

Automated Camera Layout to Satisfy Task-Specific and Floorplan-Specific Coverage Requirements

Ugur Murat Erdem, Stan Sclaroff

Computer Science Department, Boston University

Abstract

In many multi-camera vision systems the effect of camera locations on the task-specific quality of service is ignored. Researchers in Computational Geometry have proposed elegant solutions for some sensor location problem classes. Unfortunately, these solutions utilize unrealistic assumptions about the cameras' capabilities that make these algorithms unsuitable for many real-world computer vision applications: unlimited field of view, infinite depth of field, and/or infinite servo precision and speed. In this paper, the general camera placement problem is first defined with assumptions that are more consistent with the capabilities of real-world cameras. The region to be observed by cameras may be volumetric, static or dynamic, and may include holes that are caused, for instance, by columns or furniture in a room that can occlude potential camera views. A subclass of this general problem can be formulated in terms of planar regions that are typical of building floorplans. Given a floorplan to be observed, the problem is then to efficiently compute a camera layout such that certain task-specific constraints are met. A solution to this problem is obtained via binary optimization over a discrete problem space. In experiments the performance of the resulting system is demonstrated with different real floorplans.

Key words: camera placement, sensor networks, visibility, best view.

PACS:

Email addresses: merdem@bu.edu (Ugur Murat Erdem), sclaroff@cs.bu.edu (Stan Sclaroff).

1 Introduction

Computer vision in video sensor networks has become a popular research topic in recent years. Decreasing cost of associated hardware and increasing practical need for such systems are among the reasons attracting more and more researchers to focus in this area. Different visual tasks have different requirements. For an intruder detection system, complete visual coverage of the region of interest may be needed. For a multi-view reconstruction task, it may be desirable to have a minimum number of video sensors with some given angular separation. For some systems the aggregate video sensor network, depending on the specific system design and architecture, should be made fault-tolerant to camera “drop out” – for instance, the occasional failures of cameras, temporarily obstructed camera views, etc. As in cellular telephone networks, the aim is to have as much coverage as possible within a predefined region, with an acceptable level of quality-of-service. Similarly in video sensor networks, the layout of video sensors should assure a minimum level of image quality needed to satisfy certain task-specific requirements – for instance, sufficient image resolution, depth of field, servo speed for pan-tilt-zoom cameras, etc.

It is important that the camera layout satisfy task-specific requirements since this will improve a vision algorithm’s performance at runtime; yet this remains a relatively underdeveloped area of video sensor networks. The lack of interest may be linked to researchers’ tendency to focus mainly on the design of new algorithms and a clear lack of standard test beds to compare them. For instance, a well known face recognition algorithm might in theory (and perhaps even in practice) perform “well” in a single camera setup, but perform “poorly” in a multi-camera setting. It would not be fair to blame solely the algorithm for its poor performance. The real reason for poor performance might be that the camera parameters and layout do not satisfy basic algorithm requirements. Furthermore the lack of a standard testbed to evaluate the system’s performance with different camera network setups may render it difficult to understand their effect. One of the main driving forces for this work is to study and improve the effect of the off-line camera placement on the on-line machine vision system performance. The rule of thumb is, no matter how good and efficient a vision algorithm may be, it will perform miserably if there are poor decisions made in the up-front task of choosing the cameras, setting their parameters, and designing their layout in the region of interest.

2 Related Work

In Computational Geometry, extensive progress has been made in solving optimal guard location problems for a polygonal area, *e.g.*, *The Art Gallery Problem* (AGP) and its variants, where the task is to determine a minimal number of guards and their static positions, such that all points in a polygon are observed [1–3]. Even though efficient algorithms exist giving a lower bound for AGPs with simple polygons [1], the exact solution is proven to be NP-Hard.

A variant of the AGP is known as *Watchmen Tours* where guards are allowed to move inside the polygon [4–6]. The objective is to find an optimal number and route for guards guaranteeing the detection of some intruder with an unknown initial position and unlimited speed. Suzuki, *et al.* introduce another variant of watchman problem termed *boundary search* where guards are allowed to move only along the boundary of the polygon [7]. In a similar vein, *Floodlight Illumination Problems* deal with the illumination of planar regions by light sources [8,9].

Current solutions to the AGP and its variants employ unrealistic assumptions about the cameras’ capabilities that make these algorithms unsuitable for most real-world computer vision applications: unlimited field of view, infinite depth of field, and/or infinite servo precision and speed. One main aim of our work is to bridge the gap between the highly theoretical, well-established computational geometry and more realistic requirements of computer vision with real video cameras.

A survey of sensor planning methods that employ more realistic assumptions is given in [10]. In work published contemporaneously with that presented here [11], a probabilistic sensor planning framework with a “visibility analysis” is proposed which evaluates the visibility of potential subjects over possible camera configurations. While their problem definition is very similar to ours in scope, their approach differs in a number of ways. They model environments with a given object density, we model environments with a given set of coverage and cost constraints. They use a local optimization method to solve a highly non-linear constrained optimization problem, as opposed to the global optimization employed here to solve a linear optimization problem over binary variables. Any solution produced by our approach is guaranteed to be globally optimal which is not necessarily true for [11]. Moreover, our technique can handle arbitrary polygonal shapes.

There is also related work with robotic motion control for video surveillance; *e.g.*, [12] where gradient descent is employed to compute optimal locations for mobile sensing networks given some utility function over a convex polygon.

Task-based vision and camera control have a long history in computer vision

[13]. One interesting problem is determining the *next best view* – finding the next optimal camera parameter setting given the acquired visual data history for the scene under exploration [14–16]. A number of active vision methods have also been proposed for surveillance applications. For instance [17,18] use a peripheral sensor to detect the position of a moving object(s) and drive a foveal sensor to gather detailed images of the target(s). Mikic, *et al.* [19] build a camera network for an intelligent room with static and active cameras. They also employ orientation-based active camera selection criteria. In [20] tracking of humans across cameras is accomplished by selecting the next camera that gives the maximum tracking confidence. An interesting application for best view selection can be found in [21] where a central algorithm chooses and combines the best views from a camera network using cinematographic rules. In [22] the task becomes estimating the external parameters of individual cameras in a camera network with non-overlapping field of views. It is important to emphasize that in none of these systems is consideration given to the *off-line* selection and placement of the cameras to improve the *on-line* system performance.

3 Problem Definition

In this paper we pose the problem of optimal camera placement for a given region and vision task. We focus on the camera placement problem, where the goal is to determine optimal positioning and number of cameras for a region to be observed, given a set of task-specific constraints, and a set of possible cameras to use in the layout. This camera placement takes place *off-line*, for cameras that will be mounted on surfaces in an area of interest to support the task-specific requirements of *on-line* computer vision systems. In the most general (and most challenging) case, the region to be observed by cameras may be an arbitrary volumetric shape. It may be an open space or a delimited environment, or a blend of both, *i.e.*, outdoors *vs.* indoors. The region may include holes that are caused, for instance, by columns and trees, or furniture in a room that can obstruct potential camera views. It may contain an arbitrary number of static and/or dynamic objects. Furthermore, the region itself may change in time, *i.e.*, furniture or walls may be added, removed, or moved in a floor plan. Finally, one can choose from an arsenal of different types of cameras that could be used in satisfying the requirements for the specified video sensing task(s).

3.1 Cameras

For the sake of completeness we first outline some optical camera parameter definitions and their main limitations. We then describe three major video camera types employed in surveillance and compare them with respect to their optical parameters. Three crucial parameters for the current work are:

- **Field of View (FoV):** The maximum volume visible from a camera. The FoV is determined by the apex angles (azimuth and latitude) of the visible pyramidal region emanating from the optical center of the camera. This pyramid is also known as the *viewing frustum*, and can be skewed by oblique projection.
- **Spatial Resolution:** Spatial resolution of a camera is defined as the ratio between the total number of pixels on its imaging element excited by the projection of a real world object and the object's size. Higher spatial resolution captures more details and produces sharper images.
- **Depth of Field (DoF):** Depth of field is the amount of distance between the nearest and farthest objects that appear in acceptably sharp focus in an image. It is determined by the f-stop or f-number which shows the relation in between the aperture diameter and the focal length of the lens. For instance, an f-stop of f/16 shows an aperture diameter that is one-sixteenth of the focal length.

There are many types of video cameras available. They differ in the sensor element type, lens type, servo capabilities, etc. The following three are frequently used in computer vision research and applications:¹

- **Fixed Perspective Camera:** Once mounted in place, these cameras have a fixed position, orientation, and focal length.
- **PTZ (Pan-Tilt-Zoom) Camera:** These cameras can rotate around their horizontal (Tilt) and vertical (Pan) axis using remotely controlled servos. Some also have an adjustable focal length (Zoom) limited by some range. They are mounted in a fixed position in the environment.
- **Omnidirectional Camera:** These cameras have 2π horizontal FoV angle, as opposed to a pyramidal one. Despite their total FoV range, they may suffer from lens aberration effects due to small focal length and/or convex mirrors used in the setup [18,17].

¹ One additional camera type not included in this list is a mobile camera (mounted on a moving platform or robot). This type is not included because we focus on the *off-line* sensor layout problem.

Each of the mentioned cameras is defined by a set of parameters. Let the vector π_i represent the parameters that define camera i . It will contain two sub-vectors: π_i^I , the intrinsic parameters like focal length, and π_i^E , the extrinsic parameters that define the location and orientation of the camera with respect to the world coordinate system.

Furthermore we will refer to the layout planning phase of the vision system as *off-line* and to the runtime phase as *on-line*. For all three camera types the location parameters are variable during the *off-line* phase, i.e. we can place them freely (excluding positions restricted by the environment). Table 1 shows a more structured comparison of the three camera types. We label as “OFF-LINE” any parameter that is adjustable *off-line* but must remain fixed during *on-line* phase. We label as “ON-LINE” any parameter adjustable during the *on-line* phase. For instance, a PTZ camera’s zoom, FoV, DoF and orientation can be changed when the system is up and running but its location cannot.

Type	Zoom	FoV	DoF	Orient.	Location
Fixed	OFF-LINE	OFF-LINE	OFF-LINE	OFF-LINE	OFF-LINE
PTZ	ON-LINE	ON-LINE	ON-LINE	ON-LINE	OFF-LINE
Omni	BOTH	OFF-LINE	BOTH	OFF-LINE	OFF-LINE

Table 1

A comparison of the three basic video camera types considered in our problem.

3.2 Camera Placement Problem

Camera placement is an optimization problem by definition. Let \mathbf{V} be an arbitrary connected volumetric region. If \mathbf{V} is not connected then its connected parts can be treated as individual regions. Let \mathcal{T} be the given task and let \mathcal{C} be the set containing all the constraints required by \mathcal{T} . These may include spatial (*i.e.*, coverage) constraints of \mathbf{V} , temporal (*i.e.*, foveation) constraints for active cameras, quality-of-service (*i.e.*, resolution) constraints, etc. The challenge is to find where to place a set of cameras in \mathbf{V} satisfying \mathcal{C} and minimizing a given cost function $G(\cdot)$. This can be stated in a more compact form as:

$$\arg \min_{\mathbf{\Pi}} G(\mathbf{\Pi}) \text{ subject to } \mathcal{C} \text{ given } \mathbf{V} \quad (1)$$

where $\mathbf{\Pi} = \{\pi_1 \dots \pi_N\}$ and N is the optimal number of cameras to be placed. Note that this definition is an abstraction and different problem instances can be created by plugging-in different constraints, objectives and tasks. Let us give four problem instances that are consistent with the definition.

Problem 1: Given a volumetric area \mathbf{V} , find a camera set $\mathbf{\Pi}$ minimizing a given cost G such that $\forall p \in \mathbf{V}$ is visible from some camera $\pi_i \in \mathbf{\Pi}$:

$$\arg \min_{\mathbf{\Pi}} G(\mathbf{\Pi}) \text{ s.t. } \bigcup_{i=1}^{|\mathbf{\Pi}|} \Delta(\pi_i, \mathbf{V}) = \mathbf{V} \quad (2)$$

where $\Delta(\pi_i, \mathbf{V}) = \{p \in \mathbf{V} : \text{point } p \text{ is visible from camera } \pi_i\}$.

If we assume that all the cameras are omnidirectional (2π FoV), have unlimited resolution, infinite DoF and unit cost then this problem simply reduces to the Art Gallery Problem [1]. This is a simplified version of the surveillance problem, where one needs to assert that it is possible to see all points of interest in \mathbf{V} at all times. A slightly different task is when one needs to be able to foveate some active camera to any point in the region in less than some time threshold. For instance, consider a two-level surveillance system where an event like noise from an opening a door is detected by a low-resolution sensor (*e.g.*, a microphone or proximity sensor) which then sends a request to a high-resolution sensor (*e.g.*, a PTZ camera) for detailed investigation. The acceptable time window between the request and foveation depends directly on the task at hand.

Problem 2: Given a volumetric area \mathbf{V} , find a camera set $\mathbf{\Pi}$ minimizing a given cost G such that $\forall p \in \mathbf{V}$ is visible from some camera $\pi_i \in \mathbf{\Pi}$ in less than time T . In a more compact form:

$$\arg \min_{\mathbf{\Pi}} G(\mathbf{\Pi}) \text{ s.t. } \forall p \in \mathbf{V} \exists \pi_i : \Lambda(\pi_i, p) \leq T \quad (3)$$

where $\Lambda(\pi_i, p)$ gives the maximum time required to foveate camera π_i on point p . Even though the visibility of a point is guaranteed with these two problem definitions, the image quality is not. Including a minimum spatial resolution constraint addresses this problem.

Problem 3: Given a volumetric area \mathbf{V} , find a camera set $\mathbf{\Pi}$ minimizing a given cost G such that $\forall p \in \mathbf{V}$ is visible from some camera $\pi_i \in \mathbf{\Pi}$ with a given minimum required spatial resolution.

$$\arg \min_{\mathbf{\Pi}} G(\mathbf{\Pi}) \text{ s.t. } \bigcup_{i=1}^{|\mathbf{\Pi}|} \Omega(\pi_i, r) = \mathbf{V} \quad (4)$$

where $\Omega(\pi_i, r) = \{p \in \mathbf{V} : \text{point } p \text{ is visible from camera } \pi_i \text{ with spatial resolution greater than } r\}$.

Consider a computer vision system where the task is person identification by face recognition. Suppose the system is composed of a network of cameras. In order to increase the success rate and reliability of the recognition system, firstly the whole area must be visible by the camera network. Secondly

the resolution of the face image must be sufficient for the specific algorithm employed. For instance in [23] the resolution used was 60×50 (reduced to 30×25). Problem 3 addresses exactly these requirements, minimizing at the same time some cost function G , i.e. total camera network bandwidth, energy consumption or total price.

Now consider another scenario where the task is again face recognition in some given region with a camera network. Only this time different regions have different minimum resolution requirements like in an airport where security check point areas may require higher resolution coverage compared to others (which we refer to as *hotspots*). Then the task is to place a collection of cameras satisfying all coverage constraints and minimizing total cost at the same time. This problem can be defined as follows:

Problem 4: Given a volumetric area \mathbf{V} , find a camera set $\mathbf{\Pi}$ minimizing a given cost G such that $\forall p \in \mathbf{V}$ can be viewed by some camera $\pi_i \in \mathbf{\Pi}$ with some minimum spatial resolution required by p .

$$\arg \min_{\mathbf{\Pi}} G(\mathbf{\Pi}) \text{ s.t. } \exists \pi_i : R(p, \pi_i) \geq d(p) \forall p \in V \quad (5)$$

where $R(p, \pi_i)$ is the spatial resolution for point p in camera π_i , and $d(p)$ is the required spatial resolution density function.

These are only a few interesting examples of problem instances for the general camera placement problem given in Eqn. 1.

3.3 Problem Simplification

Although the discovery of an algorithm that can solve the most general case of the camera layout problem for a given volume of interest is highly-desirable, it may prove quite challenging. We therefore focus on a more manageable subclass of this general problem that can be formulated in terms of planar regions that are typical of a building floor plan, *e.g.*, Fig. 1. We will then approximate the region by a polygon. This is a valid assumption since most buildings and floor plans consist of polygonal shapes or can be approximated by a collection of polygons. The problem then becomes to efficiently compute a camera layout given a floor plan, approximated by a polygon, to be observed. As will be shown, a solution to this problem can be obtained via binary optimization over a discrete problem space.

Efficient computational geometry algorithms exist for operations involving simple polygons ², like convexity determination, area finding, triangulation,

² A simple polygon is defined as a region enclosed by a single closed polygonal chain

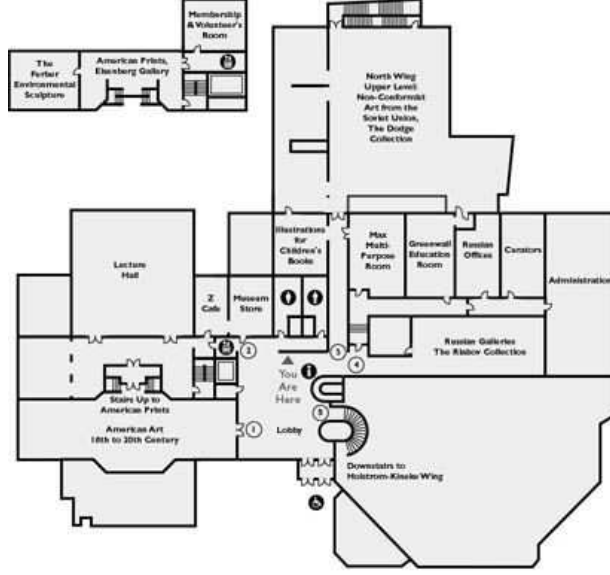


Fig. 1. A typical floor plan.

etc. However, in our work, we allow the polygon to have cavities (holes) that represent potential visibility-occluding entities in the floorplan, e.g., columns, separator wall partitions, etc. The well known linear time visibility algorithms for simple polygons [25] are not applicable for this case. We therefore formulated an angular sweep visibility algorithm that handles a polygon with holes. Detailed analysis of the algorithm is provided in the next section.

4 Visibility Algorithm

Given a simple polygon \mathbf{P}_e and simple polygonal holes \mathbf{P}_k $k=1 \dots M$ and a point X such that:

- $\mathbf{P}_k \subset \mathbf{P}_e \forall k$
- $\partial \mathbf{P}_i \cap \partial \mathbf{P}_j = \emptyset : i \neq j \forall i \forall j$
- $X \in \mathbf{P}_e \wedge X \notin \mathbf{P}_i \forall i$

where ∂ is the boundary operator, find the visibility polygon \mathbf{P}_v , *i.e.*, all points p such that,

$$\mathbf{P}_v \triangleq \max\{p : p \in \mathbf{P}_e \wedge p \notin \mathbf{P}_i \forall i \wedge \overline{Xp} \subset \{\mathbf{P}_e - \mathbf{P}_{k=1 \dots M}\}\}$$

In essence, the goal is to compute the polygonal region containing all visible points from a given point X inside a simple polygon with simple polygonal

that does not intersect itself [24].

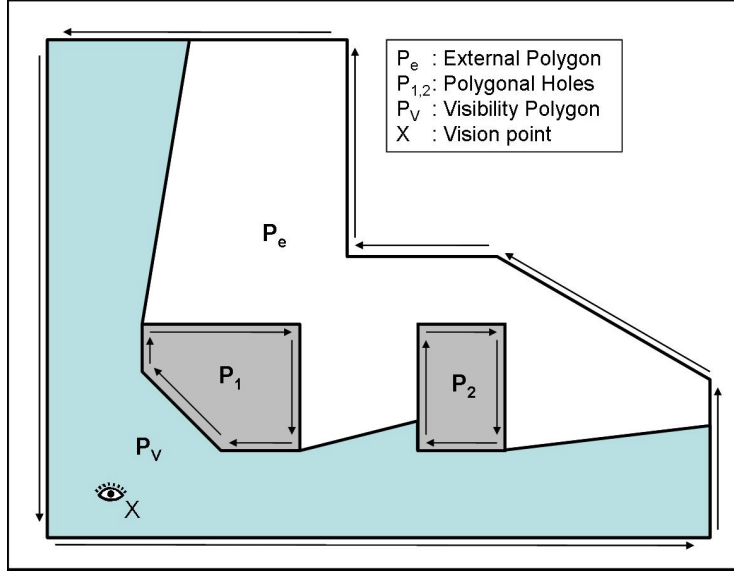


Fig. 2. A visibility polygon example.

holes. Let us first present the algorithm using a single simple polygon. The extension to a simple polygon with simple polygonal holes is straightforward and will be explained later.

The polygon is represented as an edge list in Cartesian coordinates,

$$\mathbf{ELC} \triangleq \{(vc_1^s, vc_1^e), (vc_2^s, vc_2^e), \dots, (vc_i^s, vc_i^e), \dots, (vc_n^s, vc_n^e)\}$$

where i is the edge index ordered CCW (Counter Clockwise), $vc_i^s, vc_i^e \in \mathbb{R}^2$ are the start and end vertices of the i^{th} edge in Cartesian coordinate system and $vc_i^e = vc_{i+1}^s$. This representation has a well defined interior and exterior of the polygon. The region of space on the left of any edge formed by two consecutive vertices can contribute to the polygon. The idea of the algorithm is to perform an angular sweep of the polygon with X being the center of the sweep and compute the visible line segments over the range $[0, 2\pi]$. The union of all the visible line segments is the visibility polygon \mathbf{P}_V and is the output of the algorithm. The initial step is to convert \mathbf{ELC} to its polar coordinate representation,

$$\mathbf{ELP} \triangleq \{(vp_1^s, vp_1^e), (vp_2^s, vp_2^e), \dots, (vp_i^s, vp_i^e), \dots, (vp_n^s, vp_n^e)\}$$

where $vp_i \triangleq \{\theta_i, r_i\}$, polar angle and radius of i^{th} edge's vertex respectively. This conversion makes an angular sweep possible. To prevent potential ambiguities for edges crossing $\theta = 0$, each such edge is subdivided into two edges at the intersection point $\theta = 0$. Note that only the edges satisfying $\theta_i^s < \theta_i^e$ can be fully or partially visible from X hence no other edge may contribute to \mathbf{P}_V .

Theorem 1 Given an edge list in counter clockwise order of a simple polygon \mathbf{P} and a point $X \in \mathbf{P}$, let θ_i^s and θ_i^e represent the polar angles of the start and end vertices of edge i with X being polar coordinate center. Let \mathbf{P}_V be the visibility polygon from point X . If $\theta_i^e < \theta_i^s$ then edge i cannot be part of \mathbf{P}_V .

PROOF. \mathbf{P}_V is star convex by definition.³ Any edge of \mathbf{P} creates two half-planes. Let e_i be some edge of \mathbf{P} . It is a necessary condition for star convexity (by its definition) to have the line segment connecting point X with any point of e_i , inside the half-plane containing the inward-pointing normal to the edge e_i . Consider the vector u connecting the origin of e_i to point X . Then it is necessary to have the cross product $u \times e_i \geq 0$ (\geq because we have the edges in CCW order). This is only possible when $\theta_i^e \geq \theta_i^s$. \square

Therefore all other edges where $\theta^e < \theta^s$ can be eliminated at this step. In order to sweep the polygon in CCW order, an ordered list of vertex polar angles is necessary. Let,

$$\mathbf{Q} \triangleq \{(\theta_1^s, r_1, \epsilon_1), (\theta_2^s, r_2, \epsilon_2), \dots, (\theta_n^s, r_n, \epsilon_n), (\theta_1^e, r_1, \epsilon_1), (\theta_2^e, r_2, \epsilon_2), \dots, (\theta_n^e, r_n, \epsilon_n)\}$$

be the list of polar angles (θ), radii (r) and respective edge pointers (ϵ) of all remaining vertices. In order to sweep the polygon in monotonically increasing angular order, let \mathbf{Q} be sorted in lexicographically ascending order.

Algorithm 1 constructs the visibility polygon by keeping track of the current visible edge during the angular sweep. It does so by keeping the index of the current visible edge in *ActiveEdge*. The *ActiveEdge* can only change at a polygon vertex point, hence each vertex is an *event point*. There are four main cases of events (Fig. 3):

CASE 1 The current vertex is the end vertex of *ActiveEdge* and the next edge is contiguous (Fig.3, A). In this case the current vertex is part of the visible polygon and both *ActiveEdge* and the next edge are at least partially visible from X . Current vertex is part of \mathbf{P}_V .

CASE 2 The current vertex is the end vertex of *ActiveEdge* and the next edge is *not* contiguous (Fig.3, B). The current vertex is part of the visible polygon. In order to find the next active edge it is necessary to find all the edges intersecting the half line emanating from X in the direction of the current vertex (j in Fig. 3 B). There may be more than one such edge. A

³ A subset \mathbf{P} of \mathbb{R}^n is star convex if there exists an $x_0 \in \mathbf{P}$ such that the line segment from x_0 to any point in \mathbf{P} is contained in \mathbf{P} [26].

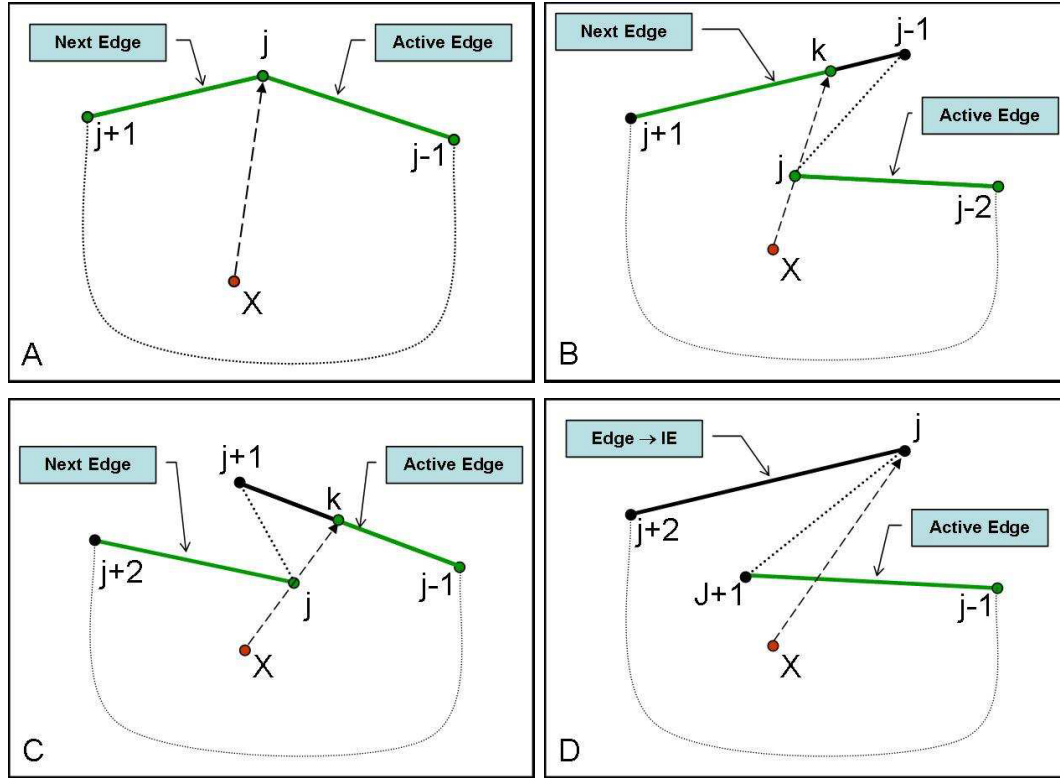


Fig. 3. Four cases of visibility events.

Algorithm 1: FINDVISIBILITYPOLYGON(\mathbf{ELC} , X)

input : Edge list \mathbf{ELC} and a point X inside the polygon.

output: The list of vertices of the visibility polygon \mathbf{P}_V for the given point X .

begin

 Convert \mathbf{ELC} to its polar coordinates, \mathbf{ELP} ;

 Prune all backward facing edges;

 Construct the edge list \mathbf{Q} ;

 Sort \mathbf{Q} in lexicographically ascending order;

 Initialize the vertex list of visibility polygon \mathbf{P}_V to the first element of \mathbf{Q} ;

 Initialize the priority queue \mathbf{IE} , the list of visited edges which may become visible, to the list of edges in \mathbf{Q} with $\theta^s = 0$ except the first element of \mathbf{Q} ;

 Set *ActiveEdge* to the first element of \mathbf{Q} ;

 /* $|\cdot|$ is the norm operator. */

for $i \leftarrow 2$ **to** $|\mathbf{Q}| - 2$ **do**

$[\mathbf{P}_V, \text{ActiveEdge}, \mathbf{IE}] \leftarrow \text{HANDLEEVENTPOINT}(\mathbf{Q}, i, \mathbf{P}_V, \text{ActiveEdge}, \mathbf{IE});$

end

priority queue (\mathbf{IE}) is used to contain these edges sorted by their radii in increasing order. The edge with closest intersection point to X is visible.

This intersection point k is part of \mathbf{P}_v and its corresponding edge (top of \mathbf{IE}) becomes *ActiveEdge*.

CASE 3 The current vertex is the start vertex of some edge other than the *ActiveEdge* and it is closer to X than *ActiveEdge* (Fig. 3, C). The intersection point k of the half line emanating from X in the direction of the current vertex with *ActiveEdge* and current vertex are parts of \mathbf{P}_v . The edge having the current vertex as its start vertex becomes *ActiveEdge*.

CASE 4 The current vertex is the start vertex of some edge other than the *ActiveEdge* and it is farther away than *ActiveEdge* from X (Fig. 3, D). The edge having current vertex as its start vertex is inserted into the priority queue \mathbf{IE} since it is a candidate for future visibility.

Cases 1 and 2 are handled by the 3rd `{if...then...}` block of the function `HANDLEEVENTPOINT`. Cases 3 and 4 are handled in the 4th block. Function `UPDATEIE` handles insertion of edges into the priority queue \mathbf{IE} sorted by the radii of the intersection points of the ray emanating from X with edges in \mathbf{IE} .

4.1 Extension to Polygons with Holes

Note that only the inside region of \mathbf{P}_e may contribute to \mathbf{P}_v . This observation is also valid for the outside regions of polygonal holes. The inside/outside definition of the polygons is strongly coupled with their edge orderings, *i.e.*, CCW for \mathbf{P}_e or CW (Clockwise) for the polygonal holes. For this reason, the aforementioned angular sweep algorithm will work correctly in the presence of holes given that their edge ordering will be set to CW. Hole edges are simply appended to the edge list, \mathbf{ELC} .

4.2 Runtime Analysis

Let m be the number of polygon edges. Converting the vertices from Cartesian to polar coordinates takes $O(m)$ time. Sorting \mathbf{Q} takes $O(m \log m)$ time. `UPDATEIE` takes $O(\log m)$ time. Since each edge can be added to and removed from \mathbf{IE} only once, the total running time of `HANDLEEVENTPOINT` is also bounded by $O(\log m)$. Since `HANDLEEVENTPOINT` is called $O(m)$ times, the total running time of `FINDVISIBILITYPOLYGON` is $O(m \log m)$.

Function HANDLEEVENTPOINT($\mathbf{Q}, i, \mathbf{P}_V, ActiveEdge, \mathbf{IE}$)

input : The current vertex index i , $\mathbf{Q}, \mathbf{P}_V, ActiveEdge$
output: $\mathbf{P}_V, ActiveEdge$
begin

 Let *NextEdge* be the edge index of the next vertex in \mathbf{Q} ;
 Let *CurrentEdge* be the edge index of the current vertex;
 Let *CurrentVertex* be the current vertex;
 Let *Ray* be the half ray emanating from the polar coordinate origin
 towards the *CurrentVertex*;

if (*CurrentEdge* == *ActiveEdge*) **then**
 if (*CurrentVertex* == Start vertex of *ActiveEdge*) **then**
 Do nothing;
 return ;
 if (*NextEdge* is contiguous) **then** */
 /*CASE 1
 push *CurrentVertex* onto \mathbf{P}_V ;
 ActiveEdge \leftarrow *NextEdge*;
 return ;
 else */
 /*CASE 2
 repeat
 | $e \leftarrow$ **pop** \mathbf{IE} ;
 until e is not totally blocked by the *ActiveEdge*;
 Find the intersection point k of *Ray* with e ;
 push *CurrentVertex* onto \mathbf{P}_V ;
 push k onto \mathbf{P}_V ;
 ActiveEdge \leftarrow e ;
 return ;

 Find the intersection point k of *Ray* with *ActiveEdge*;

if ($|k| > |CurrentVertex|$) **then** */
 /*CASE 3
 push k onto \mathbf{P}_V ;
 push *CurrentVertex* onto \mathbf{P}_V ;
 UPDATEIE (*ActiveEdge*);
 ActiveEdge \leftarrow *CurrentEdge*;
 else */
 /*CASE 4
 if (*CurrentEdge* is not totally blocked by *ActiveEdge*) **then**
 UPDATEIE (*CurrentEdge*);

end

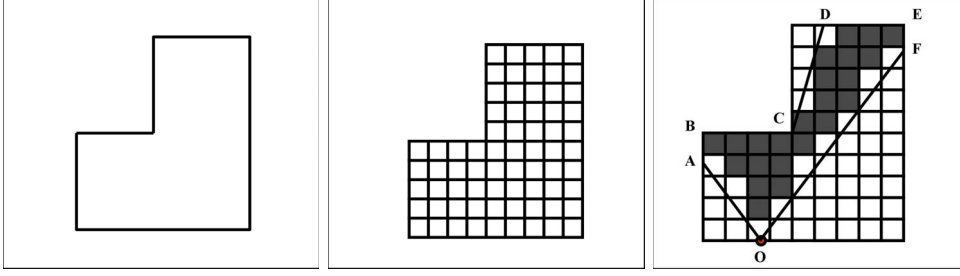


Fig. 4. **Left:**The polygon.**Middle:**Cellular representation of the polygon.**Right:**The cell coverage of a camera O with FoV limits \overline{OA} and \overline{OF} and visible polygon $\overline{OABCDEF}$. The dark cells are the visible ones from camera O .

5 Camera Placement Algorithm

The optimal camera placement problem at hand may be intractable in the continuous domain. To make solving the problem easier, we will attack it in the discrete domain. We represent the polygonal region in terms of a cellular grid. Various cellular representations are possible, for instance square or hexagonal cells. Moreover, multiresolution or single resolution grids are possible. For our implementation, we use a square cell representation at a single resolution without loss of generality. An illustration of the process of converting to a cellular grid representation is shown in Fig. 4.

We next create a set of candidate cameras ($\mathbf{\Pi}$) by sampling the camera parameters (π) which are relevant for the task at hand. The idea is to find a subset of $\mathbf{\Pi}$ which optimizes the given cost and satisfies the constraint set \mathcal{C} . For the sake of illustration suppose the task at hand is to cover the area of interest by a minimum number of cameras. Suppose furthermore that the only camera constraint is its FoV. Each camera then will have an associated coverage region (feasible region) depending on its location and orientation inside the polygonal region of interest. By superimposing this feasible region onto the previously constructed cellular grid, one can easily find the cells that overlap it. An example is shown in Fig. 4. This actually is the binary (1 for covered and 0 for not) cellular representation of the feasible region for the camera at hand. Repeating the same process for all candidate cameras in $\mathbf{\Pi}$, produces a collection of discrete feasible regions represented on the same cellular grid.

To compute the optimal subset of cameras out of $\mathbf{\Pi}$ is a combinatorial problem and may be very expensive if not dealt with carefully. Fortunately, a special optimization model, called 0-1 programming, provides a convenient way to represent this problem in matrix notation in terms of a *Set Coverage Problem* [27]:

$$\min \mathbf{c}\mathbf{x} \quad s.t. \quad \mathbf{A}\mathbf{x} \geq \mathbf{b} \quad \mathbf{x} \in \{0, 1\} \quad (6)$$

where \mathbf{A} is an $m \times n$ matrix whose i^{th} row elements are coefficients of the i^{th} linear inequality constraint, \mathbf{b} is an $m \times 1$ vector whose i^{th} element is the

right-hand-side coefficient of constraint i , \mathbf{c} is $1 \times n$ vector whose i^{th} element is the cost associated with i^{th} element of \mathbf{x} which is a $n \times 1$ vector containing n decision variables.

In the most general sense, the constraints given by \mathbf{A} and \mathbf{b} define a convex polytope *Poly* in n dimensional space that contains all the feasible solution points for the given optimization problem. The 0-1 programming tries to find a binary vector \mathbf{x}^* which yields the minimum cost function value over *Poly*. Let Q be the set containing all possible binary combinations of \mathbf{x} . Let $Q^* \subset Q$ containing only the elements of Q found inside *Poly*. Then 0-1 programming can be explained as looking for $\mathbf{x}^* \in Q^*$ giving the minimum cost function value.

The direct relationship between our problem and Eqn. 6 can be easily noted if we let each candidate camera's associated discrete feasible region vector be a column of \mathbf{A} and the cellular grid vector of \mathbf{P} be \mathbf{b} . The *Poly* defined by $\mathbf{Ax} \geq \mathbf{b}$ will then contain all the feasible combinations of candidate cameras satisfying full coverage of \mathbf{P} . We may then apply 0-1 programming to find the optimal set of cameras giving the minimum cost value. In other words the solution to the 0-1 model (Eqn.6) constructed as explained becomes the solution to our original camera location problem.

We obtain the 0-1 model representation from a given camera location problem following these steps:

Step 1 Find a representation for the given constraints as a spatial coverage.

This is the most crucial phase of the solution. If there exists a way to represent \mathcal{C} as a spatial coverage problem then it is also possible to solve it using the proposed method. Solutions for **Problem 1** and **Problem 2** differ mainly in this representation. These will be given in detail in section 5.

Step 2 Represent the polygonal region \mathbf{P} as an occupancy cellular grid. Let

$\mathbf{OG}(\mathbf{P})$ be the $h \times w$ binary matrix whose $(i, j)^{th}$ element is 1 if grid cell p with coordinates (j, i) is inside \mathbf{P} and 0 otherwise. Let us call $\mathbf{OG}(\mathbf{P})$ the occupancy grid of \mathbf{P} . Note that $h \times w$ is directly proportional to the resolution of the occupancy grid and it is an input parameter for the algorithm.

Step 3 Choose n samples from $\mathbf{\Pi}$ given the camera specifications and \mathbf{P} . Let

\mathbf{s}_k be the k^{th} sample. Note that n is an input parameter for the algorithm. For instance, depending on the task constraints, $\mathbf{\Pi}$ can be sampled over different focal lengths (multiple camera lenses), different camera orientations, locations, aperture, etc.

Step 4 For each \mathbf{s}_k find the occupancy cellular grid of its spatial coverage representation given \mathcal{C} . Since we are dealing with cameras, visibility is the top most constraint. Therefore the first step in computing the spatial coverage is the computation of the visibility polygon. Then the spatial coverage

can be found by taking the intersection of the visibility polygon and \mathcal{C} , *e.g.*, resolution constraints, FoV constraints, DoF constraints etc. Let \mathbf{S}_k be the $h \times w$ binary matrix whose $(i, j)^{th}$ element is 1 if cell grid p with coordinates (j, i) is inside the spatial coverage of \mathbf{s}_k and 0 otherwise.

Step 5 Construct the 0-1 model (Model 6). Let \mathbf{A} be:

$$\begin{aligned}\mathbf{A}_{(m=h \times w, n)} &= \{\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_i, \dots, \mathbf{a}_n\} \\ \mathbf{a}_i &= \{\mathbf{q}_1^i, \mathbf{q}_2^i, \dots, \mathbf{q}_j^i, \dots, \mathbf{q}_w^i\}^T \\ \mathbf{q}_j^i &= \{j^{th} \text{ column of } \mathbf{S}_i\}^T\end{aligned}$$

Let \mathbf{b} be:

$$\begin{aligned}\mathbf{b}_{(m=h \times w, 1)} &= \{\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_j, \dots, \mathbf{e}_w\}^T \\ \mathbf{e}_j &= \{j^{th} \text{ column of } \mathbf{OG}(\mathbf{P})\}^T\end{aligned}$$

Let $\mathbf{c} = \{c_1, c_2, \dots, c_j, \dots, c_n\}$ where c_j is the cost associated with the camera \mathbf{s}_j . This may be the price, required bandwidth, consumed energy etc. of the camera. When $\mathbf{c} = \mathbf{1}_{1 \times n}$ the solution is for the minimum number of cameras.

At this point we have all the necessary components of Eqn. 6. The last step is to solve this model using one of the well-known methods. In our case we use ‘‘Branch-and-Bound’’ [27]. Let us denote the optimal solution of the model with \mathbf{x}^* . Note that the decision variable vector \mathbf{x}^* is also an indicator vector, *i.e.* if $x_i^* = 1$ then the camera location \mathbf{s}_i is one of the optimal camera locations for the given problem instance. If $\mathbf{x}^* = \textit{infeasible}$ then there is no camera location configuration satisfying \mathcal{C} given the current sample set \mathbf{s} .

5.1 Sampling Π

Since the candidate selection is performed on continuous camera parameters and since the resulting candidate set should be a fair representation of these domains, we refer to this process as sampling of Π . The choice of a sample set has a direct effect on the solution: It becomes the domain of the optimization model. The negative effects of an *ad hoc* selection of the sample set may range from an infeasible solution to longer run-times. Furthermore, some camera parameters may be restricted to specific values or range of values due to physical constraints, *i.e.*, some locations in the region of interest may not be accessible for camera placement, orientation may be constrained depending on the location of the camera, environmental factors like vibration or temperature may prove unsuitable for some camera types, or potential camera locations may be restricted to specific locations due to aesthetic considerations. To get

an acceptable optimal solution, such issues must be taken into account during the sampling process.

Usually the sampling is carried out first on the planar locations of the cameras in the polygonal area. Then other parameters like orientation, focal length, aperture are sampled over their allowed ranges to reflect possible selections of different camera types, lens types and settings. For example, if our camera arsenal contains only fixed and omnidirectional cameras and three types of lenses differing in focal length, then we could first sample the potential locations. If we only have a limited number of mounting bracket orientations, the orientation would be sampled over these values for the fixed cameras for each sampled location (note that sampling over orientation does not apply for the omnidirectional camera). Finally we would sample over the three different focal lengths for each potential location and orientation.

It is important to note that the optimality of the solution will depend on the density of samples. In other words, the solution is optimal up to the current sample set. Usually the larger the number of sample points is, the closer is the solution of the discrete optimization to the continuous (global) optimal. However, relatively lower density sampling is generally sufficient to obtain a solution which is acceptably close to the optimal solution for the continuous problem.

5.2 Correctness

The i^{th} constraint in Eqn. 6 is:

$$\mathbf{A}_{i,1} \times x_1 + \mathbf{A}_{i,2} \times x_2 + \dots + \mathbf{A}_{i,n} \times x_n \geq b_i \quad (7)$$

Note that $\mathbf{A}_{i,j}$ represents the coverage status of the *same* grid cell by j^{th} sample: it is 1 if occupied, 0 otherwise. b_i represents the coverage of the *same* grid cell by the polygon \mathbf{P} in the same way. So, the constraint in Eqn. 7 guarantees that the grid cell represented by b_i is covered by *at least* 1 camera. Since this is true for all $w \times h$ constraints, Eqn. 6's feasible region is all the possible combinations $C(n, j)$ $j = 1 \dots n$ of \mathbf{s} which cover \mathbf{P} given \mathcal{C} . Then the solution of this model is the combination of \mathbf{s} which covers \mathbf{P} and gives the minimum value for the objective function $\mathbf{c}\mathbf{x}$.

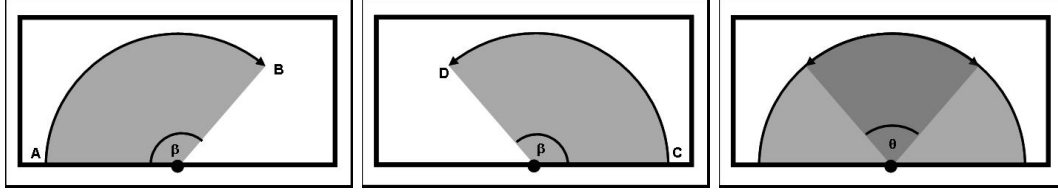


Fig. 5. Illustration of the reachable region from a camera (black disc) location on the polygon perimeter.

6 Experiments

In this section we implement Problems 2,3 and 4 defined previously and give solutions using the proposed approach. We test the system using real floorplans and use real camera specifications to determine the sampling in Π . More specifically we show how to convert the constraints of the problem definitions to area coverage constraints. The remaining steps are the same for all experiments.

For each experiment we use a grid resolution of 100×100 cells. The exact scale of the floorplans was unavailable. Assuming standard size door openings and given the grid resolution, the grid cell size is approximately 40 mm^2 . but will also increase the number of constraints in the Eqn.6. This is a tradeoff between the speed and accuracy of the discrete approximation. The algorithm is implemented using MATLAB 6.5 R13 on a Dual Athlon 1 GHz computer with 1 GB memory. running time is our custom implementation of Branch-and-Bound algorithm which can be highly optimized.

6.1 Experiment 1

Suppose the cameras are PTZ. Let us denote the maximum horizontal angular speed of a given PTZ camera with ω_h . Recall that T is the time constraint (Eqn. 3). Assume that cameras can only be placed along the perimeter of \mathbf{P} . Consider the worst case scenario. Suppose at some point in time the camera is foveated towards the minimum angle given its location. If the camera is located along an edge e , then its orientation corresponding to its minimum horizontal angle will be along e . Let e be parallel to x-axis without loss of generality. Then the minimum horizontal angle will be 0. The camera can then foveate up to orientation $\beta = T \cdot \omega_h$. Now consider the other extreme, the camera pointing to its maximum horizontal angle, π . It can foveate down to orientation $\pi - \beta$ in time T for the same reason. The intersection of the two regions formed by orientations spanning $[0, \beta]$ and $[\pi - \beta, \pi]$ is the feasible coverage for the worst case scenario and it is defined by $\theta = 2\beta - \pi$, as shown

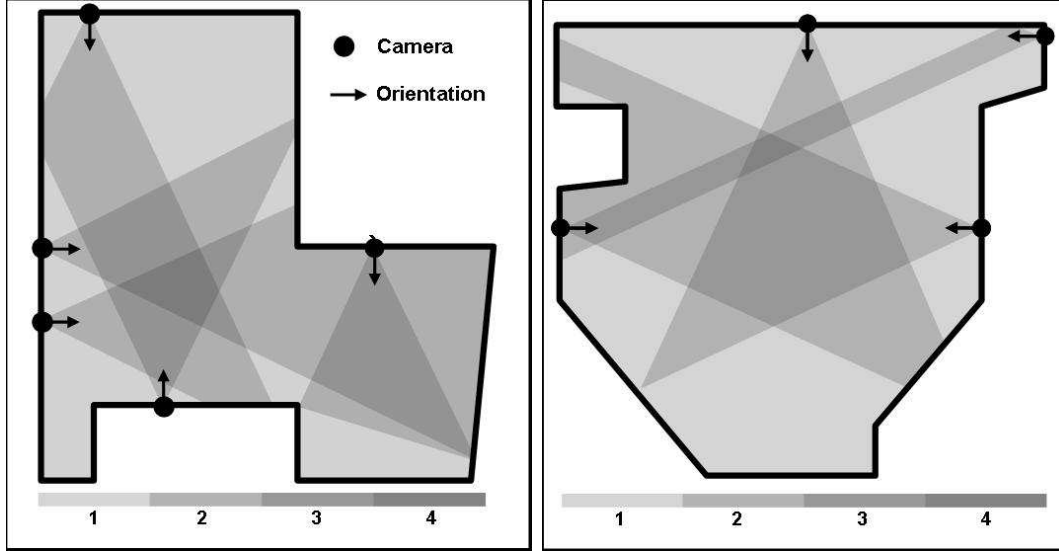


Fig. 6. Solution for problem 2 with $T = 1.5 \text{ sec}$ and $\omega_h = 80^\circ/\text{sec}$ using two real floorplans. Black discs are the optimal locations. Darker areas are covered by multiple cameras.

in Fig. 5.

Now we are able to represent the constraint defined by the time threshold T as a coverage constraint. The camera model to be placed is Sony EVI-D30 PTZ camera with $\omega_h = 80^\circ/\text{sec}$. Let $T = 1.5 \text{ sec}$, then we have $\beta = T \cdot 80^\circ/\text{sec} = 120^\circ$ and $\theta = 2\beta - 180^\circ = 60^\circ$. We sample five uniform camera locations per edge on the polygon \mathbf{P} . The solutions for two floorplans are shown in Fig. 6.

6.2 Experiment 2

In this experiment, we solve an instance of Problem 3. Given a region, the task is to setup a camera network such that every point in the region can be seen from some camera with at least a given spatial resolution. Suppose the region of interest is under surveillance and the task of the vision system is to recognize people using the face images captured by some camera of the network. It is clear that in order for the system to work reliably, sufficient discriminatory details in the face image should be visible. Furthermore, these details should be visible with at least some degree of spatial resolution to ensure accurate recognition. For example, in [23] face images from the FERET database [28] with 50×60 resolution are used for face recognition. Assuming that the average face width is around 200 mm, it would be conceivable to impose a minimum spatial resolution requirement of $\frac{50}{200}$ pixels/mm for the face recognition system under consideration. For the sake of this experiment suppose we have only one

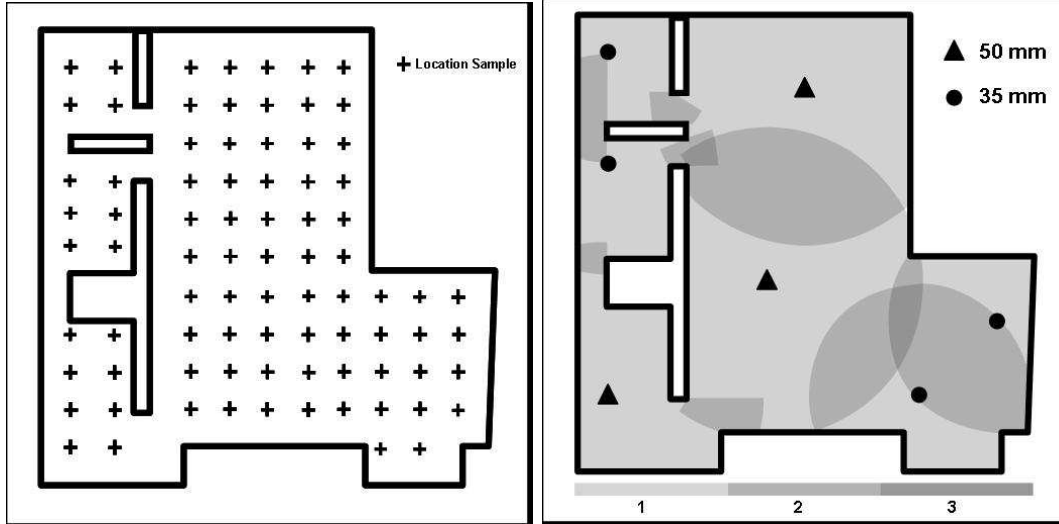


Fig. 7. Solution for problem 2. **Left:** Crosses are sampled locations. **Right:** Coverage map for optimal locations. Darker areas are covered by multiple cameras.

type of omnidirectional camera available (*i.e.*, FullView FC-1005⁴) with two focal length lenses: 35 mm and 50 mm. Different lens types will have different prices too. For the purpose of the experiment, assume the 35 mm lens costs \$100 and the 50 mm lens costs \$150. First we have to compute the feasible region of a single omnidirectional camera with a fixed focal length. Using the equation for the length of the projection of a real world object on the image plane of a camera [29] and the FC-1005 CCD specifications we get the maximum distance guaranteeing $\frac{50}{200}$ pixels/mm horizontal resolution to be $\sim 1291\text{cm}$ for 35mm lens and $\sim 1844\text{cm}$ for 50mm lens.

The resulting optimal layout for the experiment is presented in Fig. 7. Optimal total cost is $(3 \times \$150) + (4 \times \$100) = \$850$.

6.3 Experiment 3

In this experiment we assume different areas of the floorplan have different spatial resolution requirements, e.g. Fig. 8. A real world example would be a casino where game areas and cashiers may require more detailed coverage than the other areas. For this specific example, we set the detailed area spatial resolution constraint to $\frac{50}{200}$ pixels/mm and non-detailed areas to $\frac{5}{200}$ pixels/mm. All other settings and specifications are the same as the ones used in the experiment 2. The solution is given in Fig. 9. Detailed areas are chosen to be around doors and to reflect special exhibition items if the floorplan is of

⁴ <http://www.fullview.com/>

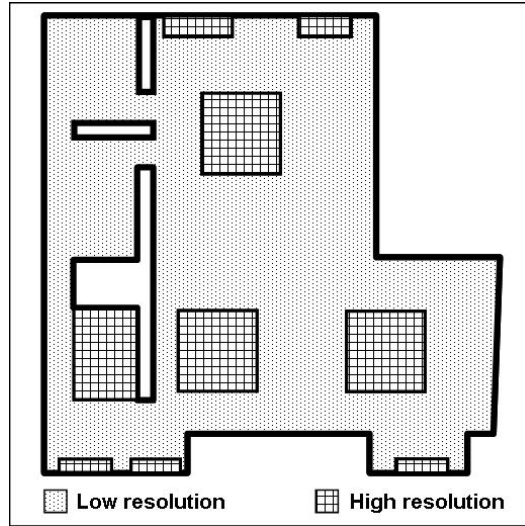


Fig. 8. Resolution constraint map. Grid areas require $\frac{50}{200}$ pixels/mm, dotted areas require $\frac{5}{200}$ pixels/mm.

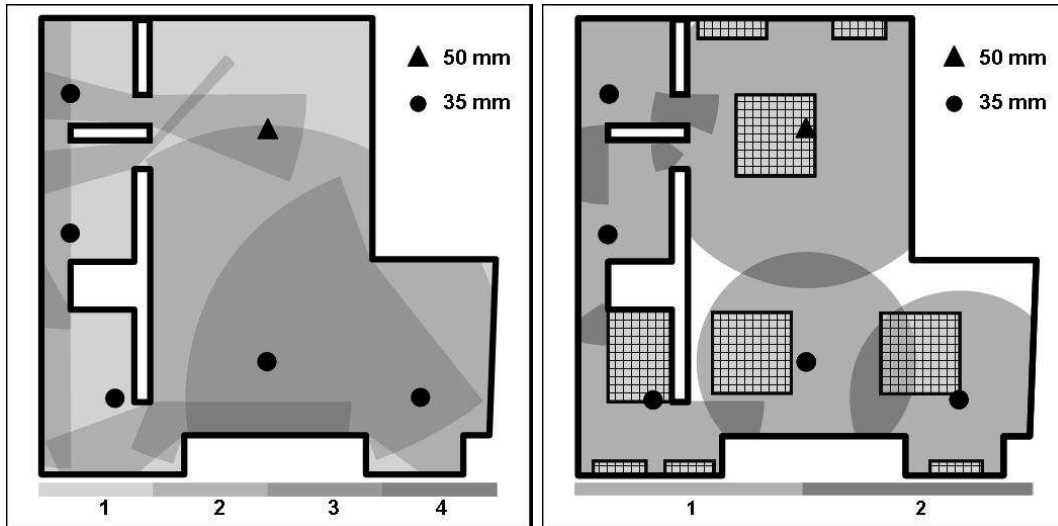


Fig. 9. Solution for problem 3. **Left:** Low resolution coverage map for optimal locations. **Right:** High resolution coverage map for optimal locations. Darker areas are covered by multiple cameras.

a museum or game tables and cashiers if it is a casino. Optimal total cost is $(1 \times \$150) + (5 \times \$100) = \$650$.

6.4 Experiment 4

The floorplan is a parking lot. The constraint is to cover the lot with at least $\frac{50}{200}$ pixels/mm resolution using the same camera and lens types as in

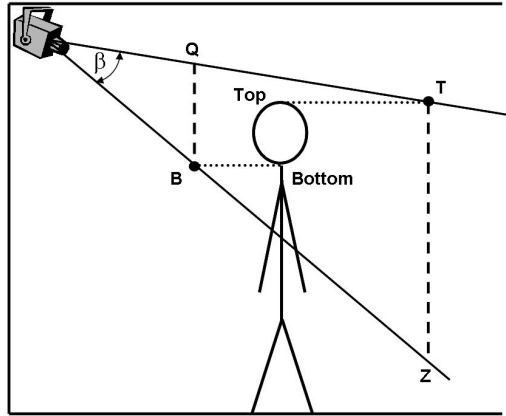


Fig. 10. Coverage limitation due to camera tilt and person's height.

Experiment 3. An additional constraint in this case is that the person's face must be in the field of view of at least one camera. The situation is illustrated in Fig. 10. Even though a region of the parking lot is within a camera's field of view, the person's face can only be visible for that camera within an annulus, which is determined by the height of the camera from the ground, its tilt angle, its vertical field of view coverage, and the expected height and face dimensions of a person. The loci of minimum and maximum distance points (B and T in Fig. 10) define the annulus. The intersection of the resolution constraint and this new constraint produces another annulus. Furthermore, the cameras can be mounted on the wall of the building or on a post depending on the sample location. This subsequently, generates another cost item. For the sake of the experiment assume the wall mount costs \$50 and the post costs \$1500. The optimal location is given and coverage obtained is shown in 11. The total cost is $3 \times (\$150 + \$50) + 5 \times (\$150 + \$1500) = \$8850$.

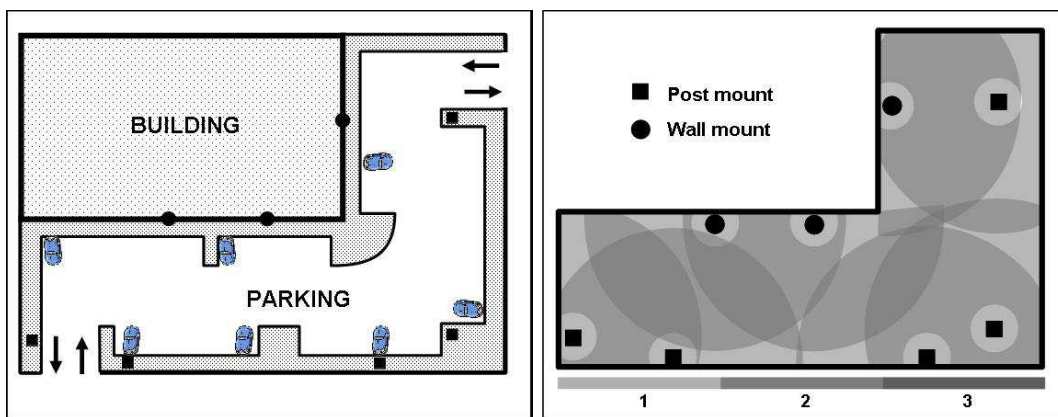


Fig. 11. **Left:** Parking lot plan and optimal camera locations. **Right:** Coverage map.

7 Conclusion

In this paper, we formulated a solution to the general task-based camera location problem, in order to satisfy coverage constraints while minimizing the total cost. An efficient angular sweep algorithm was proposed for computing the visible region for each camera; the algorithm works for polygonal regions of interest with polygonal holes. We gave four specific, real-world problem instances along with a solution based on the discretization. If the task-based constraints of the vision system are reducible to area coverage, then these constraints may also be satisfied by the solutions of our proposed method.

The formulation given in this paper is general, and its use is not limited to layout of video camera networks. We believe that coverage problems found in other applications, such as in layout of sprinkler systems, illumination problems, wireless networks, etc., can be attacked using the proposed method. Even though the presented experiments only optimize the total price, it is also possible to include other cost measures, like network bandwidth, energy consumption, etc. as long as the cost can be expressed in terms of a linear function. Furthermore it should be possible to incorporate into the model a budget constraint, and find a layout that conforms to this budget if a feasible solution exists.

To gain a tractable solution, we converted a general continuous optimization problem to a discretized binary optimization. Thus, the solution gained – while optimal for the discretization – is only approximately optimal for the continuous form. If the density of the grid approximation and the cardinality of the camera sample set are increased, then the solution of the discrete optimization tends to better-approximate the continuous (global) optimal solution. However, we found in our experiments that relatively lower density sampling is generally sufficient to obtain a solution which is acceptably close to the true optimal layout. Nonetheless, in future work, we hope to pursue solutions to the optimization in the continuous space as opposed to the discrete one, as this should yield the true optimal layout.

References

- [1] J. O'Rourke, *Art Gallery Theorems and Algorithms*, Oxford, New York, 1987.
- [2] V. Chvatal, A combinatorial theorem in plane geometry, *J. of Combinatorial Theory Series 18* (1975) 39–41.
- [3] S.Fisk, A short proof of Chvatal's watchman theorem, *J. of Combinatorial Theory Series 24* (1978) 374.

- [4] S. Carlsson, B. J.Nilsson, S. C.Ntafos, Optimum guard covers and m-watchmen routes for restricted polygons, in: Workshop on Algorithms and Data Structures, 1991, pp. 367–378.
- [5] A. Efrat, L. J.Guibas, S. Har-Peled, D. C.Lin, J. S.B.Mitchell, T.M.Murali, Sweeping simple polygons with a chain of guards, in: Symposium on Discrete Algorithms, 2000, pp. 927–936.
- [6] L. J.Guibas, J.-C. Latombe, S. M.LaValle, D. Lin, R. Motwani, A visibility-based pursuit-evasion problem, *Int. J. of Computational Geometry and Applications* 9 (4/5) (1999) 471–.
- [7] I. Suzuki, Y. Tazoe, M. Yamashita, T. Kameda, Searching a polygonal region from the boundary, *Int. J. of Computational Geometry and Applications* 11 (5) (2001) 529–553.
- [8] P. Bose, L. J.Guibas, A. Lubiw, M. H.Overmars, D. L.Souvaine, J. Urrutia, The floodlight problem, *Int. J. of Computational Geometry and Applications* 7 (1/2) (1997) 153–163.
- [9] V. Estivill-Castro, J. O’Rourke, J. Urrutia, D. Xu, Illumination of polygons with vertex lights, *Information Processing Letters* 56 (1) (1995) 9–13.
- [10] P. K.A.Tarabanis, R.Y.Tsai, A survey of sensor planning in computer vision, in: *IEEE Trans. on Robotics and Automation*, 1995, pp. 86–104.
- [11] A.Mittal, L.S.Davis, Visibility analysis and sensor planning in dynamic environments, in: *European Conf. on Computer Vision (ECCV)*, 2004.
- [12] J.Cortes, S.Martinez, T.Karatas, F.Bullo, Coverage control for mobile sensing networks, in: *IEEE Conf. on Robotics and Automation*, 2002, pp. 1327–1332.
- [13] R. Bajcsy, Active perception, in: *Proc. of the IEEE*, 1988, pp. 996–1005.
- [14] T. Arbel, F. P.Ferrie, Entropy-based gaze planning, *Image and Vision Computing* 19 (11) (2001) 779–786.
- [15] J.Maver, R.Bajcsy, Occlusions as a guide for planning the next view, *IEEE Trans. Pattern Anal. Machine Intell.* 15 (5) (1993) 417–433.
- [16] R.Pito, A solution to the next best view problem for automated surface acquisition, *IEEE Trans. Pattern Anal. Machine Intell.* 21 (10) (1999) 1016–1030.
- [17] Y.Cui, S.Samasekera, Q.Huang, M.Greifenhagen, Indoor monitoring via the collaboration between a peripheral sensor and a foveal sensor, in: *IEEE Workshop on Visual Surveillance*, 1998, pp. 2–9.
- [18] J. Batista, P. Peixoto, H. Araujo, Real-time active surveillance by integrating peripheral motion detection with foveated tracking, in: *IEEE Workshop on Visual Surveillance*, 1998.
- [19] I.Mikic, K.Huang, M. M.Trivedi, Activity monitoring and summarization for an intelligent meeting room, in: *Workshop on Human Motion*, 2000, pp. 107–112.

- [20] Q.Cai, J.K.Aggarwal, Tracking human motion in structured environments using a distributed-camera system, *IEEE Trans. Pattern Anal. Machine Intell.* 2 (1999) 1241–1247.
- [21] P.Doubek, I.Geys, T.Svoboda, L.VanGool, Cinematographic rules applied to a camera network, in: *Proc. of Omnivis, The fifth Workshop on Omnidirectional Vision, Camera Networks and Non-classical Cameras*, 2004, pp. 17–30.
- [22] A.Rahimi, B.Dunagan, T.Darrell, Simultaneous calibration and tracking with a network of non-overlapping sensors, in: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2004)*, 2004, pp. 187–194.
- [23] K. Etemad, R. Chellappa, Discriminant analysis for recognition of human face images, *J. of the Optical Society of America A* 14 (1997) 1724–1734.
- [24] M. Berg, M. Kreveld, M.Overmars, O.Schwarzkopf, *Computational Geometry*, Springer, 2000.
- [25] H. Gindy, D.Avis, A linear algorithm for computing the visibility polygon from a point, *J. of Algorithms* 2 (1981) 186–197.
- [26] E. W. W. et al., Star convex, From MathWorld–A Wolfram Web Resource.
URL mathworld.wolfram.com/StarConvex.html
- [27] L. A.Wolsey, *Integer Programming*, Wiley-Interscience, 1998.
- [28] M. P.Rauss, P.J.Philips, A.T.DePersia, Feret (face recognition technology) program, in: *25th AIPR Workshop: Emerging Applications of Computer Vision*, 1996, pp. 253–263.
- [29] S.Abrams, P.K.Allen, K.Tarabanis, Computing camera viewpoints in an active robot cell, *Int. J. of Robotics Research* 18 (1999) 267–285.