I, Zongjie Tu, hereby submit this original work as part of the requirements for the degree of Master of Science in Computer Science.

It is entitled:
Game-theoretic Multi-camera Surveillance over Arbitrary Floor Plan

Student's name: Zongjie Tu

This work and its defense approved by:

Committee chair: Prabir Bhattacharya, Ph.D.
Committee member: Raj Bhatnagar, Ph.D.
Committee member: Chia Han, Ph.D.
Game-theoretic Multi-camera Surveillance over Arbitrary Floor Plan

A thesis submitted to the
Graduate School
of the University of Cincinnati
in partial fulfillment of the
requirements for the degree of

Master of Science in Computer Science

in the Department of Electrical Engineering and Computing Systems
of the College of Engineering and Applied Science

Nov 2013

by

Zongjie Tu

Committee Chair: Dr. Prabir Bhattacharya
Abstract

Coordinated multi-resolution tracking over arbitrary floor plan is addressed using a game-theoretic approach. An enhanced radial sweep algorithm is devised to find the polygon of visibility at any point on or inside a polygon that contains vision-obstructing polygonal entities. By sampling edges of a polygon and edges of any polygonal hole inside that polygon, a two-pass 0-1 programming process is formulated to find a near-optimal set of camera samples that can dynamically cover, at a high probability, area under surveillance in the presence of camera handoffs. Radius Multiplier is introduced to handle partial visibility and is set to 1 by default to avoid insolvability of 0-1 programming problems. As a remedy to excessive redundancy triggered by camera clustering, we set camera redundancy to a fixed value of 3 for any block with concave valid area. Branch-and-cut algorithm is employed to solve 0-1 programming problems. Assigning a fixed value to Camera Redundancy of blocks with concave Valid Area, setting Radius Multiplier to a non-zero value, and utilizing secondary utility yielded better simulation results for various types of floor plans. Raising Camera Redundancy of blocks with non-concave Valid Area contributed to performance boost and in the meantime, increased the number of cameras needed.
Acknowledgements

I would like to express my sincere gratitude to my advisor Dr. Prabir Bhattacharya for helping me choose my research topic and guiding me throughout my study with unfailing patience. I am grateful for all the support my family has given me during the pursuit of my Master degree. My thanks also go to the EECS department at the University of Cincinnati, which has provided me a comfortable learning environment.
## Contents

1 Introduction ............................................................................................................................ 1  
   1.1 Problem Statement ...................................................................................................................... 1  
   1.2 Thesis Overview ...................................................................................................................... 2  

2 Related Work .......................................................................................................................... 3  
   2.1 Decentralized Game-theoretic Tracking ................................................................................... 3  
   2.2 Computation of Polygon of Visibility ........................................................................................ 5  

3 A Prototype .............................................................................................................................. 8  
   3.1 Motivation .................................................................................................................................... 8  
   3.2 Definitions .................................................................................................................................... 9  
   3.3 Assumptions .................................................................................................................................. 11  
   3.4 An Improved Algorithm ............................................................................................................ 13  
       3.4.1 Secondary Global Utility ......................................................................................................... 13  
       3.4.2 Precedence Given to Border Blocks ........................................................................................ 15  
   3.5 Simulation of Camera Functioning ......................................................................................... 15  
   3.6 Simulation of Network Functioning ........................................................................................ 16  
   3.7 Evaluation Metrics .................................................................................................................... 16  
       3.7.1 Number of Failed Handoffs ...................................................................................................... 16  
       3.7.2 Number of Complete-coverage Failures .................................................................................. 16  
       3.7.3 Number of Uncovered Blocks per Handoff .......................................................................... 16  
       3.7.4 Times of Camera Adjustment per Covering Handoff ................................................................. 17  
       3.7.5 Change of Directions in Degrees per Covering Handoff .......................................................... 17  
   3.8 Evaluation of Prototype Performance ..................................................................................... 17  
       3.8.1 Simulation Setup ....................................................................................................................... 17  
       3.8.2 Effect of Secondary Utility and Precedence Given to Border-covering Directions ............... 17  
       3.8.3 Overall Comparison of Original and Improved Algorithms .................................................... 19  
       3.8.4 Choice of the Number of Cameras .......................................................................................... 19  
       3.8.5 Limitations of Our Prototype .................................................................................................. 19  

4 Overview of Our Final Solution .......................................................................................... 21  
   4.1 Goal ............................................................................................................................................. 21  
   4.2 Contributions ............................................................................................................................ 21  

5 Enhancing the Radial Sweep Algorithm for Finding Polygon of Visibility ...................... 23  
   5.1 Flaws in the Radial Sweep Algorithm ..................................................................................... 23  
   5.2 Modifying the Radial Sweep Algorithm .................................................................................. 26  
       5.2.1 Amending TYPE1 in HandleEventPoint .................................................................................. 26
5.2.2 Amending TYPE3 in HandleEventPoint ................................................................. 26
5.2.3 Restricting polar angles of two vertices of the same edge to one period of a turn by default........ 27
5.2.4 Splitting the edge that crosses the pole ................................................................. 28
5.2.5 Defining intersection of an edge and a ray that overlaps with that edge................................. 28
5.2.6 Assigning special polar-angle values to edge vertices corresponding to a vertex-coincident pole.. 29
5.2.7 Amending initialization steps .................................................................................. 30
5.2.8 Keeping any edge containing the pole when back edges are pruned................................. 32
5.2.9 Summary .................................................................................................................. 32

6 Computation of Near-optimal Camera Quantity and Locations........................................36

7 System Design .............................................................................................................43
  7.1 Assumptions...............................................................................................................43
  7.2 Secondary Utility and Precedence Given to Border-covering Camera Directions..........44

8 Experimental Results ..................................................................................................45
  8.1 Evaluation Metrics .................................................................................................45
  8.2 Simulation Setup .....................................................................................................45
  8.3 Four Schemes of Floor Plan ..................................................................................46
    8.3.1 Scheme A .........................................................................................................46
    8.3.2 Scheme B .........................................................................................................46
    8.3.3 Scheme C .........................................................................................................46
    8.3.4 Scheme D .........................................................................................................47
  8.4 Benefit of Introducing Secondary Utility .............................................................47
  8.5 Benefit of a Setting Radius Multiplier to 1............................................................49
  8.6 Justification of Assigning the Fixed Value 3 to CR for Blocks with Concave VA.........50
  8.7 Ineffectiveness of Precedence Given to Border-covering Camera Directions..........52
  8.8 Improving System Performance by Increasing CR ...............................................52

9 Conclusion and Future Work .....................................................................................54

Bibliography....................................................................................................................55

Appendix..........................................................................................................................58
List of Figures

Fig. 1  System architecture proposed by Song et al. ..........................................................4
Fig. 2  An example of Polygon of Visibility marked in gray ...........................................5
Fig. 3  FindVisibilityPolygon function of the radial sweep algorithm. ..............................6
Fig. 4  HandleEventPoint function of the radial sweep algorithm ......................................7
Fig. 5  The HAOV of a camera .........................................................................................9
Fig. 6  A sample utility map ............................................................................................14
Fig. 7  A sample primary utility map ................................................................................14
Fig. 8  A sample secondary utility map ...........................................................................14
Fig. 9  Schemes of camera locations for simulation ..........................................................18
Fig. 10 Simulation results for Scheme B. .........................................................................18
Fig. 11 Overview of our final solution ...............................................................................22
Fig. 12 The first situation in which the radial sweep algorithm [17] will fail, i.e. the camera is
collinear with an edge .................................................................................................24
Fig. 13 The second situation in which the radial sweep algorithm [17] will fail .................24
Fig. 14 The third situation in which the radial sweep algorithm [17] will fail, i.e. the camera lies
on one edge of a hole ..................................................................................................26
Fig. 15 Pseudo-code snippet for solving the problem illustrated in Fig. 12 .......................27
Fig. 16 Pseudo-code snippet for solving the problem illustrated in Fig. 13 .......................27
Fig. 17 An example of vertex-coincident pole causing right-hand vision obstruction ......30
Fig. 18 An example of vertex-coincident pole free of right-hand vision obstruction ........31
Fig. 19 A special flaw similar to the one exemplified in Fig. 13 .......................................31
Fig. 20 Pseudo-code snippet for solving the problem illustrated in Fig. 19 .......................32
Fig. 21 FindVisibilityPolygon of the enhanced radial sweep algorithm ..........................33
Fig. 22 HandleEventPoint of the enhanced radial sweep algorithm .................................34
Fig. 23 A partially visible block ......................................................................................40
Fig. 24 Insolvability caused by a Radius Multiplier of 2 ..................................................40
Fig. 25 An example of Camera Clustering ........................................................................42
Fig. 26 An example of VA consisting of three disjoint parts ...........................................42
Fig. 27 Four schemes of floor plan over which simulation was run ...................................47
Fig. 28 Benefit of introducing secondary utility in terms of four evaluation metrics ..........48
Fig. 29 Improvement in terms of four evaluation metrics when RM is set to 1 instead of 0...50
Fig. 30 Results for Scheme D obtained when CR’s take identical and different values,
respectively ..................................................................................................................51
List of Tables

Table 1  Original algorithm versus improved algorithm...........................................................19
Table 2  Average declines in percentage thanks to secondary utility for the four evaluation metrics. ..........................................................................................................................49
Table 3  Results for Scheme D obtained when CR’s take identical and different values, respectively, as illustrated in Fig. 30. ........................................................................................................52
Table 4  Results for Scheme A obtained before and after precedence is given to border-covering camera directions..................................................................................................................53
Table 5  Results for Scheme A obtained when CR for blocks with non-concave VA is set to 3, 4 and 5, respectively. ..................................................................................................................53
1 Introduction

1.1 Problem Statement

With the emergence of network cameras, video camera networks are becoming increasingly popular. A large number of research efforts have been made to attain efficient coordination between network cameras and to maximize resource utilization [1-9]. Mittal and Davis [10] devised a solution for optimally deploying cameras, aiming at handling occlusions incurred by dynamic objects. Park and Bhat [11] provided a solution for camera handoffs in large camera networks by constructing a distributed look-up table so that the most favorable cameras for viewing a given point in the frustum of a given camera can be selected. Thanks to the distributed look-up table, both computational cost and network traffic were heavily reduced. Nevertheless neither of the solutions above applies to Pan/Tilt/Zoom (PTZ) cameras. PTZ cameras are more flexible than static ones as PTZ cameras can be dynamically reconfigured in a real-time manner. As an effort to enhance the performance of PTZ camera network, Kansal et al. [12] proposed a laser system to generate a map of static obstacles. Upon detection of an event of interest, the map is exploited to select and reconfigure the best PTZ camera for monitoring the event of interest at a high resolution. In the mean time, the other cameras are reconfigured to cover the remaining area at a low resolution. Piciarelli, Micheloni, and Foresti [13] developed a similar system by using map of activities instead of map of obstacles to aid reconfiguration of the camera network. However, neither [12] nor [13] addresses the problem of optimal offline camera placement.

The research problem involved in [12] and [13] can be generalized as tracking targets with PTZ cameras at multiple allowable resolutions, which we refer to as coordinated multi-resolution tracking.
A real-life example is as follows. There may be a suspicious-looking person in an airport hall that needs to be identified. A network camera near the suspicious-looking person must be reconfigured to focus on his or her face at a high resolution (i.e. in the zoom mode). Since the zoom-in action may cause the angle of view (AOV) of the network camera in question to shrink remarkably, other network cameras may need to adjust their directions to monitor, at a low yet acceptable resolution (i.e. in the normal mode), any new uncovered area.

Motivated by the remarks above, we have developed, utilizing PTZ cameras, a cost-effective (i.e. utilizing decentralized game-theoretic model [1]), generalized (i.e. widely applicable) and optimized (i.e. considering camera placement optimality) solution for coordinated multi-resolution tracking over an arbitrary floor plan containing obstacles.

1.2 Thesis Overview

Structure of the rest of this paper is as follows. Chapter 2 reviews related work. In Chapter 3, a preliminary prototype is depicted and then evaluated in terms of performance. Chapter 4 outlines our final solution with an overview. Chapter 5 deals with the computation of polygon of visibility by means of an enhanced radial sweep algorithm. In Chapter 6, the rationale for computing near-optimal camera quantity and locations is discussed. Chapter 7 describes the final system design. Experimental results are analyzed in Chapter 8 and conclusion is put forward in Chapter 9. An appendix at the end of this paper briefly reviews some relevant background on game theory.
2 Related Work

2.1 Decentralized Game-theoretic Tracking

A decentralized game-theoretic model can be applied to coordinated multi-resolution tracking [1,8,14]. Network cameras can be regarded as players and the size of the exclusively covered area can be deemed utility (i.e. payoff). A player always tries to maximize its utility. The strategy set for each network camera can be defined as the collection of all possible directions (e.g. from 0 to 360 degrees for an omnidirectional network camera) for each network camera. In game theory, a game can be either cooperative or non-cooperative [15]. In the former case, players form coalitions to compete with other players, and utility is defined for each coalition instead of each individual player. In the latter case, each player strategizes independently for its own interest (i.e. utility). The non-cooperative game is more suitable for the example described above than its cooperative counterpart, as every network camera is treated equally. There are no obvious criteria available to divide network cameras into coalitions of particular interests. When no network camera (i.e. player) is able to increase its area coverage (i.e. utility) by adjusting its direction (i.e. strategy) given the other network cameras’ directions, a Nash equilibrium [15] is reached. (See Appendix for the definition of Nash equilibrium.)

For the coordinated multi-resolution tracking, a Nash equilibrium can always be reached given sufficient time for camera adjustment [16]. The Nash equilibrium for coordinated multi-resolution tracking comprises the directions all network cameras have chosen, which are collectively called strategy profile [15]. Since each network camera is enforcing a pure strategy [15], in which a network camera direction is chosen with the probability of 1, the Nash equilibrium above is a pure-strategy Nash equilibrium [15]. Incidentally, as each network camera has complete information [15] regarding
strategy sets and utilities of the other network cameras, and each network camera is allowed to communicate with one another, the pure-strategy Nash equilibrium above is not a Bayesian Nash equilibrium [15].

The architecture for coordinated multi-resolution tracking proposed by Song et al. [1,8,14] is shown in Fig. 1. In their solution, each network camera watches out for any suspicious-looking object, which we refer to as moving target of interest, within its coverage while ensuring the maximization of its utility, i.e. the area that it exclusively monitors. When an action (e.g. tracking a suspicious-looking target in the zoom mode) is taken that triggers a change in a network camera (e.g. giving up original coverage), that network camera propagates its change to other network cameras within the video camera network. Each network camera receiving the change updates its local store of camera states and recalculates the direction that maximizes its own utility. If the recalculated direction differs from the current one, the recipient will also change direction and propagate that directional change into the video camera network. Although the game theoretic framework proposed in [1,8,14] has the merits of decentralization and high accuracy, Song et al. did not give quantitative evaluation of their design in terms of unsuccessful handoff, uncovered area, or cost of camera adjustment. Besides, their experimental results were specific to one particular camera setup and hence lacked generality.

Fig. 1  System architecture proposed by Song et al.
2.2 Computation of Polygon of Visibility

Erdem and Sclaroff [17] proposed a radial sweep algorithm in the context of polar coordinate system to find the polygon of visibility (PV) for an omnidirectional camera placed in a polygonal region. That polygonal region, denoted as $P_e$, may contain vision-obstructing polygons that were referred to as “holes” and that were denoted as $P_k$ ($k=1, 2, \ldots, K$). An example of PV is depicted in Fig. 2. The red circle depicts location of the observer (i.e. camera). The black line segments form the boundary of the polygonal region under surveillance. The solid blue polygons are the vision-obstructing holes. The core algorithm and function of the radial sweep algorithm, namely FindVisibilityPolygon [17] and HandleEventPoint [17], are shown in pseudo-code in Fig. 3 and Fig. 4, respectively. (We are including these two figures because we shall later refer to them in Chapter 5 as we discuss enhancing modifications. In both Fig. 3 and Fig. 4, SL stands for SortedList. For detailed definitions of SortedList and Backward Facing Edges, please refer to [17].) The core idea can be described as hitting all vertices with an imaginary rotating half line hinged at the camera, forming the outline of PV after every vertex has been visited. It should be noted that each vertex is represented by two coincident virtual vertices, as is entailed by the algorithm in [17]. The two coincident vertices belong, respectively, to the two edges that the actual single vertex joins. From this point on, we will call those virtual vertices the edge vertices. In [17], an edge vertex can be either a start vertex or an end vertex,

Fig. 2  An example of Polygon of Visibility marked in gray.
Fig. 3  FindVisibilityPolygon function of the radial sweep algorithm.

depending on the enumerating order of vertices (i.e. clockwise or counterclockwise). We will also
denote an edge as \(\{j, j+1\}\), where \(j\) is the index of start vertex and \(j+1\) is the index of end vertex, both
generated as per one particular enumerating order (i.e. counterclockwise for \(P_s\) and clockwise for
\(P_k\’s\)). The radial sweep algorithm can be used, along with 0-1 programming, to determine optimal
camera placement, which has been reduced to a Set-covering Problem [18] by grid approximation
[17]. Unfortunately, the proposed radial sweep algorithm will not work if the camera lies on one edge
of a hole or coincides with a vertex. Moreover, some special circumstances have not been considered
in that algorithm.
<table>
<thead>
<tr>
<th>Line</th>
<th>Function HandleEventPoint (CurrentVertex, ActiveEdge, PV, SL)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><strong>input</strong>: The current vertex <code>CurrentVertex</code>, the active edge <code>ActiveEdge</code>, the visibility polygon <code>PV</code> and the sorted list <code>SL</code>.</td>
</tr>
<tr>
<td>2</td>
<td><strong>output</strong>: The function operates on the input parameters.</td>
</tr>
<tr>
<td>3</td>
<td><strong>begin</strong></td>
</tr>
<tr>
<td>4</td>
<td>TYPE1</td>
</tr>
<tr>
<td>5</td>
<td>push (CurrentVertex, PV);</td>
</tr>
<tr>
<td>6</td>
<td>while SL ≠ NULL do</td>
</tr>
<tr>
<td>7</td>
<td>e ← head (SL);</td>
</tr>
<tr>
<td>8</td>
<td>if endvertexof (e) &gt; CurrentVertex <strong>then</strong></td>
</tr>
<tr>
<td>9</td>
<td>k ← intersect (e, CurrentVertex);</td>
</tr>
<tr>
<td>10</td>
<td>push (k, PV);</td>
</tr>
<tr>
<td>11</td>
<td>ActiveEdge ← e;</td>
</tr>
<tr>
<td>12</td>
<td>break;</td>
</tr>
<tr>
<td>13</td>
<td>endif</td>
</tr>
<tr>
<td>14</td>
<td>pop (SL);</td>
</tr>
<tr>
<td>15</td>
<td>endw</td>
</tr>
<tr>
<td>16</td>
<td>endif</td>
</tr>
<tr>
<td>17</td>
<td><strong>if</strong> CurrentVertex is a start vertex <strong>then</strong></td>
</tr>
<tr>
<td>18</td>
<td>k ← intersect (ActiveEdge, CurrentVertex);</td>
</tr>
<tr>
<td>19</td>
<td>e ← incidentedg eof (CurrentVertex);</td>
</tr>
<tr>
<td>20</td>
<td>TYPE2</td>
</tr>
<tr>
<td>21</td>
<td>insert (e, SL);</td>
</tr>
<tr>
<td>22</td>
<td>2.2</td>
</tr>
<tr>
<td>23</td>
<td>TYPE3</td>
</tr>
<tr>
<td>24</td>
<td>push (k, PV);</td>
</tr>
<tr>
<td>25</td>
<td>push (CurrentVertex, PV);</td>
</tr>
<tr>
<td>26</td>
<td>ActiveEdge ← e;</td>
</tr>
<tr>
<td>27</td>
<td>endif</td>
</tr>
<tr>
<td>28</td>
<td>endif</td>
</tr>
<tr>
<td>29</td>
<td>end</td>
</tr>
</tbody>
</table>

**Fig. 4** HandleEventPoint function of the radial sweep algorithm.
3 A Prototype

3.1 Motivation

Song et al. [1] adopted the aforementioned game theoretic methodology in tackling the multi-resolution tracking problem and has attained preliminary success. In their solution, cameras monitoring a rectangular area communicate their states to each other via the network. After a moving target of interest is chosen, a network camera whose zoom-mode range covers the current location of the moving target of interest is designated to track that target and predicts its location in a real-time fashion. For network cameras not designated to track any moving target of interest, the only task is to periodically calculate the optimal direction that maximizes camera utility and then adjust to that direction. A handoff between two network cameras is needed when the moving target of interest moves out of the range that zoom mode of the former network camera can cover, and enters the range that zoom mode of the latter network camera (i.e. the newly designated one) can cover. After a handoff takes place, all network cameras except the newly designated network camera calculate their respective optimal direction and then adjust to that direction. However, Song et al. [1] did not give any quantitative evaluation of their prototype. Besides, their experimental results were specific to one particular setup in terms of camera locations and hence lacked generality. To address these issues, we embarked on implementation of our own prototype [16], on which we performed simulation with different setups in terms of camera locations and evaluated the performance in a quantitative style. Next, we improved that prototype by introducing a secondary utility and giving precedence to camera directions that cover border regions. We also compared performance of our improved prototype to that proposed by Song et al. [1], again, in a quantitative style.
3.2 Definitions

The following is a list of concepts we use. Some of them were originally defined in [1].

- **Horizontal Angle of View (HAOV):** magnitude of the angle in degrees that a network camera facing a certain direction can see in the horizontal plane. Fig. 5 shows the HAOV of a network camera.

![Fig. 5 The HAOV of a camera.](image)

- **Target:** a 1 meter by 1 meter block on the ground.

- **Moving Target:** a person that is traversing the rectangular area under surveillance.

- **Moving Target of Interest:** a person that is traversing the rectangular area under surveillance and that is being tracked by one network camera in zoom mode. Moving targets except the one being tracked in zoom mode need not be explicitly assigned to any particular network camera, since the whole area under surveillance is supposed to be covered.

- **Target Utility:** the utility value of a target is 1 if tracked by at least one network camera and 0 otherwise.
• **Primary Target Utility**: the primary utility value of a target is 1 if the target is tracked by exactly one network camera or 0 otherwise.

• **Secondary Target Utility**: the secondary utility value of a target is 1 if the target is tracked by exactly two network cameras or 0 otherwise. The secondary target utility takes into account the overlapping coverage of a network camera that is shared with exactly one other network camera.

• **Global Utility**: sum of Target Utility of all targets.

• **Primary Global Utility**: sum of Primary Target Utility of all targets.

• **Secondary Global Utility**: sum of Secondary Target Utility of all targets.

• **Camera Utility**: the utility of a network camera tracking some set of targets is defined as the marginal contribution to Global Utility as a result of tracking. In other words, the camera utility is the change in Global Utility as a result of that camera tracking the set of targets against not tracking them [1]. This utility was first posed by Tumer et al. [19] as Wonderful Life Utility (WLU).

The aim of our prototype can now be stated as ensuring complete-coverage, or maximum-coverage corresponding to maximum global utility if complete-coverage is not achievable, of a large rectangular area by means of adjusting network cameras' directions dynamically while allowing one camera to, at any given time, embark on tracking a moving target of interest (e.g. a human face) in zoom mode.
3.3 Assumptions

Before we can build a prototype for simulation, we need to make reasonable assumptions regarding our scenario and assign proper values to parameters.

- The network cameras in use should be sufficient in quantity so that complete-coverage can be attained within a very short period of time for the majority (e.g. two-thirds) of handoffs as long as the cameras are deployed properly and controlled using a smart and efficient algorithm. Ideally, the system should be free from complete-coverage failure until next handoff takes place.

- We choose the monitored area to be 30 meters by 20 meters in size and at least 8 network cameras are installed on the boundaries of the rectangular area.

- The spacing between two network cameras installed next to each other along the same edge is fixed, which implies symmetry in camera deployment.

- If we choose 1 meter by 1 meter as the block size, there will be a total of 600 blocks for the 30 meters by 20 meters area. We cannot, owing to an accuracy-versus-cost tradeoff, choose too small a value for the block size (e.g. 0.1 meter by 0.1 meter) that results in much more blocks.

  - Pros of smaller blocks: more accuracy and hence fewer suboptimal decisions (i.e. directions), faster convergence (i.e. less adjustment) and fewer false-positive coverage failures (e.g. a block jointly covered by two network cameras each by half but not recognized as covered by either of the two).

  - Cons of smaller blocks: more computation, more memory usage (for utility maps), and more network traffic.
Considering network cameras are generally not “intelligent” enough to handle large amounts of computation and do not have large memory, block sizes that will result in a reasonable quantity of blocks (e.g. less than 1000) are preferable.

The capabilities of the network cameras in use should be at least equivalent to those of consumer-level PTZ cameras such as TRENDNET TV-IP600 (1/4” color CMOS sensor with a height of 2.4mm, 4.57mm focal length, 320 x 240 resolution, 4x zoom, and 46 degrees HAOV) [20]. Assuming that we can identify a human face with 50 pixels [21] in height and a human face is 0.33 meter vertically, the maximum distance (i.e. radius of coverage) for the zoom mode is

$$0.33 \times (240 + 50) \times 4.57 \times 4 + 2.4 = 12 \text{ meters.} \quad (1)$$

Suppose that the maximum distance for acceptable resolution in the normal mode is twice the zoom mode maximum distance, i.e. 24 meters, a 1.7-meter tall human will occupy about 32 pixels vertically on screen, which is an acceptable resolution for identifying an target as a human.

It should be noted that the maximum distance in either normal mode or zoom mode must be greater than the half of the shorter edge of rectangular area, otherwise complete-coverage can never be achieved and handoffs can never be successful.

There must be at least one network camera on each longer edge of the rectangular area. Otherwise unsuccessful handoffs are inclined to occur. In extreme cases in which there is no network camera on either longer edge and the maximum distance in zoom mode is shorter than half of the longer edge, handoffs will always be unsuccessful.

The network cameras operate fast enough to track any moving target in real time.
The time a network camera spends computing its optimal direction is much less than that it takes for any other network camera to change direction.

The network cameras are connected via a fast local area network such as 100Mbps Ethernet. Thus, propagation of change in camera direction and target state within the network is instant, that is, every camera always has up-to-date information regarding the whole system. In simulation, this requirement is fulfilled by shared memory, to which every camera has access.

We do not take into consideration tilting of a network camera.

### 3.4 An Improved Algorithm

We implemented two algorithms for simulation. The first is a naive implementation of the algorithm proposed in [1]. The second enhances the first by introducing secondary utility function and giving precedence to camera directions that cover border regions.

#### 3.4.1 Secondary Global Utility

We modify the original algorithm in [1] by introducing the secondary global utility, which ensures less directional reconfiguration when a camera quits collaborative monitoring and starts to track a single moving target of interest in zoom mode. The secondary target utility is evaluated only when the primary target utility is the same for two or more directions.

Primary and secondary global utilities can actually be implemented using one utility map that uses 2 instead of 1 as the maximum value for a single element. For example, the utility map for a 3 meters by 2 meters area containing 6 blocks monitored by two network cameras, namely $C_1$ and $C_2$, may look like Fig. 6.
Assume that Block (1, 2) (i.e. Row 1, Column 2) and Block (2, 1) are covered (i.e. tracked) exclusively by $C_1$, Block (2, 3) is exclusively covered by $C_2$, and Block (1, 3) is covered by both $C_1$ and $C_2$. The other 2 blocks are uncovered. The derived primary utility map is shown in Figure 7 and the derived secondary utility map in Figure 8.

Hence, the primary camera utility is 2 for $C_1$ given $C_1$’s exclusive coverage of both Block (1, 2) and Block (2, 1), and 1 for $C_2$ given $C_2$’s exclusive coverage of Block (2, 3). The secondary camera utility is 1 for both $C_1$ and $C_2$ because they cover one common block, that is, Block (1, 3).
Notions such as tertiary global utility can also be derived this way. However, such notions are of little use in our scenario since at most one moving target of interest will be tracked in zoom mode. Put another way, only one backup network camera is needed when another network camera that used to have overlapping coverage with the former suddenly gives up tracking the overlapping blocks and starts to track a moving target of interest in zoom mode.

3.4.2 Precedence Given to Border Blocks

The two camera directions that yield coverage of border blocks along the four edges of the rectangular area, namely \((H:\text{AOV} / 2)\) degrees and \((180 - H:\text{AOV} / 2)\) degrees, are given precedence (i.e. more weight) over the interior ones in our improved design. This diminishes chances of complete-coverage failures on border blocks, which are less likely to be covered than interior blocks by a random configuration of network camera directions. The default value for the precedence is, in our design, 0.5 for the two directions mentioned above. For instance, if the camera utility value for the direction of \((180 - H:\text{AOV} / 2)\) degrees is originally 5, the final camera utility value by default will be

\[
(1 + 0.5) \times 5 = 7.5.
\]  

3.5 Simulation of Camera Functioning

As is the same with [1], we mandate that only one network camera is updated at a time. In each “round”, we first generate a random sequence of network cameras. Then we select, one after another, a network camera in the order of the sequence to let that network camera perform all operations that are necessary at the time being selected. These operations include updating utility map, adjusting direction, communicating changes to other cameras, predicting the location of the moving target of interest if any, etc.
3.6 Simulation of Network Functioning

Network communication is simulated by means of shared memory. Considering the data in exchange (i.e. directional data) are not large in volume and all cameras are on a fast LAN, this simplification is acceptable.

3.7 Evaluation Metrics

The system performance can be measured by a variety of metrics. The following are the five evaluation metrics we employ, listed in the order of importance.

3.7.1 Number of Failed Handoffs

This metric signifies the number of events in which a camera tracking a moving target of interest cannot find an eligible successor to continue tracking that target in zoom mode. Put another way, the moving target of interest will be lost after a failed handoff. Occurrence of a failed handoff usually suggests the cameras in use are not insufficient in quantity. When it comes to our simulation results, this number is zero by default unless otherwise stated.

3.7.2 Number of Complete-coverage Failures

This metric indicates the number of events in which the area under surveillance is not completely covered by network cameras that operate in normal mode after a successful handoff occurs. Note that in our simulation false-positive coverage failures are excluded by eliminating any block that is jointly covered in full by two or more network cameras, each covering the block in part. This is only intended for more accurate evaluation of algorithm performance, yet not to be included as part of the actual algorithm. The actual algorithm calculation is still block-based, and will make suboptimal decisions from time to time for lack of inaccuracy.

3.7.3 Number of Uncovered Blocks per Handoff

This metric measures severity of complete-coverage failures on the whole.
3.7.4 Times of Camera Adjustment per Covering Handoff
A covering handoff is a handoff after which complete-coverage is achieved through zero or more camera adjustments before next handoff takes place. In other words, a handoff is a covering handoff if it is not followed by a complete-coverage failure.

3.7.5 Change of Directions in Degrees per Covering Handoff
This metric, together with the times of camera adjustment per covering handoff, indicates how fast the network cameras achieve a complete-coverage.

3.8 Evaluation of Prototype Performance
3.8.1 Simulation Setup
In our software simulation, we use 8 network cameras with a same value for HAOV, that is, 46 degrees. Each simulation run consists of 1000 handoffs with randomly generated moving targets of interest. Simulation is run on three schemes of camera locations, as is shown in Figure 9. Gray area signifies coverage of one or more network cameras. The darker a region is, the more network cameras covering it. Blue boxes represent moving targets and the red box the moving target of interest. Yellow boxes are drawn to highlight uncovered blocks. The network camera that is tracking the moving target of interest in zoom mode is drawn as a solid semicircle, or a solid quarter-circle if in corner, with a light-red line pointing from itself to the moving target of interest and a light-red ring indicating its maximum distance in zoom mode. Network cameras operating in normal mode are drawn as red hollow semicircles or quarter-circles.

3.8.2 Effect of Secondary Utility and Precedence Given to Border-covering Directions
Fig. 10 shows the simulation results in terms of the aforementioned evaluation metrics. It can be seen that introducing either secondary utility or precedence contributes to decline in the last four metrics we use, thus improving system performance. (All the values for the first metric, i.e. the number of failed handoffs, are zeros.) The introduction of both secondary utility and precedence generally results
in further decline except for a slight increase in the number of uncovered blocks per handoff compared to introducing precedence alone, which can be seen in Fig. 10(b). However, as our primary concern lies in the decline of complete-coverage failures, this little tradeoff is worthwhile.

---

**Fig. 9** Schemes of camera locations for simulation.

**Fig. 10** Simulation results for Scheme B.
3.8.3 Overall Comparison of Original and Improved Algorithms

Table 1 summarizes, for all three schemes, performance boosts after introducing both secondary utility and precedence. It can be seen that the improved algorithm leads to remarkable declines in three metrics and a minor decline in the last metric.

<table>
<thead>
<tr>
<th>Performance Boost</th>
<th>Scheme A</th>
<th>Scheme B</th>
<th>Scheme C</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decline in Complete-coverage Failures</td>
<td>71.29%</td>
<td>35.29%</td>
<td>97.14%</td>
<td>67.91%</td>
</tr>
<tr>
<td>Decline in Uncovered Blocks per Handoff</td>
<td>70.38%</td>
<td>38.89%</td>
<td>99.19%</td>
<td>69.49%</td>
</tr>
<tr>
<td>Decline in Camera Adjustments per Covering Handoff</td>
<td>13.48%</td>
<td>32.07%</td>
<td>22.72%</td>
<td>22.76%</td>
</tr>
<tr>
<td>Decline in Direction change in Degrees per Covering Handoff</td>
<td>-0.67%</td>
<td>19.99%</td>
<td>-2.72%</td>
<td>5.53%</td>
</tr>
</tbody>
</table>

a. The numbers of failed handoffs are all zeros and hence excluded from this table.

3.8.4 Choice of the Number of Cameras

The principal reason we chose 8 cameras is that 8 was the minimum that did not incur any failed handoff during the simulation. Moreover, for values larger than 8, there is every likelihood that the system performance will be so good as to result in zero complete-coverage failures and uncovered blocks for both algorithms that we are comparing. In such a case, the difference in system performance will, more often than not, turn out to be undistinguishable, thus concealing the improvement of our algorithm over the original. To sum up, using 8 cameras is a rational choice for our purpose.

3.8.5 Limitations of Our Prototype

Unfortunately, the prototype we proposed suffers several limitations. Firstly, the prototype employs a rectangular region rather than an irregular polygon, which is the common case for floor plans in real life. Secondly, most floor plans contain vision-obstructing entities such as pillars and wall partitions, which were not considered in the prototype. Thirdly, the camera locations were uniformly distributed
along all edges of the rectangular region, which will hardly apply to an arbitrary polygonal region.

These limitations are to be tackled in our final solution in later chapters.
4 Overview of Our Final Solution

4.1 Goal

To overcome the drawbacks of [16] and [17], we integrate an enhanced version of the radial sweep algorithm in [17] into the framework of [16] so that the prototype in [16] can be adapted for more general applications [22].

4.2 Contributions

Chief contributions of our final solution are a) enhancing the PV-finding algorithm in [17] by taking into account corner cases that have gone neglected and allowing cameras to be placed on edges of holes and on vertices; b) working out a two-pass 0-1 programming process for finding a near-optimal set of camera locations which fits into the scenario of coordinated multi-resolution tracking over an arbitrary floor plan containing vision-obstructing obstacles; and c) contriving various performance-boosting techniques whose effectiveness has been verified with different experimental setups representative of common floor plans. An overview of our solution is depicted in Fig. 11. Definitions of concepts such as Camera Redundancy will be given in later chapters.

It is worth noting that consensus-based distributed tracking as used in [8] and [14] to estimate targets’ positions is not needed for simulation and hence is out of scope of this paper. It should also be noted that the target utilities designed in [8] and [14] were intended to improve tracking resolution and accuracy in captured images and are not applicable to simulation in this paper.
Fig. 11  Overview of our final solution.
5 Enhancing the Radial Sweep Algorithm for Finding Polygon of Visibility

5.1 Flaws in the Radial Sweep Algorithm

The algorithm shown in Fig. 3 and Fig. 4 is correct in general. However, it does have several flaws that may induce errors.

- Firstly, the algorithm may fail if the camera (i.e. pole of the polar coordinate system) happens to be collinear with an edge. In Fig. 12, we assume that SL is empty when vertex $j$ is being processed. Also, we assume that the camera lies on the same line as ${j, j+1}$. Consequently, vertex $j$ will be processed before vertex $j+1$, leading to an erroneous state where ${j-1, j}$ remains ActiveEdge when vertex $j+3$ is being processed. As the ray emitting from $p$ towards $j+3$ does not intersect ${j-1, j}$ at any point, the radial sweep algorithm halts at Line 21 of Fig. 4 and can by no means proceed. One plausible solution is to deem visible the pruned edge whose vertices coincide with $j$ (i.e. $i$ in Fig. 12) and $j+1$ (i.e. $i'$ in Fig. 12), respectively. In other words, edges such as ${i, i'}$ should not be pruned at Line 5 of Fig. 3. However, this solution will not work as expected owing to lack of definition of the intersection point of two overlapping lines. Restoring ${i, i'}$ in Fig. 12 would cause the algorithm to attempt to find, at Line 21 of Fig. 4, the intersection point of ${i, i'}$ and the ray emitting from $p$ towards vertex $j+1$ when vertex $j+1$ is being processed. Since the intersection is actually a segment (i.e. ${i, i'}$) rather than a single point, the algorithm would fail.
Fig. 12  The first situation in which the radial sweep algorithm [17] will fail, i.e. the camera is collinear with an edge.

- Secondly, consider the situation in Fig. 13. After the edge vertex \( j \) is processed, SL remains empty because HandleEventPoint does not insert \( \{j-1, j+2\} \) into SL. Consequently the prospective intersection \( i' \) will be missed and hence \( \{j+1, i'\} \) will not form an edge of PV, which is erroneous. Furthermore, the algorithm will not be able to proceed at edge vertex \( j+3 \) as the ray emitting from \( p \) towards \( j+3 \) does not intersect \( \{j, j+1\} \) at any point.

Fig. 13  The second situation in which the radial sweep algorithm [17] will fail.

- Thirdly, since the sampling of camera parameters is done using a grid in [17], it is entirely possible that the some camera sample (i.e. camera deployed at a candidate location) lies on an edge of a hole. Unfortunately the radial sweep algorithm may not work when the camera is situated on one edge of a hole and does not coincide with any vertex. This is primarily due to
lack of definition of splitting method to be used for horizontal edges. Another issue is the ambiguity in getting intersection of two overlapping lines, which is not addressed in [17]. The example shown in Fig. 14 may be a case in point. In Fig. 14, the first question we are confronted with is whether we should split \( \{j+4, j\} \). On the one hand, if we choose not to split, the problem we will encounter next is how to define the intersection of the ray emitting from \( p \) towards \( j+1 \) and \( \{j+4, j\} \). In fact the intersection can be defined as any point on \( \{p, j\} \). No matter which point we choose as the intersection, the radial sweep algorithm devised in [17] will definitely end up with an invalid polygon consisting of less than three vertices. On the other hand, if we choose to split \( \{j+4, j\} \) into, say, \( \{j+4, p'\} \) and \( \{p'', j\} \), the next question will be what polar-angle value we should assign to \( p' \) and \( p'' \) respectively. Possible choices are multiples of \( \pi \) (including 0). Without delicate treatment of polar-angle assignment, however, in all likelihood the radial sweep algorithm would yield erroneous results or even stop working half way. It is worth pointing out that pruning the edge which the pole is on as suggested by [17] will not work in this situation as the entire hole will be removed if pruning is carried out in that manner.

- Fourthly, the radial sweep algorithm may not work with a camera location coinciding with a vertex. Unlike the third situation exemplified in Fig. 14, no splitting is involved in this situation. However, careful handling is still required of a vertex-coincident pole. The tricky part in dealing with a vertex-coincident pole lies in the fact that the pole sits on two edges at the same time. As orientations of both edges must be taken into account, the two edge vertices at the pole should be assigned polar-angle values that represent the two edges respectively.
5.2 Modifying the Radial Sweep Algorithm

In order for the radial sweep algorithm to work as expected under any circumstances, we modified the algorithm by eliminating flaws and taking into account corner cases. It is worth mentioning that none of the modifications below will change the complexity of the entire PV-finding algorithm, namely, $O(E \log E)$, where $E$ is the number of edges [17].

![Diagram](image)

**Fig. 14** The third situation in which the radial sweep algorithm [17] will fail, i.e. the camera lies on one edge of a hole.

5.2.1 Amending TYPE1 in HandleEventPoint

We replace Line 9 to Line 18 of Fig. 4 with the pseudo-code snippet in Fig. 15 so as to solve the problem illustrated in Fig. 12.

5.2.2 Amending TYPE3 in HandleEventPoint

To solve the problem of aforementioned second situation in which the radial sweep algorithm fails, we can simply insert the pseudo-code snippet in Fig. 16 between Line 28 and Line 29 of Fig. 4. Accordingly, we need to introduce, for each edge, a property that signifies whether an edge belongs to a hole.
if $SL$ is not empty then
  while $SL$ is not empty do
    $e \leftarrow$ head ($SL$);
    if $\text{endvertexof } (e) > \text{CurrentVertex}$ then
      $k \leftarrow$ intersect ($e, \text{CurrentVertex}$);
      push ($k, PV$);
      $\text{ActiveEdge} \leftarrow e$;
      break;
    endif
    pop ($SL$);
  endw
else
  $\text{Pole} \leftarrow \text{Location of the camera};$
  $\text{NextVertex} \leftarrow \text{head} (Q)$;
  if $\text{Pole, CurrentVertex and NextVertex are collinear}$ then
    if $\text{ActiveEdge} \neq \text{incidentedgeof} (\text{NextVertex})$ then
      push ($\text{NextVertex}, PV$);
      $\text{ActiveEdge} = \text{incidentedgeof} (\text{NextVertex})$;
      pop ($Q$);
    endif
  endif
endif

Fig. 15  Pseudo-code snippet for solving the problem illustrated in Fig. 12.

if $e$ is an edge of a hole then
  if $\text{radiusof } (k) > \text{radiusof} (\text{CurrentVertex})$ then
    insert ($\text{ActiveEdge, SL}$);
  endif
endif

Fig. 16  Pseudo-code snippet for solving the problem illustrated in Fig. 13.

5.2.3  Restricting polar angles of two vertices of the same edge to one period of a turn by default

By default, the polar angles of two vertices of the same edge are restricted to one period of a turn, i.e. $[0, 2\pi)$. An exception is that we assign a polar angle of $2\pi$, for each edge with one edge vertex on the polar axis and the other below the polar axis, to whichever edge vertex lies on the polar axis.
5.2.4 Splitting the edge that crosses the pole

It is not stated in [17] if an edge crossing the pole, i.e. edge that contains the pole in between start and end vertices, needs to be split. By trial and error, we find it necessary and convenient to have approximately half of those edges split. The edges that do not need to be split are those crossing the pole from above the polar axis or from right to left horizontally. For edges that cross the pole in other ways (e.g. from left to right horizontally), a split is needed. Nevertheless, caution needs to be exercised when polar-angle values are assigned to the two coincident edge vertices generated by the split at the pole. Otherwise effectiveness of the whole algorithm may be comprised. In our solution, we assign two special values, i.e. 0 and $2\pi$, to the two vertices emerging from the split as appropriate. Specifically, the one that serves as the start vertex is assigned 0, which implies the first vertex in the radial sweep whereas the one that serves as the end vertex $2\pi$, implying one of last few vertices (along the polar axis) in the radial sweep. These measures are intended to ensure that the radial sweep always begins running from the start vertex at the pole and terminates along the pole axis, forming a complete turn.

5.2.5 Defining intersection of an edge and a ray that overlaps with that edge

When the pole resides on an edge, the radial sweep algorithm may attempt to get the intersection of an edge and a ray that emits from the pole and overlaps with that edge. (This takes place at Line 21 of Fig. 4.) An example of this behavior can be seen in Fig. 14. Since the intersection is not a mere point but a line segment in that example, special treatment is required. Our solution is as follows. Firstly, if the edge in question (i.e. `ActiveEdge` in `HandleEventPoint`) contains the edge vertex towards which the ray is emitting (i.e. `CurrentVertex` in `HandleEventPoint`) but does not contain the pole, the intersection is defined at that edge vertex (`CurrentVertex`). Secondly, if `ActiveEdge` contains the pole but does not contain `CurrentVertex`, the intersection is defined at the pole. Thirdly, if the edge in
question contains both the edge vertex and the pole, the intersection is defined, again, at
CurrentVertex.

5.2.6 Assigning special polar-angle values to edge vertices corresponding to a vertex-coincident pole

When the camera location (i.e. location of pole of the polar coordinate system) is chosen at a vertex of
a polygon, be it \( P_e \) or any of \( P_k \)'s, the treatment of polar-angle values of the two vertices
corresponding to the pole becomes a bit intractable. In our solution, we assign special polar-angle
values only to 1) pole-coincident edge vertices whose former incident edge and latter incident edge
are both ascending, and 2) pole-coincident edge vertices whose former incident edge is descending
and whose latter incident edge is ascending to the right of former incident edge. We will explain the
concepts of “former incident edge”, “latter incident edge”, “descending”, and “ascending” with the
following example. For example, given two consecutive edges \{j-1, j\} and \{j+1, j+2\}, where \( j \) is the
end vertex that coincides with the start vertex \( j+1 \) (as well as the pole), the former incident edge is
\{j-1, j\} and the latter incident edge \{j+1, j+2\}. When it comes to “ascending”, there are two possible
situations: a) the start vertex of the edge coincides with the pole, and the end vertex of the edge has a
polar-angle value that is less than \( \pi \) but greater than 0; b) the end vertex of the edge coincides with the
pole, and the start vertex of the edge has a polar-angle value that is less than \( 2\pi \) but greater than \( \pi \). The
case with “former incident edge” can only be b) whereas the case with “latter incident edge” can only
be a). Similar to what we do when splitting an edge that crosses the pole, we assign \( 2\pi \) to the end
vertex \( j \) and 0 to the start vertex \( j+1 \). For other edge vertices that do not conform to 1) or 2), we copy,
for each edge, the polar-angle value of the edge vertex not coincident with the pole to the edge vertex
coincident to the pole. For the example above, we simply copy the polar-angle value of \( j+2 \) to \( j+1 \),
and the polar-angle value of \( j-1 \) to \( j \).
The rationale behind assigning special polar-angle values to particular kinds of edge vertex is that the right-hand vision along polar axis is obstructed by edges formed by those kinds of edge vertex. Hence, we have to ensure the algorithm starts by following the edge that blocks vision to the right. For instance, \{j, j+2\} is the vision-blocking edge in Fig. 17. Note that in Fig. 17 former incident edge \{j+5, j+10\} is descending whereas latter incident edge \{j, j+2\} is ascending to the right of former incident edge, thus satisfying the special-value criteria. We assign 0 to the polar angle of j so that the algorithm will not erroneously start building PV by including the invisible \( j+1 \) first. By contrast, the latter incident edge \{j+5, j+6\} in Fig. 18 is ascending to the left of the former incident edge \{j+2, j+1\} and hence does not lead to right-hand vision obstruction.

![Diagram](image.png)

**Fig. 17** An example of vertex-coincident pole causing right-hand vision obstruction.

### 5.2.7 Amending initialization steps

A special flaw similar to the one exemplified in Fig. 13 will emerge if we allow the pole to sit on a horizontal edge of a hole. As can be seen in Fig. 19, a derivation of Fig. 14, the radial sweep algorithm will not include \( j+2 \) in the vertices of PV after adding \( j \) and \( j+1 \) as the first two vertices of...
PV. The reason is that SL does not include \{j+2, j+3\} when the algorithm processes \(j+1\). This anomaly can be eliminated by inserting the pseudo-code snippet in Fig. 20 between Line 12 and 13 of Fig. 3.

**Fig. 18** An example of vertex-coincident pole free of right-hand vision obstruction.

**Fig. 19** A special flaw similar to the one exemplified in Fig. 13.
5.2.8 Keeping any edge containing the pole when back edges are pruned

Edges collinear with the pole are supposed to be pruned in [17]. However, pruning an edge that contains the pole will remove all edges of the hole which the pole is on, thus removing the hole altogether. Hence it is indispensable that we keep all edges that contain the pole.

5.2.9 Summary

After all modifications above have been applied, the pseudo-code of FindVisibilityPolygon and HandleEventPoint are turned into Fig. 21 and Fig. 22, respectively.

---

Fig. 20 Pseudo-code snippet for solving the problem illustrated in Fig. 19.
Line  Algorithm FindVisibilityPolygon (ELC, x)
1     input: Edge list ELC and a point x inside the polygon.
2     output: PV, the list of vertices of the visibility polygon.
3     begin
4     Convert ELC to its polar coordinates, ELP;
5     Prune all backward facing edges;
6     Construct the vertex list Q;
7     Sort Q in lexicographically ascending order;
8     SL ← NULL;
9     PV ← NULL;
10    CurrentVertex ← pop(Q);
11    ActiveEdge ← incindentedgeof(CurrentVertex);
12    push(CurrentVertex, PV);
13    NextVertex ← pop(Q);
14    VertexAfterNext ← pop(Q);
15    if CurrentVertex and VertexAfterNext are start vertices then
16       if ActiveEdge = incindentedgeof(NextVertex) then
17          if ActiveEdge is an edge of a hole then
18             insert (incindentedgeof(VertexAfterNext), SL);
19          endif
20       endif
21    endif
22    push(VertexAfterNext, Q);
23    push(NextVertex, Q);
24    while Q ≠ NULL do
25       CurrentVertex ← pop(Q);
26       HandleEventPoint (CurrentVertex, ActiveEdge, PV, SL);
27       while The top three vertices in PV are collinear do
28          delete (The 2\textsuperscript{nd} vertex in PV);
29       endw
30    endw
31    end

Fig. 21  FindVisibilityPolygon of the enhanced radial sweep algorithm.
Line | Function HandleEventPoint (CurrentVertex, ActiveEdge, PV, SL)
--- | ---
1 | **input**: The current vertex CurrentVertex, the active edge ActiveEdge, the visibility polygon PV and the sorted list SL.
2 | **output**: The function operates on the input parameters.
3 | **begin**
4 | **TYPE1**
5 | if (CurrentVertex is an end vertex) AND (incidentedgeof (CurrentVertex) == ActiveEdge) then
6 | push (CurrentVertex, PV);
7 | if SL is not empty then
8 | while SL is not empty do
9 | e ← head (SL);
10 | if endvertexof (e) > CurrentVertex then
11 | k ← intersect (e, CurrentVertex);
12 | push (k, PV);
13 | ActiveEdge ← e;
14 | break;
15 | endif
16 | pop (SL);
17 | endw
18 | else
19 | Pole ← Location of the camera;
20 | NextVertex ← head (Q);
21 | if Pole, CurrentVertex and NextVertex are collinear then
22 | if ActiveEdge ≠ incidentedgeof (NextVertex) then
23 | push (NextVertex, PV);
24 | ActiveEdge = incidentedgeof (NextVertex);
25 | pop (Q);
26 | endif
27 | endif
28 | endif
29 | endif
30 | endif
31 | **end**

**Fig. 22** HandleEventPoint of the enhanced radial sweep algorithm.
if CurrentVertex is a start vertex then
    \( k \leftarrow \text{intersect} \ (\text{ActiveEdge}, \ \text{CurrentVertex}) \);
    \( e \leftarrow \text{incidentedgeof} \ (\text{CurrentVertex}) \);
    \text{TYPE2} \quad \text{if radiusof} \ (k) < \text{radiusof} \ (\text{CurrentVertex}) \text{ then}
    \quad \text{2.2} \quad \text{insert} \ (e, \ \text{SL});
    \quad \text{endif}
    \text{TYPE3} \quad \text{if radiusof} \ (k) \geq \text{radiusof} \ (\text{CurrentVertex}) \text{ then}
    \quad \text{push} \ (k, \ \text{PV});
    \quad \text{push} \ (\text{CurrentVertex}, \ \text{PV});
    \quad \text{if } e \text{ is an edge of a hole then}
    \quad \text{if radiusof} \ (k) > \text{radiusof} \ (\text{CurrentVertex}) \text{ then}
    \quad \quad \text{insert} \ (\text{ActiveEdge}, \ \text{SL});
    \quad \quad \text{endif}
    \quad \text{endif}
    \quad \text{ActiveEdge} \leftarrow e;
    \quad \text{endif}
    \quad \text{endif}
end

Fig. 22  HandleEventPoint of the enhanced radial sweep algorithm. (Cont’d)
6 Computation of Near-optimal Camera Quantity and Locations

In [17], the camera parameters (e.g. location) are sampled to reduce computational load during optimization. As a result, the optimality of the solution is only approximate. However, as the number of samples increase, the solution gets closer to the optimal one in the continuous domain.

For the problem in [16], sampling is only needed in the spatial domain so that near-optimal camera locations can be ascertained. As the cameras we are to use are not omnidirectional, we only sample along edges of $P_e$ and each $P_k$ rather than the interior of $P_e$ excluding all $P_k$'s. Of course, one may opt to deploy cameras only on edges of $P_e$. However, that is no more than a simplified case compared to the scenario we are considering. Therefore, in this paper we do not consider the case in which only edges of $P_e$ are sampled.

The maximum sampling frequency we choose for each edge is 4 samples per meter. Put another way, the distance between two cameras is at least 0.25 meter, which is adequate for common camera sizes. For example, dimensions of the consumer-level PTZ camera TRENDNET TV-IP600 are only 11x11.2x11.2cm [20]. Sampling frequencies higher than 4 samples per meter would be inappropriate as resulting quantity for camera samples might be too large. For example, the number of camera samples is 400 for 4 samples per meter and a 30 meters by 20 meters rectangular region. Considering there might be holes within the rectangular region, it would be safer to assume twice the quantity for camera samples, i.e. 800. If we would choose a sampling frequency higher than 4 camera samples per meter, the final quantity would exceed 1000, which would be a huge computational burden for 0-1 programming. Therefore 4 samples per meter is a rational choice. It is worth noting that 4 samples per meter is only a maximum value, and it is usually not reached as the length of an edge is not
necessarily a multiple of 0.25 meter. As is the same with [17], camera samples are uniformly distributed within one edge.

After sampling is done, we compute the feasible regions [17] for all camera samples and use results of the computation as input to 0-1 programming for optimization. Unlike [17], the requirements of the problem in [16] do not entail HAOV constraint, which will be computed online after the offline computation of near-optimal camera quantity and locations. The resolution constraint is still needed as we have to ensure proper functioning of zoomed tracking.

The 0-1 programming problem in [17] is principally defined as

minimize

\[ c^T x \]  

subject to

\[ Ax \geq b \]  

and

\[ x_i \in \{0,1\} \] .

For the problem in [16], the vector \( c \) can comprise all 1’s since all locations of cameras have identical costs. \( A \) is an \( m \) by \( n \) coefficient matrix where each row corresponds to an imaginary block on the ground whereas each column corresponds to a camera sample. It is worth noting that the shape of a block is not necessarily a square on account that it may be “chopped” by one or more edges of \( P_\varepsilon \) or any \( P_k \). We refer to the portion of a block that remains in the interior of the \( P_\varepsilon \) but outside any \( P_k \) as the Valid Area (VA) of that block. Each element of vector \( b \) indicates the redundancy in camera coverage and is equal to the number of cameras that are able to cover the block in question.
From this point on, we will refer to an element of vector $\mathbf{b}$ as *Camera Redundancy* (CR). In our scenario, there are two requirements that need to be met. The first one, called *Handoff Requirement*, is to guarantee successful handoffs as defined in [16]. This requirement can be satisfied by assigning a value of 2 to each CR. The second requirement, called *Coverage Requirement*, is to decrease the number of coverage failures to the minimum level, preferably zero. Admittedly, 0-1 programming cannot serve as an immediate solution to ensure the elimination of complete coverage failures rooted in the non-omnidirectional property of cameras, since it is assumed in the original 0-1 programming problem that all cameras are omnidirectional [17]. None the less, it can be used, along with larger-than-one values assigned to CR’s, to ensure that a block can be covered, at a high probability, by at least one of a few non-omnidirectional cameras capable of covering that block. The greater the values of CR’s, the higher the probability of blocks being covered. Nevertheless, raising CR will inevitably result in an increase in the number of cameras to be deployed. Thus, a tradeoff is needed. It is also worth noting that the coefficient matrix $\mathbf{A}$ needs to be generated separately for each of the two requirements above. That is because the blocks that a camera sample is able to cover in the zoom mode are different from those in the normal mode. Usually the former ones form a subset of the latter ones as the zoom range is a subset of the normal range. For the Handoff Requirement, each column of coefficient matrix $\mathbf{A}$ is generated by the *circumferential* coverage of each camera sample operating in the zoom mode. By “circumferential”, we mean that each camera sample is treated as though it were omnidirectional. Similarly, for the Coverage Requirement, each column of coefficient matrix $\mathbf{A}$ is generated by the circumferential coverage of each camera sample operating in the normal mode. Caution needs to be exercised regarding blocks that fall into circumferential coverage. In fact not all blocks within circumferential coverage can be regarded as entirely visible to the camera sample in
question. Fig. 23 shows a block (marked in green) that is not entirely visible to a camera. The green block is not covered in its entirety due to limited HAOV of the camera, which is marked in gray. Suppose a target under surveillance has a radius of $r$. In our scenario, $r$ equates half of the block diagonal. Also suppose that the minimum distance between the center of block and the camera that implies complete visibility of the block is $m \times r$, where $m$ is a non-negative integer that we call Radius Multiplier (RM). In other words, we will regard a block as visible only when the center of block is more than $m \times r$ away from the camera. By default RM has the value 0. Although it appears reasonable to assume a non-zero RM, setting RM to a non-zero value may render Inequality (4) insolvable. It is worth mentioning that the resultant insolvability is often associated with blocks with concave VA. Fig. 24 depicts a case in point. The orange circle outlines the range in which the 0-1 programming problem is not solvable given a Radius Multiplier of 2. The blue rectangle is a vision-obstructing obstacle. One pink segment and two pink dots form the portion of edges that can see the entire VA of the green block. As the pink portion is enclosed in the orange circle, solvability can never be achieved. To resolve this issue, we force RM to be zero for any row of the coefficient matrix $A$ whose number of 1-valued elements is less than the corresponding CR, thus ruling out insolvability caused by non-zero RM. Admittedly, this is not an optimal solution, as deeming an invisible block visible might increase the chances of complete coverage failures. None the less, this compromise is acceptable as alternatives such as decreasing block dimensions are much more computationally expensive. As an effort to minimize the side effect brought about by this comprise, we set RM to a small value (i.e. 1) by default in our experiments. An alternative to the approach of circumferential coverage is to take into account camera direction in addition to camera location when spatial sampling is performed, thus eliminating uncertainty of whether a block can be covered.
However, this alternative suffers two flaws. For one thing, the number of columns in coefficient matrix $A$ may be hundreds of times as large as the original, as denser sampling is needed owing to the numerous angles within a turn (e.g. 360 degrees). This would result in much heavier burden on the computing device. For another, the solution of the 0-1 programming problem is static with respect to camera direction. Since cameras in our scenario might need to adjust to new directions during handoffs, the solution would be totally useless.

One problem associated with 0-1 programming is that Inequality (4) may not be solvable, especially when CR’s have large values (i.e. excessive demand for coverage redundancy), camera sampling frequency is low (i.e. insufficient location candidates), or blocks are oversized (i.e. probable failure to cover a block in its entirety). Accordingly, one of the following three measures can be taken to ensure solvability of Inequality (4): a) decreasing CR, b) increasing cameras sampling frequency, and c) decreasing dimensions of blocks.
The second problem is that assigning an identical value to each CR may give rise to Camera Clustering, a phenomenon in which a few cameras are deployed within a very small area intending to cover a block with concave VA. An example of this phenomenon is illustrated in Fig. 25, that is, setting CR to 4 for the green block leads to a situation in which four cameras (marked in red) cluster together along the same edge. As can be observed in Fig. 25, VA of the (green) block is concave due to the (blue) vision-obstructing entity and can only be seen in its entirety by a small portion of the (black) edge on the left. The four camera locations are chosen mainly because they are among the few camera samples within that small portion of edge. Since the four cameras cluster together and hence share most of the circumferential coverage, their usefulness during a handoff tends to be minimized. In addition, having numerous cameras deployed for the sake of merely one block is inclined to defeat our aim of maximizing resource utilization. As a remedy to this sort of wasted redundancy, we assign the fixed value 3 to any CR that corresponds to a block with concave VA. The value 3 turned out to work quite well in our experiments. Nevertheless it is worth pointing out that assigning an identical value to each CR is not the sole cause of Camera Clustering. In fact, a large RM may also cause the eligible camera samples to cluster within a small portion of an edge. For instance, Fig. 24 is essentially an extreme case in which a large RM reduces the length of aforementioned small portion of edge to zero. If we set RM to a value that is slightly less than 2, e.g. 1.5, there is every likelihood that camera clustering will occur on the right half of the pink segment. Hence, assigning a fixed value to any CR that corresponds to a block with concave VA can only rule out the case in which Camera Clustering is solely triggered by the intention to cover blocks with concave VA.

The third problem is similar to the second one but is even trickier. Since $P_e$ and $P_k$’s can be polygons of any shape, it is entirely possible that VA of a block is divided into disjoint parts by two or
more edges of $P_e$ and $P_k$’s. As can be seen in Fig. 26, VA of the (green) block consists of three parts, which are invisible as a whole from any single point outside the (blue) vision-obstructing entity. There are three approaches to cope with this situation. The first approach is to further divide VA of the block into three sub-VA’s, each corresponding to one of the three separate parts. However, this approach is against our objective of approximating optimality using large units (i.e. blocks). Besides, it will considerably complicate block handling. The second approach is to decrease block dimensions to the extent that every block has monolithic convex or concave VA. This approach will not work either on account that the computational cost will increase sharply due to rapid rise in quantity of blocks caused by shrink of block dimensions. The third approach is to ignore the situation altogether as if the block in question did not exist. Although the third approach seems to be unsafe at first glance, our experimental results have demonstrated that it has worked decently.

![Fig. 25 An example of Camera Clustering.](image)

![Fig. 26 An example of VA consisting of three disjoint parts.](image)

Similar to [17], in which the branch-and-bound algorithm [23] was adopted, we use the branch-and-cut algorithm [23] to solve the 0-1 programming problem. Specifically, GLPK (GNU Linear Programming Kit) [24] has been employed as the computing device.
7 System Design

The system design in this paper extends the one in [16] in two aspects. On the one hand, the enhanced radial sweep algorithm allows cameras to be placed at any allowable position, even on edges and vertices. On the other hand, the proposed two-pass 0-1 programming process for finding a near-optimal set of camera locations is tailored to the needs of coordinated multi-resolution tracking over an arbitrary floor plan in the presence of vision-obstructing obstacles.

7.1 Assumptions

As is the same with [16], before building a model for system simulation, we need to make reasonable assumptions regarding the scenario and designate proper values to parameters. As the scenario in our final solution differs a lot from the one in [16], some assumptions in [16] such as those related to rectangular shape do not apply any more and new assumptions like spatial sampling frequencies of camera location need to be added.

- There should be sufficient cameras so that complete-coverage can be achieved within a very short period of time for the majority of handoffs. We set dimensions of the bounding rectangle that completely encloses the polygon under surveillance to 30 meters by 20 meters.

- The block size is 1 meter by 1 meter.

- The capabilities of the cameras in use should be at least equivalent to those of consumer-level PTZ cameras such as TRENDNET TV-IP600 (1/4" color CMOS sensor with a height of 2.4mm, 4.57mm focal length, 320 x 240 resolution, 4x zoom, and 46 degrees HAOV) [20]. Unlike [16], no additional constraint needs to be explicitly imposed on the maximum distance in the zoom mode to ensure success of handoffs because successful handoffs are implied by
fulfillment of the Handoff Requirement during 0-1 programming. However, if Inequality (4) is not solvable no matter how intrinsic parameters (e.g. Radius Multiplier) are tweaked, chances are that cameras with more advanced specifications (e.g. larger zoom range) must be used in order to fulfill the surveillance task.

- The network cameras operate fast enough to track any moving target in real time.
- The time a camera spends computing its optimal direction is much less than that it takes for any other camera to change direction.
- The cameras are connected via a fast local area network such as 100Mbps Ethernet.
- We do not take into consideration tilting of a camera.

7.2 Secondary Utility and Precedence Given to Border-covering Camera Directions

In [16], the boost in system performance is brought about by secondary utility and precedence given to border-covering camera directions. Secondary utility improves system performance during handoffs by taking into account the overlapping coverage of a camera that is shared with another camera. Since secondary utility is irrelevant to floor plan, it can be applied to an arbitrary polygonal region under surveillance without any modification. But precedence given to border-covering camera directions no longer works as vision of cameras may be obstructed by “holes” of arbitrary polygonal shapes. Hence the probability of an interior block being covered is not likely to differ drastically from that of a border one. This argument will be verified against experimental results which will be given later. Unless otherwise stated, in this paper no camera direction takes precedence over others, and border blocks and interior blocks are treated equally.
8 Experimental Results

8.1 Evaluation Metrics

We use, for evaluation purposes, the last four metrics of the five proposed in [16], namely, the number of Complete-coverage Failures, the number of Uncovered Blocks per Handoff, the number of times of Camera Adjustment per Covering Handoff, and Change of Directions in Degrees per Covering Handoff. The first of the original five metrics, that is, the number of Failed Handoffs is no longer needed as successful handoffs are guaranteed by fulfillment of the Handoff Requirement, which amounts to saying that the number of Failed Handoffs is always zero. The four metrics in use as listed above are in descending order with respect to importance.

8.2 Simulation Setup

In our simulation, the camera quantity and locations are determined by a two-pass optimization process. The first pass is intended to meet the Handoff Requirement whereas the second pass the Coverage Requirement. Output of the first pass is fed as input to the second pass. The second pass assumes that cameras output by the first pass already exist. In the first pass, all CR’s have the value of 2. In the second pass, CR’s corresponding to blocks with concave VA are assigned a fixed value of 3. Other CR’s are assigned the value of 3 by default but this value may be increased as per experimental needs. As is the same with [16], each simulation run consists of 1000 handoffs with randomly chosen moving targets of interest, and HAOV is 46 degree for all cameras.
8.3 Four Schemes of Floor Plan

Simulation is run over each of the following four schemes of floor plan, as is shown in Fig. 27. Gray area signifies coverage of one or more cameras. The darker a region is, the more cameras covering it. Blue boxes represent moving targets and the red box represents the moving target of interest. Yellow boxes highlight uncovered blocks. The camera that is tracking the moving target of interest in the zoom mode is drawn as a (partial) solid red circle, with a light-red line pointing from itself to the moving target of interest and a translucent red overlay indicating its zoom range. Cameras operating in the normal mode are drawn as (partial) hollow red circles. Solid blue polygons represent vision-obstructing entities. The blue boxes (i.e. moving targets) are present only for demonstration purposes and can be ignored.

8.3.1 Scheme A

Scheme A is the simplest of all four schemes. In Scheme A, \(P_e\) is a rectangle and the only \(P_k\) (i.e. \(P_1\)) is a rectangle placed at the center of \(P_e\). The number of blocks in this scheme is 560. A real life example of this scheme is a museum hall exhibiting skeleton of a dinosaur.

8.3.2 Scheme B

\(P_e\) of Scheme B is a rectilinear polygon. Scheme B has two holes, namely \(P_1\) and \(P_2\). \(P_1\) is a bar-like rectangle placed on the left in Fig. 27(b) whereas \(P_2\) resembles a golf club in shape and is placed on the right in Fig. 27(b). The number of blocks in this scheme is 494. This floor plan, in which all edges meet at right angles, is characteristic of an art gallery.

8.3.3 Scheme C

In Scheme C there are two triangles, i.e. \(P_e\) and \(P_1\). \(P_e\) is an isosceles triangle whereas \(P_1\) is a scalene triangle placed near the center of \(P_e\). The number of blocks in this scheme is 316. This scheme may be applied to a garden with a pavilion in the middle.
Fig. 27 Four schemes of floor plan over which simulation was run.

8.3.4 Scheme D

This scheme is the most complicated of the four. $P_e$ in this scheme is a concave polygon with a lot of edges, short and long. Inside $P_e$ there are two polygonal holes, namely, $P_1$ and $P_2$. $P_1$ is a concave polygon with a sharp spike on the left whereas $P_2$ is a convex polygon which is significantly larger than $P_1$. The number of blocks in this scheme is 453. Although this scheme is rather unlikely as an indoor floor plan, an outdoor paintball field may adopt such a scheme.

8.4 Benefit of Introducing Secondary Utility

The impact of secondary utility is evaluated in Fig. 28. (All CR’s are assigned the identical value 3. The number of cameras in use is 8 for Scheme A, 21 for Scheme B, 12 for Scheme C, and 24 for Scheme D.) As can be observed in Fig. 28, secondary utility has brought about significant declines in all four metrics, which implies remarkable performance boost. Average declines in percentage for all four metrics, which were derived from data in Fig. 28, are shown in Table 2. These results agree with
those in [16], confirming that secondary utility is beneficial not only to surveillance of a rectangular region but also to surveillance of a polygonal region with polygonal holes in it.

Fig. 28  Benefit of introducing secondary utility in terms of four evaluation metrics.
Table 2  Average declines in percentage thanks to secondary utility for the four evaluation metrics.

<table>
<thead>
<tr>
<th>Performance Boost</th>
<th>Average Decline in Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decline in Complete-coverage Failures</td>
<td>58.3%</td>
</tr>
<tr>
<td>Decline in Uncovered Blocks per Handoff</td>
<td>72.5%</td>
</tr>
<tr>
<td>Decline in Camera Adjustments per Covering Handoff</td>
<td>29.2%</td>
</tr>
<tr>
<td>Decline in Direction change in Degrees per Covering Handoff</td>
<td>45.6%</td>
</tr>
</tbody>
</table>

8.5 Benefit of a Setting Radius Multiplier to 1

Fig. 29 shows that the 1-valued RM enjoys advantage over 0-valued RM in terms of overall performance. (All CR’s are assigned the identical value 3. Secondary Utility is in use. The quantity of cameras yielded by 0-valued RM is the same as that yielded by 1-valued RM for all schemes except for Scheme A. In Scheme A, 0-valued RM yields a set of 10 cameras whereas 1-valued RM yields a set of 8 cameras.) The only anomaly was observed at the Uncovered Blocks per Handoff for Scheme A, where 0-valued RM appears to have outperformed 1-valued RM. However, given that the solution gained from 0-valued RM uses 10 cameras whereas the solution gained from 1-valued RM only uses 8, the comparison for Scheme A is simply unfair. In fact, when we tried setting CR to 3 for blocks with concave VA and 5 for other blocks, the solution gained from 1-valued RM used 10 cameras as well, but yielded a value around 0.1 for Uncovered Blocks per Handoff. The value 0.1 is far better than the result obtained from 1-valued RM for Scheme A, which is approximately 3.8. (Incidentally, when we tried setting CR to 3 for blocks with concave VA and 5 for other blocks, the solution gained from 0-valued RM used 14 cameras, which again exceeds that of 1-valued RM and cannot be used for fair comparison.) Therefore, we may safely arrive at the conclusion that 1-valued RM is prone to yield more favorable results than 0-valued RM.
Fig. 29  Improvement in terms of four evaluation metrics when RM is set to 1 instead of 0.

8.6 Justification of Assigning the Fixed Value 3 to CR for Blocks with Concave VA

Among all four schemes, Scheme D is the only one that has a sharp spike (on $P_1$), which is prone to Camera Clustering. Hence we will only experiment with Scheme D with a view to verifying the
benefit of assigning the fixed value 3 to CR for blocks with concave VA. Two cases are depicted in Fig. 30. In Case 1, CR for all blocks is set to 4 for all blocks whereas in Case 2, CR is set to 3 for blocks with concave VA and 5 for blocks with non-concave VA. (Secondary utility is in effect for both cases. RM is 1 in each case. Case 1 and Case 2 have the same quantity of cameras in use, which is 31. Recall that the quantity of cameras is determined by a two-pass 0-1 programming process.) It can be seen that Camera Clustering is alleviated on the leftmost edge in Fig. 30(b) as compared with the case in Fig. 30(a). Unlike Case 1, Case 2 has no Complete-coverage Failures according to Table 3. As a result, Uncovered Blocks per Handoff is also zero for Case 2. Admittedly, Case 1 outperforms Case 2 in terms of Camera Adjustments per Covering Handoff and Direction change in Degrees per Covering Handoff. None the less, since Complete-coverage Failures and Uncovered Blocks per Handoff outweigh the other two metrics in importance, the results in Table 3 do bolster the argument that assigning the fixed value 3 to CR for Blocks with Concave VA contributes to improvement in overall performance.

Fig. 30  Results for Scheme D obtained when CR’s take identical and different values, respectively.
Table 3  Results for Scheme D obtained when CR’s take identical and different values, respectively, as illustrated in Fig. 30.

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Case 1: CR=4</th>
<th>Case 2: CR=3 for blocks with concave VA; CR=5 otherwise.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complete-coverage Failures</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>Uncovered Blocks per Handoff</td>
<td>0.005</td>
<td>0</td>
</tr>
<tr>
<td>Camera Adjustments per Covering Handoff</td>
<td>0.650</td>
<td>1.342</td>
</tr>
<tr>
<td>Direction change in Degrees per Covering Handoff</td>
<td>39.6</td>
<td>66.6</td>
</tr>
</tbody>
</table>

a. Secondary utility is in effect. RM is 1. In either case, 31 cameras are in use.

8.7 Ineffectiveness of Precedence Given to Border-covering Camera Directions

In [16], the precedence of border-covering camera directions has proved useful in improving performance of the camera network monitoring a rectangular region. However, this is not the case with a polygonal region with polygonal holes in it. For example, when we give precedence to border-covering camera directions in Scheme A, we obtain the results as shown in Table 4. It is obvious that the performance is actually degraded with respect to all four metrics when precedence is in effect. These results justify abandoning precedence in surveillance of a polygonal region with polygonal holes in it.

8.8 Improving System Performance by Increasing CR

One may have noticed, in the light of earlier experimental results, that the system performance in Scheme A is rather poor. This is because the default value for CR’s (i.e. 3) has yielded a set of cameras that is insufficient in quantity. As we stated earlier, the probability of blocks being covered
rises as CR values increase. As can be observed in Table 5, system performance improves monotonously as the values of CR’s for blocks with non-concave VA are increased. (Recall that CR’s for blocks with concave VA are fixed at 3.) Nevertheless it can also be seen that the number of cameras in use also increases, which, again, conforms to what we stated earlier. The user is left with the freedom to tweak CR values as per his or her budget, that is, available funds for purchasing cameras.

Table 4  Results for Scheme A obtained before and after precedence is given to border-covering camera directions.

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Without precedence</th>
<th>With precedence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complete-coverage Failures</td>
<td>330</td>
<td>375</td>
</tr>
<tr>
<td>Uncovered Blocks per Handoff</td>
<td>4.83</td>
<td>4.91</td>
</tr>
<tr>
<td>Camera Adjustments per Covering Handoff</td>
<td>2.46</td>
<td>2.63</td>
</tr>
<tr>
<td>Direction change in Degrees per Covering Handoff</td>
<td>137.56</td>
<td>182.02</td>
</tr>
</tbody>
</table>

a. Secondary utility is in effect. RM is 1. All CR’s are assigned the identical value 3.

Table 5  Results for Scheme A obtained when CR for blocks with non-concave VA is set to 3, 4 and 5, respectively.

<table>
<thead>
<tr>
<th>Metrics</th>
<th>CRN=3 Using 8 cameras</th>
<th>CRN=4 Using 10 cameras</th>
<th>CRN=5 Using 12 cameras</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complete-coverage Failures</td>
<td>330</td>
<td>14</td>
<td>0</td>
</tr>
<tr>
<td>Uncovered Blocks per Handoff</td>
<td>4.83</td>
<td>0.09</td>
<td>0</td>
</tr>
<tr>
<td>Camera Adjustments per Covering Handoff</td>
<td>2.46</td>
<td>1.96</td>
<td>0.51</td>
</tr>
<tr>
<td>Direction change in Degrees per Covering Handoff</td>
<td>141.61</td>
<td>118.05</td>
<td>31.93</td>
</tr>
</tbody>
</table>

a. Secondary utility is in effect. RM is 1. CR’s for blocks with concave VA is fixed at 3.

b. CRN stands for Camera Redundancy for blocks with Non-concave VA.
9 Conclusion and Future Work

We implemented a simulation system of camera network for coordinated multi-resolution surveillance by incorporating a combination of an enhanced radial sweep algorithm and a two-pass 0-1 programming process into a decentralized game-theoretic framework proposed in [1,8,14]. In the first place, we put forward a prototype for preliminary evaluation and gained encouraging results in terms of numerous evaluation metrics. Next, we provided remedies to flaws in the radial sweep algorithm given by [17]. In the meantime, we adapted the radial sweep algorithm to cases in which a camera can be placed at any point of edges of the polygonal region under surveillance and edges of polygonal holes inside the polygonal region. Then we described the two-pass 0-1 programming process used to obtain a near-optimal set of camera locations. Concepts such as Camera Redundancy, Radius Multiplier and Valid Area are proposed to facilitate formulation of our performance-boosting techniques. We also took advantage of the secondary utility as an enhancing technique. Later we reviewed, on a concise basis, simulation assumptions and evaluation metrics as defined in [1] and [16]. In our experiments, we ran a series of simulation over four schemes which are representative of a variety of floor plans. The effectiveness of our performance-boosting techniques was verified against experimental results.

In this paper, we considered pure-strategy Nash equilibrium, where each camera has complete information regarding characteristics of the other cameras. As future work we shall explore the possibility of utilizing Bayesian Nash equilibrium to enhance system performance. We shall also try to find a way to tackle dynamic occlusion (i.e. moving objects occluding each other) that best fits our coordinated surveillance scenario.
Bibliography


Appendix

To make this paper self-contained we briefly review the definition of Nash equilibrium. Further details regarding Nash equilibrium can be found in [25] and [15].

Nash equilibrium is a situation in which no player in a game involving two or more players can increase its utility by changing its strategy alone, given the strategies that the other players are using. More formally, Nash equilibrium corresponds to a strategy profile in which each strategy is a best response to the other strategies in the strategy profile, where the best response is defined as the strategy that yields the most favorable result for a player. The concept of Nash equilibrium has been extended to Bayesian Nash equilibrium where the information each player has regarding characteristics of the other players is incomplete.