Geometric Approach to solving the Art Gallery Problem on a Polygon with Holes

Alma Ćemer, Emin Mulaimović, Nedim Omeragić

University of Sarajevo, Faculty of Science, Bosnia and Herzegovina

{alma.cemer, emin.mulaimovic, nedim.omeragic}@student.pmf.unsa.ba

***Abstract*— The Art Gallery Problem(AGP) is one of the fundamental problems in computational geometry. The Problem is to try and determine the minimum number of cameras that can observe the polygonal Art Gallery. While there are numerous algorithms that have been proposed for simple polygons, addressing the AGP on polygons with holes becomes more complex. In this study we present a geometric approach to try and find an optimal solution for AGP on polygons with holes. We aim to achieve maximum coverage with the least amount of cameras.**

***Keywords* — Art Gallery problem; Genetic Algorithm; Triangulation;**

I. Introduction

AGP [1] is one of the fundamental problems in computational geometry, it is an NP-hard [3] problem [2] that plays a pivotal role in various applications, including surveillance and security systems[4]. The problem is to ascertain the minimum number of cameras required to cover a polygonal art gallery. While various algorithms have been devised for polygons without holes [5] [6] [7], addressing the AGP on polygons with holes becomes a harder problem. Recognizing the NP-hard nature of the AGP, it becomes evident that obtaining an optimal solution in a practical time frame is unattainable.

NP-hard problems necessitate the exploration of approximation algorithms. This study presents a geometric approach to AGP on polygons with holes. The fundamental goal of our investigation is to develop an algorithm that maximizes coverage within a given polygon using a fixed number of cameras as well as different types of cameras, using geometric strategies as the primary means for achieving this objective.

The first section gives a brief introduction, a definition of the problem, and motivation for solving it. The second section examines existing literature with a specific focus on AGP within the context of polygons with holes. This literature review shows us key insights, methodologies, and challenges explored by previous studies. Section three details the algorithms used for achieving a near-optimal solution to AGP on polygons with holes, emphasizing geometric principles. The fourth and final section shows the empirical outcome of our implemented algorithms, showing coverage percentages on many different polygons. In the last section, the paper's summary and significance of the paper are described, and avenues for future work and research are given.

II. Related Work

In reference [8], Wang and her colleagues achieved a great approximation of the AGP on a campus that was represented as a polygon with holes. They employed Particle Swarm Optimization on pixels to get this solution. In our approach, we aim to find an approximation using a geometric algorithm that does not depend on pixels. Our algorithm will only utilize pixels for calculating coverage, as we have not yet found a polynomial algorithm for determining the surface area of a union of n convex polygons.

III. Case Study

In this study we will first define our cameras. We will be using two types of cameras: regular cameras and circle cameras. Regular cameras have a radius of 150px and an angle of 1 radian. Circle cameras have a radius of 64px and an angle of 2π. These dimensions were used to try and look as close as possible to the ones used in [8]. You can see an example of a regular camera in Figure 1 and an example of a circle camera in Figure 2.

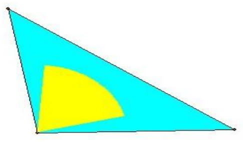

Fig 1. Regular camera

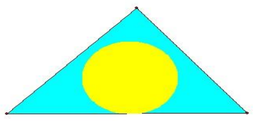

Fig 2. Circle camera

We are currently tasked with covering a polygon that includes holes, utilizing a specific type of camera. To achieve this, we aim to triangulate the polygon, enabling us to incorporate triangles into our algorithm. The preference is for the triangles to be highly similar, justifying the use of Delaunay triangulation. We will adopt a Constrained Delaunay triangulation as defined in [9], wherein our constraints will encompass all edges that cut through the holes.

To calculate the coverage and visualize the solution, we need to intersect the cameras with the polygons and determine the total surface area covered. The intersection will be performed using the ray casting algorithm as described in [8], as illustrated in Figure 3.

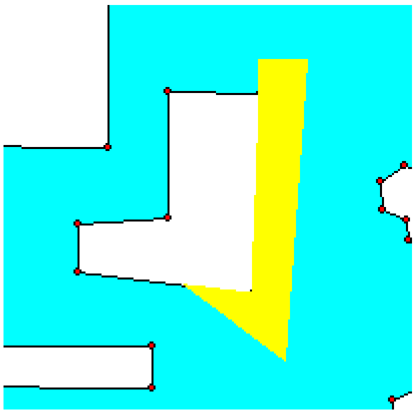

Fig. 3 camera intersection

The covered area will be computed by counting all the yellow pixels, and to obtain the percentage, we will divide by the surface area of the campus. The surface area of the campus is determined by calculating the surface area of the polygon and subtracting the surface area of the holes. All these polygons are simple, and their surface area can be calculated using the shoelace algorithm[10].

Firstly, we explore Greedy algorithms[11] for generating cameras and determining their positions in a triangulated polygon.

Algorithm 1 utilizes a Greedy method for coloring triangle vertices. After coloring vertices of Delaunay triangles in three colors, one color is selected to represent the camera position. The camera direction is the bisector of the opposite side of the triangle as shown in Figure 1. Disadvantage of the Color vertices algorithm is a high camera count especially if there are a lot of small Delaunay triangles.

Fig. 1 Regular camera

Algorithm 1 Color vertices

```
vertexColors = empty unordered_map
do polygon Delaunay triangulation

for each triangle in Delaunay triangulation do
  for each  vertex in triangle do
    if vertex is not colored do
      for each neighbour in triangle do
        find colors of neighbors
      end
    end
  end
end
put camera in selected vertex color
```

That fact motivates the idea to look at areas of triangles and develop an Area-based algorithm. Algorithm 2 examines the area of the current triangle under consideration to determine camera position. Cameras are positioned and directed the same as in the previous algorithm. If the triangle area falls between 1500 and 6000, a single camera is positioned, otherwise, for areas over 6000, two cameras are assigned. Given our observation that the coverage area of our cameras is approximately 6000 pixels, it is reasonable to position two cameras on triangles surpassing 6000 in elevation, and one camera on triangles exceeding 2000. Through experimentation, we found that camera placements at elevations of 1500 and 6000 yielded optimal results under our tested conditions. Similar to the Color Vertices Algorithm, the disadvantage of this approach is that there is still a potential for a large number of cameras.

Algorithm 2 Area-based

```
do polygon Delaunay triangulation

for each triangle in Delaunay triangulation do
  if triangle area > 1500 do
    put  regular camera on triangle vertex
  end
  if triangle area > 6000 do
    put second regular camera on another triangle vertex
  end
end
```

To address the issue of camera count, we introduce algorithms that take the number of cameras as input. If cameras are not positioned properly, low camera count will lead to a lower percentage of covered polygon area.

Algorithm 3 is an optimized Area-based algorithm. Area constraints are adapted to camera coverage. The algorithm sorts triangles to prioritize camera placement on triangles with larger surface areas before those with smaller ones. The circle camera (Figure 2) has an approximate coverage area of 3000, while the regular camera's coverage area is slightly larger, exceeding 6000. Cameras are positioned based on the conditions outlined in the pseudocode. Through experimentation, we have identified nearly optimal camera placement boundaries for multiple cameras on triangles. However, the algorithm's performance is suboptimal, as coverage area depends on constraints, and camera overlap remains unavoidable using this approach. (Prepraviti uslove za dimenzije kamera)

Fig. 2 Circle camera

Algorithm 3 Advance area-based

```
do polygon Delaunay triangulation
sort Delaunay triangles by area
```

```
for each triangle in Delaunay triangulation do
    if camera number has reached do
        break
    end
    if triangle area between 1000 and 6000 do
        put circle camera on triangle centroid
    end
    if triangle area > 8000 do
        put regular camera on triangle vertex
    end
    if triangle area > 12000 do
        put second regular camera on another triangle vertex
    end
    if triangle area > 16000 do
        put third regular camera on last available triangle vertex
    end
end
```

To improve the Advanced area-based algorithm, we introduce a new algorithm named Check positions that considers every possible camera position in a triangle vertices and selects the one with the best progress in covering the polygon. Progress is defined as the improvement in polygon coverage achieved through this method. Positions that are being checked include the centroid of the triangle for the circled camera and all 3 vertices of the triangle for regular cameras.

Algorithm 4 Check positions

```
do polygon Delaunay triangulation
sort Delaunay triangles by area

for each triangle in Delaunay triangulation do
    if camera number has reached do
        break
    end
    check position for camera (triangle vertices and centroid)
    if camera position achieves significant covering do
        put camera in position
    end
end
```

In Algorithm 4 checking the progress uses a function that calculates the total coverage of cameras at every step. If placing a camera doesn't change the coverage more than we specify, we do not place the camera there and continue with the algorithm. Unlike the algorithms before, Algorithm 4 is not purely geometric because the coverage is calculated by counting pixels. This severely slows this algorithm down but also provides higher coverage.

Lastly, using coverage as progress motivates us to try using a genetic algorithm[12] to solve this problem. Possible points to place the cameras are going to be vertices and centroids of the Delaunay triangles. Regular cameras will point as illustrated in Figure 1. The algorithm begins with the creation of an initial population of candidate camera positions. These positions are randomly selected from the vertices and centroids of Delaunay triangles within the polygon. The top-performing individuals from the initial population are selected based on their coverage capabilities. The selected individuals undergo recombinations. Specifically, half of the cameras from one individual are combined with the other half from another, creating a new individual. This is repeated until sufficient coverage is reached. The pseudo code for this algorithm is given in Algorithm 5.

Algorithm 5 Genetic AGP

```
$population = initPopulation(triangles, populationSize)$
while true do
    $scores = evaluateCovereges(population, Polygon)$
    $topIndividuals = selectTop(population, score)$
    if $max(scores) > convergenceThreshold$ do
        break
    end
    $newPopulation = [\,]$
    for $i \in 1, \frac{len(topIndividuals)}{2}$ do
        $parent1 = topIndividuals[i]$
        $parent2 = topIndividuals[i + \frac{len(topIndividuals)}{2}]$
        $child1 = crossover(parent1, parent2)$
        $child2 = crossover(parent2, parent1)$
        $novaPopulation.add(child1)$
        $novaPopulation.add(child2)$
    end
    $population = newPopulation$
end
return $topIndividuals[0]$
```

IV. Results

Our priority was addressing the challenge of maximizing the coverage inside a polygonal area. We attempted to do this with as few cameras as possible. To illustrate these results, we reference the polygon depicted in Figure 4 that represents a campus in [8]. This polygon has almost 200 vertices and it is concave with concave holes. Figure 4 also presents the Delaunay triangulation of the previously mentioned polygon, further emphasizing its complex geometric structure.

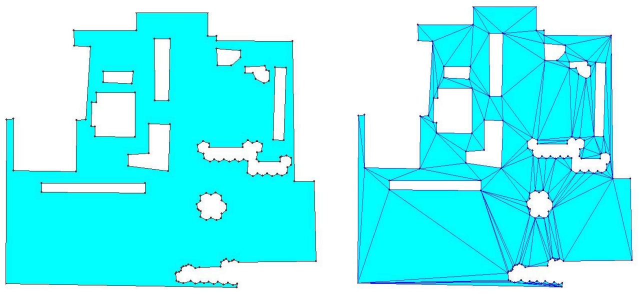
Fig. 4 Campus 1 example

Area covered, number of cameras and the time it took our algorithms to find a solution can be seen in Figure 6.

	Area covered	Num of cameras	Time
Color vertices	100%	174	10ms
Area-based	94.40%	98	10ms
Advanced area-based	61.96%	16	50ms
Check positions	82.40%	16	3min
Genetic AGP	86.20%	16	42min

Fig. 5 Results to the Art Gallery problem on campus 1solved by Color vertices, Area-based, Advanced area-based, Check positions and Genetic AGP

Aside from doing our testing on a very complex polygon described in [8] and depicted in Figure 4. We have decided to use a simpler polygon to show the strengths of some of these algorithms. That simpler polygon and its Delaunay triangulation is given in Figure 6. Results from our algorithms on this simpler polygon are given in Figure 7.

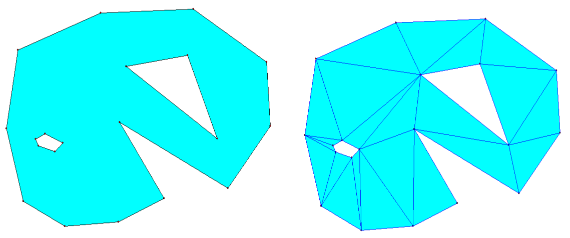
Fig. 6 Campus 2 example

	Area covered	Num of cameras	Time
Color vertices	93%	21	5ms
Area-based	94.40%	98	10ms
Advanced area-based	81.29%	16	40ms
Check positions	74.56%	8	2min
Genetic AGP	96.15%	16	42min

Fig. 7 Results to the Art Gallery problem on campus 2 solved by Color vertices, Area-based, Advanced area-based, Check positions and Genetic AGP

Lastly, we will show our results for Algorithms 3 4 and 5 illustrating their respective strengths and weaknesses. The result of Algorithm 3 is given in Figure 8. We see that the algorithm did well on a simpler campus but had problems in the complex one mostly because of higher variance in Delaunay triangles.

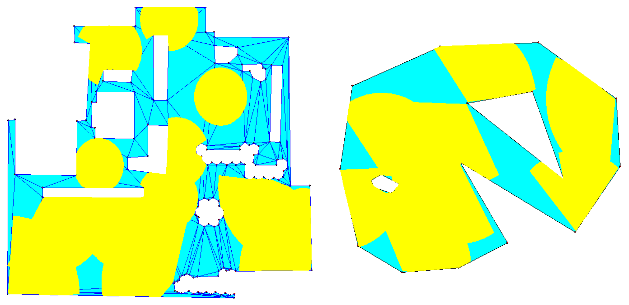
Fig. 8 Check positions on campus 1 and campus 2

Results for Algorithm 4 are illustrated in Figure 9. This algorithm did not share any problems with Algorithm 3 because it accounted for coverages.

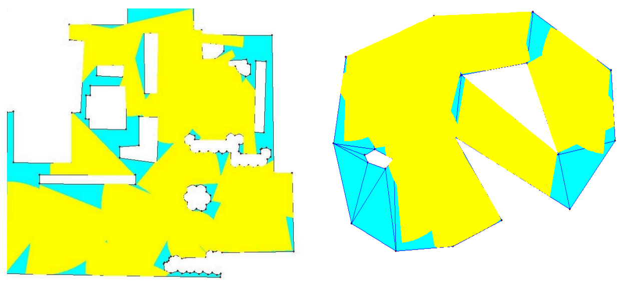
Fig. 9 Advanced area-based on campus 1 and campus 2

Algorithm 5 did an excellent job as can be seen in Figure 10, the only caveat being its long runtime.

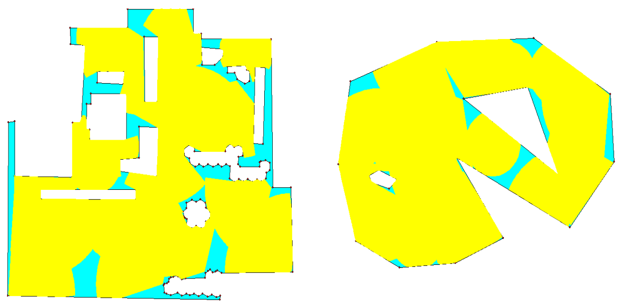
Fig. 10 Genetic AGP on campus 1 and campus 2

V. Conclusion

During this study we developed algorithms to try and tackle AGP on polygons with holes. Our plan was to create a geometric based algorithm that doesn't rely on pixel count and finds a near optimal solution. We created 3 algorithms namely Algoritam 1, 2 and 3 to solve this problem. On a more complex problem these algorithms didn't give a satisfactory solution so we compromised on using pixels only to ascertain respective coverages and improve these algorithms. With these improvements we created Algorithms 4 and 5.

In summary, Algorithm 5 gives the best result but is also computationally intensive. Algorithm 4 is almost as good albeit a lot less computationally intensive. However, both Algorithm 5 and Algorithm 4 exhibit a departure from our geometric approach, they become more computationally intensive, particularly at higher resolutions. Conversely, Algorithm 3 yields a satisfactory outcome with remarkable speed. This algorithm also scales well to higher resolution because it doesn't rely on pixels.

References

[1] Abrahamsen, Mikkel, Anna Adamaszek, and Tillmann Miltzow. "The art gallery problem is ∃ ℝ-complete." ACM Journal of the ACM (JACM) 69.1 (2021): 1-70.

[2] Lee, D., and Arthurk Lin. "Computational complexity of art gallery problems." IEEE Transactions on Information Theory 32.2 (1986): 276-282.

[3] Knuth, Donald E. "Postscript about NP-hard problems." ACM SIGACT News 6.2 (1974): 15-16.

[4] Chesnokov, Nicole. "The Art Gallery Problem: An Overview and Extension to Chromatic Coloring and Mobile Guards." Unpublished manuscript, 2018.

[5] Kröller, Alexander, Mahdi Moeini, and Christiane Schmidt. "A novel efficient approach for solving the art gallery problem." WALCOM: Algorithms and Computation: 7th International Workshop, WALCOM 2013, Kharagpur, India, February 14-16, 2013. Proceedings 7. Springer Berlin Heidelberg, 2013.

[6] Bonnet, Édouard, and Tillmann Miltzow. "An approximation algorithm for the art gallery problem." arXiv preprint arXiv:1607.05527 (2016).

[7] Ghosh, Subir Kumar. "Approximation algorithms for art gallery problems in polygons." Discrete Applied Mathematics 158.6 (2010): 718-722.

[8] Wang, X., Zhang, H., & Gu, H. "Solving Optimal Camera Placement Problems in IoT Using LH-RPSO." IEEE Access, vol. 8, 2020, pp. 40881-40891. doi:10.1109/ACCESS.2019.2941069.

[9] De Floriani, L., & Puppo, E. "An on-line algorithm for constrained Delaunay triangulation." CVGIP: Graphical Models and Image Processing, vol. 54, no. 4, 1992, pp. 290-300. ISSN 1049-9652.

[10] Setiawan, Adi, Eko Sediyono, and Tundjung Mahatma. "CALCULATION OF CENTRAL JAVA PROVINCE REGION AREA USING SHOELACE FORMULA BASED ON THE GADM DATABASE." BAREKENG: Jurnal Ilmu Matematika dan Terapan 16.2 (2022): 597-606.

[11] DeVore, Ronald A., and Vladimir N. Temlyakov. "Some remarks on greedy algorithms." Advances in computational Mathematics 5 (1996): 173-187.

[12] Lambora, Annu, Kunal Gupta, and Kriti Chopra. "Genetic algorithm-A literature review." 2019 international conference on machine learning, big data, cloud and parallel computing (COMITCon). IEEE, 2019.