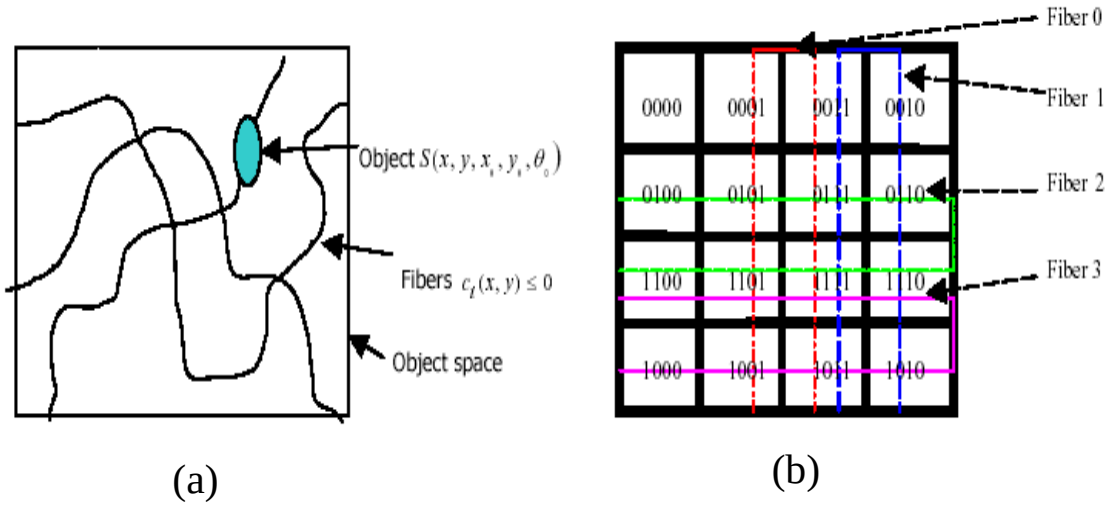

Figure 2.1: The configuration of the fiber sensor network. (a) random deployment without pad layer or cell segmentation. The object space is a 2D space. A fiber will be triggered if it is covered by the object. (b) with pad layer or cell segmentation coded by 2D gray code. The arrows indicate how each fiber should be deployed.

(x, y), the signature vector can be written as,

$$M_p(x, y) = \left\{ \begin{array}{l} h_1(x, y) \\ \vdots \\ \vdots \\ h_l(x, y) \end{array} \right\} \tag{2.3}$$

For a point object, the only non-zero element in this vector is the response of the fiber at the location of the point object. In the real world, the fiber is not an ideal line and the object can not be an ideal point. An object can be approximated as a point object if its size is about the same as the width of the fiber. Most other objects should be regarded as extended objects. For an extended object, the signature vector is written as,

$$M(x_0, y_0, \theta_0) = \left\{ \begin{array}{l} m_1(x_0, y_0, \theta_0) \\ \vdots \\ \vdots \\ m_l(x_0, y_0, \theta_0) \end{array} \right\} \tag{2.4}$$

Note that M is now the function of both the object's location and orientation. The relationship between general sensor signature $M(x_0, y_0, \theta_0)$ for an extended object and $M_p(x_0, y_0)$ for a point object is described by

$$M(x_0, y_0, \theta_0) = \bigcup_{\forall(x,y), S(x,y,x_0,y_0,\theta_0)=1} M_p(x, y) \tag{2.5}$$

and

$$m_i(x_0, y_0, \theta_0) = \bigcup_{\forall(x,y), S(x,y,x_0,y_0,\theta_0)=1} h_i(x, y) \tag{2.6}$$

where $\cup$ denotes inclusive OR. The distribution of signature vectors across the object space is called the signature field. Several signature fields for each fiber sensor are drawn in Fig.2.2 and the model of a human foot used in Fig.2.2 is shown in Fig.2.3. In our model in Fig.2.3, we assume an even contact pattern between the human foot and the floor since for localization purposes, one only needs to assume the contact pattern occupies a finite area, not to find out its actual shape.

In general, for a non-circular object, the coverage of fibers changes with both the position and orientation of the object. This makes a fiber web a good candidate for estimation of both object location and orientation. However, in this paper we will limit our attention to using the fiber web for object localization. Physically, we can introduce a pad layer between the fiber web and the object so that the object will apply pressure to the pad and the pad will transfer the pressure evenly to all the fibers under it. The pad lowers the sensitivity to object orientation, but it reduces the possibility of the object missing the fiber sensor.

From Eqn(2.5) and (2.6), it seems there would be some problems when mapping the location of an extended object to a signature vector because different combinations on the right hand side can yield the same signature vector, resulting in ambiguity. On the other hand, when an extended object rotates about its center of mass, the combination on the right hand sides of eqn(2.5) and (2.6) changes, which means an object can be mapped to several signature vectors even if its center of mass does not move. If we don't pay attention to these problems, even if the signature vectors of point objects are guaranteed to differ from each other from pad to pad, the signature vector of an extended object might be the same when the object stays in different locations. Specifically for a walking human, the two feet will touch the ground alternatively or simultaneously. Correspondingly, a human can be treated as a single object when his feet are within one cell or an extended object when his

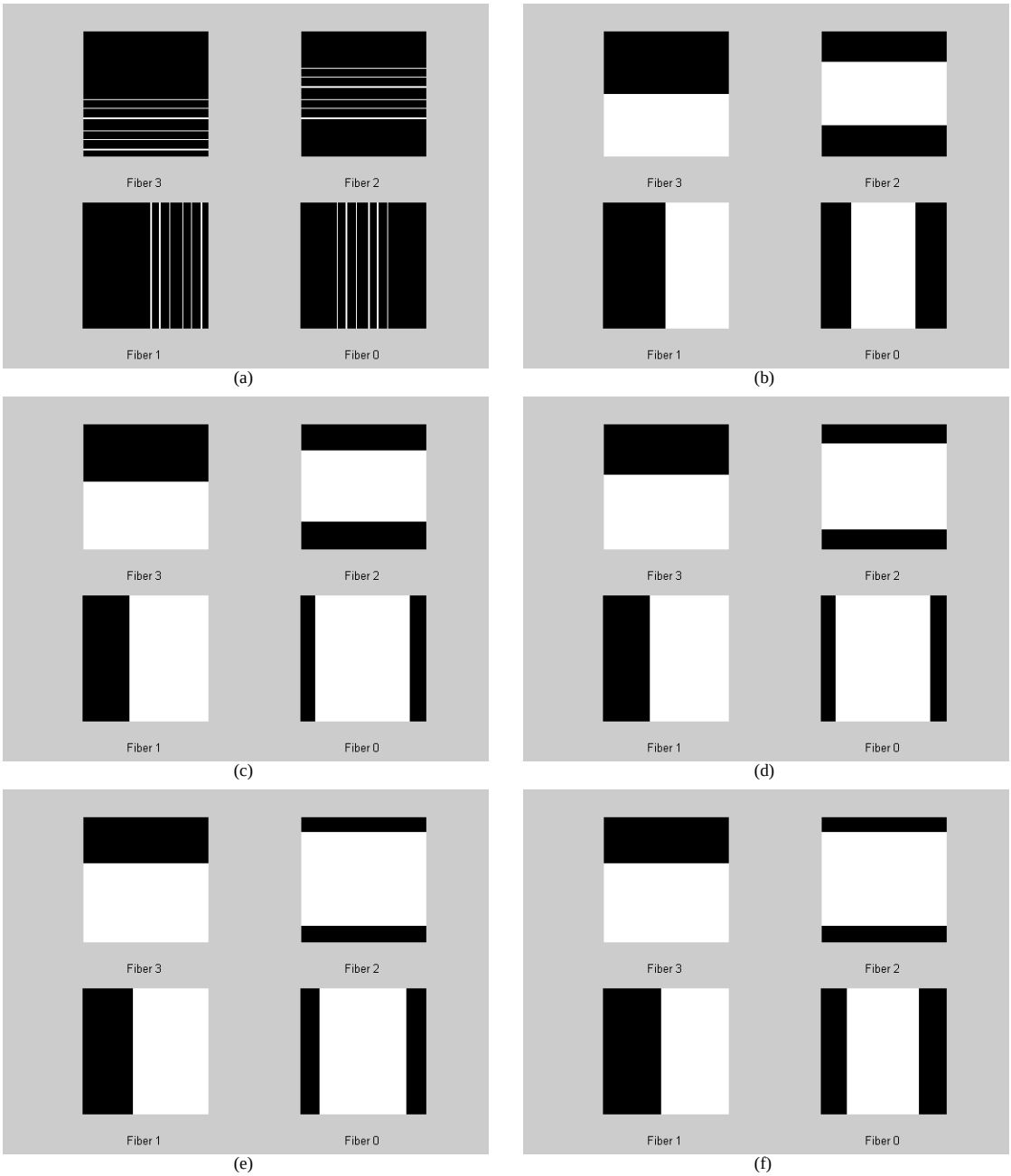

Figure 2.2: The signature field for each of the four fiber channels. The object space is 4 feet by 4 feet. (a) For a point object. According to eqn (2.4). This one actually shows the layout of each fiber in the object space. The fibers are deployed parallelly within the object space. (b) For a circular object whose diameter is 0.4 foot. (c) For a human foot model tilted at 0°. The human foot modeled is shown in Fig.2.3 (d) For a human foot model tilted at 30° (e) For a human foot model tilted at 60°(f) For a human foot model tilted at 90°.

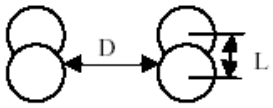

Figure 2.3: The model of a human foot. Each foot consists of two circles separated by L and the separation between the two feet is D.

feet occupy multiple cells in a small neighborhood. The same problem for general extended object tracking occurs in human tracking as well. Thus an effective object space coding scheme is required to solve the potential problems mentioned above.

2.3 Optimized Coding For Single Object Localization

To accomplish the multiplexed sampling, the space is segmented into discrete cells on a rectangular grid. The size of each cell determines the resolution. The points within cell $c(i,j)$ satisfy $g_{ij}(x,y) \leq 0$, where the $g_{ij}(x,y) \leq 0$ describes the shape of the pad. All the points within the same cell $c(i,j)$ have common sensor signature vector $M(i,j)$ for a point object, where (i,j) denotes the spatial coordinate of the cell and $M(i,j) \in (0,1)^l$. After we discretize the object space with a pad, for an extended object, the sensor signature vector is given by

$$M(x_0, y_0, \theta_0) = \bigcup_{\forall(i,j),\exists(x,y),S(x,y,x_0,y_0,\theta_0)=1 \& g_{ij}(x,y)\leq 0} M(i,j) \tag{2.7}$$

By comparing equation (2.5) and (2.7), we find the pad layer reduces the number of elements in the right hand side of eqn (2.5), which makes the overall sensor response

more predicable.

To locate a point object, one should carefully assign a unique signature vector $M(i,j)$ to the cell (i,j). Many coding schemes are possible, but a self-revealing code is preferred to prevent an expensive search in the look-up table to identify a cell index given a sensor signature vector. A binary code with l sensors can detect up to $2^l - 1$ object states. Each bit in the codeword for a cell can correspond to a fiber passing through the cell.

For an extended object, the right hand side of eqn (2.7) may have several terms, corresponding to sensor signature vectors of several adjacent cells. Since the final sensor output is the Boolean sum of these sensor signatures, and it will correspond to the signature of a particular cell, proper coding will locate this cell within a small neighborhood enclosing all those cells included in eqn (2.7) and the approach will provide consistent and controllable localization error.

The size of the neighborhood should be minimized to reduce localization error. In a rectangular grid, a good candidate for the neighborhood would be a square. Assume the size of the minimum square enclosure for the object is $k \times k$ cells regardless of its location. It is obvious $k > 1$ for an extended object because it could lie on the boundary between cells. There are $\sum_{i=1}^{k^2} \begin{pmatrix} k^2 \\ i \end{pmatrix} = 2^{k^2} - 1$ possible combinations within any $k \times k$ square enclosure while we only have k^2 sensor signatures within this square. We want to design a coding scheme where the result of the boolean sum operation of all the $2^{k^2} - 1$ combination is always one of the k^2 sensor signatures.

Assume all the signature vectors within a $k \times k$ square form a set Λ_2. The above statement requires the set Λ_2 to be closed with the Boolean sum operation. Notice that every $k \times k$ square within the whole object space should satisfy this condition. This problem seems daunting if we start from a 2D object space. However, we can

show that the problem can be reduced to one dimension by finding a qualifying 1D coding scheme which will allow the construction of the corresponding 2D coding scheme.

Consider a 1D signature vector set $\Lambda_1 = \{q_{0,}q_{1,}\ldots q_k\}$ consisting of k adjacent elements that satisfy $q_i \cup q_i \in \Lambda_1, (1 \leq i, j \leq k)$, where q_i is a binary codeword with fixed length and $i = 1, ...k$. We design the corresponding 2 dimensional code by forming the set

$$\Lambda_2 = \Lambda_1^2 = \left\{ \begin{array}{cccc} q_1q_1 & q_1q_2 & \cdots & q_1q_k \\ q_2q_1 & q_2q_2 & \cdots & q_2q_k \\ \cdots & \cdots & \cdots & \cdots \\ q_kq_1 & q_kq_2 & \cdots & q_kq_k \end{array} \right\} \tag{2.8}$$

Here q_iq_j indicates a compound codeword by putting codeword q_i and q_j with the specified order.

By simple calculation one can verify $q_{i_1}q_{j_1} \cup q_{i_2}q_{j_2} = (q_{i_1} \cup q_{i_2})(q_{j_1} \cup q_{j_2}) \in \Lambda_1^2 = \Lambda_2, (1 \leq i_1, i_2, j_1, j_2 \leq k)$. The proof above indicates 2D coding can be constructed by assigning two 1D-coding schemes to separate coordinates.

Suppose our codeword system has l bits. We first put part of the 2^l codewords along an array $(q_1, q_2, ..., q_i)$, where $i \leq 2^l$. Note that this is not a set, the order of the codewords is critical to our application. For arbitrary k, if any adjacent t $(1 \leq t \leq k)$ codewords in this line form a set closed with the Boolean sum operation, we call it a k-neighbor-local code. If the locations of these k elements are arbitrary, (i.e. these elements are not necessarily adjacent to each other) the problem can be solved efficiently by the use of single user tracking superimposed codes [18]. However, since we only require these k elements to be within a small neighborhood, we have more degrees of freedom in our design and expect a higher code efficiency (more available codewords given a fixed number of bits), the single

user tracking superimposed code becomes a suboptimal solution. We have generated several examples and . One example of 3-neighbor-local code is shown in Fig.2.4 with $l = 4$. Only 15 out of 16 binary codes can be used in this case. We have observed that when k increases, the percentage of usable $l - bit$ codewords for k-neighbor-local code decreases. This is understandable since increasing k imposes more constraints on the coding design.

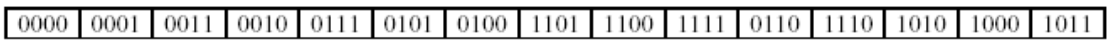

Figure 2.4: A 1D 3-neighbor-local code with $l = 4$.

We have not found a general optimal solution for arbitrary k. However, when $k = 2$, we are able to find an optimal solution by making use of all 2^l codewords and we will demonstrate that $k = 2$ is the optimal choice for our human tracking application. If the adjacent codeword only changes 1 bit, one can always guarantee

their Boolean sum is closed within this 2-element set. Such a code is the popular gray code. A binary reflected gray [25] code corresponds to a Hamiltonian circuit of an l-hypercube so that no codeword is wasted. For a typical human localization scenario where the result might be used to orient video cameras or a microphone, the resolution comparable to the width of a human waist is sufficient. This leads to a cell size of about 1.5 feet by 1.5 feet. With these dimensions, the length of 2 cells is a little greater than a walking human step length [63], as a walking human won't occupy more than 2×2 cells in most cases. Thus a 2D gray code with cell size 1.5 feet by 1.5 feet is a cost effective choice for indoor human tracking. In addition, with 2D gray code, the coding efficiency is maximized, an n bits codeword can be used to encode 2^n cells. A fast decoding algorithm exist with the computational complexity $O(n^2)$, which is extremely useful for large scale application.

A 2D gray code constructed according to Eqn (2.8) is shown in Fig.2.1(b) for $l = 4$. This construction is one example of the available 2D gray codes. It was selected for its simplicity of construction from its 1D counterpart. It is easy to be implemented with line sensors since any bit in the codeword will be constant along a row or a column. This allows us to configure the corresponding fiber sensor along its row or column. The localization error when using a 2D gray coding scheme is at most 1 cell, as we will demonstrate later.

2.4 Experiment Result and Discussion

We use a commercially available E1000 jacked plastic fiber as our fiber sensor. Each channel connects a photo darlington(IF-D93) detector to a fiber optic LED (IF-E96) light source. The light source/detector pair has a low unit cost and can be assembled by simple plug and play. The fibers go through a small mat, as shown in Fig.2.5.

The holes in the mat and the crossing fibers help to creates microbending along the fiber when there are objects on it. The fibers are woven densely enough to simulate a pad layer so that every human step can be detected. We set the threshold at 90 percent of the original transmission efficiency, which corresponds to about 60 lb of weight distibuted over an area about the size of a human foot. A carefully selected threshold value prevents false triggering from pets or other small objects. Although we design our system for human tracking, an object with similar size and weight to a human will be tracked as well.

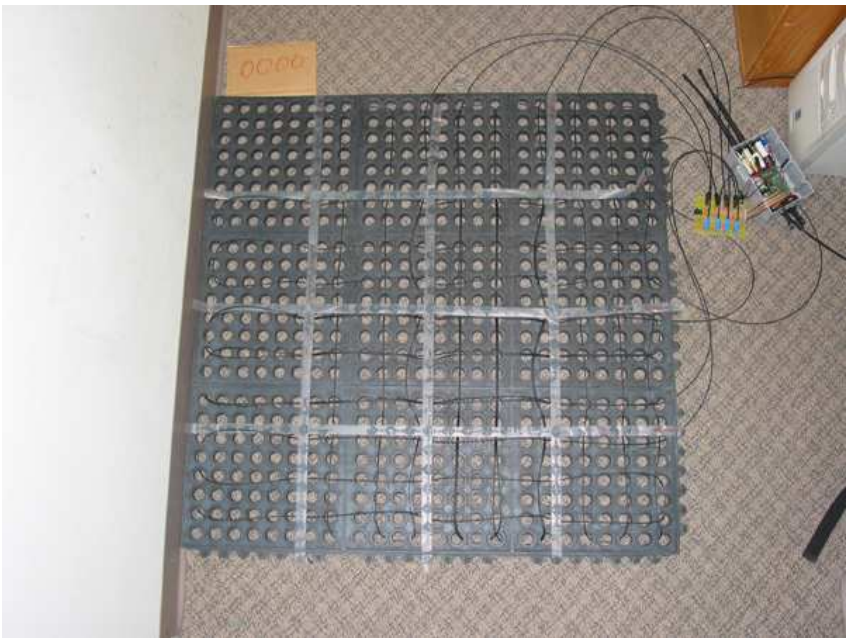

Figure 2.5: A fiber mat system with 4×4 cells woven on a single mat. A complete signal channel consists of LED, fiber, darlington receiver, A/D board and wireless unit.

Along a particular fiber channel, the detected signal in the photodarlington is transmitted into a microcontroller, which performs A/D conversion, then transimitted wirelessly to a computer for centralized processing. The wireless transmission

is handy when the mat is far from the processing unit or the designs call for multiple fiber mats to act as a distributed sensor network. For a multiple fiber mat configuration, some of the data processing can be shifted to a local unit to further decrease the bandwidth requirement in wireless transmission.

The fiber wired mat is one of our solutions for the construction of a portable, robust fiber floor. One advantage in the design lies in the fact that it can be transportted to another location without another calibration. Another design concept is to weave fiber on a hardware net, as shown in Fig.2.6. The grid structure on the hardware net improves the fiber floor sensitivity to pressure. A design concept is twisted fiber cable as shown in Fig.2.7. Compared to the previous two configurations, the twisted setup can help fibers generate microbending without an external grid structure so that it can be deployed in a much more flexible way. However, with this design, one will need to do the fiber weaving and calibration every time it is moved to a new location.

The 2D gray code discussed in the last section was employed to weave an experimental fiber mat with 4×4 cells. We use the fiber mat to track the human by detecting his footsteps. Each foot is modeled as two circles close to each other with little overlap, as shown in Fig.2.3. At any time, there could be one or two feet detected due to the nature of human walking. When there are two, based on the discussion in the previous section, we assume the two feet are always within a square of 2×2 cells. Fig.2.8 shows the software interface for such system. The blue dot indicates the current position of the human being tracked.

To illustrate how the system works, a person walks on the mat along a cell sequence $(0001) \rightarrow (0111) \rightarrow (1101) \rightarrow (1011)$ as shown in Fig.2.9(a). The raw signals from 4 fibers during his walk are shown in Fig.2.10. We find the noise level is about 2 percent of the transmission when there is no object on the fiber. On

Figure 2.6: A fiber floor weaved on a hardware net.

Figure 2.7: The configuration of a twisted fiber pair. The twisted fibers do not rely on external stucture to generate microbending. One of the two cables can be the regular wire, or they can both be fibers to provide redundancy for a single signal channel.

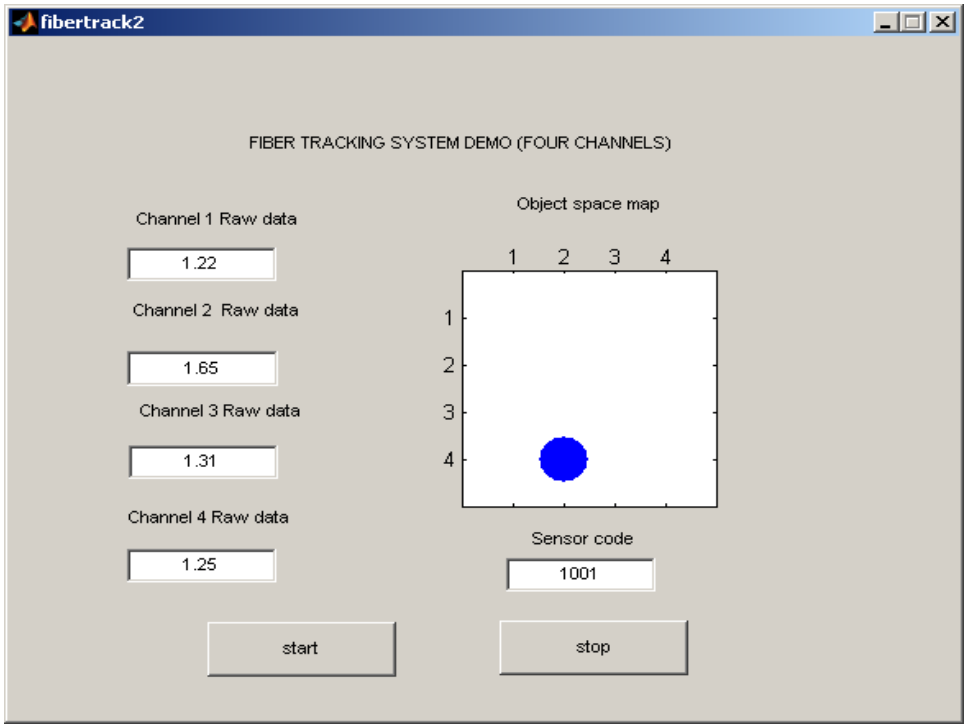

Figure 2.8: The system interface showing the raw data and the location of the object being tracked.

the other hand, for humans who weigh from 130lb to 180lb, the transmission in the fibers will typically drop between 85 percent and 60 percent of the original value. Thus the threshold for each fiber channel is set about 0.1 volt (corresponding to 90 percent) below the average reading with no object present. The corresponding sensor signature of Fig.2.10 is shown in Fig.2.11. For each fiber, the sensitivity to pressure is a function of location and local weaving condition. The pressure from the human is a function of walking speed, stance time, foot-floor contacting area, etc. All these variations explain why the raw sensor output shown in Fig.2.10 is not a constant when the corresponding fiber is pressed by human footsteps.

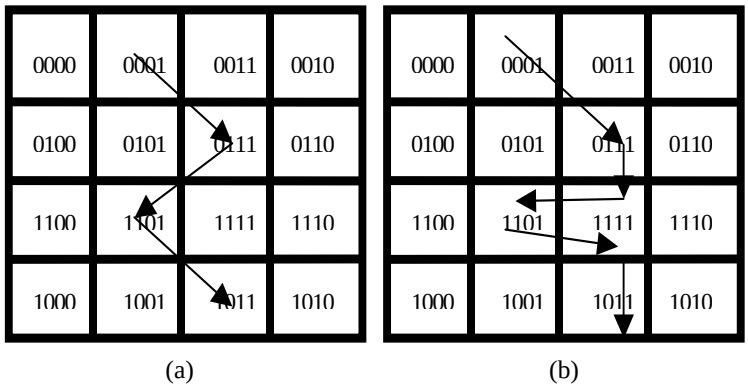

Figure 2.9: The original and reconstructed object locus. (a) Original object locus (b) Reconstructed object locus.

The detected signature sequence read from Fig.2.11 is (0000) → (0001) → (0111) → (1111) → (1101) → (1111) → (1011) → (0000). The signature sequence is mapped back to reconstruct the human locus, as shown in Fig.2.9(b). Note that the detected sequence is not an exact reproduction of the original sequence. The difference can be explained by understanding the human's walking

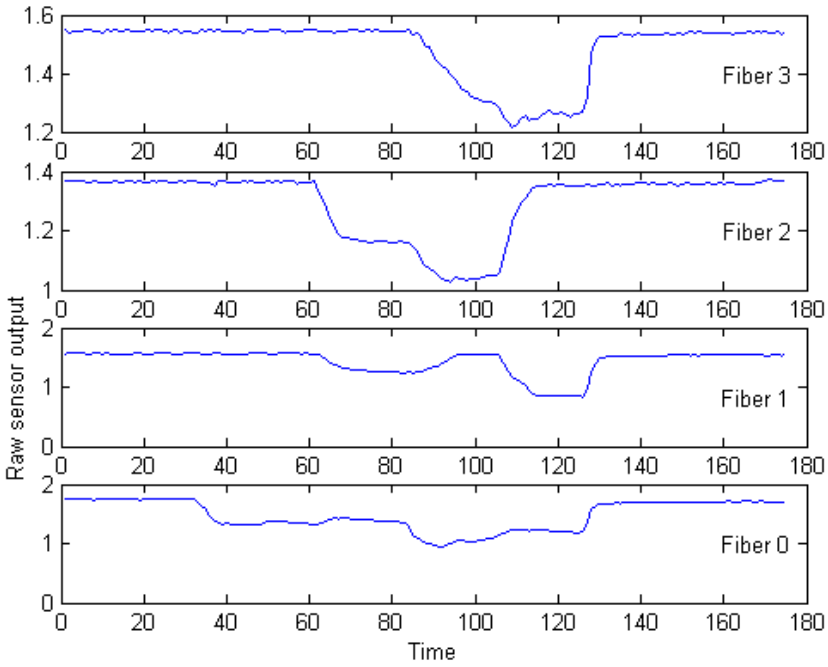

Figure 2.10: The raw sensor signals from four fiber channels.

style. When he enters the monitored area, he places his right foot on cell (0001). Then in the following order he places his left foot on cell (0111), lifts his right foot to step on cell (1101) while his left foot remains in cell (0111). There will be a short time frame when both of his feet are on the mat, producing an output of $0111 \bigcup 1101 = 1111$,which is confirmed by the next output signature. As the walker continues on, he lifts his left foot and move it towards cell (1011). Because only his right foot is in contact with the cell (1101), the next signature sequence in the detected sequence is (1101). Continuing on the walking trajectory, he next places his left foot on cell (1011). Because both feet are on the mat. The signature is $1011 \bigcup 1101 = 1111$, confirmed by the next signature in the sequence. Finally he lifts his right foot and moves off the mat. The next signature (1011) indicates the last position of his left foot and signature (0000) indicates he moves off the mat.

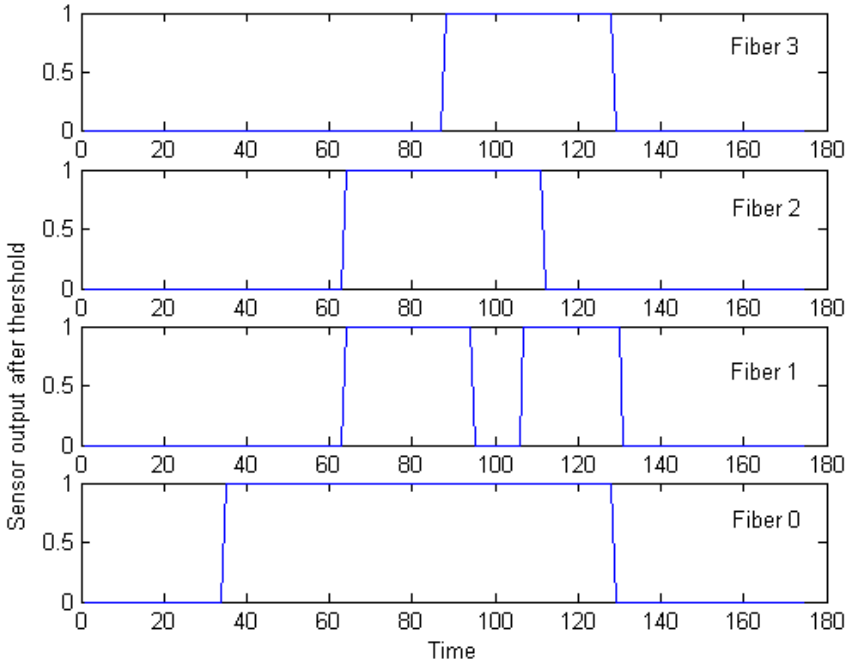

Figure 2.11: The detected signature sequence after threshold. The corresponding locus is mapped in Fig.2.9(b).

The above explanations are illustrated in Fig. 2.12.

Several observations can be made from the experiment results:

- When feet are confined within one cell, the tracking result is exact.
- When the human steps fall on different cells, but within a square containing 2×2 cells, the tracking result can be any cell within that square.
- The reconstructed locus not only corresponds to the actual locus, but it is also related to that particular human's walking style. Specifically, it depends on how often his two feet land on the floor simultaneously.
- The reconstructed locus is within an envelope determined by a series 2×2 squares spanned by successive cells in the original locus.

The observations indicate there are trade-offs between system resolution and robustness. A smaller cell size leads to finer resolution, but the system might malfunction when the object size is greater than expected. A greater cell size can lead to a more robust system at the price of lower resolution. For objects with a given size, however, the maximum tracking error is fixed at the maximum of cell size and object size. For a general system designed with k-neighbor-local-code, the cell size and value of k are two important parameters to be considered. There are two design rules for determining their values : $minimum - object - size \approx cell - size$ and $maximum - object - size \approx cell - size \times k$, where the minimum or maximum object size refers to the lower and upper bounds of the object size which may appear in the object space. A cell size smaller than minimum object size does not improve system resolution since the final tracking result will indicate any cell covered by the object. A cell size much greater than the minimum object size will yield too poor a resolution for tracking small objects. On the other hand, if $k \times cellsize$ is greater

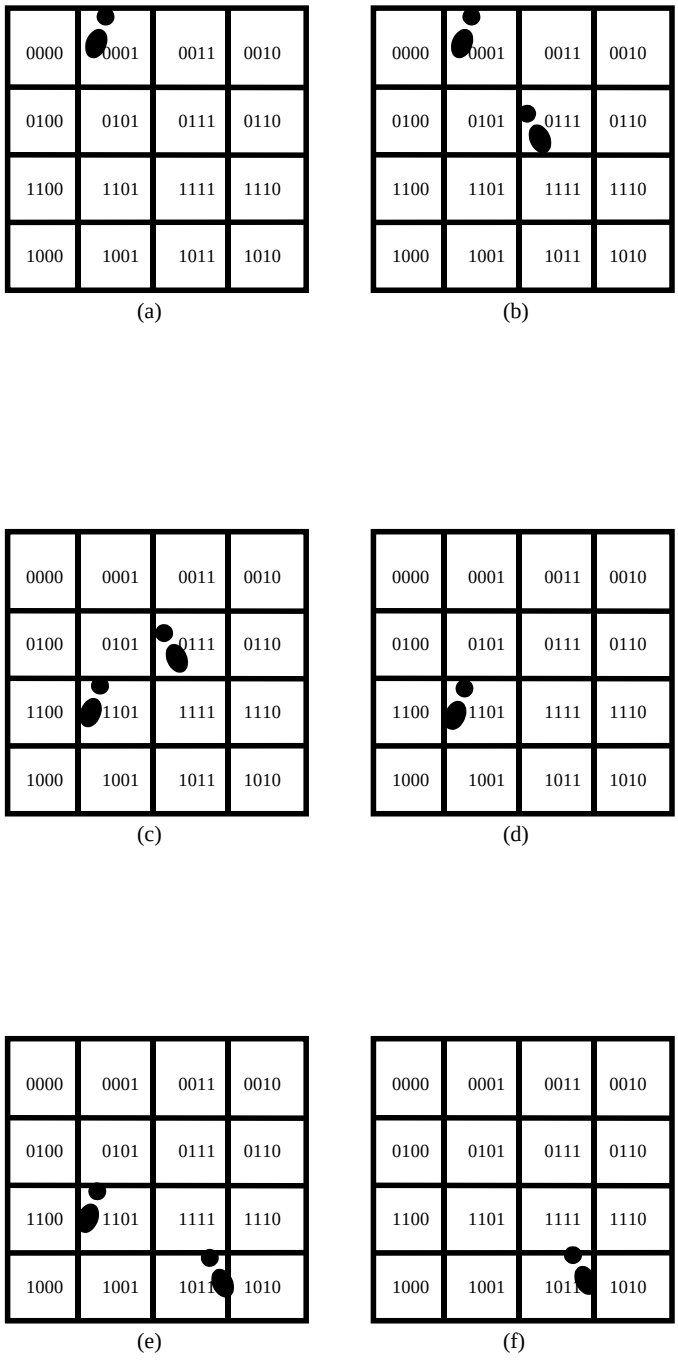

Figure 2.12: The reconstruction of human footsteps. The corresponding detected signatures for each subgraph are (a) (0001), (b) (0111), (c)(1111), (d) (1101), (e) (1111), (f) (1011).

than maximum object size,we will have over-designed the system and its sensing efficiency will be lowered. We will have to use more than the minimum required number of sensors. If the $k \times cellsize$ is smaller than the maximum object size, we will not get correct tracking results since the combination of the signatures from those cells covered by the object will be unpredictable. Other factors including cost and feasibility of manufacturing will also affect the system design.

For our human tracking system, the minimum object size equals the size of one foot and the maximum object size equals the maximum distance between the two feet when a human is walking. Here, we set the waist size as the minimum object size because it is a better indicator of the human's projection on ground. Also, 2 is the miminum value for k because no matter how small the size of an extended object, it is probable it will cover the boundary of two or more cells.

The experiment result and analysis actually shows another promising application of the fiber mat—human gait analysis. Recall that even if the human walks along a fixed locus, different walking habits will result in different reconstructed object loci. By comparing the original locus and reconstructed locus, one is able to measure the stance time, step length, double-support time, swing time, swing-to-stance ratio, and velocity. A similar gait analysis can be performed by a commercially available gait mat containing a large number of floor sensors [1]. However, the multiplexing design described here is a competitive low cost solution with a system resolution that can be tuned for specific needs.

Although the size of the demonstration fiber mat system is only 4×4, the scalability is not a problem. The computation time is $O(n^2)$ for n sensors so that there is no dimension curse. The layout is separable along each coordinate and is line sensor friendly as discussed in previous section.

Still, there remain some implementation issues. We find that the existance of

finite response time and relaxing time for microbending leads to certain undesired intermediate states. For example, by carefully examining Fig.2.11(b), one can see the falling edges of the fibers 0,1,3 near the 130th time step are neither vertical nor of the same slope, which means the signals do not drop to a logic '0' at the same time. Theoretically, the detected signature is supposed to change from (1011) to (0000) immediately; however, intermediate states such as 0011 appear, but disappear quickly. One simple solution is to filter out this short time state which a human walking at normal speed won't produce. The other potential problem apprears when the fiber sensors operate for a long time. The light transmission efficiency of the fibers will drift due to such causes as previous usage, change of fiber curvatures, change of fiber LED or darlington light efficiency, temprature shift, etc. Error results if we don't monitor the light level shift continuously and adjust the threshold correspondingly. One solution is to use a motion sensor to detect any intruders so that the fiber system can calibrate automatically when there is no human being in the monitored area.

2.5 Conclusion

A single fiber serves as a binary line sensor to detect the existence of a pressure field. Its line sampling makes it easy to modulate a sensor receiver pattern. The space is effectively segmented into cells by using the volume sampling method.A fiber web is built by object space coding with a 2D gray code for single object localization. Each cell is assigned a unique codeword by weaving the fiber in the object space. When objects trigger the fiber web, a codeword is detected and a look-up table is used to find the corresponding position. The object is localized after current measurement data is available so this fiber web corresponds to a static identification of object

states.

A fiber sensor web with such a channel coding design can have many advantages as compared to traditional counterparts. It has a multiplexing nature so that each sensor will be triggered by multiple object states. It is an efficient design, such that for N object states, only $\log_2(N)$ number of sensors and computations are required. Such efficient design also ensures only non-redundant information is transmitted in a wireless system to reduce bandwidth requirement. Finally, a large group of traditional channel coding designs can be readily employed in sensor systems.

The issue of deployment complexity is not addressed in this thesis. When the scale of the object space is large enough, say, $1000 \times 1000 ft^2$, it is hard to deploy a fiber web according to a structured coding scheme. In such a case, a random coding scheme is preferred. One can simply toss the fiber to reduce deployment cost and calibrate the system response once the fiber is deployed. A combination of a random and a structured hybrid-coding scheme may also be a good choice. The structured coding can serve as a first level to localize the object with low resolution, while in a small neighborhood, random coding could serve as a second level to improve the resolution.

Chapter 3

Multiple Object Localization with Superimposed Code

3.1 Introduction

In the previous chapter we introduced the fiber web as our test-bed for volume sampling. A 2D gray code was introduced as an example of geometric object space coding to localize a single volume object. Here we want to further generalize the coding scheme so that it will enable the system to localize multiple objects simutaneously. Fortunately, we find that the concept of the groups testing code, which was invented and applied before in other reasearch areas can be applied here.

Group testing enables the efficient identification of a very small number of active states among a large number of possible states by testing the states in groups instead of one by one. Since all tests are performed simultaneously and the group membership is decided in advance, the gnostic subject is referred to as nonadaptive group testing. Group testing or superimposed codes, as it is also known, was first introduced by W. H. Kautz and R. C. Singleton in 1964 [40]. Group testing has wide application in medical, chemical and electrical testing, coding, drug screening, pollution control and multi-access channel sharing [23].

Let us describe in more detail an example from communications. In a con-

ventional multi-access channel, multiple users share the same channel to exchange information. In order to avoid conflicts, active users identify themselves by transmitting a signature codeword before sending out their actual message. Codewords transmitted simultaneously are combined in the form of a bitwise OR in the communication channel. The receiver identifies the active users by inspecting the combined codewords and assigns an order to active users. A naive signature codeword is a codeword with length equal to the number of total users with the bit corresponding to each active user set to "1". However, in the common scenario when the number of active users is far less than the total number of users, a more efficient coding is achieved using nonadaptive group testing codes.

The isomorphism of a multi-access channel and a binary sensor array makes it possible to apply superimposed codes in multiple object localization as well. Consider a binary sensor array, in which each active object state triggers a selected group of binary sensors. Equivalent to a multi-access channel, the information associated with each object state and measured by the sensor array is a binary vector or codeword. When multiple active object states coexist, the final sensor output is the bitwise OR of the binary vectors. The original active object states can thus be identified by checking a look-up table when the final sensor output is unique for all possible combinations of active states.

The sensor array design based on nonadaptive group testing is an example of multiplexing sensor design, in which each sensor monitors a number of object states. We have been working on such multiplexing sensors by modulating the sensor receiver pattern [10, 27, 58, 59]. In particular, we have demonstrated a fiber web deployed according to 2D Gray code to localize an extended source in the previous chapter. [69]. Fiber, when used as a line sensor, allows the designer great flexibility to change its receiver pattern by weaving it in the object space. Here we

demonstrate the application of superimposed codes in sensor array design with a fiber web. Other applications of superimposed codes include file comparison [47], pattern matching [32], DNA screening [5] and multiaccess channel [44].

3.2 Design of Group Testing Code

Suppose that there are up to k objects coexisting in a space consisting of n positions, each of which may be occupied by no more than one object. To detect the positions of the objects, we use a sensor array with m binary sensors. A sensor returns 1 if an object is present in any position monitored by that sensor and 0 otherwise. Consider the $m \times n$ binary matrix T with $T_{ij} = 1$ denoting that sensor i monitors position j and 0 otherwise. Each row of matrix T corresponds to the receiver pattern modulation of that particular sensor. Each column corresponds to the sensors in the sensor array monitoring that position. In addition, column T_j can be seen as the codeword assigned to object position j. The state of the object space is described by a binary vector of length n with a 1 at position j denoting the presence of an object there. The state of the sensor outputs is described by a binary vector z of length m that is the element-wise inclusive logical OR of the columns T_j where position j is occupied by an object(i.e. sensor j is active.)

$$z = \bigcup_{j \in \text{Active}} T_j. \tag{3.1}$$

The elements of z are the independent binary measurements of each sensor.

In order to be able to discriminate the positions of up to k objects simultaneously present, the inclusive logical OR of all subsets of up to k columns of encoding matrix T must be distinct from each other. The set of codewords (or equivalently

the matrix T with those codewords as columns) with such a property is called the *uniquely decipherable code* of order k, abbreviated as UD_k [40]. The matrix is also called $\overline{k}$-separable matrix, as its columns form the UD_k code.

If we further require that the inclusive logical OR of up to k columns does not *cover* any other columns, then the matrix T becomes a k-disjunct matrix. A codeword vector x covers codeword vector y if and only if the elementwise $x \cup y = x$. The columns of a k-disjunct matrix form a ZFD_k code, the *zero-false-drop code* of order k [40]. A ZFD_k code is also a UD_k code, but not vice versa.

A ZFD_k code provides fast decoding, so the decoding time complexity can be reduced from $O(n^k)$ to $O(k)$ [31]. This property may be extremely useful if one is designing a large-scale sensor network. There are also universal construction algorithms for ZFD_k codes with a fixed distance between codewords. The guaranteed distance between any pair of codewords is useful for error detection and correction [57].

However, the ZFD_k code sacrifices coding efficiency to achieve the above-mentioned properties. A UD_k code does not provide fast decoding, but it can achieve higher sensing efficiency due to the fewer constraints on code construction. A UD_k code is ideal for small or medium scale applications where measurements are reliable. One needs to balance these factors to determine which code matrix is suitable for a specific application. Our primary goal is to design an encoding matrix T to maximize $\frac{n}{m}$, the ratio of monitored positions n to number of sensors m. By doing this, we can reduce the number of sensor channels, system complexity, cost, and bandwidth requirements.

There have been a number of algorithms for the construction of UD_k and ZFD_k codes since their first introduction in 1964 [40]. Macula introduces a uniform way of constructing a class of ZFD_k codes with distance 1 [46]. Ngo and Du provide

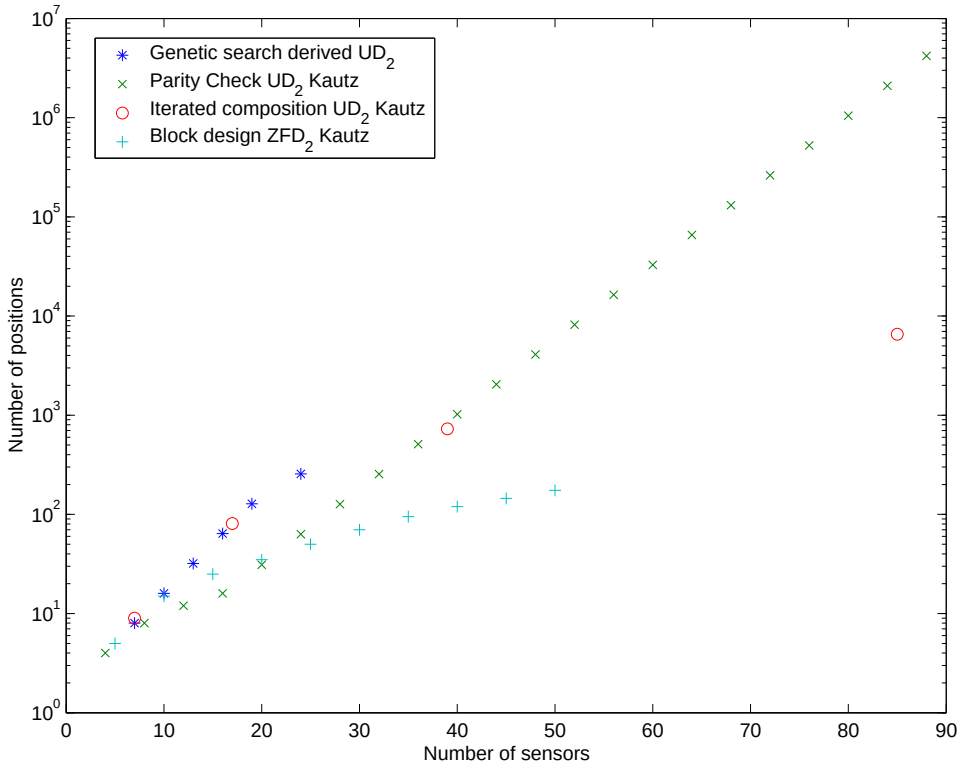

Figure 3.1: A comparison of the coding efficiency of several algorithms for $K = 2$

the construction of two families of k-disjunct matrices with general distance [57]. There are construcion algorithms for UD code with the discrete value of k, but not for arbitrary k. When $k = 2$, among those reported algorithms, the parity check based UD_2 construction algorithm of Kautz and Singleton has the highest efficiency. For small sizes of m, the iterated composition UD_2 method of Kautz and Singleton is superior. However, for larger m, their efficiency drops, and moreover, the construction is only available for sparse values of m. Fig. 3.1 shows a comparison of the coding efficiency of several algorithms. The ZFD_2 class of codes cannot compete with the other algorithms. The iterated composition of UD_2 algorithms is actually attractive before been overpassed by the parity check UD_2 when the number of sensor increases.

For our experimental demonstration, we localize up to 2 people simultaneously

within a moderate number of positions. The floor is segmented into 64 cells forming a square grid. Each cell is assigned a unique codeword from UD_2, the uniquely decipherable code of order 2. Since the number of positions is not large and the number of people present simultaneously is only two, we use a genetic search algorithm to find the most efficient UD_2 code to encode the object space of 64 positions. It turns out that only 16 sensors are required to localize up to 2 objects in 64 positions. It can be seen in Fig. 3.1 that the instances generated by our genetic search match or surpass the efficiency of all other construction algorithms.

The *fitness function* of the genetic search is the ratio of the number of unique sensor states over the number of all possible states of interest. In our case, the number of source states of interest are $2081 = \binom{64}{0} + \binom{64}{1} + \binom{64}{2}$ denoting the presence of up to 2 sources. A fitness value of 1 for an individual declares the finding of a UD_2 coding matrix.

The search starts with a generation of 40 individuals that are either random binary matrices or rows of non-optimal $\overline{2}$-separable matrices. A population of 120 individuals is generated by exchanging random pairs of rows between individuals (parents) to get new individuals (children). Random mutations may also flip a bit of an individual. The new generation is the set of 40 individuals from the population with best fitness function values. The search repeats the generation-population cycle until an individual with fitness value equal to one is found or an upper bound of iterations is exceeded.

To our knowledge, this is the first time a genetic algorithm search has been used in the construction of a $\bar{2}$-separable matrix. Although an estimate to the upper bound of columns given a fixed number of rows has proven to be $n(2, m) = O(2^{\frac{m}{2}}))$, no general construction method exists. Fig. 3.2 shows a pictorial representation of the $\overline{2}$-separable matrix we used for encoding the 64 space positions with 16 bit

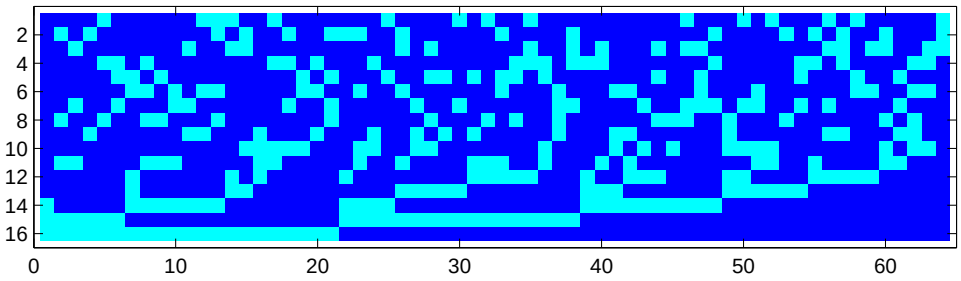

Figure 3.2: A pictorial representation of the $\overline{2}$-separable matrix we used for encoding the 64 space positions with 16 bit codes. Light colored tiles correspond to 1 and dark tiles to zero.

codes. The columns of the matrix have been reordered in reverse lexicographic order. Fig. 3.3 shows a pictorial representation of the unique sensor states for all possible object states with up to 2 active positions. These states are the signatures which will reconstruct the positions of the objects later.

There are several $\overline{2}$-separable matrices of optimal size. One can determine certain matrices as more attractive when the weight of the rows or the columns does not exceed a bound, or when the total number of non-zero elements is minimal. This is generally true because the more 1 in the matrix, the more deployment cost will be required since we need to modulate the receiver pattern of the sensor to make it monitor the corresponding region. In addition, the distribution of the 1 and 0 elements affects the ease and / or cost of fabrication. In our fiber floor sensor application, there are also considerations regarding the assignment of codes to specific cells in order to minimize the total length of the fiber used.

There are in total $2^{16} = 65,536$ available codes and only 2081 codewords used. If due to noise, a detected codeword is not one of the 2081 valid codewords, it is rejected. Since only about 3.2% of the total codewords are used, it is very probable that erroneously detected codewords are among the unused codewords.

The UD_2 codes do not have the error detecting property of ZFD_2 codes to

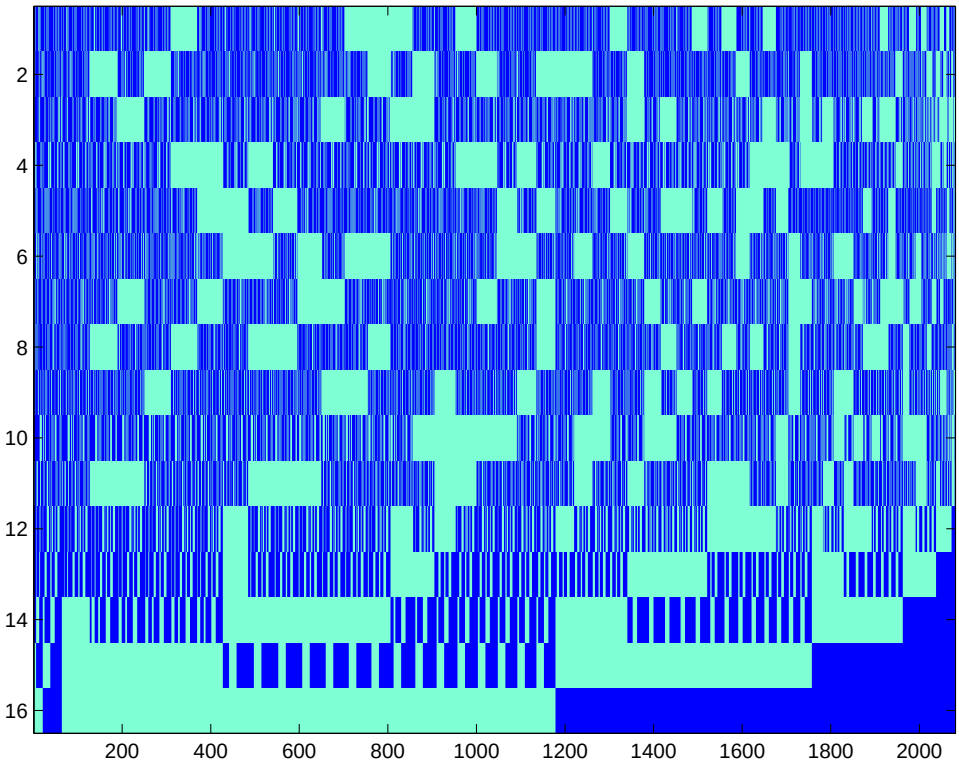

Figure 3.3: A pictorial representation of the unique sensor states for all possible object states with up to two active positions. Light colored tiles correspond to 1 and dark tiles to zero.

generate a code that does not belong to the code family whenever more than 2 objects are present in the object space. This is because according to the definition of 2-disjunct matrix, the union of any column vectors up to 2 columns does not contain any other columns. When there are more than 2 objects, the detected code is the union of more than 2 columns and contains more than 2 columns. This code can not belong to the ZFD_2 code family. For UD_2 code, we can only guarantee the union of any column vectors up to 2 columns is distinct. The union of more than 2 columns can be the same as one of the code in the UD_2 code family. Thus, when there are more than two objects, a code corresponding to two of the total objects might be generated. In addition, the fact that the object state is beyond the design capability of the sensor system may pass undetected. The above discussion applies to arbitrary k.

3.3 Experiment Result

The human model and fiber sensor response model are discussed in the previous chapter about single object detection. With a little modification of the experimental setup, we localize up to 2 people in 64 positions with 16 fiber channels. A hardware net layer is now used where the fibers are tied to form cells of 1.5×1.5 feet. The fiber cables are spread to produce the desired microbending when there is pressure due to somebody stepping on a cell. The floor is topped with a regular rug. This design makes the fiber floor net easier to deploy and move.

Fig. 3.4 displays a schematic diagram of the sensor array setup. Note that because each microcontroller can only accept 8 signal channels, we need 2 micro controllers simultaneously. The calibration of the threshold can be associated with an additional motion sensor. When the motion sensor detects there is no action

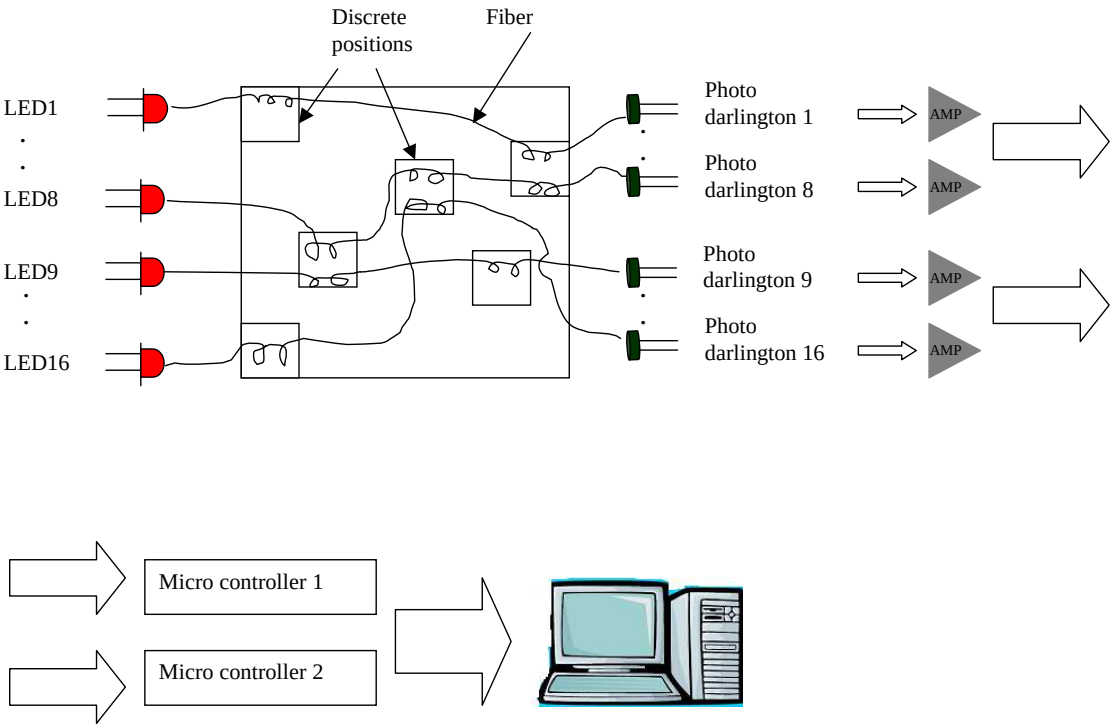

Figure 3.4: A schematic diagram of the sensor array consisting of fiber optic LEDs, the fiber cable arranged in cells, the light detectors, the signal amplifiers, the microcontrollers and the personal computer. The fibers beyond the discrete positions are grouped into bus lines so that they won't be triggered if someone steps on their intersections.

in the field, the system begins an automatic process of self-calibration, including threshold tuning. Once a person is detected, the system stops calibration and starts the tracking process. After thresholding, the binary data from these channels are grouped to form a binary vector. The detected binary vector is then decoded by a look-up table. However, unlike the 2D gray code used before to localize a single object, the UD_2 code does not have a universal fast-decoding algorithm. One can either employ a binary tree structure to speed table look-up by using more memory, or employ a hash table to balance memory usage and computation speed. In general, the overall computational requirements for the signal post processing of the fiber sensor array are still minimal and can be directly implemented in a micro-controller without the need for a personal computer. Fig. 3.5 shows the graphical user interface of our fiber tracking system. Fig. 3.6 contains a series of snapshots showing the detected trajectory of 2 humans moving simultaneously.

3.4 Conclusion

With this work, we demonstrate the deployment of fiber sensors with a coding design based on superimposed codes. We design the coding scheme and the fiber web to localize up to 2 people simultaneously. The sensor system detects the inclusive logical OR of the unique codewords assigned to each cell. The resulting codeword indicates the number of people and their positions. Both the coding design and hardware can be easily generalized to include more objects and more positions.

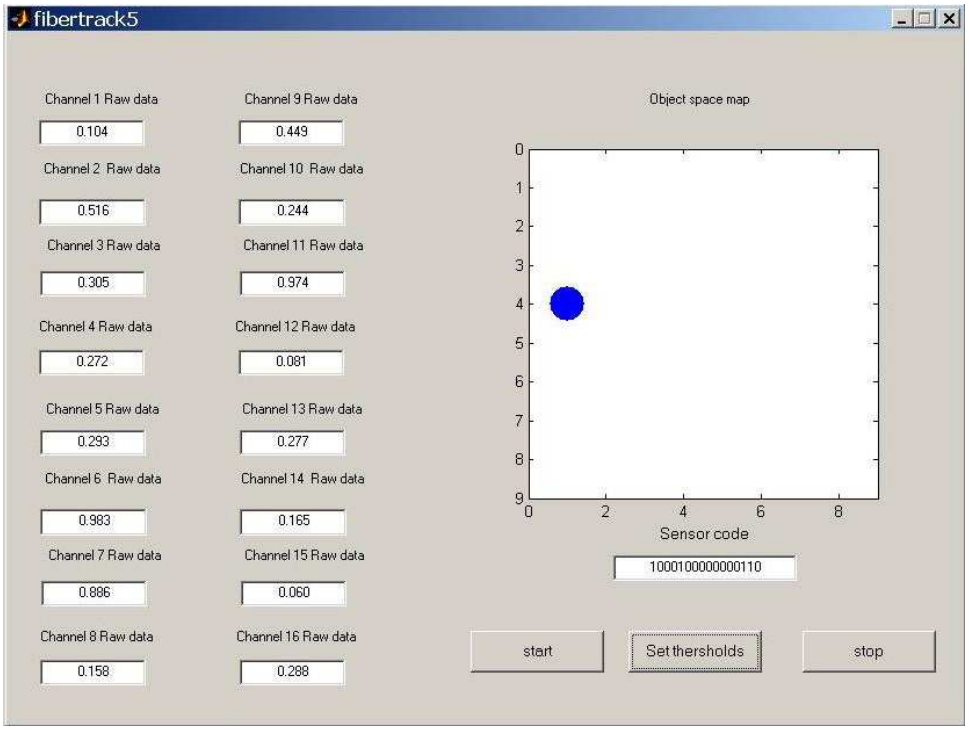

Figure 3.5: The screen shot of the interface for fiber tracking system.

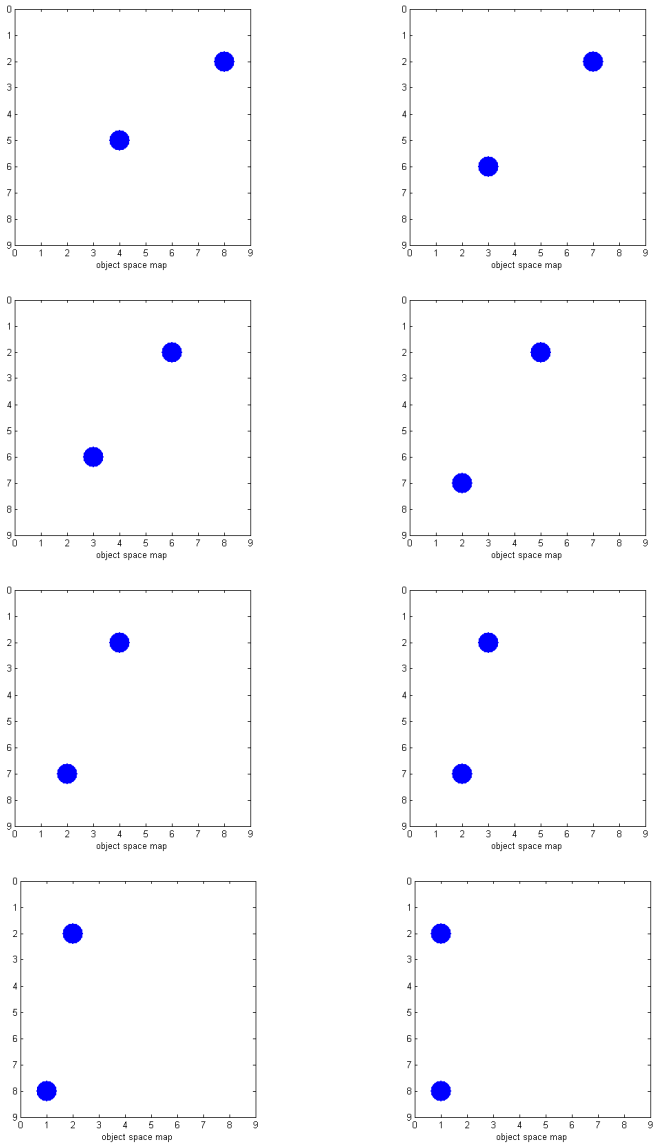

Figure 3.6: The snap shots of the space map when 2 objects move simultaneously. One moves to the left through a horizontal line and the other moves towards the left down corner.

Chapter 4

Object Tracking With Boundary Sampling

4.1 Introduction

The concept of distributed sensors that fire when a person or object crosses a geometric structure, as defined for example by laser beam, is familiar to fans of spy and crime cinema. Surprisingly little academic analysis has appeared as to the optimal deployment and capacity of this class of sensor networks, perhaps accounting for the universal ability of Hollywood spies to penetrate these systems. While we do not resolve this problem here, we do describe the object localization capacity of a particular class of geometric sensors. We hope by this incremental analysis to seed continuing scholarly interest in the design and analysis of such sensors.

In general, design of a sensor network for localization and tracking involves both spatial sensor deployment and receiver pattern modulation. Deployment strategies were previously considered in [30,62,67] assuming circular receiver pattern. Related work on sensor deployment and path identification used path exposure as a merit function to determine the quality of sensor deployment for target detection [17, 52–54]. Our group previously looked at receiver pattern design in [27, 58, 69, 70]. The aforementioned works employ a static sensing mode that tries to localize the

object based on an instantaneous detected signal vector/codeword, not a temporal sequence. Every point in the object space is thus encoded with a binary codeword by proper sensor deployment.

The physical interface of a sensor system is determined mappings between a target configuration and a physical distribution in the target embedding space and between the target distribution and the sensor state. In the case of photography, for example, the target is an object in a 3D embedding space and the mapping from the target to a physical distribution occurs when the target radiates or scatters light. The mapping from the light to the sensor state is mediated by lenses such that the sensor state is proportional to the light density on the target.

With interest in computational distributed sensor networks, novel conceptions of the target-field and field-sensor mappings have arisen. Using various approaches to sensor deployment, optical and electronic interfaces and object radiation, it is possible to design sensors that are sensitive to objects occupying specific points, average values of objects covering a volume or objects crossing a curve or a surface. Expanding the concept of the object embedding space to include hypercubes of spatial, spectral, coherence, polarization and even chemical information, it is possible to imagine many interesting relationships between the sensor state and the target state.

The sensor receiver pattern is a general approach to describe the object field-sensor data mapping. The receiver pattern describes the points in the object space to which a given sensor is sensitive. For example, a pixel on a camera may be sensitive to all points along a specific ray extending from the camera or a motion sensor may be sensitive to all objects moving in a specific volume.

The goals of sensor system design are to maximize the capacity of the system relative to operational metrics (such as probability of detection, localization and

classification) while minimizing system complexity, computational complexity and latency, deployment and operations costs and power consumption. Many different classes of sensors and algorithms may be applied to a sensing problem, ultimately sensor design must decide between these approaches. The sensor receiver pattern is the most fundamental determinant of the capabilities of a particular sensing approach.

As an example of a novel approach to sensor system design and analysis, this paper considers boundary sensors. While the concept of a boundary sensor is by no means novel, detailed analysis of the tracking and localization capacity of networks of boundary sensors has not, to our knowledge, been previously presented. A boundary sensor in N dimensions is triggered when an object crosses an N-1 dimensional structure. Along the number line, a boundary is determined by a discrete set of points. In two dimensions a boundary is a curve or a set of curves. In three dimensions a boundary is a surface. The detection of boundary crossing can be realized by video cameras [7, 68], pyroelectric sensors [22, 27],laser beams and optical fibers. When the object moves in the object space continuously, it triggers a series of boundary sensors. The object can then localized based on detected sequence of activated boundary sensors, which will be called a *sensor sequence*. An example is shown below in Fig.4.1

In contrast with staring sensor networks, the bounary sensor network uses temporal sequences of sensor events to encode spatial information. A spatial location can be reached by multiple paths generating distinct sequences, but ideally each sequence corresponds uniquely to the current object location. In order to localize a moving object based on the detected sequence, each distinct path must be associated with a unique sensor sequence. (We consider all paths through the same sequence of boundary defined regions to be identical.) A one-to-one mapping from

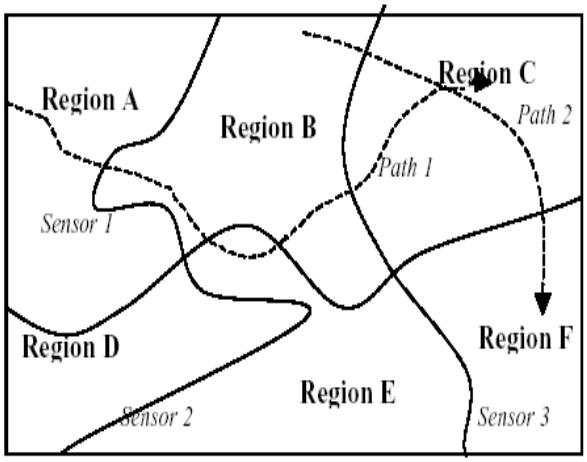

Figure 4.1: An object space monitored by the boundary sensor network. An object entering the object space along path 1 induces sensor sequence (2113), while path 2 induces sensor sequence (31).

sequences to paths requires proper deployment of the boundary sensors. In general, it is also desired in that deployment scheme multiplex sensors so that we can track the object in as many regions as possible with as few sensors as possible. At the same time, certain system robustness should be introduced to combat noise and system error.

The primary results presented in this paper are limits on the localization and sensor sequence generation capacity of boundary sensor networks. We define m to be the number of boundary sensors in the network and n to be the length of the sequence used to track the object state. We assume that

(a) There is at most one object moving in the object space.

(b) At any instant the object can only trigger up to one boundary sensor.

(c) The boundary sensor will not miss any boundary crossing event.

(d) A given boundary sensor defines the boundary between a given object location and at most one other object location. (The object cannot change location by triggering the same sensor twice in immediate succesion. Events in which the same sensor is triggered twice in a row drops that sensor from the current sequence.)

4.2 One dimensional boundary sensor systems

In one dimension, a boundary sensor triggers when a target passes a point. Our model allows a single sensor point to be assigned detection points along the number line. The model is motivated in part by an understanding of the 1D boundary segmentation as a projection of a 2D map. In two dimensions we will allow the boundary to curve arbitrarily. In crossing a 2D segmentation in 1D, we could cross the same curve multiple times, thus creating the possibility of firing the same sensor at multiple points. In a particularly simple case, one can imagine the use of crossed 1D boundary structures for localization on a 2D space. This approach is illustrated in Fig.4.2. The x and y axes are each segmented by arrays of line sensors. By connecting these line sensors using curves outside the object space, we create 1D boundary sensors with repetitive coding for each sensor. Allowing a single sensor to curve through the space multiple times allows us to minimize the number of sensor used.

Whether or not it is actually deployed in 2D as illustrated in Fig.4.2, a 1D boundary sensor may be represented graphically by a segmentation of the number line. A line and a number of sensor denoted as points spread over the line are shown in Fig. 4.3. Each points stand for a 'probe' from a particular sensor and a sensor can have multiple probes. Our goal is to arrange these points so that given m

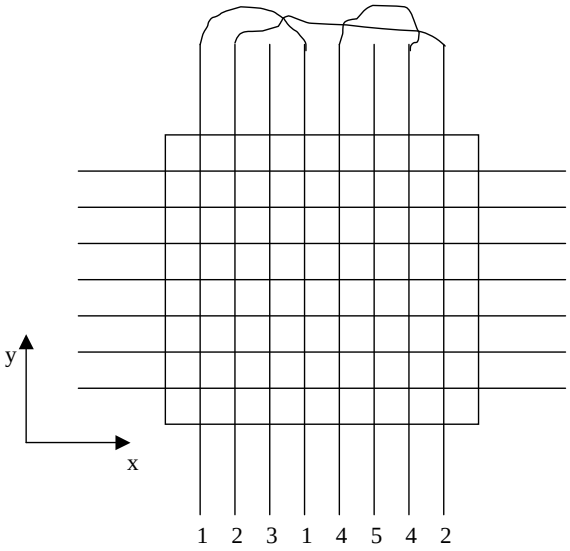

Figure 4.2: An example of the separable Cartesian gird design. The parallel lines are the boundary sensors. Lines are connected at one end if they are the same sensor. The numbers of the sensors are marked at the other end of the sensors. The sensors parallel to x axis have the identical numbering but are not marked in this figure. During operation, the x, ycoordinates of the object are identified separately.

sensors and a desired sensor sequence of length n, every consecutive point sequence of length n is unique among all such sequences deployed along this line. For example, in Fig. 4.3, given $m = 5$, $n = 3$, one can list all the consecutive point sequences of length 3: 123, 231, 314, 145, 454, 542. We will discuss bi-directional sequence later. We find all such point sequences are unique. Thus the deployment sequence $D = (12314542)$ satisfies our requirement, where D consists of all the sensor probes along the line from left to right and $D \in [1, 2..., m]^l$, l is the number of sensor probes. However, this deployment sequence is clearly not the longest one. We can add another 5 to the deployment sequence to form a longer one that satisfies our requirement. A longer deployment sequence means one can use the same amount of sensor resources to monitor a larger space. Our design goal is to find the longest possible sequence for a given length n. We will start from the formal definition of the deployment sequence.

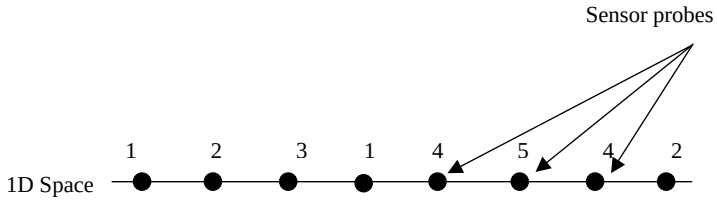

Figure 4.3: The line model of 1D boundary sensor deployment. The line indicates the 1D space. The points stand for the probe of a particular sensor. Each sensor can have multiple probes.

Definition 4.2.1. *A* Deployment Sequence *is a arrangement of sensor points in 1D space described by a number sequence wherein each number stands for a sensor.*

Definition 4.2.2. *A* Sensor Sequence *of length n is an n symbol subsequence of a deployment sequence, corresponding to the sensor firings that would be detected as an object moved between corresponding locations on the number line.*

As an object moves along the number line, sensors in the deployment sequence fire and segments of the deployment sequence are recorded by the sensor system. If the system must determine the location of the object based on a single firing, each sensor point must be assigned to an independent sensor. As we consider longer sequences of sensor firings, however, it becomes possible to localize the object even when single sensors correspond to multiple edges between object locations. The localization capacity of the 1D sensor system (i.e. the maximum number of unique object locations that can be achieve along the number line) is characterized by

the maximum length of a deployment sequence under the constraint that every subsequence of length n is unique.

Definition 4.2.3. *A* sequence graph *for a system with m sensors based on sensor sequences of length n is a directed graph consisting of*

- *A vertex corresponding to every possible sensor sequence of length $n-1$.*
- *Directed edges between all vertices such last $n-2$ numbers of the sensor sequence of the originating vertex are the same as the first $n-2$ numbers of the sensor sequence of the terminating vertex.*

Under these conditions each edge corresponds to a sensor sequence with length n and a trail in the sequence graph corresponds to a deployment sequence.

Lemma 4.2.4. *A sequence graph has $m(m-1)^{n-2}$ vertices. The in-degree and out-degree for each vertex are both $m-1$. The total number of edges is $m(m-1)^{n-1}$.*

Proof. According to assumption (d), no consecutive numbers within a sensor sequence should be the same; thus, the number of vertices equals the number of sensor sequences with length $n-1$. After the first sensor in a sensor sequence is determined from the m candidats, there are $m-1$ candidates for each successive sensor in the sequence. The total number of allowable sequences of length n is calculated as $m \cdot \underbrace{(m-1)(m-1)\cdots(m-1)}_{n-2} = m(m-1)^{n-2}$.

Since outgoing and incoming edges match subsequences of length $n-2$, the number of such matches is determined by the number of different first or last number in the length $n-1$ vertex sequences. Since we have assumption (d), only $(m-1)$ choices are available. The total number of edges is calculated as $\frac{2m(m-1)^{n-2}\cdot(m-1)}{2} = m(m-1)^{n-1}$ □

Lemma 4.2.5. *A sequence graph is strongly connected.*

Proof. We need to prove there is a directed path from any vertex to another vertex.

Suppose the last sensor of the starting vertex is A and the first sensor of the ending vertex is B. We will find a directed path from the starting vertex to the ending vertex based on the rules mentioned before for two different situations. Note that once we reach a vertex whose sensor sequence ends at B, it is trivial to find the rest of the path to reach the ending vertex whose sensor sequence begins with B.

1. when $A = B$, the corresponding path is $(\ldots A) \Rightarrow (\ldots B) \Rightarrow \cdots (B \ldots)$
2. when $A \neq B$, the corresponding path is $(\ldots A) \Rightarrow (\ldots AB) \Rightarrow (\ldots B) \cdots (B \ldots)$

□

Lemma 4.2.6. *A directed Euler circuit exists for a directed graph iff (if and only if) the graph is strongly connected and each vertex has an equal number of in-degree and out-degree [9]*

Theorem 4.2.7. *The maximum achievable number of locations identified by a boundary sensor system with m sensors using sensor sequences of length n such that each sensor sequence is unique in the deployment sequence along either positive or negative direction is*

$$P(m,n) = \begin{cases} m, n = 1 \\ m(m-1)^{n-1}, n \geq 2 \end{cases} \tag{4.1}$$

Proof. When $n = 1$, we will construct a graph with the vertices corresponding to the m sensors and edges connecting the sensors. All the vertices are connected by undirected edges since any two different sensors can be adjacent to each other. A deployment sequence that maximizes sensing efficiency corresponds to a path that

transverses all the vertices. Such path is a Hamiltonian path [9]. It is trivial to find a Hamiltonian path in a complete graph. For example, the darkened edges in Fig.4.4 are a Hamiltonian path corresponding to the deployment sequence (135462).

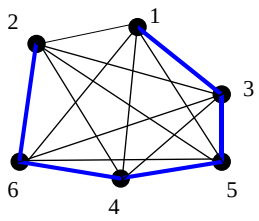

Figure 4.4: A graph with vertices for 6 sensors and edges for potential sensor boundaries for $n = 1$. The darkened path is one of the Hamiltonian paths of this graph.

When $n > 1$, from Lemma 4.2.4, 4.2.5, and 4.2.6, it is found that a directed Euler circuit exists for the sequence graph. The Euler circuit is a special kind of trail with the same starting and ending vertex. Thus we can generate a optimal deployment sequence utilizing all the $m(m-1)^{n-1}$ sensor sequences based on the Euler circuit. □

Fig.4.5 shows an example when $n = 4, m = 3$. Fleury's algorithm is employed to find an Euler circuit in the graph. [45] Note that the Euler circuit derived is not unique. One can start from another vertex or take another edge during the trail to form another deployment sequence.

Theorem 4.2.8. *The number of distinct deployment sequences derived from the sequence graph for m sensors and sensor sequence of length n is*

$$D = (m-1)!^{m(m-1)^{n-2}} det(K) \tag{4.2}$$

where K is the principal submatrix obtained from the Laplacian [16] of the sequence graph by deleting any row and the corresponding column.

Proof. From the BEST formula [20, 60], the number of Euler trails in a digraph is given by $\underbrace{(d_1-1)(d_2-1)\dots(d_i-1)}_{number-of-vertices} det(K)$, where d_i is the number of outdegree for ith vertex. Inserting the values presented in lemma 4.2.4 to the BEST formula , we obtain Eqn(4.2). □

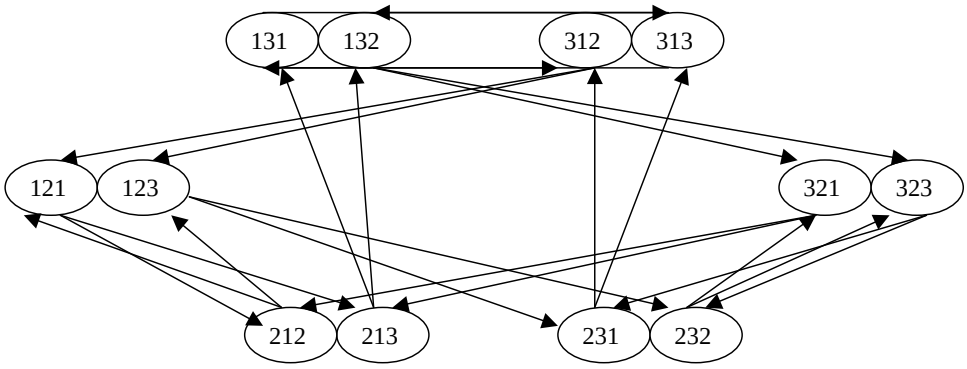

Figure 4.5: The sequence graph for $m = 3, n = 4$. The total number of edges is $3 * (3-1)^{4-1} = 24$. An Euler trail can be found within this graph that corresponds to a deployment sequence to track an object in 24 positions with 3 sensors and at least 4 consecutive measurements. One of the deployment sequence we get from this Euler circuit is (121232312313121321313232121).

In theorems 4.2.7 and 4.2.8 we have derived the number of locations and deployment sequences we can achieve in 1D with m sensors and n detection events. It is useful to extend these results to more constrained deployment sequences that are robust against *false crossings*. A false crossing occurs when an object triggers a boundary sensor but does not finish the crossing. The object could touch the boundary and trigger the sensor but remain in the same region as it was before triggering the sensor or it could cross in a complicated multiple trigger manner as induced by the walking pattern of legged objects. In a robust system we want to detect false crossings by sacrificing little sensing efficiency; therefore, we will need to apply another constraint on the sensor sequence arrangement, which is stated as assumption (e)

(e) No alternating sensor in a sensor sequence can be the same.

We incorporate ssumption (e) in a new definition for an *enhanced sequence graph.*

Definition 4.2.9. *An enhanced sequence graph is a sequence graph excluding deployment sequences in which sensors repeat in less than two steps.*

Lemma 4.2.10. *A enhanced sequence graph has* $m(m-1)(m-2)^{n-3}$ *vertices. The in-degree and out-degree for each vertex is both* $m-2$*. The total number of edges is* $m(m-1)(m-2)^{n-2}$

Proof. For the first sensor in the sensor sequence, we still have m candidates. After the first sensor is chosen, the second should not be the same as the first one according to assumption (f), so we have $(m-1)$ choices. For the *ith* sensors in the sensor sequence, it can neither be the same as the $(i-1)$th sensor nor the $(i-2)$th sensor according to assumption (g), so it only has $m-2$ choices. That is the same as the in-degree or out-degree for any vertex in the enhanced sequence graph. Since each vertex corresponds to a sensor sequence of length $n-1$, the number of such sensor sequences (same as the number of vertices) now is $m \cdot (m-1) \cdot \underbrace{(m-2)(m-2)\cdots(m-2)}_{n-3} = m(m-1)(m-2)^{n-3}$. □

Lemma 4.2.11. *An enhanced sequence graph is strongly connected when* $n = 2$ *or* $n \geq 3$ & $m \geq 4$

Proof. The proof is trivial when $n = 2$.

When $n \geq 3$ & $m \geq 4$, suppose the last two sensors in the starting vertex are A_1A_2 and the first two sensors of the ending vertex are B_1B_2. We will find a directed path from the starting vertex to the ending vertex based on the rules mentioned before for six different situations. Note that once we reach a vertex whose sensor sequence ends at B_1B_2, it is trivial to find the rest of the path to reach the ending vertex whose sensor sequence begins with B_1B_2.

1. When $A_1 \neq B_1, B_2, A_2 \neq B_1, B_2$, the corresponding path is $(\ldots A_1 A_2) \Rightarrow (\ldots A_1 A_2 B_1) \Rightarrow (\ldots A_2 B_1 B_2) \Rightarrow \cdots \Rightarrow (B_1 B_2 \ldots)$

2. When $A_1 = B_1, A_2 \neq B_1, B_2$, suppose there is A_3 where $A_3 \neq B_1, B_2, A_2$Then the corresponding path is $(\ldots B_1 A_2) \Rightarrow (\ldots B_1 A_2 A_3) \Rightarrow (\ldots A_2 A_3 B_1) \Rightarrow (\ldots A_3 B_1 B_2) \cdots \Rightarrow (B_1 B_2 \ldots)$

3. When $A_1 = B_2, A_2 \neq B_1, B_2$, suppose there is A_3 where $A_3 \neq B_1, B_2, A_2$, Then the corresponding path is

4. $(\ldots B_2 A_2) \Rightarrow (\ldots B_2 A_2 A_3) \Rightarrow (\ldots A_2 A_3 B_1) \Rightarrow (\ldots A_3 B_1 B_2) \cdots \Rightarrow (B_1 B_2 \ldots)$

5. When$A_2 = B_1, A_1 \neq B_1, B_2$, then the corresponding path is $(\ldots A_1 B_1) \Rightarrow (\ldots A_1 B_1 B_2) \Rightarrow \cdots \Rightarrow (B_1 B_2 \ldots)$

6. When $A_2 = B_2, A_1 \neq B_1, B_2$. suppose there is A_3 where $A_3 \neq B_1, B_2, A_1$

then the corresponding path is $(\ldots A_1 B_2) \Rightarrow (\ldots A_1 B_2 A_3) \Rightarrow (\ldots B_2 A_3 B_1) \Rightarrow (\ldots A_3 B_1 B_2) \Rightarrow \cdots \Rightarrow (B_1 B_2 \ldots)$

Based on the discussion above, we find if $m \geq 4$, there is always a directed path from any vertex to another vertex. □

Theorem 4.2.12. *The maximum achievable number of locations identified by a boundary sensor system with m sensors using sensor sequences of length n such that no sensor appears with less than two intervening distinct sensors is*

$$P(m,n) = \begin{cases} m, n = 1 \\ m(m-1)(m-2)^{n-2}, n \geq 2, m \geq 4 \end{cases} \tag{4.3}$$

Proof. When $n = 1$ the proof is the same as in theorem 4.2.7. When $n \geq 2, m \geq 4$, we will construct an enhanced sequence graph. Then from lemma 4.2.10 and 4.2.11, an Euler circuit exists for an enhanced sequence graph when $m \geq 4$, which corresponds to a deployment sequence monitoring $m(m-1)(m-2)^{n-2}$ regions. □

Theorem 4.2.13. *The number of deployment sequences derived from the enhanced sequence graph for m sensors and sensor sequence of length n is*

$$D = (m-2)!^{m(m-1)(m-2)^{n-3}} det(K) \tag{4.4}$$

We note in theorem 4.2.7 and 4.2.12 that the sensor sequence is unique in the deployment sequence along either positive or negative direction, but it is not globally unique. We find that the algorithm to find the Euler circuit in a graph can be modified to find a deployment sequence with globally unique sensor sequence in an enhanced sequence graph. We will travel in the enhanced sequence graph in the way that once a directed edge is traversed, the edge with mirror sensor sequence of the traversed one will be removed to prevent it from being included in the trail in the future. In this way, we can guarantee that each sensor sequence is globally unique.

Lemma 4.2.14. *By developing a trail with the traveling and discarding method, exactly half of the edges in an enhanced sequence graph will be traversed when $n \geq 3$ & $m \geq 4$.*

Proof. Here we present a constructive proof showing an algorithm of how to find a trail by *traveling and discarding* method.

Select a vertex v and begin to build a directed trail starting at v. For every edge added to this trail, remove the edge whose corresponding sensor sequence is the reversal of the one added to the trail to form a "mirror" trail. Proceed until you can go no further. Now one of the following must hold

- We returned to v, forming a circuit C^*.
- We ended at a vertex w other than v.

If the latter condition holds, then w has different in- and out- degree, and one of the following conditions must hold:

1. Either none or an equal number of inward and outward edges of w have been removed, and the algorithm halts with 'proof' that G has no directed Euler circuit, which contradicts our statement before.

2. One more out edge of w was removed than the in edges. We assume the vertex right before w is u and that before u is t. In this case the sensor sequence of either u or t must be the reversal of the sensor sequence of w. Supposed the sensor sequence of u is $(A_1A_2 \dots A_{n-2})$ and that of t and w are $(B_1B_2 \dots B_{n-2})$ and $(C_1C_2 \dots C_{n-2})$. For the first case we have $(A_2 \dots A_{n-2}) = (C_1 \dots C_{n-3})$ and $(A_1A_2 \dots A_{n-2}) = (C_{n-2} \dots C_2C_1)$,which will lead to $A_i = A_{i+2}$ or $A_i = A_{i+1}$,$(3 \leq i \leq n-2)$, contradicting the rule that no consecutive or alternating sensor should be the same. For the latter case, we have $(B_1B_2 \dots B_{n-2}) = (C_{n-2} \dots C_2C_1)$, $(A_1 \dots A_{n-3}) = (B_2 \dots B_{n-2})$ and $(A_2 \dots A_{n-2}) = (C_1 \dots C_{n-3})$, which again leads to $A_i = A_{i+2}$ or $A_i = A_{i+1}$,$(3 \leq i \leq n-2)$, contradicting to the rule that no consecutive or alternating sensor should be the same.

Thus, the latter condition does not hold in any case, so we will "grow" C by adding C^* to it.

Now, starting at v, proceed along C. One of the following must hold:

- We find a vertex u on C which still has edges leaving it.
- We find no such vertex, and C contains half of the edges of G. The other half was removed when we grow C.
- We find no such vertex, and C does not contain half of the edges of G.

In the third case, we have proven that G is disconnected; there is no directed path from any vertex of C to any vertex not on C, so we can halt with a proof that G has no Euler circuit, contradicting with lemma 4.2.11 before. In the second case,

C is the trail we want to find, so the algorithm halts in the happy state. In the first case, we let $v = u$, and we return to Step 1, to continue to grow our circuit.

The final trail C will traverse exact half of the edges of the graph, corresponding to a deployment sequence containing $\frac{m(m-1)(m-2)^{n-2}}{2}$ sensor sequences. This is the maximum since adding any sensor sequence will make it no longer globally unique.

□

Theorem 4.2.15. *The maximum achievable number of locations identified by a boundary sensor system with m sensors within which every consecutive sensor sequence of length n is globally unique in the deployment sequence is given by*

$$P_u(m,n) = \begin{cases} m, n = 1 \\ \frac{m^2-m}{2}, m odd, n = 2 \\ \frac{m^2-2m+2}{2}, m even,, n = 2 \\ \frac{m(m-1)(m-2)^{n-2}}{2}, n \geq 3, m \geq 4 \end{cases} \tag{4.5}$$

Proof. When $n = 1$, the proof is trivial.

When $n = 2$, we can generate a sequence graph, it contains undirected edges. Now an edge corresponds to two sensor sequences that are mirrors of each other. In this graph an Euler trail corresponds to a deployment sequence among which every sensor sequence is globally unique because each undirected edge can only be traversed once.

Lemma 4.2.16. *An Euler trail exists for a undirected graph iff there are exactly 0 or 2 vertices of odd degree. [9]*

When m is odd, a complete graph with m vertices satisfying lemma 4.2.16 has the maximum number of edges $\frac{m(m-1)}{2}$. When m is even, we need to remove $\frac{m-2}{2}$ edges from a complete graph to satisfy lemma 4.2.16. Thus, the total number of edges is $\frac{m^2-2m+2}{2}$. An example with $m = 5$ is shown in Fig.4.6.

When $n \geq 3$ & $m \geq 4$, we will travel in an enhanced sequence graph by

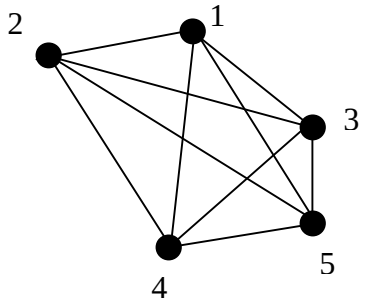

Figure 4.6: The complete graph when $m = 5, n = 2$. An Euler trail can be found in such a graph to construct a deployment sequence in which the sensor sequence is globally unique along both directions. One example is (1234524135)

the traveling and discarding method to get the deployment sequence with length $\frac{m(m-1)(m-2)^{n-2}}{2}$.

□

Proof. The proof is similar to that in theorem 4.2.8 by substituting the values in lemma 4.2.10 □

In Fig.4.7 We use the algorithm discussed before to travel in an enhanced sequence graph and find the deployment sequence containing both globally unique sensor sequences and one whose sensor sequences are unique only along positive or negative directions.

As discussed above, 1D boundary sensor networks arranged orthogonally in the plane may be used to localize sources in 2D. This Cartesian arrangement will fail to localize along one axis, however, if the the object moves along a line perpendicular to that axis. To solve this problem, a third parallel set of straight boundary sensors can be included so that they will coincide with the diagonals, as shown in Fig.4.8.

Theorem 4.2.17. *In a deployment graph formed by a triangle grid so that each straight line corresponds to a distinct sensor, object paths that share common sensor sequences of length* $n(n \geq 2)$ *must circle the same vertex.*

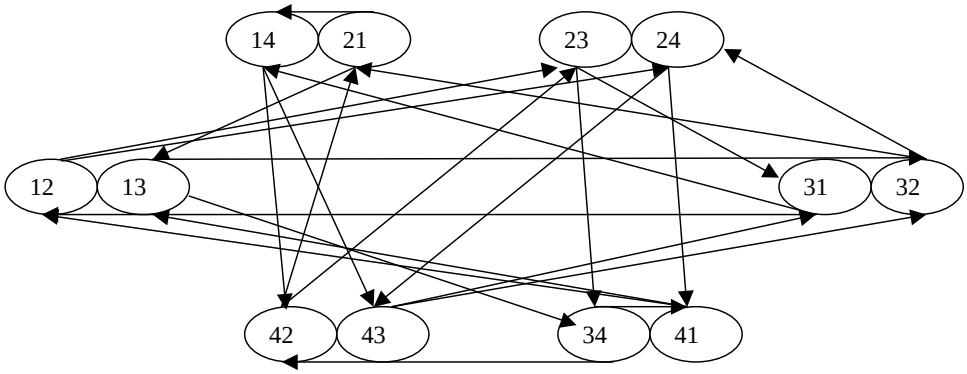

Figure 4.7: The enhanced sequence graph for $m = 4, n = 3$. The total number of edge is $4 * (4 - 1) * (4 - 2)^{4-1} = 24$. Again, an Euler trail can be found within this graph that corresponds to a deployment sequence to track an object in 24 positions with 4 sensors and at least 3 consecutive measurements. One deployment sequence is (142314324124312341342132 14), whose length (number of sequences of length 3) is 24. The deployment sequence whose sensor sequences are globally unique based on the traveling and discarding algorithm is (14312342314214), whose length is $24/2 = 12$.

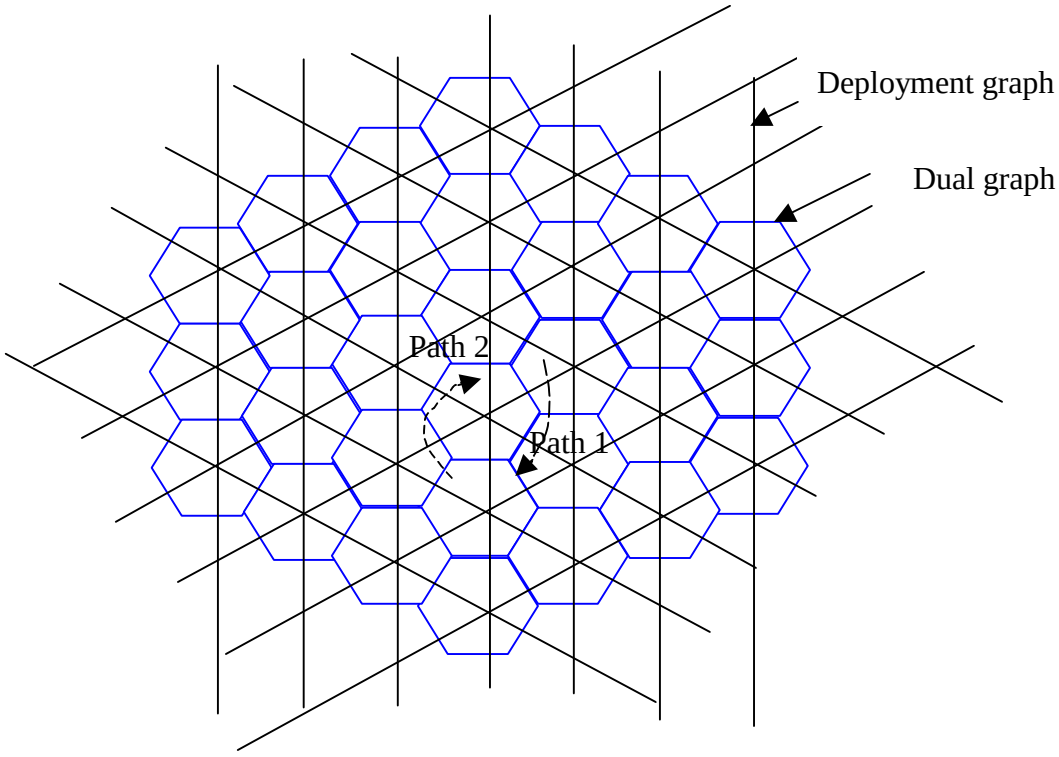

Figure 4.8: The deployment graph (boundary not drawn) and dual graph for the triangle grid.

Proof. We will first map the object path in the deployment graph into the path along edges in the dual graph. We number each of the three parallel sets of boundary sensors with independent of number set a, b and c. ($a = 1, 2, \ldots, b = 1, 2, \ldots, c = 1, 2, \ldots$). Next in the dual graph we assign a 3-element coordinate (a, b, c) to each vertex, where the value of each element is determined by the number of the corresponding edge. (The number of the edge is determined by the number of boundary sensors to which it corresponds). Fig.4.9 shows the coordinate of the vertices of one of the hexagon in the dual graph. We notice that the vertices which have the same value for 2 out of 3 elements in their coordinate can only be on the two ends on a diagonal.

If two paths are different in the dual graph, at least at one particular step their vertices should be different. Assume they are v_1 at path 1 and v_2 at path 2. Since in the next step they will trigger the same sensor, at least one of their coordinate elements should be the same, and this element will not change after this step. Without losing generality, let's assume after this step, $v_1 = (a, b_1, c_1)$ and $v_2 = (a, b_2, c_2)$. Now starting from v_1,v_2, since they are going to trigger another sensor, another element in the coordinate should be the same. (Keep in mind that in the path of the object is not allowed to directly turn back to trigger the same sensor again; otherwise, a new tracking sequence will begin). Without losing generality, we assume it is the second element, thus we can rewrite the coordinate of v_1,v_2 as $v_1 = (a, b, c_1)$and $v_2 = (a, b, c_2)$. Now since v_1,v_2 share two common elements, they must be on the two ends of the diagonal of a hexagon. The same analysis can be applied to the following vertices in the paths. We will find that if two paths share the common sensor sequence, they have to be the same, or part of the two paths have to be symmetric on a particular hexagon (like path 1 and path 2 in Fig.4.8). □

Thus theorem 4.2.17 guarantees that the tracking error for a triangle grid is localized within a small neighborhood.

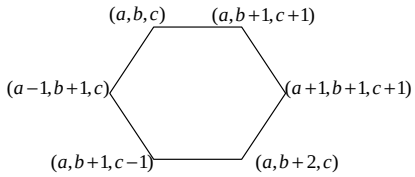

Figure 4.9: The coordinate system for the dual graph of a triangle grid. If two vertices have 2 coordinate elements in common, they have to be on the two ends of the diagonal of a hexagon.

4.3 Conclusion

We propose a new paradigm for a sensor network which tracks moving objects based on detected sensor signal sequences in the time domain. A complete analysis is given for the 1D case and based on graph theory, we find the most efficient boundary sensor design and give the construction method for several cases. This 1D boundary sensor has potential application areas such as tracking automobiles along a street, machine position along an axis or human position in a plane.

Chapter 5

2D Boundary Sensor Array

5.1 Introduction

In two dimensions, a boundary sensor is associated with a curve in the plane. The sensor fires when an object crosses over the curve. Such sensors may be implemented using optical fibers, piezo-electric or pneumatic pressure sensors, light beams or motion sensors. As in the 1D case, an object moving continuously through the object space to form a path passing $n + 1$ regions, triggers n boundary sensors to form a sensor sequence of length n. If there is one to one mapping between the object paths and the sensor sequences, we are able to localize the final location of an object without ambiguity.

5.2 2D Random Deployment Scheme

Design of an efficient 2D deployment strategy for a boundary sensor array is a daunting problem. We begin discussion here by considering the performance of quasi-random 2D deployments. The unique mapping between an object path and a sensor sequence in a random deployment is not guaranteed. However, if the ratio of the total number of paths to the total number of available sensor sequences is much smaller than 1, we expect with high probability there is such unique mapping.

Moreover, we also hope this ratio will further decrease when we increase m and n so that less ambiguity can be achieved by a little more cost.

Definition 5.2.1. *A deployment graph is a graph that describes the original layout of a boundary sensor network in 2D space. In the deployment graph, the object space boundary is considered as one vertex. The intersections between boundary sensors are also vertices. Two vertices are connected with an edge if there is a common sensor segment between them.*

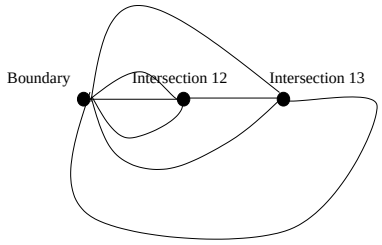

Figure 5.1: The deployment graph of Fig.4.1.

Lemma 5.2.2. *A deployment graph is a finite, connected, planar graph.*

Proof. From the construction of deployment graph, this is obvious. □

Lemma 5.2.3. *The dual graph [29] of a deployment graph is a simple, finite,connected, planar graph.*

Proof. It is obvious that the dual graph is a finite, connected, planar graph. We assume that the boundaries associated with 2D sensors do not end within the object space. By this assumption, two adjacent regions can only share one common boundary. The regions in the deployment graph correspond to vertices in the dual graph and the boundaries correspond to edges in the dual graph. Thus at most one edge connects any two vertices in the dual graph, which means the dual graph is simple. □

For example, the dual graph of Fig.5.1 is shown in Fig.5.2

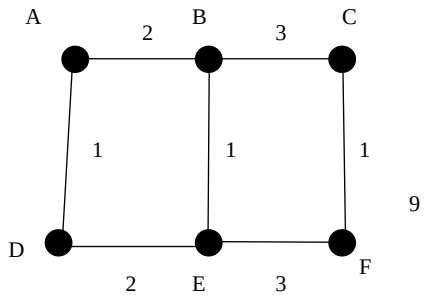

Figure 5.2: The dual graph for the deployment graph shown in Fig.5.1. The numbers here stand for the sensors, and the characters stand for the regions in Fig.4.1

Lemma 5.2.4. *If G is a simple, finite planar graph with v vertices ($v \geq 3$), then G has at most $3v - 6$ edges [29].*

Lemma 5.2.5. *The upper bound of the average degree of a simple, finite, planar graph is 6.*

Proof. Average degree $\leq \frac{2(3v-6)}{v} = 6 - \frac{12}{v} \leq 6$ □

Lemma 5.2.6. *The upper bound of the total number of object paths passing $n+1$ regions is $O\left(V(m)\left(\frac{5}{2}\right)^n\right)$, where $V(m)$ is the number of vertices in the dual graph.*

Proof. An object path from one region to another in the deployment graph corresponds to a path from and to the corresponding vertices in the dual graph. We have $V(m)$ choices to start. At each step in average we have at most $6 - 1 = 5$ choices to extend the path. Since we can extend the path at both ends, the new longer path will be counted twice. Thus we have at most $O\left(V(m)\underbrace{\left(\frac{5}{2}\right)\cdots\left(\frac{5}{2}\right)}_{n}\right) = O\left(V(m)\left(\frac{5}{2}\right)^n\right)$ paths. □

Lemma 5.2.7. *The number of sensor sequences of length n is $O(m^n)$*

Proof. The number of sensor sequences is given by $S_{curve}(m,n) = m(m-1)^{n-1} = O(m^n)$ □

Two assumptions are made to further estimate the order of $V(m)$:

(a) The deployment graph is a simple graph (There could be multiple edges between the vertex for object space boundary and other vertices, but the number of such edges is small compared with the total number of edges.)

(b) The lay out of each boundary sensor is assumed to be approximated by polynomial curve with order up to K. (This assumption is based on fiber sensors, there could be other assumptions for other sensor forms.)

Lemma 5.2.8. *If G is a connected planar graph with v vertices, r faces and e edges, then its dual graph G^* has r vertices, v faces and e edges. [2]*

Lemma 5.2.9. *In the deployment graph, the number of vertices is in the same order as the number of regions.*

Proof. Assume the number of vertices, edges and regions in the deployment graph are v, e and r. According to Euler's formula [2] , we have $v - e + r = 2$. From lemma 5.2.4 and lemma 5.2.8 we have $e \leq 3v - 6$ (derived from the deployment graph) and $e \leq 3r - 6$ (derived from the dual graph) Combine these three, and we get $\frac{v}{2} + 2 \leq r \leq 2v - 4$. Thus we have $O(v) = O(r)$. □

Lemma 5.2.10. *Two polynomial curves with order up to K have at most K intersections.*

Theorem 5.2.11. *The upper bound of the ratio of the total number of paths $P_{curve}(m,n)$ passing $n+1$ regions to the total number of sensor sequence $S_{curve}(m,n)$ of length n is*

$$\frac{P_{curve}(m,n)}{S_{curve}(m,n)} = O\left(\left(\frac{5}{2m}\right)^n V(m)\right) \tag{5.1}$$

where $V(m)$ is the number of vertices in the dual graph or number of regions in the deployment graph. If we further assume the deployment graph is a simple graph and the deployment of each boundary sensor can be described by a polynomial curve with order up to K, Eqn(5.1) becomes

$$\frac{P_{curve}(m,n)}{S_{curve}(m,n)} = O\left(\left(\frac{5}{2m}\right)^{n-2}\right) \tag{5.2}$$

Proof. Eqn(5.1) can be derived from lemma 5.2.6 and 5.2.7.

From lemma 5.2.10, we find $V(m) = r = O(v) = O\left(K\begin{pmatrix} m \\ 2 \end{pmatrix}\right) = O(m^2)$. Substituting $V(m)$ into Eqn(5.1) yields Eqn(5.2) □

Theorem 5.2.11 confirms our our speculation stated earlier this section. When $m \geq 3$ and $n \geq 3$ (which is easy to be met in real world applications), we find that $P_{curve}(m,n)$ will be far smaller than $S_{curve}(m,n)$ when m or n increase. Since each path is always associated with a sensor sequence (but not vice versa), a pool of sensor sequences with far larger quantity than that of the paths means we have great possibility to make each path associate with a distinct sensor sequence.

When both the boundary sensor and the object path are straight lines, theorem 5.2.11 can be further simplified.

Theorem 5.2.12. *When both the boundary sensor and object path are limited to straight lines that extend to infinity*

$$\frac{P_{curve}(m,n)}{S_{curve}(m,n)} = O\left(m^{4-n}\right) \tag{5.3}$$

Proof. Let L be a set of m line (boundary sensor). We dualize L to a set $L^*$of points. The duality transform preserves the above-below relationship, i.e., if a point p lies above (resp.\ below, on) a line l, then the line p^*dual to p lies above (resp.\ below, on) the point l^* dual to l. Let H be the set of $O(m^2)$ lines connecting every pair of

points in L^*; each line in H is dual to an intersection point of lines in L. The lines in H induce a planar subdivision, denoted by $A(H)$, whose vertices are intersection points of H. The edges are the maximal portions of lines in H that do not contain any intersection points, and the faces are the maximal connect regions that do not intersect any line of H. Let p and q be two points that lie on the same edge or face of $A(H)$ Then the lines p^* and q^* intersect the lines of L in the same order because all intersection points of L lie on the same side of p^* and q^*. Note p^* and q^* are straight object paths and they should be regarded as the same path if they satisfy the above conditions. Hence, there are $O(m^4)$ different object path. Combine this result with lemma 5.2.7 then we get Eqn (5.3) □

Corollary 5.2.13. *When the boundary sensors are straight lines and object paths are straight lines passing $n+1 (n \leq m)$ regions. We have*

$$\frac{P_{curve}(m,n)}{S_{curve}(m,n)} = O\left((m-n+1)m^{4-n}\right) \tag{5.4}$$

Proof. The number of straight-line paths passing $n+1$ regions along a particular line is $m-n+1$, from which theorem 5.2.12 yields Eqn(5.4) □

5.3 System Data Structure and Simulation

In a boundary sensor array, we store the regions associated with each boundary sensors in the structure of a series of region pairs. Two regions in a region pair face each other across the corresponding sensor. Fig.5.3 shows one example of random deployment with 6-laser diode/photodiode pair as boundary sensors. The regions associated with sensor (a) are stored as $Sensor(a) = [(1,2),(6,7),(9,10),(11,12),(14,15),(19,20)]$. When an object is estimated to be at a particular region due to previous measurement and a new sensor is triggered, the system will check all the associated regions with that sensor to find a region pair containing the current region as well as the

next possible region for the object. In most cases the object will keep moving forward thus the next possible region should be the other region in the region pair. In rare cases when there is a false crossing, the object stays in the same region. When there is an unreliable measurement so that the sensor misses the object crossing event (due to the fact that sometimes the object tries to avoid the sensor), which could lead to the result that we cannot find the current region in the associated region pairs of the triggered sensor (refer to the data structure of the boundary sensor). We need to start over the tracking procedure to clear unreliable measurements. In the real-time tracking scheme, we will keep a queue of possible current regions in descending order indicating descending possibility. This queue is initiated with all the associated regions of the first triggered sensor. Every time a new sensor is triggered, depending on the situation either the object will cross that sensor, stay in the same region, or there will be no valid next region because of unreliable measurement, the queue will be updated with new regions added and some old ones replaced or removed. If the queue gets empty, the tracking will start over with the queue refilled by all the regions associated with the current sensor.

In Table 5.1 we show a simulated tracking result when an object follows the path in Fig.5.3. In the table, each row corresponds to the possible regions after the *nth* sensor in the sensor sequence is triggered. The first region in each row is the most possible tracking result while the one at the end is the least possible. We assume the more false crossing an object makes along a path, the less possible the final region will be as the tracking result. The object makes 2 false crossing at region 6. In general, the final regions will always be two adjacent regions since we don't know if there is a false crossing during the last step.

The data structure used in boundary sensor tracking scheme use less storage space and processing time as compared to that of static sensor. It is easy to find

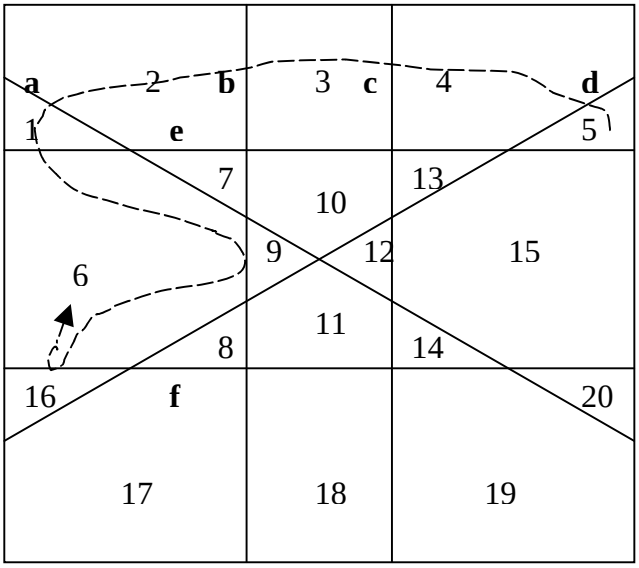

Figure 5.3: A sensor array deployment scheme with six laser boundary sensors.

Table 5.1: A simulated tracking sequence. The first column of each row shows at that particular time step which sensor is triggered. From the second column the object's possible positions are listed in descending possibility.

Step 1,sensor d	4	5	6	8	9	10	11	12	13	15	16	17
Step 2,sensor c	3	13	14	15	10	12	4	11				
Step 3,sensor b	2	7	8	3	10	11						
Step 4,sensor a	1	6	9	12	2	7	10	11				
Step 5,sensor e	6	1	7	2	3	10						
Step 6,sensor b	9	10	3	2	7	6						
Step 7,sensor f	16	6										

that the storage space is determined only by the number of regions. The searching time is determined by the number of regions associated with each sensor , which is $O(m)$. Besides, for the static sensor array demonstrated in the previous chapter, the multiplex sensing is carried out in the spatial domain (which means a sensor can sense multiple regions at the same time). The relationship between the object location and the signature code needs to be recorded in a lookup table. The required storage space and decoding time is determined by the size of the lookup table. Sometimes these requirements can be reduced by a delicate coding scheme. For a boundary sensor network, however, the multiplex sensing is carried out in the temporal domain (which means a sensor can sense multiple steps in a short path). This characteristic enables us to decode a sensor sequence bit by bit at each time step. We don't need to store the relationship between the object paths and the sensor sequences, but only the associated regions with each boundary sensor. Furthermore, this temporal domain multiplexing makes the system's detecting power (maximum number of regions detectable) grow with the length of the sensor sequences. On the other hand, when errors occur, the boundary sensor array needs n time step to completely converge to the correct tracking result while the static sensor array can correct it immediately in the next time step. In Table 5.2 we show some comparison between these two types of sensor arrays.

We did some calculations to study the deployment scheme Fig.5.3 to verify the theoretical result. In Fig.5.4 we plot the relationship between length of sensor sequence n and $\log\left(\frac{P_{curve}(m,n)}{S_{curve}(m,n)}\right)$. According to Eqn (4.5), this should be a straight line, which is confirmed by Fig.5.4. The larger the slope of this line, the shorter the length of sensor sequence required to map to a unique sequence.

We are interested in how long the sensor sequence should be so that the tracking result can converge to one distinct final region. Note that sometimes even if a

Table 5.2: A comparison between the boundary sensor and the static sensor on system requirements and sensing capability.

Sensor type (Binary sensor)	**Boundary sensor**	**Static sensor**
Sensing type	Region boundary	Region interior
Multiplexing type	Temporal domain	Spatial domain
Maximum number of positions monitored with m binary sensors.	$O(m^n)$ with sensor sequence of length n	$O(2^m)$
Storage space required for m sensors, P positions	$O(P)$	$O(P)$ or $O(2^m)$, depending on algorithm used.
Decoding time required for m sensors.	$O(m)$	$O(m)$ to $O(P)$, depending on code used.
Time steps required for error correction	n	1

sensor sequence corresponds to multiple paths, these paths may still lead to the same final region, which further reduces the ambiguity for object tracking. The simplest example is by comparing two paths: (1) object A triggers a sensor and come back to the original region. (2) object B comes from another region and triggers the same sensor A that triggered; then object B enters object A's region. These two paths are different because they start from different regions, however, they share the same sensor sequence and end up with the same final region. The calculation result is shown in Fig.5.5. It is found that the number of paths per sensor sequence indeed drops dramatically when the length of the sensor sequence grows. However, this number does not go to 1. The reason is that in the simulation we allow arbitrary paths such as circles. For example, in Fig.5.3 if an object starts from region 16 and circles clockwise to region 6,8... it will always trigger the same sensor sequence as the path that starts from region 17 and circles counterclockwise to region 8,6... as long as their lengths are the same. There are always some sensor

sequences that correspond to more than one final regions by this way no matter how long the sensor sequence is. But in general, an object, especially a walking human, will seldom follow this kind of path. We observe numerically in general that once the length of the sensor sequence reaches 6 or 7, the number of the final regions will converge to 1. (When false crossing is allowed, the final regions are two adjacent regions).

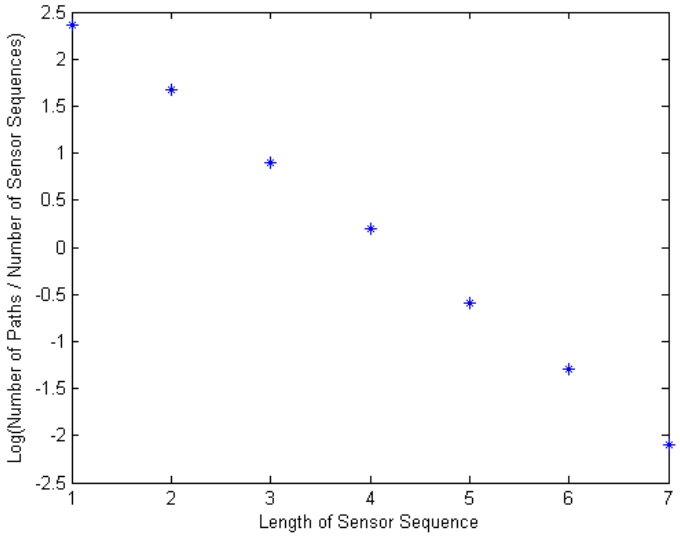

Figure 5.4: The log (number of paths / number of sensor sequences) versus the length of sensor sequence.

5.4 Conclusion

For random deployments of a boundary sensor array in a 2D domain, we give the upper bound on the ratio of the number of paths to the number of sensor sequences, where the path and sensor can be either curves or straight lines. We find in both cases, this ratio decreases rapidly when the number of sensors or the length of the object path increases, indicating great possibility to have each path associated with

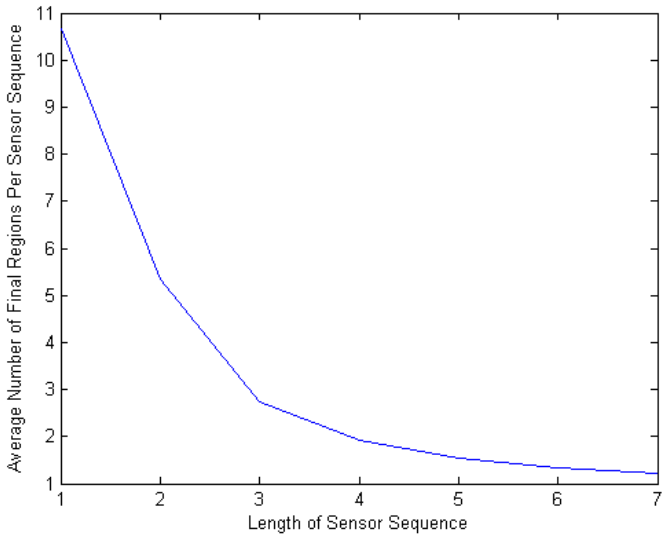

Figure 5.5: The average number of final regions per sensor sequence versus length of sensor sequence.

a unique sensor sequence. A simulation was run to confirm the theoretical analysis. We propose the data structure of the boundary sensor and compare it with the static sensor discussed in the previous chapters. It is found that the boundary sensor requires less decoding time and storage space by putting less demand on the error correction capability.

Chapter 6

Probabilistic Partial Boundary Sampling in 3D Space

6.1 Introduction

In the previous chapters we discussed the design of boundary sensor arrays in 1D and 2D space. When it is in the 3D space, however, the situation can be quite different. The reason is that we don't have a good sensor candidates to create a continuous surface so that they can serve as boundaries in 3D space. Even if we have certain kind of surface sensor, such as spread fan-shape laser beam, since it is not guaranteed to completely cut a 3D object space, as shown in Fig. 6.1, it is not an ideal boundary. The same situation happens with line sensors as discussed before. A vivid example of line sensors can be found in the movie Entrapment. In these cases, these sensors actually serve as partial boundary sensors. When an object starts from a region to the adjacent region separated by a partial boundary, it has a certain probability to trigger that partial boundary sensor, depending on the actual path it takes. If we deploy such a boundary sensor array in the space and an object travels in the space, we will also detect a sensor sequence in the temporal domain. In order to reconstruct the original path, a statistical model must be employed.

The partial boundary sensor model also applies to other scenarios. For the

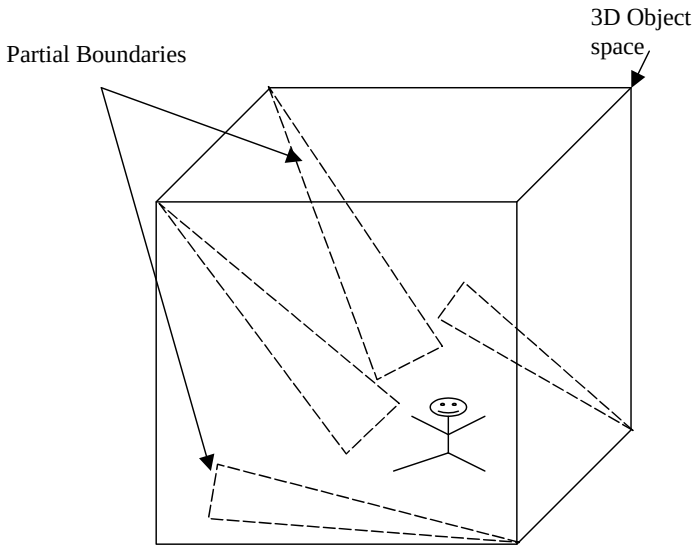

Figure 6.1: A number of triangle shaped surface serve as sensor probe to partially segment the 3D space. The sensor is triggered when the object cut the triangle pattern.

boundary sensor arrays discussed for 1D and 2D cases, when there are errors in the measurement, such as those due to the system noise or outside disturbance, the measurement becomes a erroreous measurement with certain probability P_{cross} so that a statistical model is required as well. Or, if we use fibers deployed on the ground as boundary sensors to track human beings, there are chances the human can miss the fiber boundary when he moves around. This is similar to the 3D partial boundary. In general, the boundary sensors we discussed in chapters 4 and 5 can all be included in the 3D partial boundary framework since they are special cases where $P_{cross} = 1$ and the 3D space collapses to 2D or 1D space. The partial boundaries can also be generalized to small volumes as long as the sum of all these small volume is far less than the volume of the object space. One should notice the difference from volume sampling. In volume sampling, the sum of all the segmented volumes are equal to the volume of the object space.

We will not discuss the design of partial boundary sensor array in this thesis, but we will propose a model and leave the solution as a challenging open problem.

6.2 Model of 3D Partial Boundary Sensor

Assuming a 3D object space $\Re^3$, a number of volumes V_{ij} float in this space. Each volume V_{ij} serves as a jth probe of a particular sensor M_i. The number of sensors is usually far less than the number of volumes. An object with volume $\aleph$ moves within this object space $\Re^3$. A sensor M_i is triggered if the object volume $\aleph$ intersects with a probe volume V_{ij}. When the object moves along a continuous trajectory, a sensor sequence $S_l = [M_{i1}, M_{i2}, M_{i3}...]$ will be recorded, and we are supposed to reconstruct the object trajectory based on the detected sensor sequence.

Since the volumes V_i do not necessarily segment the object space into isolated regions, we still need to find a way to segment the 3D object space into discrete regions. Two regions are considered adjacent to each other if they share a common edge or both are adjacent to a common probe volume. Only when the object travels between two virtual regions which are both adjacent to a probe volume V_{ij} will it be possible for the object to intersect volume V_{ij}. The challenge here is to find a way to define regions and estimate the value of P_{cross}. After that, for a given path ϕ_k and a sensor sequence S_l, one can calculate the conditional probability $P(S_l|\phi_k)$. Then calculate $P(\phi_k|S_l)$ for all paths ϕ_k by Bayesian model: $P(\phi_k|S_l) = \frac{P(S_l|\phi_k)P(\phi_k)}{P(S_l)}$. The path with the largest $P(\phi_k|S_l)$ will be the reconstructed path.

Bibliography

[1] http://www.photonicdetectors.com.

[2] M. Aigner and G. M. Ziegler. *Three Applications of Euler's Formula.* Springer-Verlag, Berlin, 1998.

[3] A. Armitage, Binnie T.D., J. Kerridge, and Lucy Lei. Measuring pedestrian trajectories using a pyroelectric diferential infrared detector. In *Sensors and their Applications XII*, pages 143–149, 2003.

[4] Doran J. Baker and Allan J. Steed. Electronic scanning spectrometer for measurements of rapidly changing spectra. *Applied Optics*, 7(11):2190–2194, 1968.

[5] E. Barillot, B. Lacroix, and D.Cohen. Theoretical analysis of library screening using an n-dimensional pooling strategy. *Nucl. Acids Res*, 19:6241–6247, 1991.

[6] Harrison H. Barrett and Kyle J. Myers. *Foundations of image science.* Wiley-Interscience, Hoboken, NJ, 2004.

[7] P. Bellutta, N. Ancona, and T. Poggio. A floor boundary sensor for autonomous robot navigation. In *Int. Joint Conf. on Artificial Intelligence, 12th IJCAI*, 1991.

[8] J. W. Berthold, W. L. Ghering, and D. Varshneya. Design and characterization of a high temperature fiber optic pressure transducer. *IEEE J. of Lightwave Tech.*, 5:870– 876, 1987.

[9] B. Bollobs. *Graph Theory: An Introductory Course.* Springer-Verlag, New York, 1979.

[10] David J. Brady, Nikos P. Pitsianis, and Xiaobai Sun. Reference structure tomography. *JOSA A*, 21(7):1140–1147, 2004.

[11] P. Buchhave and C. H. Church. A wide range, rapid scanning spectrometer for emission and absorption measurements. *Applied Optics*, 7(11):2200–2204, 1968.

[12] Q. Cai and J. K. Aggarwal. Tracking human motion in structured environments using a distributed-camera system. *IEEE Trans on PAMI*, 2(11):1241–1247, 1999.

[13] T. M. Cannon and E. E. Fenimore. Tomographical imaging using uniformly redundant arrays. *Applied Optics*, 18(7):1052–1057, 1979.

[14] T. M. Cannon and E. E. Fenimore. Coded aperture imaging - many holes make light work. *Optical Engineering*, 19(3):283–289, 1980.

[15] K. Chakrabarty, S. S. Iyengar, H. Qi, and E. Cho. Coding theory framework for target location in distributed sensor networks. In *Proc. International Symposium on Information Technology: Coding and Computing*, pages 130–134, 2001.

[16] F. R. K Chung. *Spectral Graph Theory.* Amer. Math. Soc, Providence, RI, 1997.

[17] Thomas Clouqueur, Veradej Phipatanasuphorn, Parameswaran Ramanathan, and Kewal K. Saluja. Sensor deployment strategy for detection of targets traversing a region. *Mob. Netw. Appl.*, 8(4):453–461, 2003.

[18] Mikls Csros and Mikls Ruszink. Single user tracing superimposed codes. In *IEEE Symposium on Information Theory*, page 255, 2004.

[19] S.P. Davis, M.C. Abrams, and J.W. Brault. *Fourier transform spectrometry.* Academic Press, 2001.

[20] N.G. de Bruijn and T. van Aardenne-Ehrenfest. Circuits and trees in oriented linear graphs. *Simon Stevin*, 28:203–217, 1951.

[21] M. R. Descour, C. E. Volin, E. L. Derenaiak, T. M. Gleeson, Hopkins M. F., D. W. Wilson, and P. D. Maker. Demonstration of a computed-tomography imaging spectrometer using a computer generated hologram dispenser. *Appl. Opt.*, 36:3694–3698, 1997.

[22] B. C. Dougherty, R. M. Bunch, R. E. Lukens, J. Nagel, and W. L. Wolfe. Passive infrared device for detection of boundary crossings. United States Patent Application 20040129883, 2004.

[23] D-Z Du and F. K. Hwang. *Combinatorial Group Testing and its Applications.* World Scientific, second edition, 2000.

[24] B. Ford, M. R. Descour, and R. M. Lynch. Large-image-format computed tomography imaging spectrometer for fluorescence microscopy. *Opt. Express*, 9:444–453, 2001.

[25] M. Gardner. *The Binary Gray Code*. New York, 1986.

[26] A. F. H. Goetz, G. Vane, J. Solomon, and B. N. Rock. Imaging spectrometry for earth remote sensing. *Science*, 228:1147–1153, 1985.

[27] U. Gopinathan, D. J. Brady, and N. P. Pitsianis. Coded apertures for efficient pyroelectric motion tracking. *Optics Express*, 11(18):2142 – 2152, 2003.

[28] N. Griffith and M. Fernstrom. Litefoot: A floor space for recording dance and controlling media. In *Proceedings of the 1998 International Computer Music Conference*, pages 475–481, 1998.

[29] F. Harary. *Graph Theory*, pages 376–379. Addison-Wesley, MA, 1994.

[30] Chi-Fu Huang and Yu-Chee Tseng. The coverage problem in a wireless sensor network. In *ACM Intl Workshop on Wireless Sensor Networks and Applications*, 2003.

[31] S. H. Huang and F. K. Hwang. When is individual testing optimal for nonadaptive group testing? *SIAM Journal on Discrete Mathematics*, 14(4):540–548, 2003.

[32] P. Indyk. Deterministic superimposed coding with applications to pattern matching. In *Proceedings of the 38th IEEE Symposium on Foundations of Computer Science*, pages 127–136, 1997.

[33] K. Itoh, T. Inoue, and Y. Ichioka. Interferometric spectral imaging and optical three-dimensional fourier transformation. *Jap. J. Appl.Phys.*, 29:1561–1564, 1990.

[34] Paradiso J., Abler C., K. Hsiao, and Reynolds M. The magic carpet:physical sensing for immersive environments. In *Proceedings of CHI 97,ACM*, pages 277–278, 1997.

[35] T. Fisli J. Urbach and G. Starkweather. Laser scanning for electronic printing. In *Proceeding of IEEE*, pages 597 – 618, 1982.

[36] Jr. Decker John A. Hadamard-transform image scanning. *Applied Optics*, 9(6):1392–1395, 1970.

[37] D. Johnson and D. Dudgeon. *Array Signal Processing: Concepts and Techniques*. Prentice Hall, Englewood Cliffs, NJ, 1993.

[38] Avinash C. Kak and Malcolm Slaney. *Principles of computerized tomographic imaging*. IEEE Press, New York, 1988.

[39] J. Kauppinen and J. Partanen. *Fourier transforms in spectroscopy.* Wiley, 2001.

[40] W. H. Kautz and R. C. Singleton. Nonrandom binary superimposed codes. *IEEE Trans. on Information Theory*, 10:363–377, 1964.

[41] Jon Kerridge, Alistair Armitage, David Binnie, and Lucy Lei. Monitoring the movement of pedestrians using low-cost infrared detectors: Initial findings. In *Proceedings US Transport Research Board Annual Meeting*, 2004.

[42] A. D. Kersey, M. A. Davis, H. J. Patrick, M. LeBlanc, C. G. Askins K. P. Koo, M. A. Putnam, and E. J. Friebele. Fiber grating sensors. *J. Lightwave Technol.*, 15:1442C1462, 1997.

[43] S. Khan, O. Javed, Z. Rasheed, and M. Shah. Integrated vision sensor for detecting boundary crossings. In *Proceedings- Eighth IEEE International Conference on Computer Vision*, pages 331–336, 2001.

[44] J. Komlos and A. G. Greenberg. An asymptotically fast nonadaptive algorithm for conflict resolution in multiple access channels. *IEEE Trans. Information Theory*, 31:302–306, 1985.

[45] E. Lucas. *Rcrations Mathmatiques.* Gauthier-Villares, Paris, 1891.

[46] A. J. Macula. A simple construction of d-disjunct matrices with certain constant weights. *Discrete Math*, 162:311–312, 1996.

[47] Tom Madej. An application of group testing to the file comparison problem. In *9th Int. Conf. on Distr. Computing Systems*, pages 237–243, 1989.

[48] D. L. Marks, R. Stack, A. J. Johnson, D. J. Brady, and D. C. Munson. Cone-beam tomography with a digital camera. *Applied Optics*, 40(11):1795–1805, 2001.

[49] D. L. Marks, R. A. Stack, and D. J. Brady. Three-dimensional coherence imaging in the fresnel domain. *Applied Optics*, 38(8):1332–1342, 1999.

[50] D. L. Marks, R. A. Stack, D. J. Brady, D. C. Munson, and R. B. Brady. Visible cone-beam tomography with a lensless interferometric camera. *Science*, 284(5423):2164–2166, 1999.

[51] L. McElligott, M. Dillon, K. Leydon, B. Richardson, M. Fernstrom, and J.A. Paradiso. Forse fields - force sensors for interactive environments. In *UbiComp 2002, LNCS 2498*, pages 168–175, 2002.

[52] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. Srivastava. Coverage problems in wireless ad-hoc sensor networks. In *Proceedings of INFOCOM*, pages 1380 – 1387, 2001.

[53] S. Meguerdichian, F. Koushanfar, G. Qu, and M. Potkonjak. Exposure in wireless ad-hoc sensor networks. In *Proceedings of MOBICOM*, page 139C150, 2001.

[54] S. Meguerdichian, S. Slijepcevic, V. Karayan, and M. Potkonjak. Localized algorithms in wireless ad-hoc networks: Location discovery and sensor exposure. In *Proceedings of MOBIHOC*, pages 106–116, 2001.

[55] D. R. Miers, D. Raj, and J. W. Berthold. Design and characterization of fiber-optic accelerometers. In *Proceedings of SPIE*, volume 838, page 314, 1987.

[56] Mark A. Neifeld and Premchandra Shankar. Hadamard-transform image scanning. *Applied Optics*, 42(17):3379–3389, 2003.

[57] H. Ngo and D. Du. New constructions of non-adaptive and error-tolerance pooling designs. *Discrete Math*, 243:161–170, 2002.

[58] P. Potuluri, U. Gopinathan, J. R. Adleman, and D. J. Brady. Lensless sensor system using a reference structure. *Optics Express*, 11(8):965–974, 2003.

[59] P. Potuluri, M. Xu, and D. J. Brady. Imaging with random 3d reference structures. *Optics Express*, 11(18):2134 – 2141, 2003.

[60] C.A.B. Smith and W.T. Tutte. On unicursal paths in a network of degree 4. *Amer. Math. Monthly,*, 48:233–237, 1941.

[61] W. B. Spillman and R. L. Gravel. Moving fiber optic hydrophone. *Optics Lett.*, 5:30–33, 1980.

[62] D. Tian and N. D. Georganas. A coverage-preserving node scheduling scheme for large wireless sensor networks. In *ACM Intl Workshop on Wireless Sensor Networks and Applications*, 2002.

[63] A. Turner, C. Mottram, and A. Penn. An ecological approach to generative design. In *Gero, JS ed Design Cognition Computing '04*, pages 259 – 274, 2004.

[64] E. Udd, W. Schulz, D.V. Nelson J. Seim, and A. Makino. Transverse fiber grating strain sensors based on dual overlaid fiber gratings on polarization maintaining fiber. In *Proceedings of SPIE*, volume 3330, page 253, 1998.

[65] J. Wang and S. Singh. Video analysis of human dynamics: a survey. Real Time Imaging (in press), 2003.

[66] Zhaochun Xu, Zhanglei Wang, Michael E. Sullivan, and David J. Brady. Multimodal multiplex spectroscopy using photonic crystals. *Optics Express*, 11(18):2126–2133, 2003.

[67] F. Ye, G. Zhong, S. Lu, and L. Zhang. A robust energy conserving protocol for long-lived sensor networks. In *Intl Conf. on Distributed Computing Systems*, 2003.

[68] S. Zahnd, P. Lichtsteiner, and T. Delbruck. Integrated vision sensor for detecting boundary crossings. In *Proceedings - IEEE International Symposium on Circuits and Systems*, pages 376–379, 2003.

[69] Yunhui Zheng, David J. Brady, M. E. Sullivan, and B.D. Guenther. Fiber optical localization by geometric space coding with 2d gray code. *To appear in Applied Optics*, 2005.

[70] Yunhui Zheng, Nikos P. Pitsianis, and David J. Brady. A fiber sensor floor web for multi-person tracking. *Summited to IEEE Sensor Journal*, 2004.

Biography

Yunhui Zheng was born in Fuzhou, China on September 21,1974. He spent his childhood in the No.2 Primary school in Fuzhou and later entered the No.1 Middle school in Fuzhou. Exempted from the national entrance examination, he joined Zhejiang University in September 1993. There, during the first two years, he was in the honored class to receive the top level education in math, physics and engineering. Later, he continued his college years in the department of Optical Engineering. In September 1997, exempted from the national entrance examination again, he joined the graduate school at Zhejiang University and received a Master's degree in Optical Engineering in March, 2000. He immediately came to the graduate school at the University of Illinois, Urbana-Champaign in Fall 2000 and received another Master's degree in Electrical Engineering under Professor David Brady's supervision. He then moved with Professor Brady to Duke University to continue his graduate study and research in optical sensor systems. He is expected to receive his Ph.D. degree in Electrical and Computer Engineering in 2005.

Publications

Yunhui Zheng, David J. Brady, M. E. Sullivan, and B.D. Guenther. Fiber optical localization by geometric space coding with 2d gray code. *To appear in Applied Optics*, 2005.

Yunhui Zheng, Nikos P. Pitsianis, and David J. Brady. A fiber sensor floor web for multi-person tracking. *Submited to IEEE Sensor Journal*, 2004.

Yunhui Zheng, David J. Brady, and Pankaj K. Agarwal. Boundary sensor: An analysis based on graph theory. *Manuscript in preparation for ACM Transaction of Sensor Network*, 2005.

Yunhui Zheng. Information theoretic design and optimization of imaging system. In *Master thesis*, University of Illinois at Urbana Champaign, 2001.

Proceedings and Conference Talks

S.D. Feller, Y. Zheng, E. Cull, and D.J. Brady. Tracking and imaging humans on heterogeneous infrared sensor arrays for law enforcement applications. In *In sensors, and Command, Control, Communications, and Intelligence (C31) Technologies for Homeland Defense and Law Enforcement,The International Society for Optical Engineering*, pages 212–221, 2002.

Yunhui Zheng, David J. Brady, B. D. Guenther, and Nikos P. Pitsianis. Fiber optical web for object localization by geometric space coding. In *OSA Annual Meeting 2004*, Rochester, NY, 2004.

Yunhui Zheng, Mike Sullivan, and David J. Brady. Differential tracking system. In *OSA Annual Meeting 2003*, Tuscon,AZ, 2003.