CAMERA PLACEMENT MEETING RESTRICTIONS OF COMPUTER VISION

A Thesis

Submitted to the Faculty

of

Purdue University

by

Sara Aghajanzadeh

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science

May 2020

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF THESIS APPROVAL

Dr. Yung-Hsiang Lu, Chair

School of Electrical and Computer Engineering

Dr. David Ebert

School of Electrical and Computer Engineering

Dr. Jennifer Neville

School of Computer Science

Approved by:

Dr. Dimitrios Peroulis

Head of the School Graduate Program

To my family.

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Yung-Hsiang Lu for his continuous guidance, care and support. I would like to express my gratitude to my committee members, Dr. Ebert and Dr. Neville. And I appreciate all that I learned from my teammates in the HELPS laboratory and cherish the time we spent together learning and working in the lab.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

SYMBOLS

α	Angle
C	Set of cameras
$L(.)$	Cost function
$\Omega(.)$	Coverage function
P	Floor plan polygon region
p	Point
R	Restriction of resolution subject to camera specifications
T	Detection and tracking task

ABBREVIATIONS

AOV Angle of View

FOC Field of Coverage

FOV Field of view

PPF Pixels Per Foot

ABSTRACT

Aghajanzadeh, Sara M.S., Purdue University, May 2020. Camera Placement Meeting Restrictions of Computer Vision. Major Professor: Yung-Hsiang Lu. Professor.

In the blooming era of smart edge devices, surveillance cameras have been deployed in many locations. Surveillance cameras are most useful when they are spaced out to maximize coverage of an area. However, deciding where to place cameras is an NP-hard problem and researchers have proposed heuristic solutions. Existing work does not consider a significant restriction of computer vision: in order to track a moving object, the object must occupy enough pixels. The number of pixels depends on many factors (How far away is the object? What is the camera resolution? What is the focal length?). In this study we propose a camera placement method that identifies effective camera placement in arbitrary spaces, and can account for different camera types as well. Our strategy represents spaces as polygons, then uses a greedy algorithm to partition the polygons and determine the cameras' locations to provide desired coverage. The solution also makes it possible to perform object tracking via overlapping camera placement. Our method is evaluated against complex shapes and real-world museum floor plans, achieving up to 85% coverage and 25% overlap.

1. INTRODUCTION

Placing surveillance cameras automatically is considered to be a challenging task and has been proven to be an NP-hard problem [1]. In the past, both heuristic and approximation algorithms [2] are studied extensively to resolve the issue of automated camera placement. Most of the previous approaches are either limited to the trade-offs between coverage and costs or the prior knowledge given by security experts [3].

Particularly, the limitations of computer vision is often neglected within the design of automated camera placement. For instance, in an object identification application, objects need to occupy a certain number of pixels for being successfully identified via computer vision [4]. In other words, neglecting the needs of computer vision within the design of automated camera placement could result in poor performance for applications using computer vision technologies. Such an observation motivates this study to propose a greedy solution to achieve a functional balance between the coverage, resolution, and the requirements of computer vision applications. The difficulty of this paper lies in *how to identify and consider the computer vision requirements within the designs of automated camera placement.*

The problem of automated camera placement is a variation of the art gallery problem [5], which is a well-studied visibility problem in computational geometry. Given the layout of a museum, such as the Louvre, as illustrated in fig 1.1, the goal of art gallery problem is to monitor a gallery with the minimum number of guards so that the whole floor plan can be observed by guards.

The same concept also applies to the problem of automated camera placement, as the goal is to monitor an area with the minimum number of surveillance cameras. The difference is that each camera has a different field of coverage (FOC) according to the camera type. For example, the more expensive omnidirectional or pan-tilt-zoom

Fig. 1.1.: Floor plan of Louvre Abu Dhabi museum is used for the evaluation of our method (`source: https://www.archdaily.com`).

(PTZ) camera can provide larger coverage with fewer cameras, while more static cameras, while cheaper, are required for covering the same area.

Large public areas, such as museums, airports, and shopping malls, are monitored by hundreds, if not thousands, of security cameras [6]. Aimed to reduce human intervention, smart edge devices equipped with cameras are utilized in automated surveillance systems to gather visual data and process the data with computer vision techniques, such as object detection or tracking [7–9]. Erdem et al. [10] note that placing surveillance cameras at the proper locations is important for effective object detection and tracking. Existing work on camera placement assumes that a surveillance camera can see infinitely far away (similar to the original "art gallery problem"). Realistically, computer vision is ineffective when objects are too far away and too small [4]. Even human eyes end up having to squint to see things far away! Most tracking applications struggle to detect objects at low resolutions (below 10 pixels per foot) [11]. Generally speaking, when the number of pixels of a bounding box is less than 400, the resolution is considered low [12]. More importantly, visibility is not sufficient for automated persistent tracking.

This restriction is further complicated by the fact that surveillance cameras come in all types: differing focal lengths and camera resolutions all have an impact. The camera placement problem has been proven to be an NP-hard problem; thus, instead of seeking an optimal solution, the placement techniques seek an approximate near-optimal solution [13].

This paper proposes a fast algorithm to determine where to place surveillance cameras to achieve good coverage of a space. This novel algorithm accounts for the restrictions of computer vision by considering the *effective* field of coverage (FOC) of a camera based on the camera's specifications and the effective range (distance from camera) required to successfully identify and track objects. This algorithm accepts a polygon representation of the space and divides the area into smaller polygons of the same size, equal to the camera's field of coverage.

Since each subpolygon may have an arbitrary shape and the camera's field of coverage (FOC) is a triangle, the 100% coverage may not be achieved. A greedy strategy is utilized to find the camera locations that can satisfy the requirements for each subpolygon. Each subpolygon has at least one camera for surveillance. The effectiveness of the proposed greedy solution is evaluated through experiments on the real-world floor plan of the Louvre Abu Dhabi museum. The experiments show that the proposed solution has consistent results, always above 77% coverage and below 25% overlap for any n-sided polygon.

The rest of this document is organized as follows. Chapter 2 discusses the research background, related works, and research contribution. Chapter 3 states the problem and definitions. Chapter 4 describes the design of the proposed greedy solution. Chapter 5 then presents the experimental results. Finally, Chapters 6 and 7 present concluding remarks.

2. BACKGROUND AND CONTRIBUTIONS

2.1 Computer Vision and Object Tracking

To effectively monitor a wide area and track individuals, the video streams from multiple cameras are needed as well as the suitable number or type of cameras [14]. As most of existing computer vision techniques demand high-resolution video stream for offering good performance, pan-tilt-zoom (PTZ) cameras are more favorable than fixed cameras since PTZ cameras can pan their FOC in the range of 360°at a fixed position. However, PTZ cameras are more expensive than fixed cameras. Thus, this work refers to fixed cameras as a case study for deploying cameras automatically with the considerations of computer vision requirements.

Object tracking in video streams is a crucial part of computer vision and is widely used in various real-world applications, including human-computer interaction, autonomous vehicles, surveillance, and security [15]. Object tracking involves detecting, recognizing, and tracking an object in video streams. The effectiveness of object tracking is affected by noise in video streams, such as illumination and scale changes, and occlusion. In addition, most tracking algorithms have difficulty detecting objects at extremely low resolutions, such as 10 pixels per foot (PPF) [11]. An example is given in fig. 2.1. In fig. 2.1 (a), the object in the bounding box has at least a resolution of 20 pixels per foot, thus the object of interest can be tracked easily. In fig. 2.1 (b), the object is too far away from the camera and has a resolution below 10 pixels per foot, making it difficult to track effectively.

As the authors state [16], visibility is not sufficient for automated persistent tracking, and some overlap between the cameras' FOCs should be secured for a successful handoff. As a result, a properly placed camera network can produce significant benefits for subsequent computer vision tasks, such as tracking [17].

(a) (b)

Fig. 2.1.: Comparison of the number of pixels per object used by an object tracker.

With above observations, *the proposed greedy solution in this study aims to provide overlapping FOCs between adjacent cameras as to achieve an effective balance between coverage, resolution, and tracking requirements.*

2.2 Automated Camera Placement Problem

The problem of automated camera placement has captured the attention of the research community for quite some time. Horster et al. [18] have determined the camera positioning such that coverage is maximized for different camera types as well as a single camera type. Bisagno et al. [19] have proposed the use of reconfigurable cameras that can dynamically adapt their field of view (FOV), resolution, and position. The goal of Bisagno's study was to determine coverage and target resolution by focusing attention on critical areas of a crowd while ensuring an acceptable level of attention on less critical areas, resulting in a trade-off between coverage and resolution. Similarly, Yabuta et al. [13] have proposed a method for camera placement considering camera specifications, such as visual distance, visual angle, and resolution. Yabuta's method involves dividing a scene into regions and weighting the regions according to importance. Here, there is a trade-off between coverage and cost (*i.e.*, number of cameras). And Altahir et al. [20] propsed a dynamic programming solution that relies on human experts to determine camera locations, thus, camera placement is not fully automatic.

Table 2.1 presents the specifics of proposed solutions with no comparison purposes because the solutions are applicable in different use cases. Table 2.2 compares the related work. To our knowledge, this is the only work that combines computer vision restrictions, real-world floor plans (arbitrary shape polygons: regular, irregular, convex, concave, orthogonal, non-orthogonal), and automated placement into one system.

Table 2.1.: The specifics of various proposed solutions

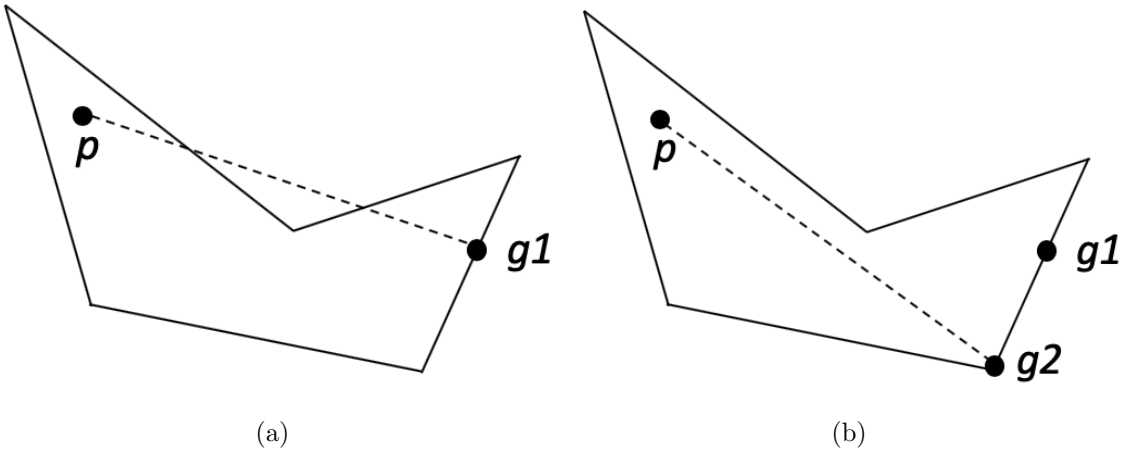
Technique	Scene	Dimension	FOC Layout	Camera Type	Coverage	Overlap
Erdem et al. [10]	Cartesian grid	2D	Triangle and circle	Fixed and omnidirectional	Limited	Variant
Horster et al. [18]	Simple rectangle region	2D	Triangle	Not real camera	Limited	Variant
Bisagno et al. [19]	Cartesian grid	3D	Cone and sphere	PTZ, Omnidirectional	Full	Variant
Yabuta et al. [13]	Simple rectangle region	2D	Triangle	Not real camera	Limited	Variant
Altahir et al. [20]	Cartesian grid	2D	Trapezoid and Triangle	Fixed	Limited	Variant
Proposed Solution	Cartesian grid	2D	Triangle	Fixed	Limited	Variant

Table 2.2.: Comparisons of related works and the proposed solution

Technique	Computer Vision Considerations	Complex (real-world) Floor Plans	Fully Automated Placement
Erdem et al. [10]	✓		✓
Horster et al. [18]			✓
Bisagno et al. [19]			✓
Yabuta et al. [13]			✓
Altahir et al. [20]		✓	
Proposed Solution	✓	✓	✓

2.3 Art Gallery Problem

The camera placement problem is part of a group of visibility problems referred to as the art gallery problem (AGP); here, an art gallery is represented by a polygon, and guards are represented by points in a set G. However, it is also possible to place the guards anywhere inside the polygon. The art gallery problem is solved when guards are placed such that every point inside the polygon is visible from at least one guard. Fig. 4.2 illustrates an example of AGP.



(a) (b)

Fig. 2.2.: (a) A polygon without full coverage because guard g1 cannot see point p. (b) A polygon with full coverage when guard g2 is added, but it is not an optimal solution as two guards here are redundant.

To the authors' knowledge, no existing solution has been developed that can provide an optimal solution (i.e., using the least number of cameras) to *fully* cover polygons of arbitrary shapes. As stated previously, AGP is np-complete because a suggested solution can be verified in polynomial time, but finding the optimal solution is only achievable if all possible solutions are tested. The majority of theoretical solutions assume unlimited field of coverage and infinite visibility; thus, existing theoretical works cannot be applied in a real deployment [10]. Prior heuristic works reduce the problem based on restrictions and relevance to various applications.

Although a significant amount of research has been conducted in the theoretical aspects of the problem, few studies have been devoted to automated placement with the consideration of vision technologies. In fact, automation could be useful in cases of frequent change in deployment plans. This paper proposes a greedy solution for determining an effective placement of cameras for monitoring an area with a target resolution sufficient for computerized tracking of individuals.

2.4 Contributions

This work has the following major contributions. First, it presents a solution for placing cameras such that vision tasks like tracking can be performed effectively. To bridge the gaps in the current evaluation platforms, this paper introduces the *minimum resolution threshold* as a new metric to consider computer vision requirements. Second, this is the first solution for camera placement considering vision requirements with the ability to handle complex (real-world) floor plans. The proposed solution converts floor-plans into a 2-dimensional polygons for automatic camera placement. A limitation to most prior works is that they are only used on polygons with orthogonal angles. The proposed solution can handle such a drawback and accept arbitrary shapes of polygons, convex, concave, orthogonal, non-orthogonal, except self-intersecting polygons (not applicable in majority of locations). Third, the paper demonstrates that the solution is effective for fulfilling the coverage and tracking requirements of computer vision tasks. The solution considers camera specifications and then automatically determines the position and directions of the cameras. The overall coverage and amount of overlap achieved is also derived. The evaluation shows this method has consistent coverage, approximately 77% of the total area.

3. PROBLEM DESCRIPTION

As stated previously [1], the solution for 100% coverage with optimal number of cameras is computationally intractable, so this paper does not produce full coverage. Instead, this work aims to achieve a balance between coverage and resolution.

3.1 Definitions

In the camera placement problem, several factors need to be considered. We define and discuss these factors as follows.

- *Fixed camera* sometimes referred to as static camera has a fixed position and orientation after being mounted.
- *Camera FOC* depends on the lens type and image sensor size. A normal value for the horizontal angle of view is around 50°but it can range up to 110°for specific cameras and give us a wider FOC.
- *Coverage* is defined as the percentage of the polygon area covered by the camera FOC. The amount of coverage differs from domain to domain; for example, some places may choose 100% coverage threshold while others may not.
- *Resolution* is similar to coverage in terms of domain requirements, but there are often constraints on how small an object monitored is allowed to be. Resolution is one of the many factors that can affect tracking success.

3.2 Camera Specifications

A fixed-lens surveillance camera has a set viewing angle, and the focal length of the camera is permanently set. Such cameras are comparatively less expensive and

are thus suitable options for locations in which the floor plan is not likely to change. First we must understand that field of view (FOV) and angle of view (AOV) are different and we can see that clearly in fig. 3.1.

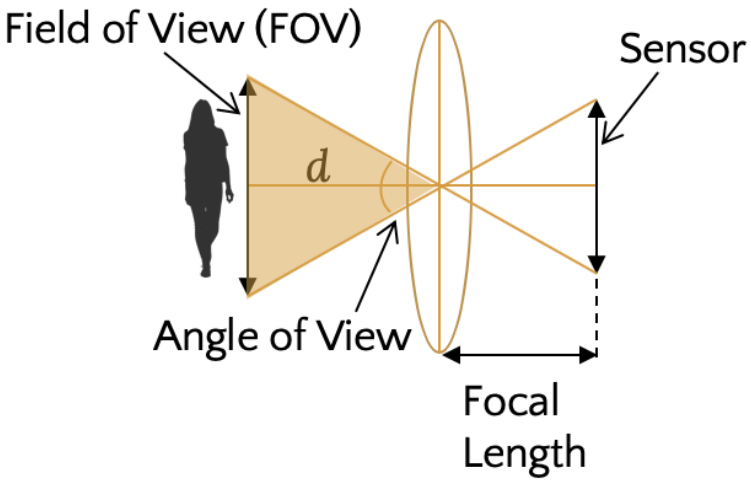


Fig. 3.1.: Camera model specifications: angle of view (AOV) α and distance d limited for resolution requirement and field of view (FOV) $= \frac{\text{Horizontal camera resolution}}{\text{PPF}} = \text{FOV}$.

AOV is affected by the focal length of the lens and the size of the sensor in the camera. The smaller the focal length of the lens is, the greater the AOV is, as shown in fig. 3.2. In addition, the size of the image sensor has an impact on the AOV. For

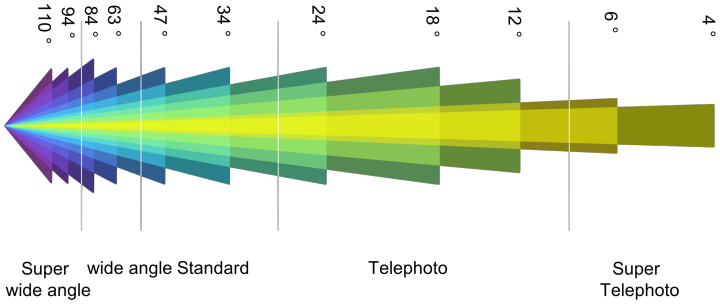


Fig. 3.2.: Difference in angle of view for different focal lengths [21].

example, a 1/3" image sensor provides a wider viewing angle than a 1/4" image sensor. Another important factor is camera resolution. Although the camera's viewing angle remains constant, its FOC can change according to the required distance of objects

being tracked (i.e. PPF threshold). Based on the observation from Section 1, this paper sets a threshold of 20 PPF. Given the threshold, we determine the distance from a camera to a target individual that is required for successful detection and tracking. We use Equation 3.1 to compute this distance.

$$\text{distance} = \frac{\text{focal length} \times \text{horizontal camera resolution}}{\text{chip width} \times \text{resolution threshold}} \tag{3.1}$$

Table 3.1 illustrates the difference in distance produced by a 720 HD camera and a 1080 HD camera. The camera distance in Table 3.1 can be interpreted as follows. To maintain a 20 PPF resolution at 56 ft from the camera, a 2.8 mm lens is required. The angle of view is the angle that can be expected when the camera has a 1/3" image sensor with 4.8mm chip width, and a given focal length.

Table 3.1.: The camera distance is calculated using Equation 3.1. The coverage area is the area of a triangle that is calculated using $Area = \frac{1}{2} \times b \times h$, where h is the camera distance and b is the maximum horizontal field of view calculated using $\frac{\text{Horizontal camera resolution}}{\text{PPF}} = \text{FOV}$ for each camera type.

Focal Length	Angle of View	Camera Distance (1920 x 1080)	Camera Distance (1280 x 720)	Field of Coverage Area(in square ft)
2.8 mm	109°	56 ft	37.3 ft	2688
3.6 mm	74°	72 ft	48 ft	3456
4.0 mm	67°	80 ft	53 ft	3840
6.0 mm	42°	120 ft	80 ft	5760
8.0 mm	32°	160 ft	106.66 ft	7680
12.0 mm	22°	240 ft	160 ft	11520
25.0 mm	11°	500 ft	333.33 ft	24000

3.3 Problem Formulation

The problem is formulated in terms of a planar region, which is typical of a building floor plan, denoted by P. In this paper, the term area denotes a physical 2D space. The most commonly used scene representation method is a Cartesian grid of points in a 2D plane [22]. A third dimension would add great complexity to the algorithm and would not be of use in most polygon areas. Recent sensor placement studies formulate the problem in a 2D space; that is, cameras are usually deployed

in a 3D space to monitor a 2D plane [23]. In addition, a 3D space is used when PTZ cameras are included; whereas, we consider fixed cameras, as they are less costly. According to Horster et al. [18], there can occur practical situations where regions do not restrict the camera FOC; for example, a table in an office environment. Thus, in this study, static obstacles are not considered. In terms of walls and blocks, we would be aware that at least one camera is required per location that is separated by a wall from another location (*e.g.*, two cameras are required for two separate rooms).

Letting detection and tracking tasks be denoted by T and restrictions including resolution and distance be denoted by R, and given coverage function $\Omega(.)$, the requirement for our solution would be to place a set of cameras C in region P satisfying restrictions R by using an effective number of cameras, denoted by cost *L(.)*. Mathematically, we can formulate the camera placement problem as follows:

$$\underset{C}{\operatorname{argmin}} L(C) \quad | \quad \bigcup_{c \in C} \Omega(c, \alpha, d, r) \tag{3.2}$$

where $\Omega(c, \alpha, d, r)$ is $\{\, p \in P$ where point p is visible from camera c with viewing angle α facing objects within distance d satisfying spatial resolution $r. \,\}$

3.4 Flow of Camera Placement

Fig. 3.3 presents a flowchart of the proposed approach. The input to the algorithm are the desired floor plan as a mesh of 2-dimensional grid points, producing a mapping between them to create the original polygon, and the camera specifications. The main computational steps are: 1) Initialization procedure - computing camera FOC based on the camera model, and splitting the polygon into equal areas based on the FOC; and 2) Placement procedure - placing cameras using a greedy strategy in each subpolygon. The output of the program provides the total number of cameras, their locations, % covered, and % overlapped.

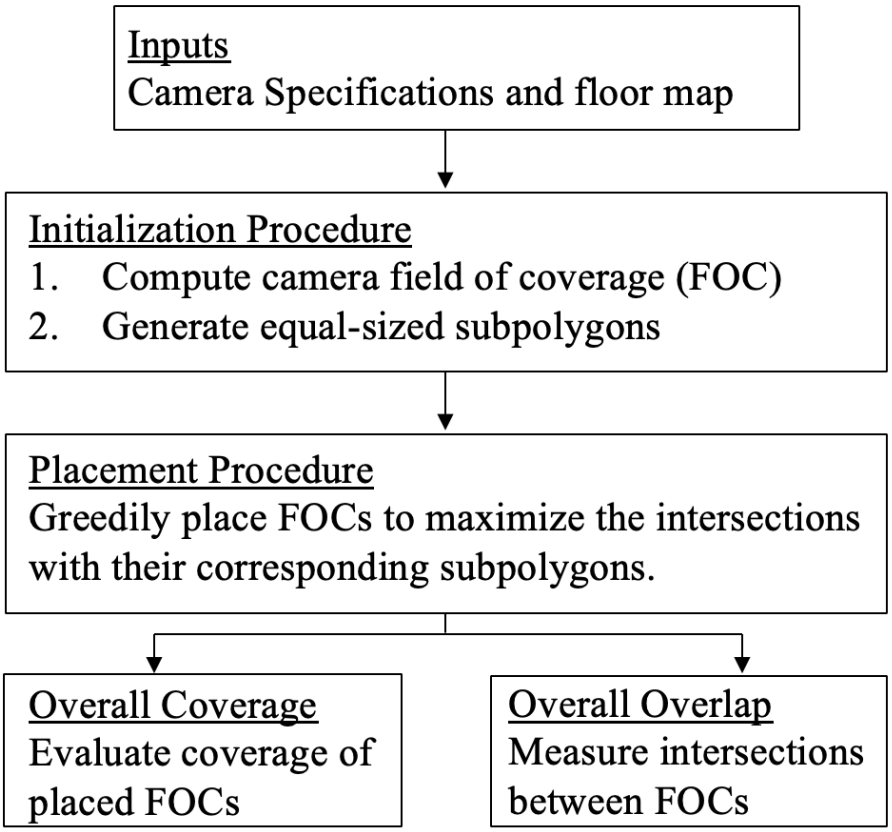


Fig. 3.3.: Flow chart of the proposed approach.

4. AUTOMATIC CAMERA PLACEMENT MEETING RESTRICTIONS OF COMPUTER VISION

4.1 Overview

In this paper, one goal is to bridge the gap between the theoretical computational geometry and more realistic requirements of computer vision involving real cameras. To achieve this goal, we offer a new method for automatic camera placement by considering both the resolution and inter-camera coverage metrics of computer vision requirements. The space is then divided into regions, denoted as subpolygons, of equal size subject to camera FOC. The algorithm begins by placing one camera per subpolygon and then measuring the overall coverage and overlap. The algorithm requirement is to produce effective coverage of the region while considering the limitations of computer vision.

4.2 Initialization Procedure

4.2.1 Find Field of Coverage (FOC)

Given a pixels per foot (i.e. in our case PPF is 20) threshold, we can determine the distance from a camera to a target individual that is required for successful tracking. The input to this module would be camera model specifications including camera viewing angle and focal length. Given the distance, we can compute the camera's FOC, whose area is a triangular shape, as illustrated in fig. 3.1. The complexity of this module is O(1) with output camera FOC and area that will be used in the next module to generate equal-sized subpolygons.

After converting a floor plan into a 2-dimensional n-sided polygon, it is theoretically possible to consider all grid points as possible camera positions; however, it

is not practical or efficient due to the increased computational overhead. Our novel approach involves splitting the space into equal regions based on the camera FOC, which is the coverage (meeting restrictions of computer vision) provided by a single camera and is derived in advance from the cameras' specifications. Thus, each divided subpolygon is created with the same area. However, there can be a remainder when dividing the original polygon into subpolygons of equal areas. This occurs because the number of subpolygons created with the same area may not equal the entire area of the original polygon. We use Equation 4.1 to determine the area of the main polygon. It is the Gaussian formula used to determine the area of any regular, irregular, convex, or concave (but not self-intersecting) n-sided polygon whose vertices are given by their Cartesian coordinates in the plane.

$$\text{Area}_{\text{polygon}} = \left| \frac{(x_1 y_2 - y_1 x_2) + ... + (x_n y_1 - y_n x_n)}{2} \right| \tag{4.1}$$

where x_i and y_i are the coordinates of a vertex. Next, we use the algorithm in [24], offering a closed-form solution to splitting a polygon into any number of equal areas, to divide the main polygon into n subpolygons with areas equal to $\frac{\text{Area}_{mainPolygon}}{n}$. The process is displayed in fig. 4.1.

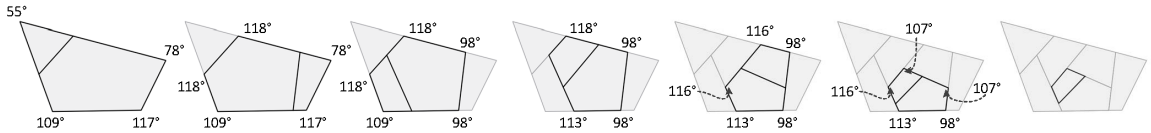


Fig. 4.1.: Steps showing the original polygon (far left) being split into subpolygons with equal areas. The area is based on the calculated FOC subject to camera viewing angle α and distance d required to maintain successful tracking.

4.2.2 Equal-Size Subpolygon Generation

The algorithm [24] solves the problem of splitting polygon into equal-sized subpolygons while ensuring minimum length of line based cuts. Below, we will summarize the key aspects of the algorithm, which is open-source and used as a a starting point

for the greedy camera placement algorithm in this work. The algorithm applies a divide and conquer methodology by first calculating the area of the main polygon, $\text{Area}_{mainPolygon}$, and the target area of each of the n subpolygons, hence, $\frac{\text{Area}_{mainPolygon}}{n}$. Second, splitting the main polygon into two subpolygons - smaller one with area of $\frac{\text{Area}_{mainPolygon}}{n}$, and bigger one with area of $\frac{(n-1)}{n}\text{Area}_{mainPolygon}$, and last, continue using the same approach on the bigger sub polygon until it reaches the desired area.

Base Case with 4-sided polygon

- The cut is a line, so it would start from one edge and end at another edge. For instance, a base case 4-sided polygon would have 6 possible edge pairs, $\binom{4}{2}$, 3 of which are shown in fig. 4.2 (a).
- The problem is reduced to making a cut on a given edge pair in order to split the main polygon into subpolygons. Let's say we would split with line from edge AD to edge BC.
  - The bisector m of the edge pair is determined, shown in fig. 4.2 (b).
  - At an angle perpendicular to the bisector m, we locate the projected vertices on their corresponding opposite edge (i.e., point G and point H). We ignore those that don't lie on the opposite line segments (i.e., point A and point D). Points B and C are then connected to their corresponding valid projection points H and G, shown in fig. 4.2 (b).
  - As a result, the polygon is now a combination of 2 triangles and 1 trapezoid. Thus, if the minimum cut exists, it must lie within these triangles or the trapezoid, and we can find out where it lies by a simple area check. Given Area_{AHB} and Area_{CDG} and Area_{HGCB} , the minimum cut would lie in
    - left triangle if $\frac{\text{Area}_{mainPolygon}}{n} < \text{Area}_{AHB}$,
    - right triangle if $\frac{\text{Area}_{mainPolygon}}{n} < \text{Area}_{AHB} + \text{Area}_{HGCB}$,

  - trapezoid if $\frac{\text{Area}_{mainPolygon}}{n} > \text{Area}_{AHB}$ and $\frac{\text{Area}_{mainPolygon}}{n} < \text{Area}_{AHB} + \text{Area}_{HGCB}$

- If the minimum cut lies in the left triangle, one end of the line lies on the vertex B and the other end can be found by linear interpolation of points A and H based on the target area. Since the area of the subpolygon is $\frac{\text{Area}_{mainPolygon}}{n}$, the interpolated point will be $A + \frac{\text{Area}_{mainPolygon}}{n \times \text{Area}_{AHB}}(H - A)$, as shown in fig. 4.2 (c).
- If the minimum cut lies in the trapezoid, then linear interpolation based on the target area would be used to find the two points of the split line. In the case of the trapezoid the end points are not fixed as in the case of triangle where one end point was fixed as a vertex. However, to meet the minimum length requirement, the split line must always be perpendicular to the bisector of the edge pair. Consequently, one end point is the linear interpolation between points H and G, and the other is the interpolation between points B and C, as shown in fig. 4.2 (d).

General Case with n-sided polygon

For a given edge, we check the remaining edge to determine a potential split line. For instance, in the case of concave polygons, a potential cut line could fall outside the polygon or intersect with polygon edges. In such cases, a verification step is considered to ignore those potential split lines. The approach to determining the minimum cut line for a given edge pair is easily extended to the general n-sided case.

Time Complexity

The iterative approach takes the minimum cut at each step and ignores the other possibilities, thus the results will be fast and has cubic time complexity for convex polygons because checking every edge pair results in quadratic complexity, and for

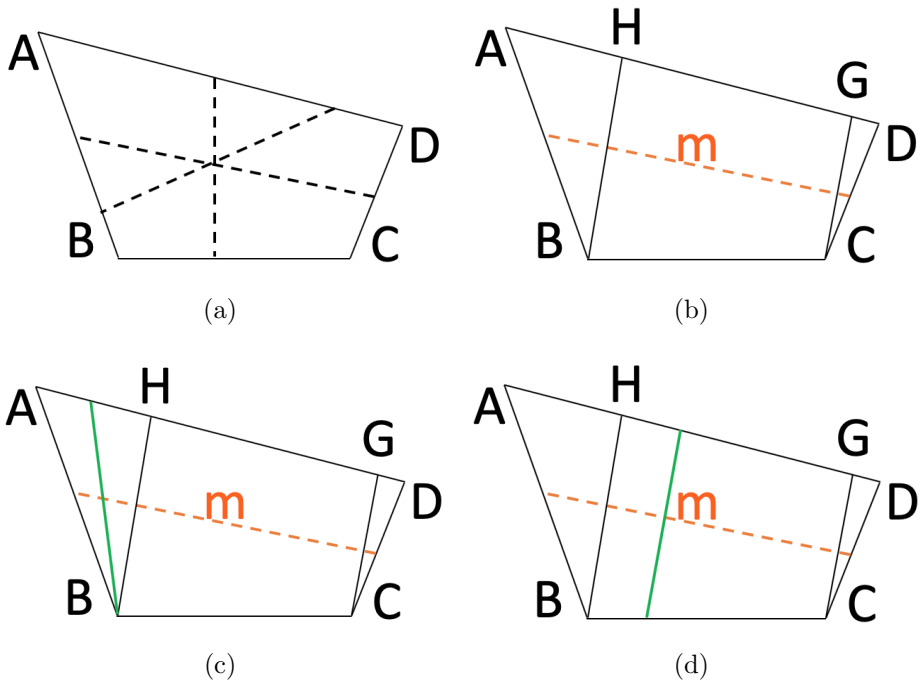


Fig. 4.2.: (a) Potential cuts: from edge AD to edge AB, from edge AD to edge BC, from edge AB to edge DC.
(b) Projection of B is H, and projection of C is G.
(c) Minimum cut (green) for a given edge pair lies in the left triangle, target area $\frac{\text{Area}_{mainPolygon}}{n} < \text{Area}_{AHB}$.
(d) Minimum cut (green) for a given edge pair lies in the trapezoid, target area $\frac{\text{Area}_{mainPolygon}}{n} > \text{Area}_{AHB}$ and $\frac{\text{Area}_{mainPolygon}}{n} < \text{Area}_{AHB} + \text{Area}_{HGCB}$.

a given edge pair, the algorithm involves calculating area and performing linear interpolation to determine split lines. For concave polygons, an additional verification step mentioned in section 4.2.2 leads to quartic time complexity.

4.3 Placement Procedure

Next, for every vertex of the subpolygons, the level of intersection between the camera's FOC and the subpolygon is computed, and the FOC is rotated such that it lies inside the polygon and provides the best intersection between the FOC and its corresponding subpolygon. Fig. 4.4 summarizes the various possibilities for the rotated FOC. The input to this module is the camera FOC, and it will place the

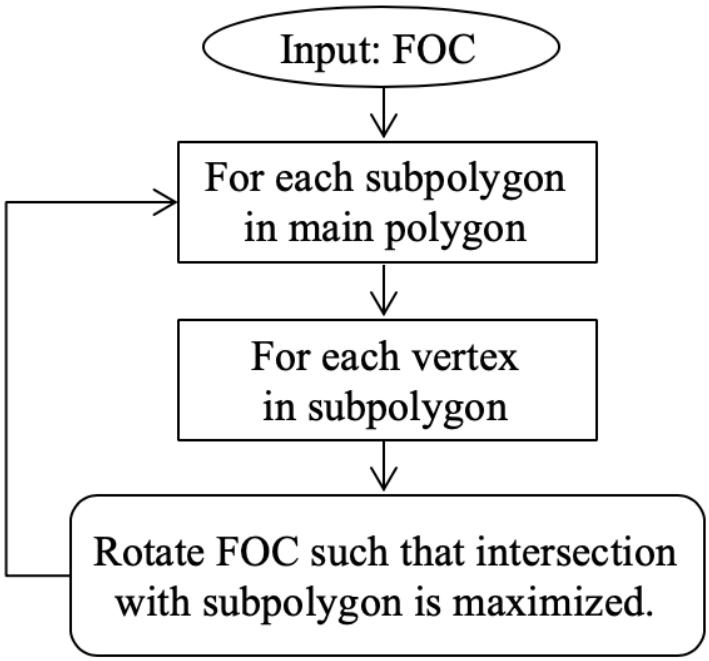


Fig. 4.3.: Flow chart 3.3 - Greedy Camera Placement Procedure

camera FOC in a greedy manner at a possible candidate vertex of the subpolygons. For every possible position in each subpolygon, the algorithm calculates the highest intersection of the current FOC (*e.g.*, camera placed at that vertex) and adds it to a local placement list. Before proceeding to the next subpolygon, it obtains the FOC that results in the maximum coverage among elements of the local placement list and adds the final FOC and its position to the overall placement list. The algorithm repeats the procedure for all subpolygons. The complexity of this module is $O(n^2)$. In the end, each subpolygon has a camera FOC placed in *one* of its vertices. Thus, each subpolygon has one camera as its effective candidate located in the best position that provides the highest coverage for each subpolygon and effective coverage for the original polygon. The total number of cameras is equal to the total number of subpolygons.

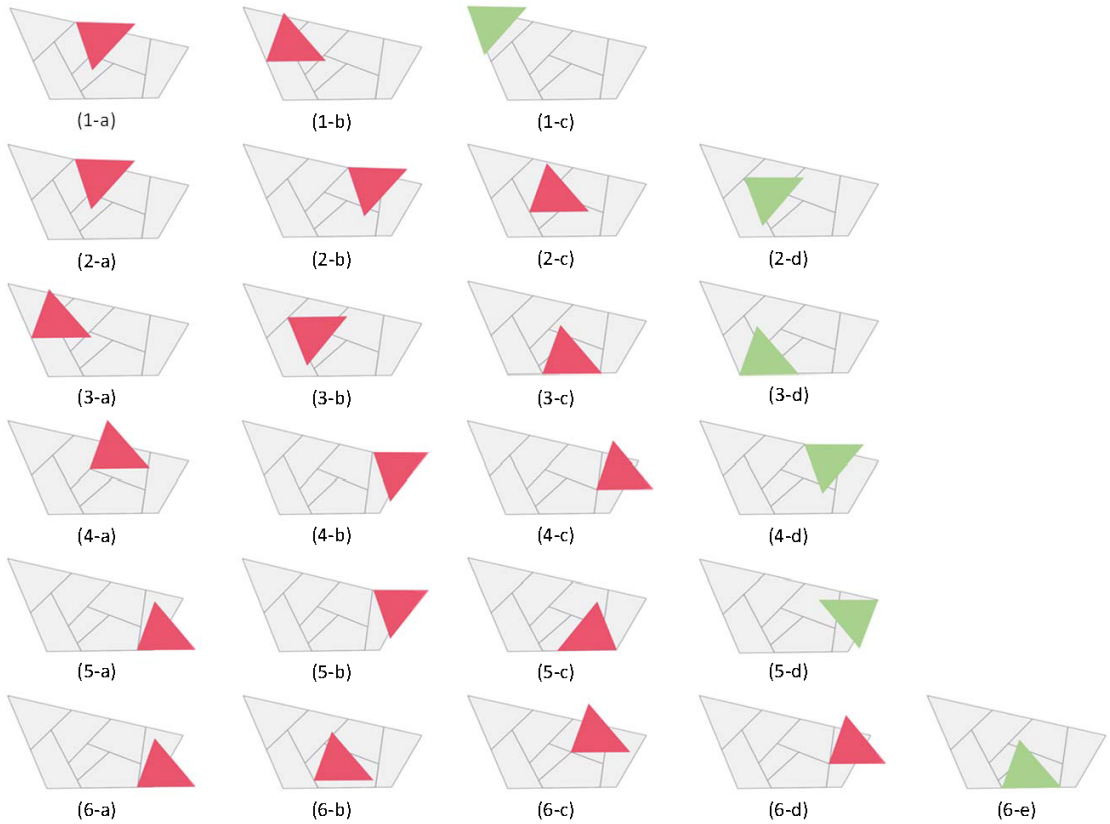


Fig. 4.4.: For every subpolygon, the solution examines each vertex [a-e] and computes the coverage gained by placing camera FOC at that vertex. The vertex with the highest coverage score is chosen for each subpolygon. Red color designates ineffective placements whereas green color designates the effective placement. As an example, (1-a) camera FOC does not intersect with its own subpolygon at all, thus the solution proceeds to the next vertex (1-b), where camera FOC intersects with the subpolygon over a small area. The solution proceeds to the next vertex (1-c), where highest intersection is achieved. So, it picks this vertex. The same procedure continues for all the other subpolygons until each subpolygon has at least one FOC placed at one of its vertices. Please note this figure only shows some possibilities rather than a full 360 rotations for all vertices.

4.4 Measure Overall Coverage

This module will compute the total area of coverage achieved by the cameras' FOCs. The module receives as input the Cartesian coordinates of the main plane, as well as the list of camera FOCs obtained in 4.3. The module has 3 main variables: number of points inside the main polygon, number of points covered inside the main

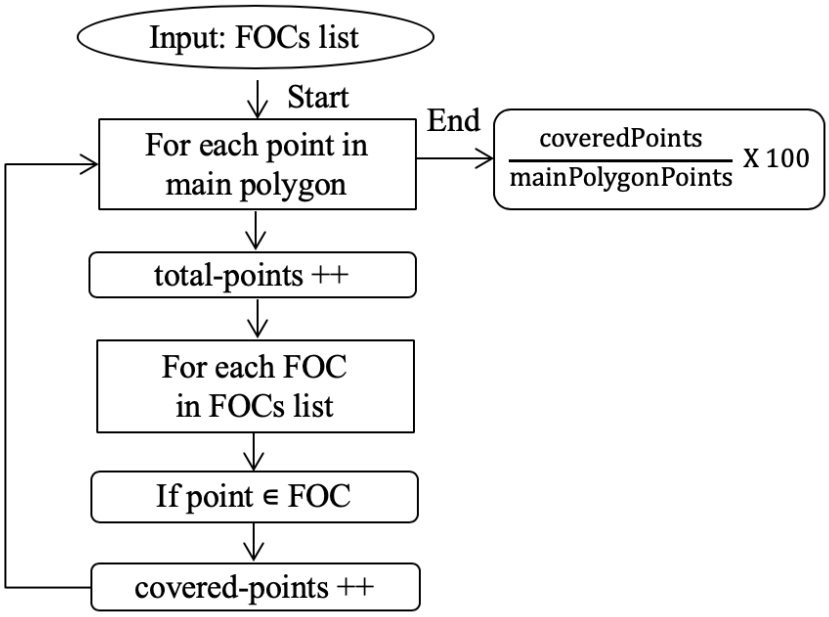


Fig. 4.5.: Flow chart 3.3 - Overall Coverage

polygon, and the coverage score. It starts by initializing all variables to 0 values. For each pixel point inside the main polygon, it increments the number of points inside the main polygon variable by 1. And it checks whether the point falls inside a camera FOC by iterating through the camera FOCs list. For each point inside the main polygon, it verifies whether the point falls inside a camera FOC. If yes, it increments the number of points covered inside the main polygon variable by 1. By the end of this procedure, the algorithm has captured all pixel points inside the main polygon that are covered. In the end, it returns the coverage score that is equal to $\frac{covered-points}{total-points} \times 100$. The complexity of this module is O(n^2).

4.5 Evaluate Overall Overlap

This module will evaluate the total area of overlap achieved by the cameras' FOCs. The module receives as input the list of camera FOCs. The module has 3 main variables: an arbitrary shape polygon named intersection, a list of intersections (arbitrary shape polygons), and overlap score. The module consists of a nested loop

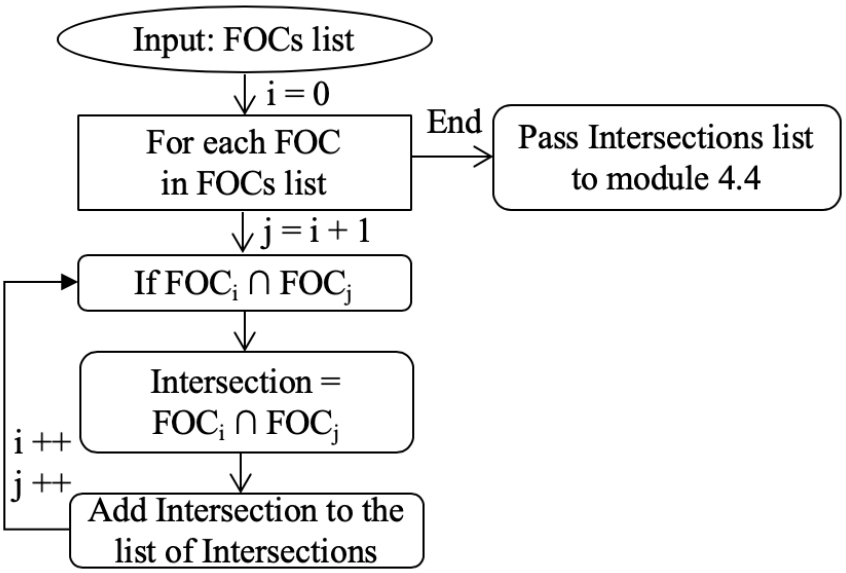


Fig. 4.6.: Flow chart 3.3 - Overall Overlap

to verify whether each camera FOC intersects with other camera FOCs, and save the intersection as an arbitrary shape polygon. It appends all intersection polygons to the intersections list, and it passes the intersections list to measure overlap areas. Furthermore, it returns the overlap score computed via 4.4 module. The complexity of this module is O(n^2) in addition to the computational complexity of 4.4 module.

5. EVALUATION

The proposed greedy solution provides a functional balance between coverage and resolution requirement. The greedy approach offers relatively high efficiency and exhibits a low runtime. The computational complexity is quadratic for the subpolygons and their corresponding vertices. It seeks the candidates that satisfy the objective locally rather than identifying a global optimum. Please note that, since inputs from human experts are required for previous camera placement methods and the sources of previous camera placement methods are not available, the proposed method is compared to case scenarios where placement is ensured via best placement manually by humans. For our evaluation, we use different types of cameras and evaluate the proposed solution in two different scenarios: base case evaluation and complex case evaluation.

5.1 Base Case Evaluation

Fig. 5.1 compares the ideal solutions with 100% coverage without overlap, plotted by humans with the automated solutions. Fig. 5.1 (a) uses a 1080p HD fixed camera with angle of view $\alpha = 74°$, and focal length $f = 3.6$mm, and distance $d = 72$ ft limited for resolution requirement, respectively. Thus, the camera's FOC covers an area of 3,456 square feet. Both solution (manually and automated) use the same number of cameras (10 cameras), since the original polygon is divided based on the camera FOC. Our method provides less than 100% coverage (83% coverage) because the subpolygons created by the algorithm do not always match triangle shapes of the camera FOC; as a result, the proposed method has overlaps (9% overlap), which could be potentially useful for tracking applications during camera hand-off.

Fig. 5.1 (b) uses a 1080p HD fixed camera with angle of view $\alpha = 67°$and focal length $f = 4.0$mm and distance $d = 80$ ft limited for resolution requirement, respectively. Thus, the camera's FOC covers an area of 3,840 square feet. Both solution (manually and automated) use the same number of cameras (10 cameras), since the original polygon is divided based on the camera FOC. Our method provides 82% coverage and 12% overlap.

Fig. 5.1 (c) uses a 1080p HD fixed camera with angle of view $\alpha = 42°$and focal length $f = 6.0$mm and distance $d = 120$ ft limited for resolution requirement, respectively. Thus, the camera's FOC covers an area of 5,760 square feet. Both solution (manually and automated) use the same number of cameras (10 cameras), since the original polygon is divided based on the camera FOC. Our method provides 80% coverage and 7% overlap.

Fig. 5.2 compares the ideal solutions with 100% coverage without overlap, plotted by humans with the automated solutions. Fig. 5.2 (a) uses a 1080p HD fixed camera with angle of view $\alpha = 74°$, and focal length $f = 3.6$mm, and distance $d = 72$ ft limited for resolution requirement, respectively. Thus, the camera's FOC covers an area of 3,456 square feet. Both solution (manually and automated) use the same number of cameras (15 cameras), since the original polygon is divided based on the camera FOC. Our method provides less than 100% coverage (85% coverage) because the subpolygons created by the algorithm do not always match triangle shapes of the camera FOC; as a result, the proposed method has overlaps (14% overlap), which could be potentially useful for tracking applications during camera hand-off.

Fig. 5.2 (b) uses a 1080p HD fixed camera with angle of view $\alpha = 67°$and focal length $f = 4.0$mm and distance $d = 80$ ft limited for resolution requirement, respectively. Thus, the camera's FOC covers an area of 3,840 square feet. Both solution (manually and automated) use the same number of cameras (15 cameras), since the original polygon is divided based on the camera FOC. Our method provides 81% coverage and 19% overlap.

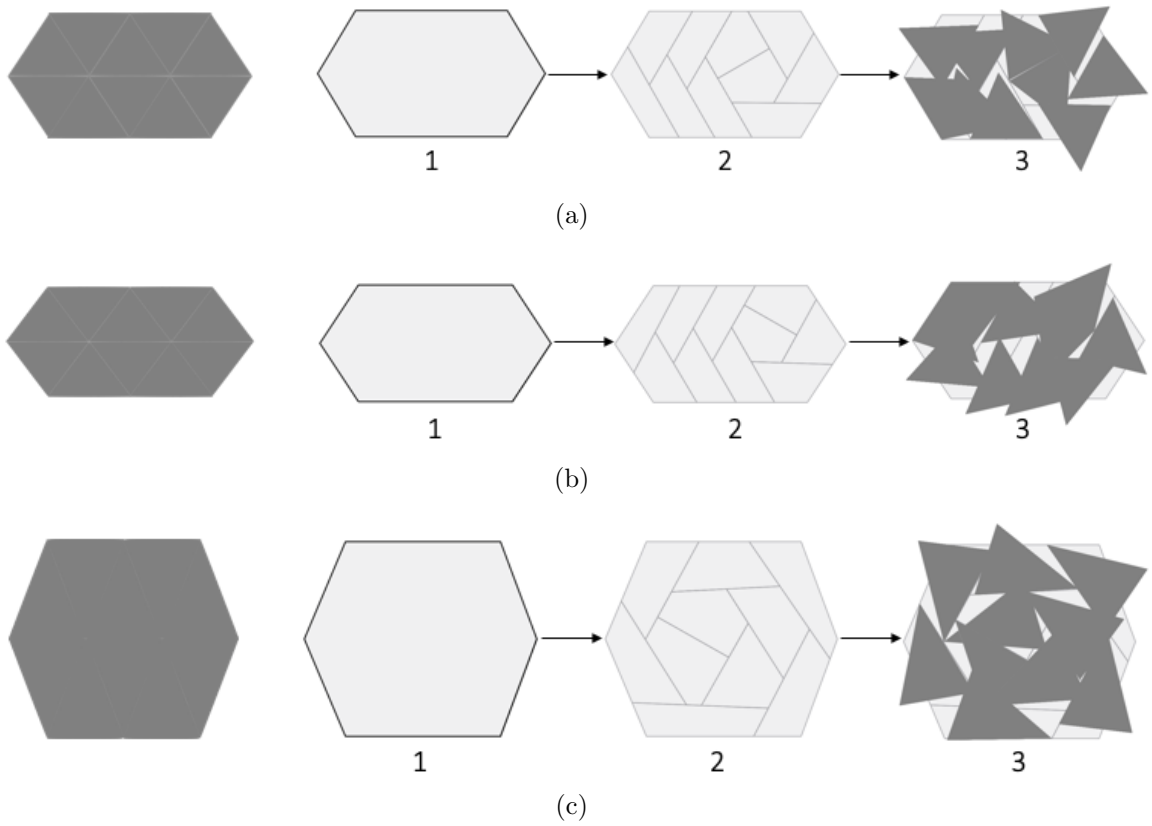


Fig. 5.1.: (Left) Best possible solution achieving 100 % coverage with 10 cameras. (Right) Our method receives the main polygon in (1) and divides the main polygon into equal areas subject to the camera FOC in (2), and greedily places the cameras in (3). (a) 10 cameras with angle of view $\alpha = 74°$and focal length $f = 3.6$mm achieves 83% coverage and 9% overlap. (b) 10 cameras with angle of view $\alpha = 67°$and focal length $f = 4.0$mm achieves 82% coverage and 12% overlap. (c) 10 cameras with angle of view $\alpha = 42°$and focal length $f = 6.0$mm achieves 80% coverage and 7% overlap.

Fig. 5.2 (c) uses a 1080p HD fixed camera with angle of view $\alpha = 42°$and focal length $f = 6.0$mm and distance $d = 120$ ft limited for resolution requirement, respectively. Thus, the camera's FOC covers an area of 5,760 square feet. Both solution (manually and automated) use the same number of cameras (15 cameras), since the original polygon is divided based on the camera FOC. Our method provides 80% coverage and 10% overlap.

Additionally, fig. 5.3 (a-f) shows the performance of the algorithm as the number of edges increase, resulting in more complex polygons and requiring more cameras.

Our solution stabilizes at approximately achieving above 75% coverage and below 25% overlap with increasing number of cameras.

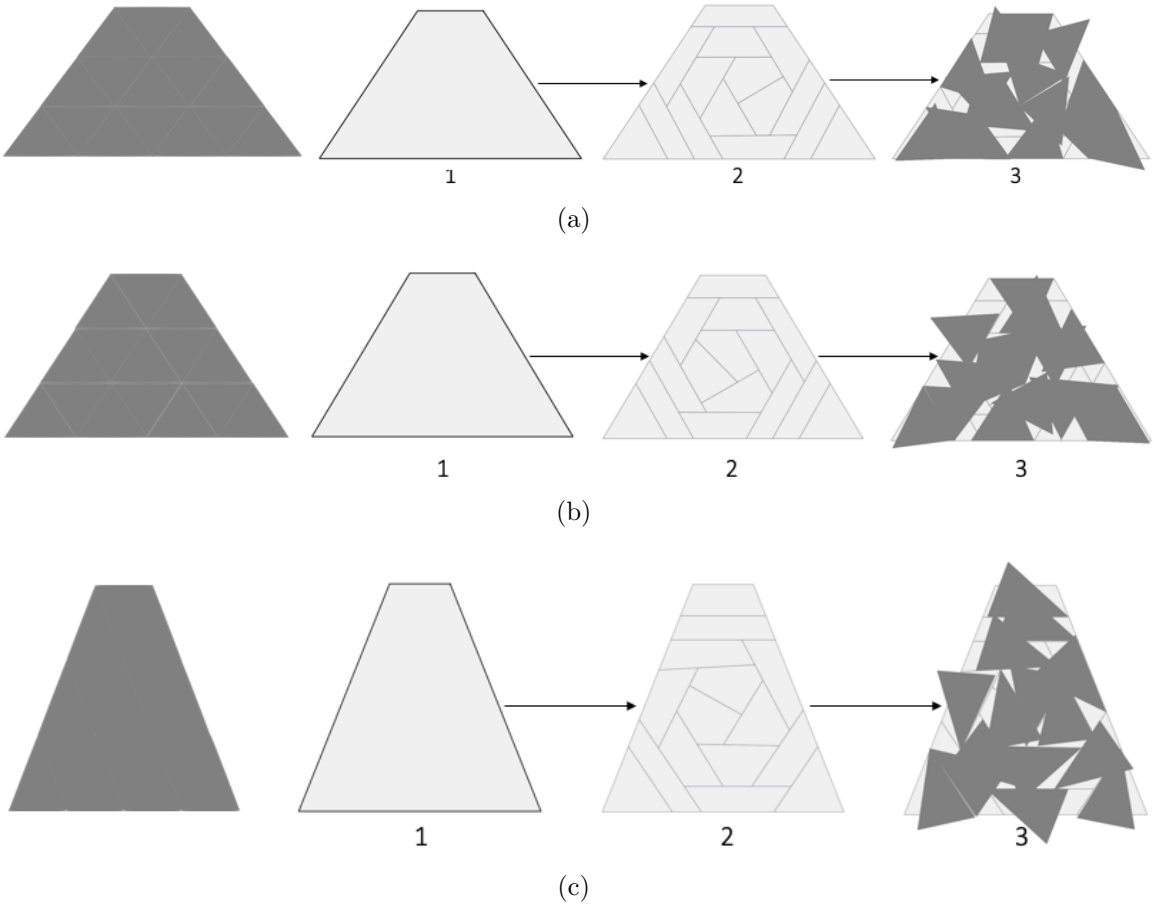


Fig. 5.2.: (Left) Best possible solution achieving 100 % coverage with 15 cameras. (Right) Our method receives the main polygon in (1) and divides the main polygon into equal areas subject to the camera FOC in (2), and greedily places the cameras in (3). (a) 15 cameras with angle of view $\alpha = 74°$and focal length $f = 3.6$mm achieves 85% coverage and 14% overlap. (b) 15 cameras with angle of view $\alpha = 67°$and focal length $f = 4.0$mm achieves 81% coverage and 19% overlap. (c) 15 cameras with angle of view $\alpha = 42°$and focal length $f = 6.0$mm achieves 80% coverage and 10% overlap.

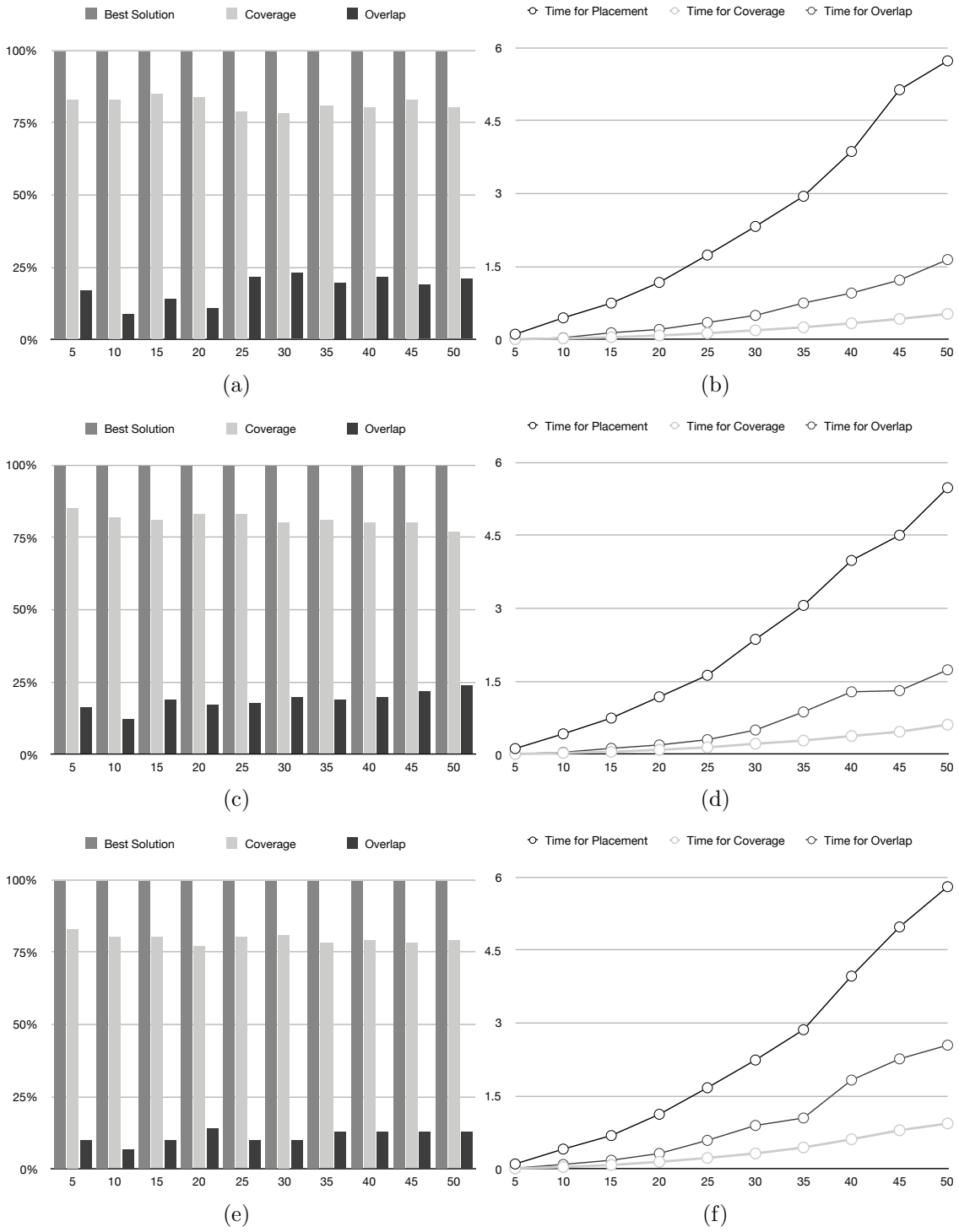


Fig. 5.3.: (Left) Evaluation of coverage and overlap with increasing number of cameras as a result of more complex polygon (more edges and vertices) (Right) Evaluation of time (in seconds) for placement, coverage and overlap calculation. (a-b) cameras with angle of view $\alpha = 74°$and focal length $f = 3.6$mm. (c-d) cameras with angle of view $\alpha = 67°$and focal length $f = 4.0$mm. (e-f) cameras with angle of view $\alpha = 42°$and focal length $f = 6.0$mm.

5.2 Complex Case Evaluation

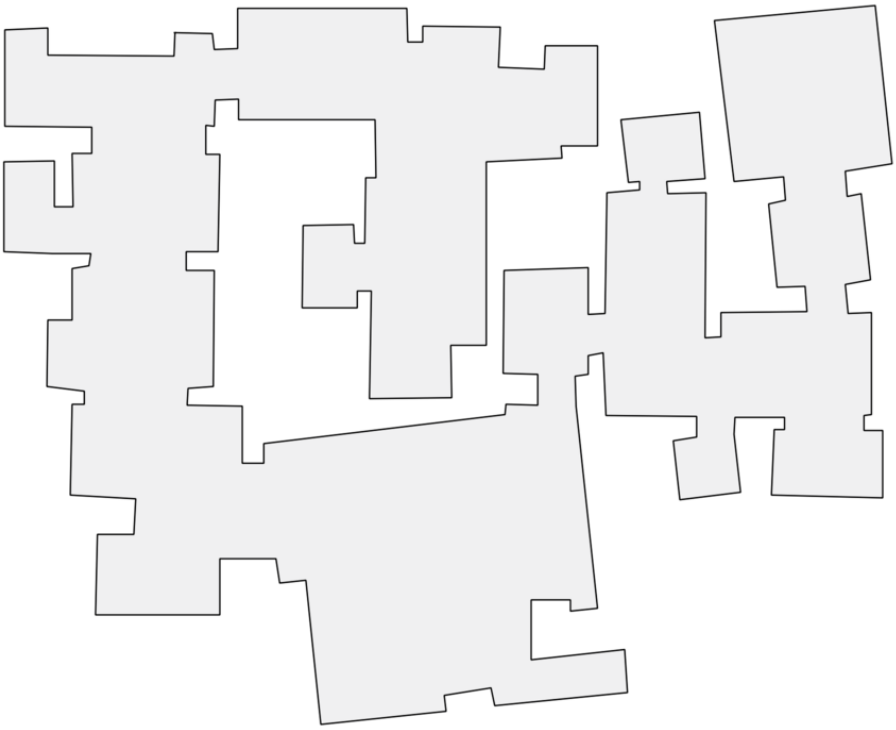
In this section, we present our approach applied to the floor plan of the Louvre museum shown in fig. 5.4 (a), and fig. 5.4 (b) shows the result of feeding the floor plan to the proposed solution. The results in Section 5.2 displays the placement of different cameras such that effective coverage is achieved and limitations of computer vision applications are addressed.

Fig. 5.5 (a) illustrates when the main polygon is divided into equal subpolygons based on the camera specifications. Fig. 5.5 (b) uses a 1080p HD fixed camera with angle of view $\alpha = 74°$, and focal length $f = 3.6$mm, and distance $d = 72$ ft limited for resolution requirement, respectively. Thus, the camera's FOC covers an area of 3,456 square feet. There are a total of 119 cameras resulting in 84% coverage and 13% overlap. Moreover, fig. 5.6 (a) illustrates when the main polygon is divided into equal subpolygons based on the camera specifications. Fig. 5.6 (b) uses a 1080p HD fixed camera with angle of view $\alpha = 67°$, and focal length $f = 4.0$mm, and distance $d = 80$ ft limited for resolution requirement, respectively. Thus, the camera's FOC covers an area of 3,840 square feet. There are a total of 107 cameras resulting in 84% coverage and 13% overlap. Furthermore, fig. 5.7 (a) illustrates when the main polygon is divided into equal subpolygons based on the camera specifications. Fig. 5.7 (b) uses a 1080p HD fixed camera with angle of view $\alpha = 42°$, and focal length $f = 6.0$mm, and distance $d = 120$ ft limited for resolution requirement, respectively. Thus, the camera's FOC covers an area of 3,840 square feet. There are a total of 71 cameras resulting in 82% coverage and 6% overlap. Table 5.1 summarizes the findings.

Table 5.1.: Complex Case Evaluation Details

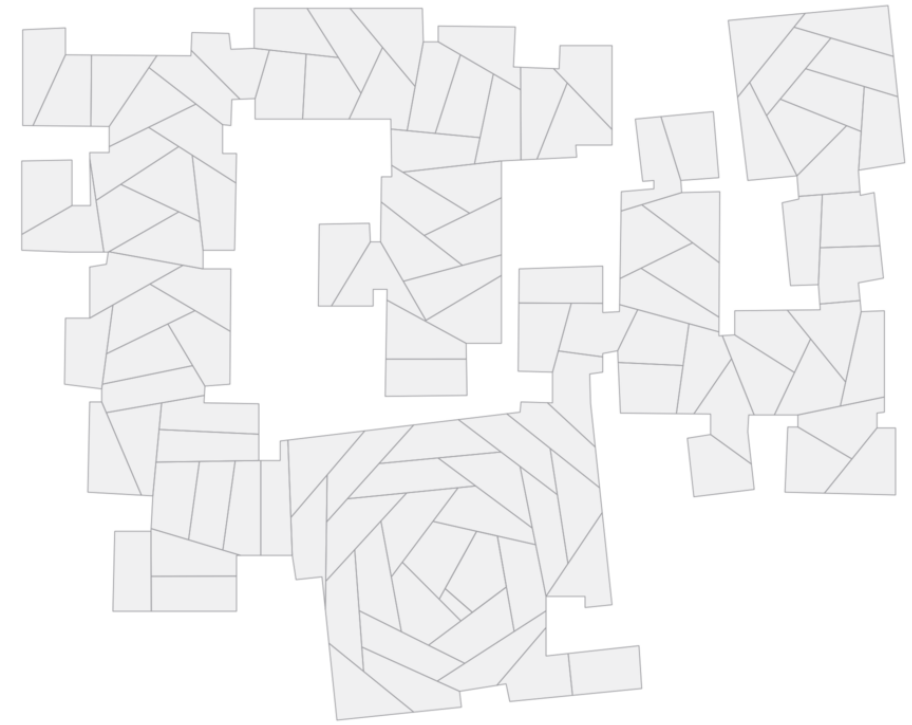
Layout	Number of Cameras	Camera Specifications				Coverage	Overlap	Overlap Time (Sec)	Coverage Time (Sec)	Placement Time (Sec)
		Focal Length	Angle of View	Camera Distance (1920 x 1080)	FOC Area					
Fig. 5.5	119	3.6 mm	74°	72 ft	3456	84%	13%	8	4	32
Fig. 5.6	107	4.0 mm	67°	80 ft	3840	84%	13%	6	3	28
Fig. 5.7	71	6.0 mm	42°	120 ft	5760	82%	6%	4	2	15

(a)

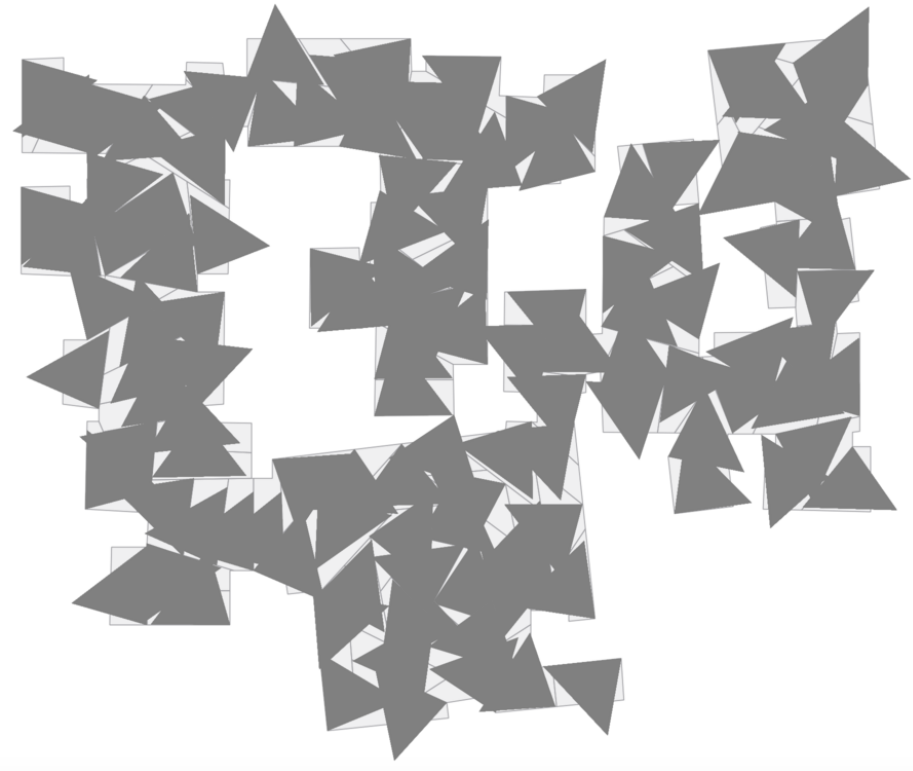


(b)

Fig. 5.4.: Floor plan of Louvre Abu Dhabi museum (https://www.archdaily.com) is input to the proposed solution.

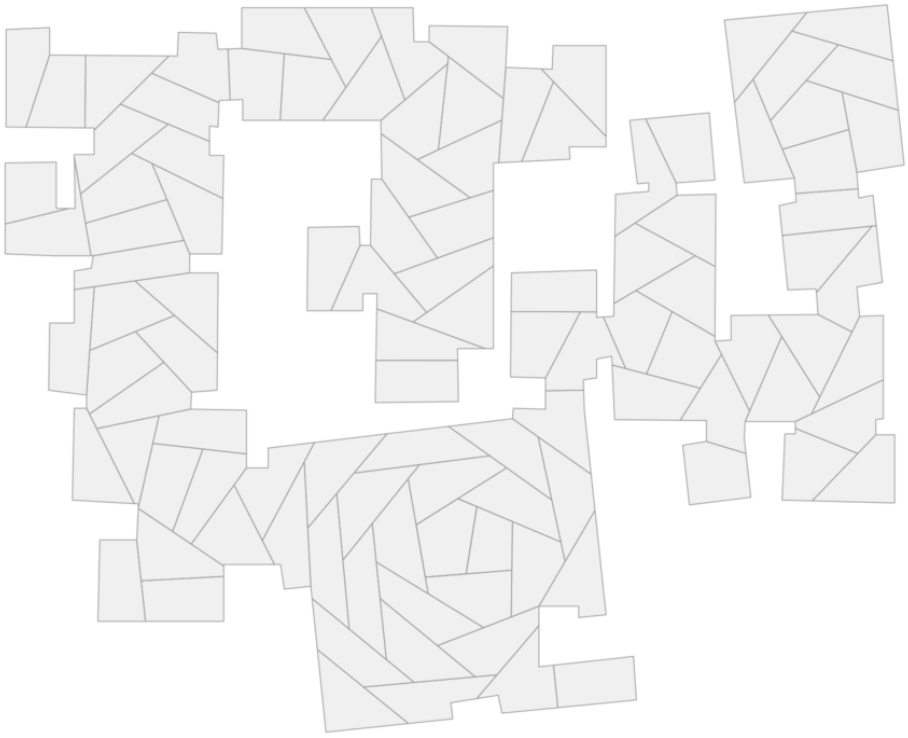


(a)

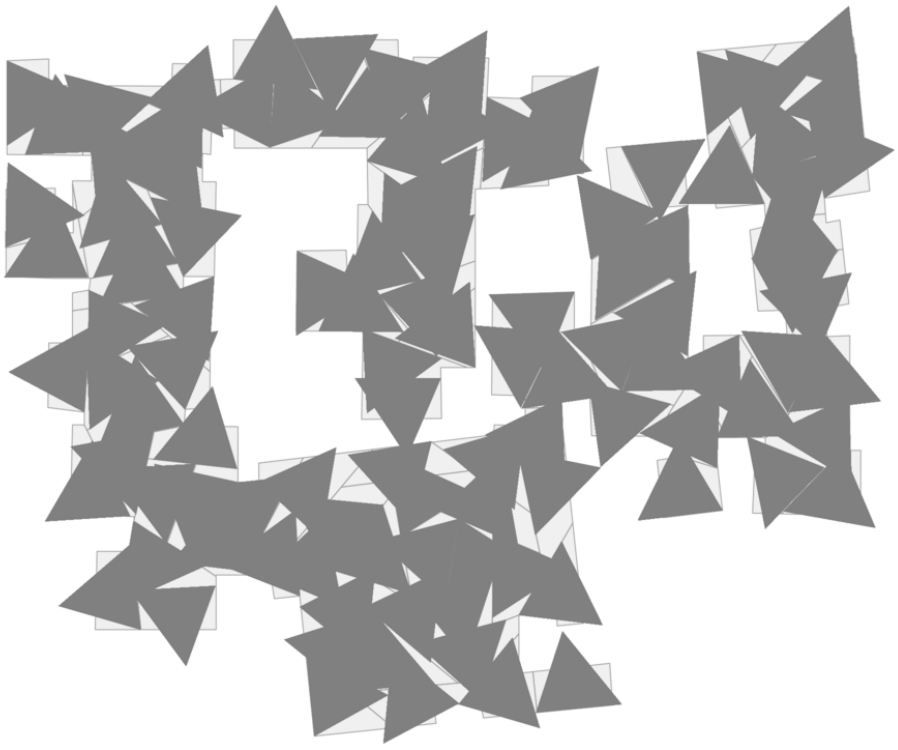


(b)

Fig. 5.5.: (a) The main polygon is split into subpolygon with equal areas based on camera FOC subject to camera specifications. (b) 119 FOCs with viewing angle $\alpha = 74°$and distance $d = 72\ ft$ are greedily placed in each subpolygon. 84% of the area is covered and 13% overlap is achieved.

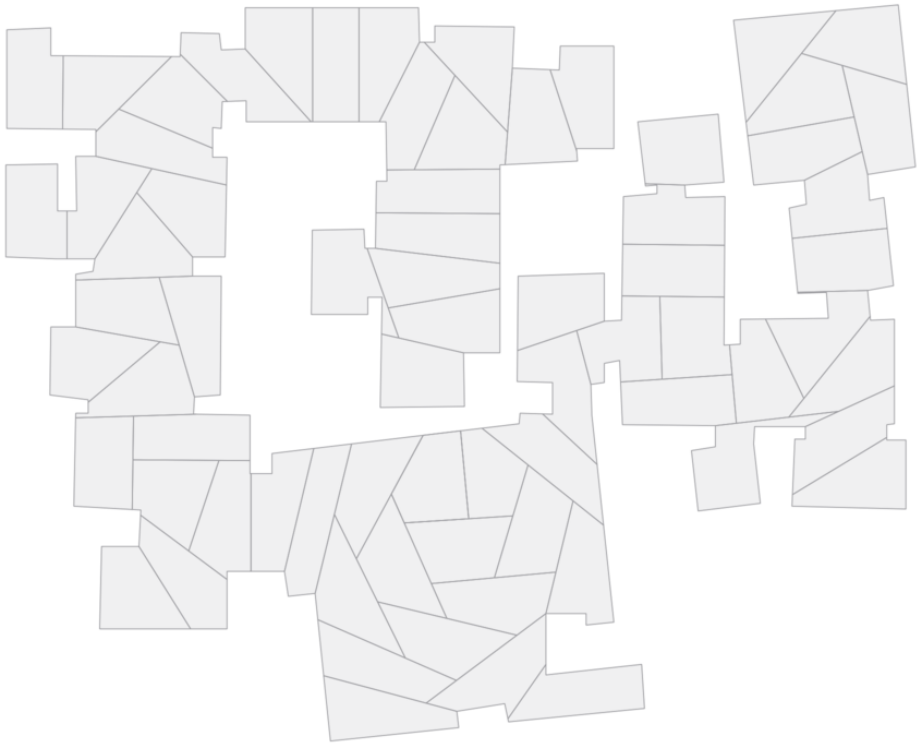


(a)

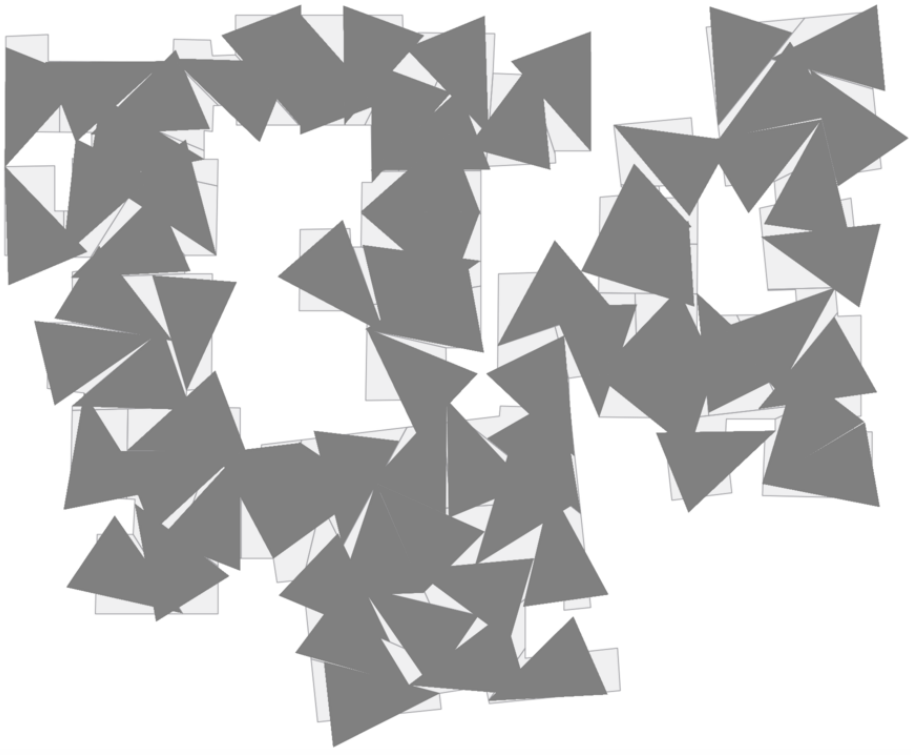


(b)

Fig. 5.6.: (a) The main polygon is split into subpolygon with equal areas based on camera FOC subject to camera specifications. (b) 107 FOCs with viewing angle $\alpha = 67°$and distance $d = 80\ ft$ are greedily placed in each subpolygon. 84% of the area is covered and 13% overlap is achieved.



(a)



(b)

Fig. 5.7.: (a) The main polygon is split into subpolygon with equal areas based on camera FOC subject to camera specifications. (b) 71 FOCs with viewing angle $\alpha = 42°$and distance $d = 120\ ft$ are greedily placed in each subpolygon. 82% of the area is covered and 6% overlap is achieved.

6. SUMMARY

Compared with prior works on the surveillance camera placement problem, our automated placement method takes into consideration the realistic constraints of computer vision, making our algorithm suitable for real-world deployment. Our solution aims to eliminate the gap between theoretical computational geometry and the realistic requirements of computer vision by ensuring both the minimum required resolution and the camera angle of view coverage are satisfied during camera placement. To achieve above goals, the proposed greedy solution partitions the main polygon into fixed size subpolygons and then cameras are greedily placed within the subpolygons. The proposed solution is implemented and evaluated on a real-world floor plan. The evaluation results show that the greedy solution can achieve above 77% coverage and below 25% overlap for spaces (i.e.,polygons) of different shapes.

7. FUTURE WORKS

There are some possible improvements to consider; for example, adding support for 3-dimensional spaces and more camera models with PTZ (pan, tilt, zoom) capabilities, as a result, considering certain blockage of views such as walls and pillars, in the case of a museum, large exhibits. The current greedy camera placement algorithm can be executed, placing 3- dimensional camera triangles (i.e., cones) instead of the current 2-dimensional FOCs; Consequently, such an improvement enables the use of PTZ camera models and provide greater capabilities by considering occlusion objects.

REFERENCES

REFERENCES

[1] S. Eidenbenz, C. Stamm, and P. Widmayer, "Inapproximability of some art gallery problems," in *Proceedings of the 10th Canadian Conference of Computational Geometry*, 1998, pp. 64–65.

[2] S. Eidenbenz, C. Stamm, and P. Widmayer, "Inapproximability results for guarding polygons and terrains," *Algorithmica*, vol. 31, no. 1, pp. 79–113, 2001.

[3] K. Tarabanis and R. Y. Tsai, "Computing viewpoints that satisfy optical constraints," in *Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, June 1991, pp. 152–158.

[4] R. Cucchiara, "Multimedia surveillance systems," in *Proceedings of the Third ACM International Workshop on Video Surveillance and Sensor Networks*, 2005.

[5] R. Bodor, A. Drenner, P. Schrater, and N. Papanikolopoulos, "Optimal camera placement for automated surveillance tasks," *Journal of Intelligent and Robotic Systems*, vol. 50, Oct 2007.

[6] S. Soro and W. Heinzelman, "A survey of visual sensor networks," *Advances in Multimedia*, vol. 2009, 2009.

[7] G. L. Foresti, P. Mahonen, and C. S. Regazzoni, "Multimedia video-based surveillance systems: Requirements, issues and solutions," *Springer Science and Business Media*, 2012.

[8] C. Micheloni, B. Rinner, and G. L. Foresti, "Video analysis in ptz camera networks - from master-slave to cooperative smart cameras," *IEEE Signal Processing Magazine*, vol. 27, no. 5, pp. 78–90, 2010.

[9] M. Shah, O. Javed, and K. Shafique, "Automated visual surveillance in realistic scenarios," *IEEE MultiMedia*, vol. 14, no. 1, pp. 30–39, Jan 2007.

[10] U. M. Erdem and S. Sclaroff, "Automated camera layout to satisfy task-specific and floor plan-specific coverage requirements," *Computer Vision and Image Understanding*, vol. 103, no. 3, pp. 156 – 169, 2006, special issue on Omnidirectional Vision and Camera Networks.

[11] N. Jiang, W. Liu, H. Su, and Y. Wu, "Tracking low resolution objects by metric preservation," in *CVPR 2011*, June 2011, pp. 1329–1336.

[12] Y. Wu, J. Lim, and M. Yang, "Online object tracking: A benchmark," in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 2411–2418.

[13] K. Yabuta and H. Kitazawaand, "Optimum camera placement considering camera specification for security monitoring," in *2008 IEEE International Symposium on Circuits and Systems*, May 2008, pp. 2114–2117.

[14] A. Bakhtari, M. Mackay, and B. Benhabib, "Active-vision for the autonomous surveillance of dynamic, multi-object environments," *Journal of Intelligent and Robotic Systems*, vol. 54, no. 567, 2009.

[15] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Computing Surveys (CSUR)*, no. 4, 2006.

[16] Y. Yao, C. Chen, B. Abidi, D. Page, A. Koschan, and M. Abidi, "Can you see me now? sensor positioning for automated and persistent surveillance," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 40, no. 1, pp. 101–115, 2010.

[17] J. Liu, S. Sridharan, and C. Fookes, "Recent advances in camera planning for large area surveillance: A comprehensive review," *ACM Comput. Surv.*, vol. 49, no. 1, pp. 6:1–6:37, May 2016.

[18] E. Hörster and R. Lienhart, "On the optimal placement of multiple visual sensors," in *Proceedings of the 4th ACM International Workshop on Video Surveillance and Sensor Networks*, 2006, pp. 111–120.

[19] N. Bisagno, N. Conci, and B. Rinner, "Dynamic camera network reconfiguration for crowd surveillance," in *Proceedings of the 12th International Conference on Distributed Smart Cameras*, 2018, pp. 4:1–4:6.

[20] A. A. Altahir, V. S. Asirvadam, N. H. B. Hamid, P. Sebastian, N. B. Saad, R. B. Ibrahim, and S. C. Dass, "Optimizing visual surveillance sensor coverage using dynamic programming," *IEEE Sensors Journal*, vol. 17, no. 11, pp. 3398–3405, 2017.

[21] "What is focal length," *StudioBinder*, Feb 2019. [Online]. Available: https://www.studiobinder.com/blog/focal-length-camera-lenses-explained/

[22] A. A. Altahir, V. S. Asirvadam, N. H. Hamid, P. Sebastian, N. Saad, R. Ibrahim, and S. C. Dass, "Modeling multicamera coverage for placement optimization," *IEEE Sensors Letters*, pp. 1–4, Dec 2017.

[23] Y.-G. Fu, J. Zhou, and L. Deng, "Surveillance of a 2d plane area with 3d deployed cameras," *Sensors*, vol. 14, pp. 1988–2011, 2014.

[24] S. Khetarpal, "Dividing a polygon in any given number of equal areas," 2014. [Online]. Available: http://www.khetarpal.org/polygon-splitting/

APPENDIX

A. IMPLEMENTATION

The software for this work is open source and available at
https://github.com/SaraAghajanzadeh/poly-split

VITA

VITA

Sara Aghajanzadeh is a graduate student in the School of Electrical and Computer Engineering of Purdue University. She graduated with a bachelor of science degree in Computer Information Technology and minor in Management from Purdue University. She is the lead author of "Observing Human Behavior through Worldwide Network Cameras" APA book chapter. She is a co-author of "Dynamic Sampling in Convolutional Neural Networks for Imbalanced Data Classification" IEEE MIPR 2019 and "See the World through Network Cameras" IEEE Computer 2019. Her research interest lies in embedded and low-power computer vision.