In the blooming era of smart edge devices, surveillance cam- eras have been deployed in many locations. Surveillance cam- eras are most useful when they are spaced out to maximize coverage of an area. However, deciding where to place cam- eras is an NP-hard problem and researchers have proposed heuristic solutions. Existing work does not consider a signifi- cant 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 iden- tifies effective camera placement in arbitrary spaces and can account for different camera types as well. Our strategy rep- resents spaces as polygons, then uses a greedy algorithm to partition the polygons and determine the cameras’ locations to provide the desired coverage. Our 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% cov- erage and 25% overlap.

*Index Terms*— Computational Geometry, Computer Vi- sion, Camera Placement

**1. INTRODUCTION**

Smart edge devices equipped with cameras are utilized in au- tomated surveillance systems to gather visual data and pro- cess the data with computer vision techniques, such as ob- ject detection or tracking. Erdem et al. [1] note that plac- ing 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 prob- lem”). Realistically, computer vision is ineffective when ob- jects are too far away and too small [2]. Even human eyes end up having to squint to see things far away! Most track- ing applications struggle to detect objects at low resolutions (below 10 pixels per foot) [3]. An example is given in Fig. 1. In Fig. 1 (a), the object in the bounding box has a resolu- tion of 20 pixels per foot, thus the object of interest can be tracked easily. In Fig. 1 (b), the object is too far away from the camera and has a resolution below 10 pixels per foot, mak- ing it difficult to track effectively. Generally speaking, when the number of pixels of a bounding box is less than 400, the resolution is considered low [4]. More importantly, visibil- ity 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.

 (a) (b) 

**Fig. 1**: Comparison of the number of pixels per object used by an object tracker.

The camera placement problem has been proven to be an NP-hard problem; thus, instead of seeking an optimal so- lution, the placement techniques seek an approximate near- optimal solution [5]. Most of the previous approaches are either limited to the trade-offs between coverage and costs or the prior knowledge given by security experts [6]. 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 ef- fective 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. 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. Since each subpolygon may have an arbi- trary shape and the camera’s field of coverage is a triangle, the 100% coverage may not be achieved. The effectiveness of the proposed greedy solution is evaluated through experi- ments 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.

**2. BACKGROUNDANDCONTRIBUTIONS**

The problem of automated camera placement has captured the attention of the research community for quite some time. Bis- agno et al. [7] used reconfigurable cameras that can dynami- cally adapt their field of view (FOV) and resolution to provide coverage 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 resolu- tion. Yabuta et al. [5] proposed a method considering camera specifications and a trade-off between coverage and the cost (i.e., the number of cameras). And Altahir et al. [8] propsed a dynamic programming solution that relies on human experts to determine camera locations.

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. The majority of theoretical solutions assume unlim- ited field of coverage and infinite visibility; thus, existing the- oretical works cannot be applied in a real deployment [1]. Prior heuristic works reduce the problem based on restrictions and relevance to various applications. Although a significant amount of research has been conducted, few studies have been devoted to automated placement with the consideration of vi- sion technologies. This paper proposes a greedy solution for determining an effective placement of cameras for monitor- ing an area with a target resolution sufficient for computerized tracking of individuals.

**3. GREEDYCAMERAPLACEMENT**

As stated previously [9], the solution for 100% coverage is computationally intractable, so this paper does not produce full coverage. Instead, this paper aims to achieve a balance between coverage and resolution. Fig. 2 presents a flowchart of the proposed approach. The input to the algorithm are the desired floor plan and the camera specifications. The main computational steps are: 1) Initialization procedure - comput- ing camera FOC based on the camera model, and splitting the polygon into equal areas based on the FOC; and 2) Place- ment 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.

**3.1. Camera Placement Procedure**

This paper uses fixed cameras and sets the threshold as 20 pix- els per foot based on the observation from Section 1. Given the threshold, we determine the distance from a camera to a target individual that is required for successful detection

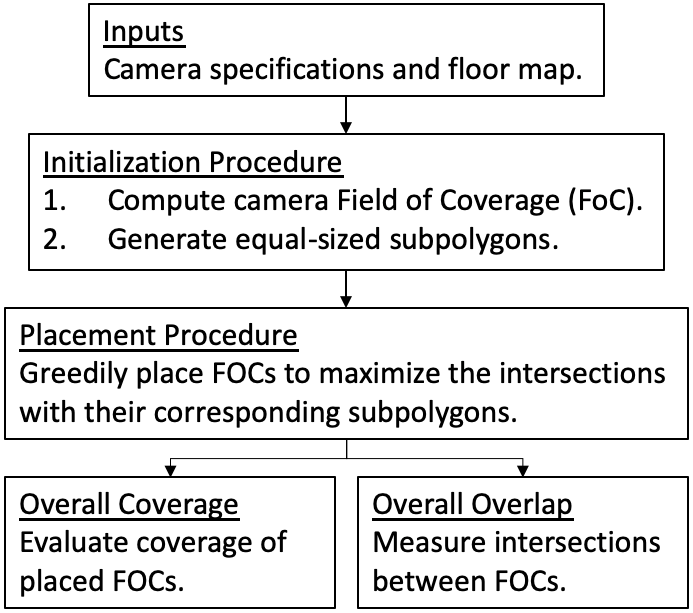


Fig. 2: Flow chart of the proposed approach.

and tracking. We use Equation 1 to compute this distance. Fig. 3 shows the relationship between the camera specifica- tions. Given the distance, we compute the FOC, which is the area of a triangle calculated using Area = 12 × b × h, where h is the camera distance and b is the maximum horizontal field of view calculated using horizontal camera resolution

= FOV distance=focallength×horizontalcameraresolution (1)

for each camera.

resolution threshold

chip width × resolution threshold

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 the algorithm in [10], 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

AreamainP olygon to n

. The process is displayed in Fig. 4.

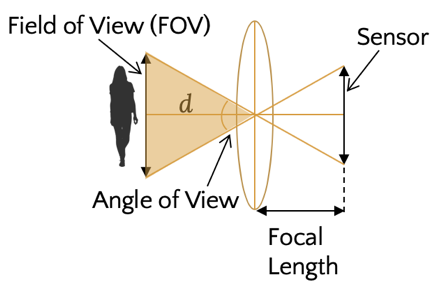
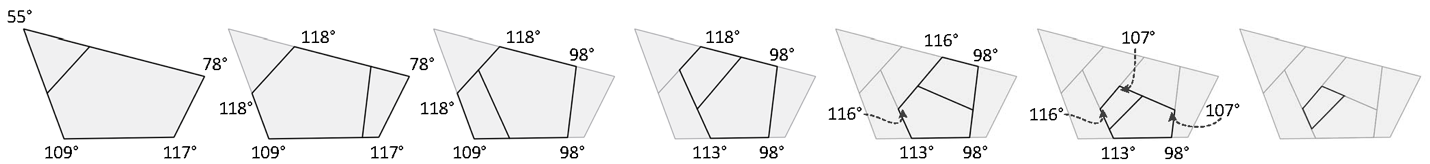
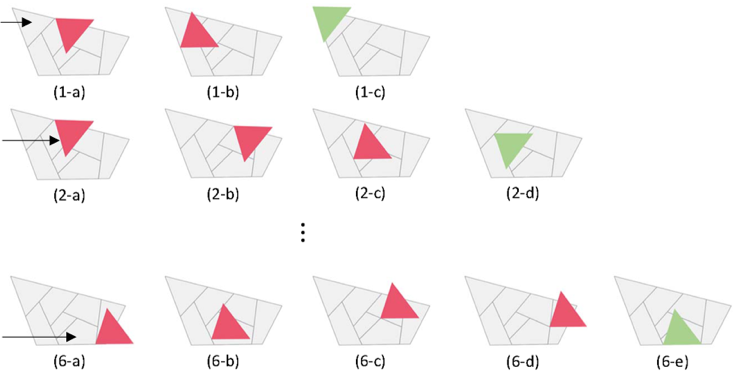


Fig. 3: The input camera specifications.



**Fig. 4**: 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.

For every vertex of the subpolygons, the level of intersec- tion between the camera’s FOC and the subpolygon is com- puted, and the FOC is rotated such that it lies inside the poly- gon and provides the best intersection between the FOC and its corresponding subpolygon. Fig. 5 summarizes the various possibilities for the rotated FOC.

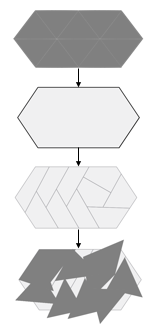
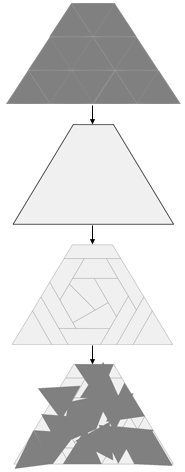


**Fig. 5:** For every subpolygon, indicated by the arrow, our so- lution examines each vertex [a-e] and computes the coverage gained by placing camera FOC at that vertex. The vertex with the highest coverage score is chosen for each subpolygon. Red color designates ineffective placements whereas green color designates the effective placement. As an example, (1- a) camera FOC does not intersect with its own subpolygon at all, thus our solution proceeds to the next vertex (1-b), where camera FOC intersects with the subpolygon over a small area. Our 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.

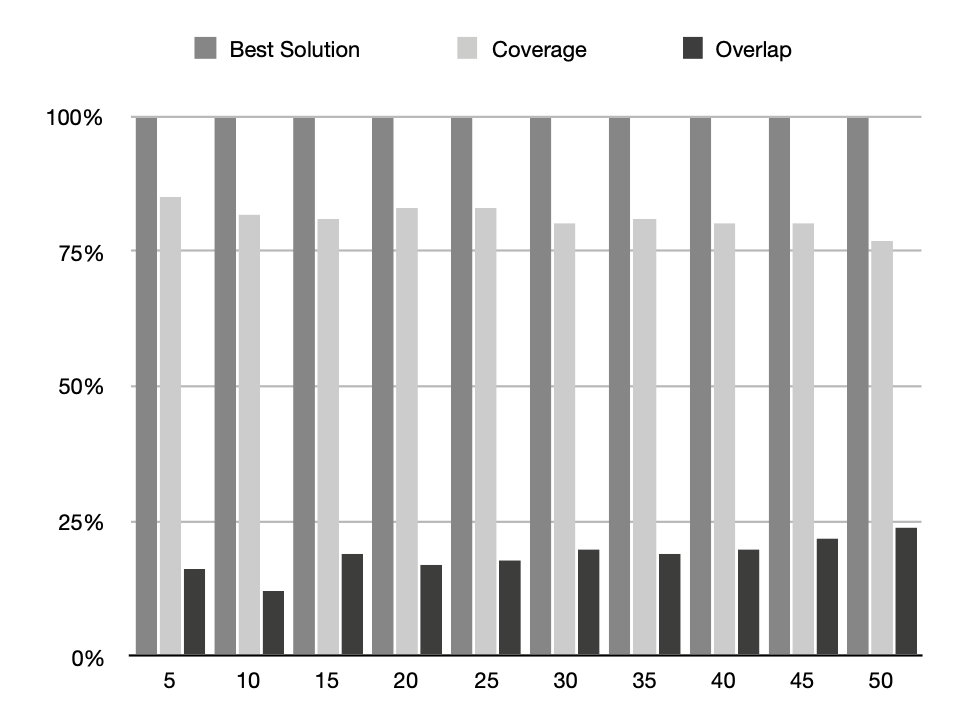
**4. EVALUATION**

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 polyno- mial, O(n2) for the subpolygons and their corresponding ver- tices. It seeks the candidates that satisfy the objective locally rather than identifying a global optimum. For our evaluation, a 1080p HD fixed camera with angle of view α = 67°and focal length *f* = 4.0mm, distance *d* = 80 ft limited for resolution re- quirement is used. Thus, the camera’s FOC covers an area of 3,840 square feet. This section evaluates the proposed method in two different scenarios.

For a base case scenario, shown in Fig. 6, we compare the ideal solutions with 100% coverage without overlap, plot- ted by humans, with the automated solutions. In Fig. 6 (a) both solutions (manually and automated) use the same num- ber of cameras (10 cameras), since the original polygon is di- vided based on the camera FOC. Our method provides less than 100% coverage (82% coverage) because the subpoly- gons created by the algorithm do not always match triangle shapes of the camera FOC; as a result, the proposed method has overlaps (12% overlap), which could be potentially useful for object tracking applications during camera hand-off.

 (a) (b) 

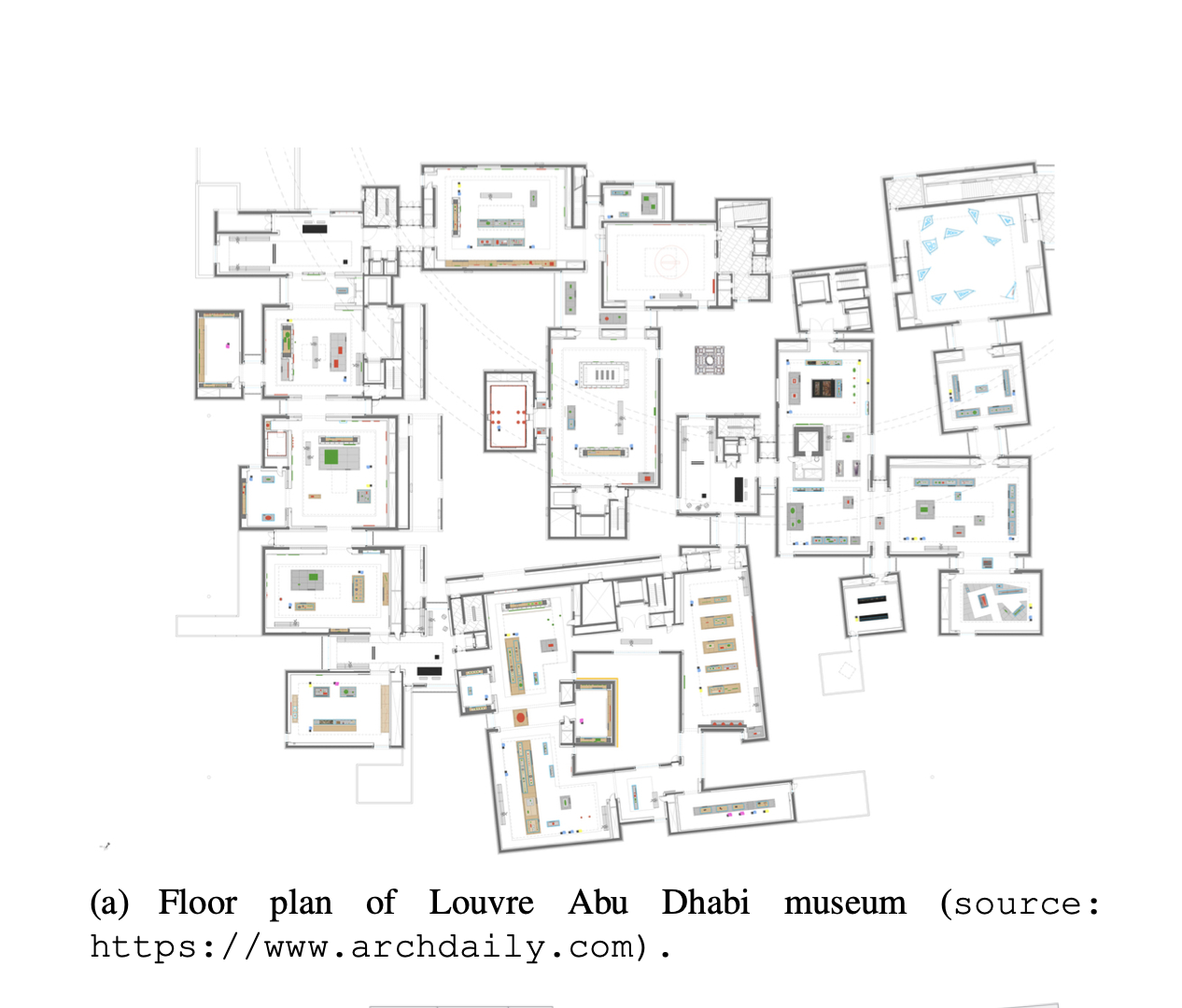
**Fig. 6**: (Top) Best possible solution achieving 100 % cover- age with 10 cameras in (a) and 15 cameras in (b). Dark grey triangles are cameras with angle of view α = 67°and focal length *f* = 4.0mm. (Bottom) Our method receives the main polygon, divides the main polygon into equal areas subject to the camera FOC, and greedily places the cameras in every subpolygon. (a) 10 cameras are placed to achieve 82% coverage and 12% overlap. (b) 15 cameras are placed to achieve 81% coverage and 19% overlap. Similarly, in Fig. 6 (b) both solutions (manually and auto- mated) 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. Furthermore, Fig. 7 shows the performance of the algorithm as the number of edges increase, resulting in more com- plex polygons and requiring more cameras. Our solution stabilizes at approximately achieving above 77% coverage and below 25% overlap with increasing number of cameras.

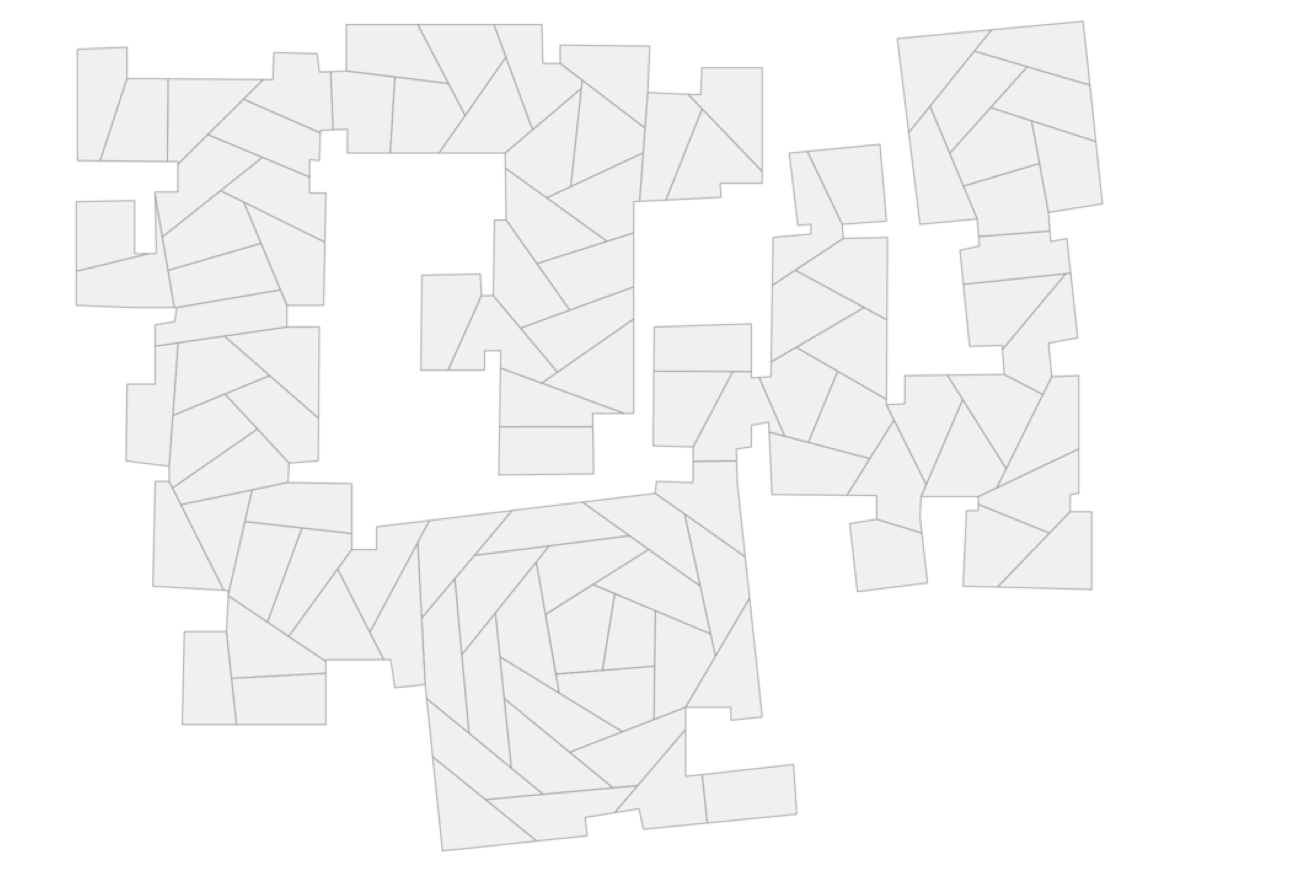


**Fig. 7**: Evaluation of coverage and overlap with increasing number of cameras as a result of more complex polygon (more edges and vertices)

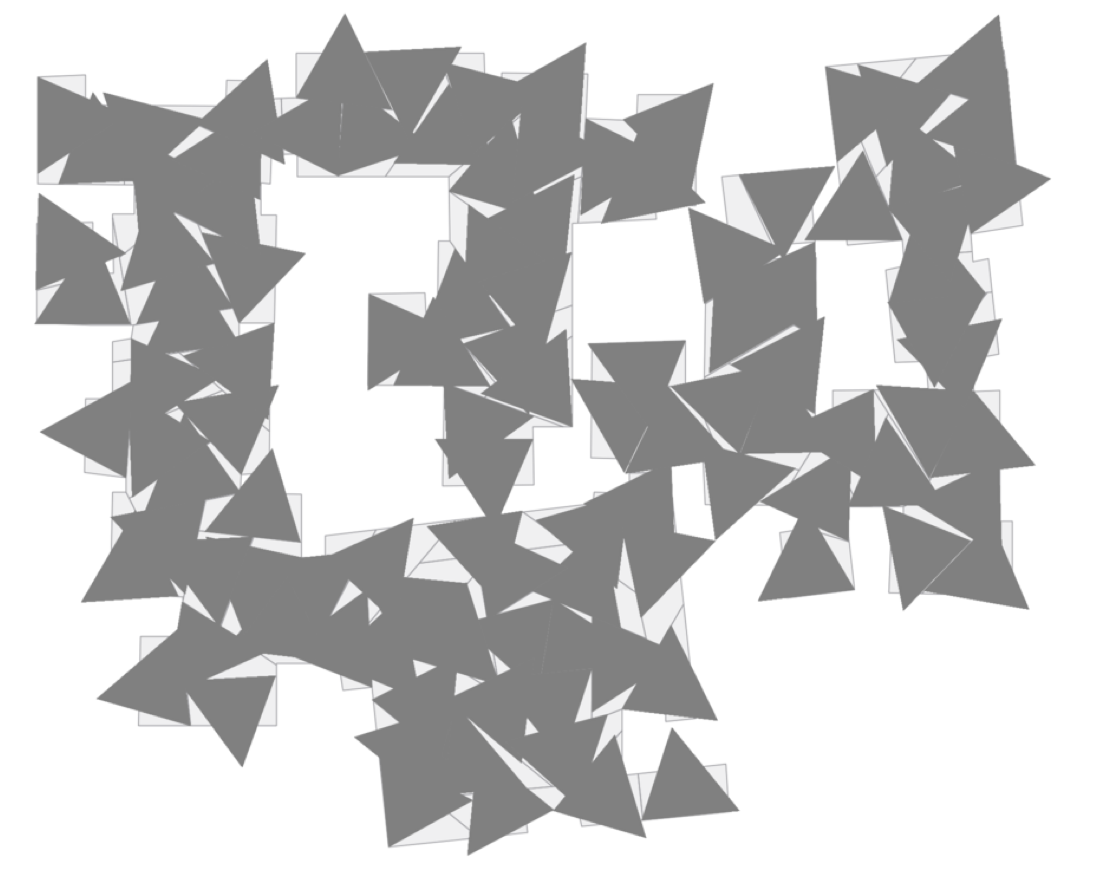
Next, for a complex case scenario, we apply our method to the floor plan of the Louvre Abu Dhabi museum shown in Fig. 8 (a). Fig. 8 (b) shows the result of dividing the floor plan programmatically into equal subpolygons based on the camera specifications, and Fig. 8 (c) illustrates the placement of camera FOCs such that effective coverage is achieved and requirements of computer vision applications are met.

**5. CONCLUSION**

Compared with prior works on the surveillance camera place- ment problem, our automated placement method takes into consideration the realistic constraints of computer vision, making our algorithm suitable for real-world deployment. Our solution eliminates the gap between theoretical compu- tational 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 so- lution 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 of different shapes. The software for this work is open source and available at https://github.com/SaraAghajanzadeh/poly-split



(b) Original polygon is split into subpolygons with equal areas. Area is based on the camera FOC subject to camera viewing an- gle α and distance *d* required to maintain successful detection and tracking.



Greedy placement of cameras with angle of view α = 67°and focal length *f* = 4.0mm. Dark grey triangles represent 107 cameras. 84% of the area is covered and 13% overlap is achieved.