Genetic Algorithms Applied to the Searching of the Optimal Path in Image-based Robotic Navigation Environments

Fernando Martínez Santa, Fredy H. Martínez Sarmiento, Holman Montiel Ariza
Universidad Distrital Francisco José de Caldas
Bogotá, Colombia

***Abstract*—This paper describes an optimal-path finding strategy based on Genetic Algorithms, applied to mobile robots in static navigation environments. This strategy starts from an image or plan of the environment and is supported by some different image processing algorithms, mainly the image skeletonization. Three different strategies were tested, changing the domain of the optimization target function for the Genetic Algorithm, the first domain was all the points of the environment image less the obstacles or walls, the second domain was similar but using an image with the obstacles dilated, and the final domain was only the points of the skeleton image. The last tested domain is from 99.4% to 99.6% smaller than the others, that implied reductions from 95% to 96% in the overall execution time of the strategy. Likewise, three skeletonization algorithms were tested in order to use the one with less execution time in this proposal. Finally, the proposed path planning strategy was tested on the same environment changing the initial and final points giving as result a valid and optimized path for the mobile robot in all the tested cases, and an overall average optimization time less than 2 minutes. This last, validates this proposal for robotic navigation applications with static obstacles.**

***Keywords*—Genetic algorithms; optimal path; optimization; robotic environment; mobile robots, image skeletonization**

I. Introduction

Although robotics has been one of the areas with the most ongoing research over the years, today, it continues to expand its fields of application due to the incorporation of automation in daily life. One example of this is mobile robots, which are now looking to help people in daily tasks; this situation implies that mobile robots must be capable of a path in different environmental conditions or grounds. Hence, navigation has become an important robotics process, allowing a mobile robot to know its location and plan and follow a path, preventing collision with obstacles.

According to recent studies [1], there are some approaches to the solution of the navigation problem in robotics based on deterministic and non-deterministic algorithms, which focus on optimal path planning [2] and collision prevention [3]. When reviewing from the path planning point of view, where a route is planned in a given environment reducing the total cost associated with the trajectory, it is necessary to consider whether local path planning [4] or global path planning [5], [6] is being performed. Thus, different methods have been developed, classified into classical approaches and heuristic [1] or reactive approaches [7]. In recent years, heuristic or reactive approaches have been the most used due to their robustness to handle the uncertainty present in the environment and real-time navigation problems. Some heuristic or reactive methods that have been used are Genetic Algorithm (GA) [8], [9], Ant Colony Optimization (ACO) [10], [11], Particle Swarm Optimization (PSO) [12], [13], [14], [15], Neural Networks (NN) [16], Fuzzy Logic (FL) [17], Dijkstra algorithm [18], A* algorithm [19], among others.

Genetic algorithms (GAs) are part of evolutionary computing, one of Artificial Intelligence techniques that exist today. GA is a meta-heuristic method for solving searching and optimization problems, where a new population is generated from the fitness value of the previous generation. It is based on the phenomenon of natural selection and genetic operations such as mutation and crossover [20]. Some studies have shown that GA is a robust search method that requires little information about the environment to achieve reducing path length and producing smoother path for robot navigation, with some limitations in convergence rate and time-consuming process [21], [22].

Therefore, the aim of this research is to propose a path planning strategy for mobile robots based on digital image processing and Genetic Algorithms. The main idea is exploring the GA as selection and optimization tool for searching a valid and optimized path for a mobile robot in its environment, starting from an image or plan of that environment.

The remainder of this paper is organized as follows. In Section II, a brief description of the robot environment is given, likewise the description of the data flow (pipeline) of the proposed strategy; the pre-processing image operations for the environment image are presented in Sections II-A, II-B and II-C; Section II-D gives the first process of the path planning, the computing of the skeleton of the image. Section II-F describes the application of a GA for finding an optimal path. Section III shows the experiments and results and the Section IV gives our final conclusions.

II. Methodology

The path planning proposal for mobile robots shown in this document, is supported by digital image processing [23], [24] and it is based on an initial image of the navigation environment, which can be a plan of the room or a photo taken from above. For the scope of this article the initial images are as the one shown in Fig. 1 which represents a plan of a building floor where the mobile robot has to navigate. In that initial image the floor walls are shown in *black* and the free-navigable space are shown in *white*, likewise the p_s represents the initial or starting point of the robot and p_e the final or ending point.

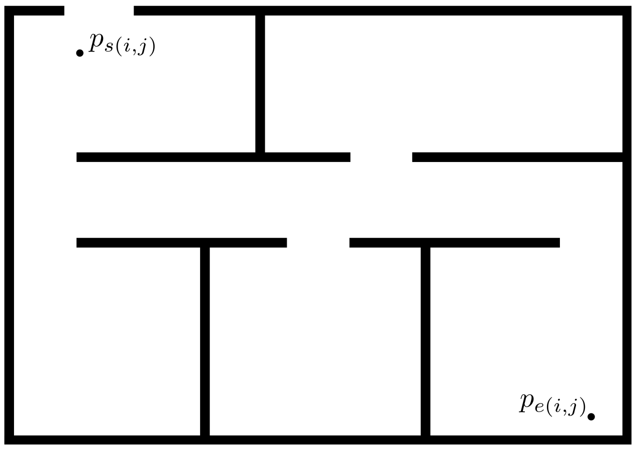

Fig. 1. Navigation Environment for the Mobile Robot.

The proposed path planning strategy is composed of the following steps: first, a resizing process is applied to the environment image in order to reduce and standardize the input. Second, the resized image is turned into binary to be compatible with the next steps of the process. After, two images are calculated, the first one is an image with the walls expanded (as wide as the radius of the robot), this image will be used to determine possible collisions in the next steps. The other image is the skeleton [25], [26] (medial axis) of the free-navigable space, this one is used for calculating the final path as far as possible to the obstacles (walls). Then, based on the skeleton of the environment image, the array of all the possible navigable points is stored. Finally, a short path is found using a Genetic Algorithm (GA) as optimizing and/or searching algorithm. The GA uses both the image skeleton and the walls dilated image to find the optimal path. The complete pipeline of proposed strategy is shown in the diagram of the Fig. 2. The overall path planning strategy was implemented by using *Python 3* programming language and mainly the *Scikit Image* module for the digital image processing operations. Next subsections show a detailed explanation of each of these steps.

A. Image scaling

In order to reduce the computing time of the overall path planning strategy, the images to work with have to be the small as possible, due to that, the resolution of the input image and therefore all the generated and used images is limited to $700k$ pixel, specifically images of *1000x700* pixels. This last is very important mainly for the GA, due to some image processing operations are part of the fitness function or target function, that implies it has to be iterated very much times until reach the convergence value. The exponential increasing of the execution time in image processing algorithms where the resolution of the input image increases, will imply a very high execution time for the overall strategy, for that reason is very important to limit the input image resolution. In the scaling or resizing operation that was applied to the input image, no anti-aliasing

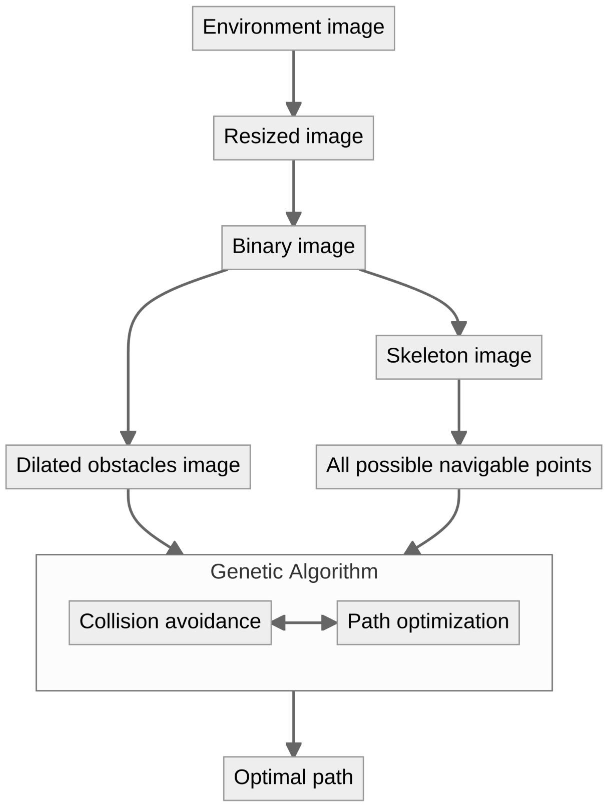

Fig. 2. Pipeline of the Proposed Path Planning Strategy.

method was used in order to not affect the original shape of the border objects in the environment.

B. Image Binarization

Once scaled, the image is normalized and binarized, this is achieved first by turning all the pixel values (bytes) of the image into normalized values from 0 to 1, by means of a simple product. After that, a threshold operation is performed following the eq. 1, where I and J are the maximum dimensions of the normalized image A and the output binary image B, likewise Th is the threshold defined as 0.5. At the end of this process, the image B has the obstacles in *black* (False) and the navigable space in *white* (True), as shown in Fig. 3.

$$\forall i \in I, j \in J \quad : \begin{cases} A_{ij} > Th \rightarrow B_{ij} = \textit{True} \\ A_{ij} \leq Th \rightarrow B_{ij} = \textit{False} \end{cases} \tag{1}$$

C. Obstacle Dilation

In order to keep the robot to a safe distance from the obstacles or walls of the environment, that distance is calculated from to the maximum radius (measure from the center to the maximum distance to this one) of the robot following the eq. 2, where r_d is the dilation radius, r_m is the robot maximum radius and Δr is a radius tolerance defined as 10%. The resultant r_d is represented in pixel units and it is rounded to the floor.

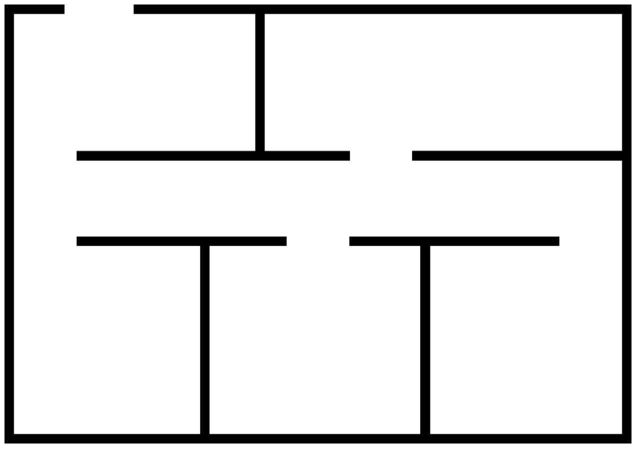
Fig. 3. Resized and Binarized Image of the Environment.

$$r_d = \lfloor r_m + \Delta r \rfloor \tag{2}$$

In a binary image, the dilation process is achieved over the *white* objects, so it is necessary to invert the image to obtain the desired *white* obstacles. At the end, the obtained image is inverted again to obtain an image with the dilated *black* obstacles. After inverting the obstacles binary image, a morphology dilation operation is performed by means of using a $2D$ convolution between that image and a disk shape a radius equivalent to r_d. After that, the image is inverted back, and the result is shown in Fig. 4, where the area of the obstacles or walls are expanded because of the dilation operation. All of this, pretends to avoid collisions between the robot with the walls due to the maximum radius of the robot r_m was taken into account.

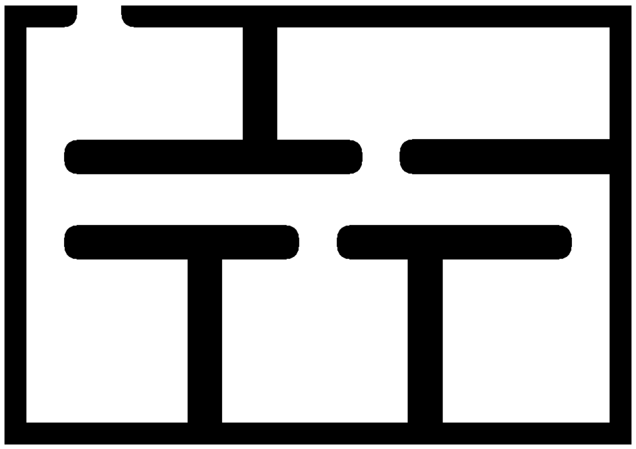
Fig. 4. Dilated Obstacles Image.

D. Skeletonization Algorithm

The image skeletonization algorithms pretend to find a medial axis of the shapes on a binary image applying an iterated and controlled image erosion operation until obtained a thin line [27]. This obtained line represents the medial axis of each object in the image. There are some different proposed algorithms about such as the one proposed by Zhang et al. [28], the Lee's proposal (et al.) [29], and other medial axis operation. For the scope of this paper, three different skeletonization algorithms were tested: the standard skeletonization (Zhang), the Lee's skeletonization and a standard medial axis obtaining algorithm. The main aim is to recognize which of them is the fastest in order to be applied in this proposal. The Fig. 5 shows the resultant execution times of the tested skeletonization algorithms. The tests was applied over an *1000x700* image that contains a possible navigation environment for a mobile robot. A total of 5 tests were done and the average execution time of each algorithm is the shown in Fig. 5. Then, according to those results, for this proposal only the Zhang skeletonization algorithm is applied for obtaining the medial axis points of the environment image, as reference for the navigation.

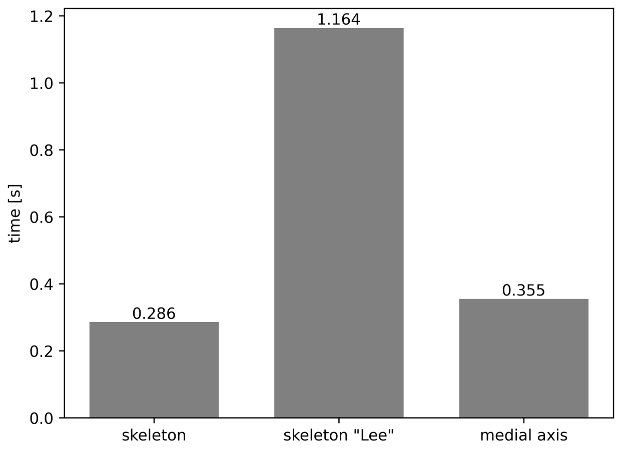

Fig. 5. Skeletonization Algorithms Execution Times.

The Zhang's skeletonization operation is applied to the resized environment image in order to obtain a thin line in the middle distance between obstacle or walls [30]. The Fig. 6 shows the resultant skeleton (in white) of the environment image, likewise the Fig. 7 shows original environment image plus the inverted skeleton image. This last image was obtained by means of applying logical *not* and logical *and* operations. As shown in the Fig. 7, the skeleton corresponds to the middle distance (medial axis) between halls walls and rooms walls, so it is a good reference for a free-collision navigation of the mobile robot.

E. Navigable Points

Once the environment image skeleton is obtained, all the points of the medial axis E_{ij} are stored in a bi-dimensional array P where each point $P_{k(x,y)}$ is a possible point of the final path. The storage operation of the skeleton set E starts from the skeleton image (see Fig. 6) and looks for the pixels in *white* (True), as the eq. 3 summarizes.

$$\forall k, i \in I, j \in J \quad : E_{ij} = \mathit{True} \rightarrow P_k = (i, j) \tag{3}$$

F. Genetic Algorithm

For calculating and optimizing the path from the navigable points array, Genetic Algorithms (GAs) are used as

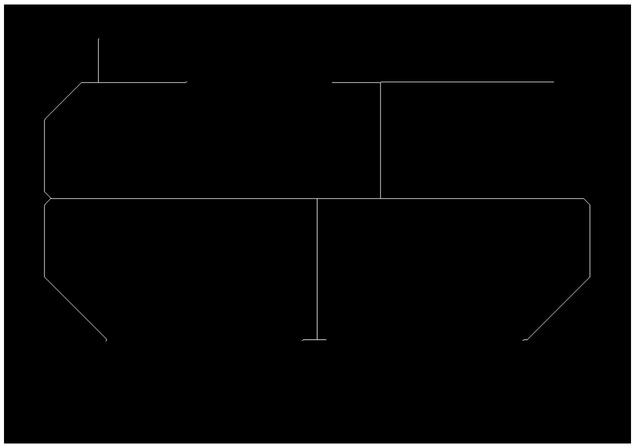

Fig. 6. Skeleton of the Environment Image.

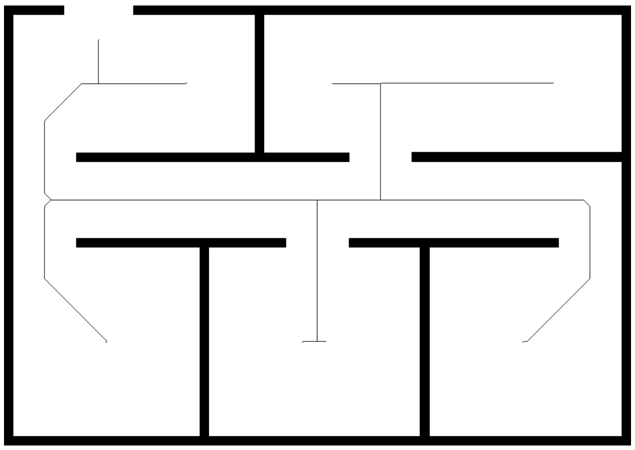

Fig. 7. Image of the Navigation Environment Plus its Skeleton.

optimization/searching tool [20] in order to find the shortest path using as reference the navigable points array obtained from the environment skeleton image, and at the same time avoiding collisions with obstacles and walls using the dilated obstacles image previously mentioned.

1) Collision Avoidance: The avoidance of possible collisions between each segment of the final path and the obstacles or walls is achieved applying some binary image operations, specifically a binary exclusive disjunction *XOR* between the dilated walls image and the same image with the specific segment drawn in *white* as shown in Fig. 8, where p_1 and p_2 are the points of the path segment. When there is a collision, a white segment line will be down over an obstacle or wall, producing that the two images are different. The eq. 4 has to be accomplished by all the pixels of both images for validating the non-collision condition. Where again, I and J are the valid set of indices in the dilated obstacles image D and the copy of the same image plus the drawn line L.

$$\forall i \in I, j \in J \quad : D_{ij} \not\leftrightarrow L_{ij} = \mathit{False} \tag{4}$$

2) Fitness Function: For starting the searching and optimization process, the starting point p_s and the ending point p_e

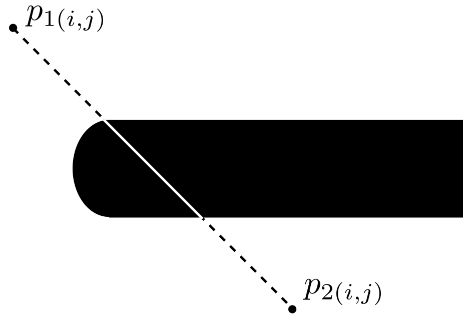

Fig. 8. Drawn Line for Testing the Collision with the Obstacles (walls).

are appended to the navigable points array P. Once completed the navigable points array, it is necessary to define the *fitness function* or *target function* to optimize $f(X)$, which depends on the solution set X shown in eq. 5.

$$X = \{x_0, x_1, x_2 \ldots x_n\} \tag{5}$$

where each x represents a reading index of the navigable points set P which is based on the skeleton of the environment image. Then, the *fitness function* depends on the summation of the distances of each segment $\overline{p_i p_{(i+1)}}$ from p_s to p_e, that also accomplishes the eq. 4, the complete definition is shown in eq. 6.

$$f(X) = \sum_{i=0}^{n} |\overrightarrow{p_{x_i} p_{(x_i+1)}}| \; + \; d_c \tag{6}$$

Listing 1: *fitness function* Implementation on Python 3.

```
def obj_function(X):

    d = 0

    route_x ,
    route_y = build_route(  X,
                            start_p ,
                            end_p    )

    for i in range(len(route_x)-1):

        dx = route_x[i] - route_x[i+1]
        dy = route_y[i] - route_y[i+1]
        d += np.sqrt( dx**2 + dy**2 )

        imt = (imd == True)
        rr , cc = line(  route_y[i],
                         route_x[i],
                         route_y[i+1],
                         route_x[i+1]   )
        imt[rr , cc] = True

        imc = imt == imd
        if sum(sum(imc)) != imd.size:
            d += (max_i + max_j)
    return d
```

In the eq. 6, $|\overrightarrow{p_{x_i}p_{(x_i+1)}}|$ is the distance of a segment between two nearby navigable points, and d_c is a penalty distance which is added to the total distance if the current segment produces a collision according to the eq. 4. This penalty distance d_c corresponds to the summation of the image dimension maximum values $A_{x_{max}} + A_{y_{max}}$.

The implementation of the GA *fitness function* in *Python 3* is shown in the listing 1, where the function **build_route** appends the starting ans ending points to the rest of points of the path.

3) Implementation: The implementation of the GA in this proposal, follows the parameter shown in Table I, where it is important to highlight that the mutation probability was increased to 0.2 in order to accelerate the convergence.

TABLE I. Genetic Algorithm Parameters

Parameter	value
maximum iteration number	None
population size	100
mutation probability	0.2
elitism ratio	0.01
crossover probability	0.5
parents portion	0.3
crossover type	uniform
mutation type	uniform by center
selection type	roulette
max. iteration without improvement	10
dimension	8
variable type	integer
function timeout	5s

The genome of the GA is simply defined as the complete set X, taking into account that $\forall i \in \{0, 1, 2 \ldots n\} : x_i \in \mathbb{E}$, where each gene corresponds to each x_i variable, it means $gen_0 = x_0$, $gen_1 = x_1$ etc, being each gene an integer variable.

All the tests of the proposed path planning strategy were done on a simple laptop with the following features: CPU AMD Athlon Gold 3150U @ 2.400GHz with 2 hardware cores, GPU AMD ATI Picasso, RAM 12 GB and main drive SSD. These test were run on the GNU/Linux distribution Ubuntu 20.04.3 LTS x86_64 and Python 3.8.10.

III. Results

As described in the section II-D, three different skeletonization algorithms were tested: the standard one in *Scikit Image* (Zhang's), the Lee's version and the standard *Scikit Image* medial-axis detection algorithm. The Zhang's version showed to take only the 24.6% of the time that the Lee's version took and 80.5% of the time that the medial-axis detection took, as shown in Fig. 5.

The first tested approach applied the GA to search the optimal path directly on the binary image of the plan of the environment, that means that all the pixels of the *white* area in the Fig. 3 (around $617 \cdot 10^3$ points for a *1000x700* image) compose the fitness function domain for the GA. This first approach took around 46 minutes to execute. In the second test, the GA was applied to the obstacles dilated image, reducing the *white* area therefore the fitness function domain to $465 \cdot 10^3$ points, likewise the execution time was reduced to around 35 minutes. None of those approaches had acceptable execution times, then a third approach was proposed using the skeleton of the image instead of all the possible navigable area. This last approach reduced the domain to just $2.5 \cdot 10^3$ points and the execution time to around 1 minute 46 seconds.

As previously said, the overall algorithm took around 1 minute 46 seconds finding an optimal path with the parameters given to the GA. This time is the median of 10 test done with the same conditions, where only the starting and the ending points were changed.

Four different results are shown in Figures from 9 to 12, where for all the cases the starting point is the same, but the final point was changed to the different rooms in the environment plan. As shown in Fig. 11 it is specifically difficult for the GA to find valid and short paths when the starting and the ending points are near. Fig. 11 shows how the path took king of wrong direction and came back to the correct path, and also taking around 4 minutes to reach the convergence. This strange behaviour probably happens due to the fixed number of points (always 8) configured in the GA, which did not present problems for larger paths because in those there are more space between points.

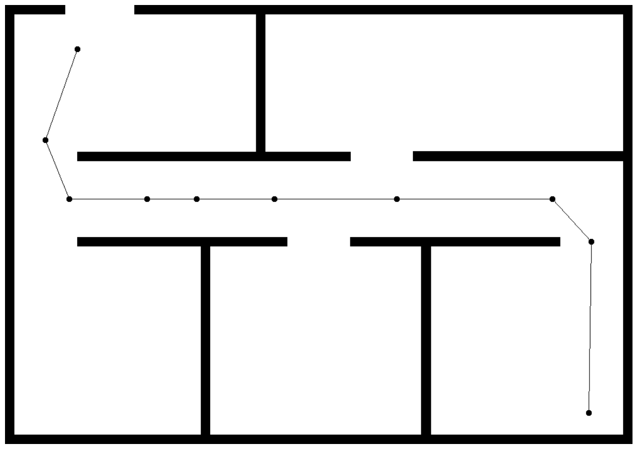
Fig. 9. Resultant Optimized Image.

IV. Conclusion

The speed of Zhang skeletonization algorithm makes it feasible to be applied on environments with moving obstacles, due to the GA would not take very much time recalculating the new image skeleton when any of the obstacles moves.

The execution times of the proposed strategy were reduced from 95% to 96%, reducing the fitness function domain by means of using the image skeleton instead of the original image or even the image with the obstacles and walls dilated. This significantly reduction makes this proposal a hundred percent applicable for static obstacles environments and gets close to real time applications.

As a technical recommendation, the execution time is able to be reduced changing the input parameters of the GA in order to accelerate its convergence, but having the risk to obtain a local minimum (not exactly the "best" as shown in Fig. 11).

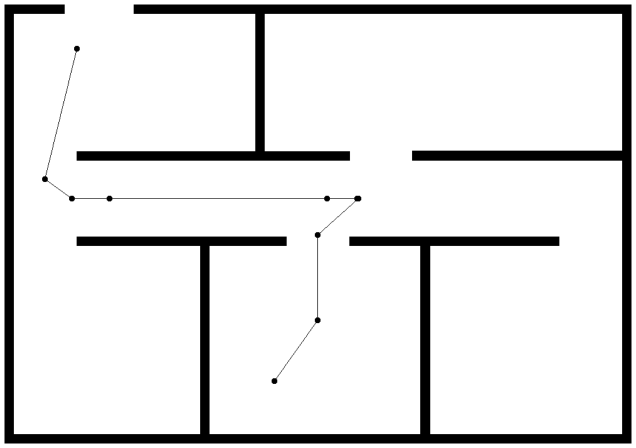
Fig. 10. Resultant Optimized Image, Case 2.

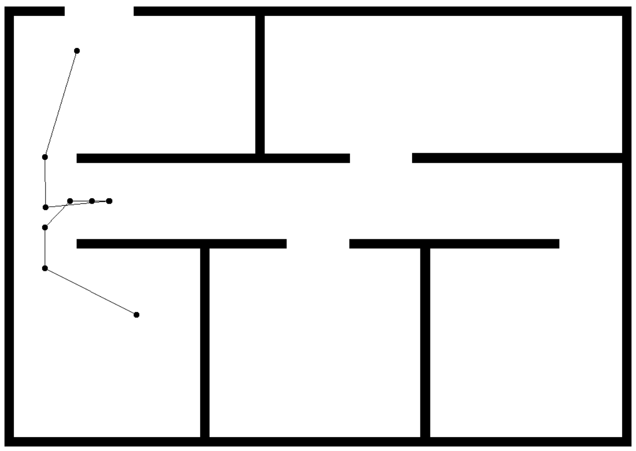
Fig. 11. Resultant Optimized Image, Case 3.

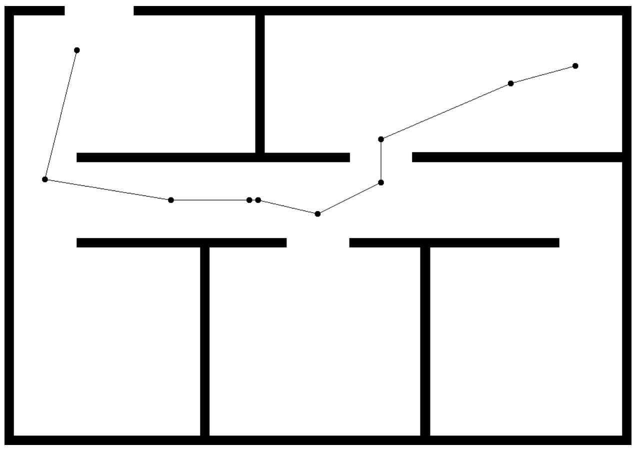
Fig. 12. Resultant Optimized Image, Case 4.

As future work, an adaptable and automatic change of the total number of points of the path is proposed in order improve the convergence times and the optimal result when the starting and ending points are close as the example shown in Fig. 11.

Acknowledgment

This work was supported by Universidad Distrital Francisco José de Caldas, specifically by the Technology Faculty. The views expressed in this article are not necessarily endorsed by Universidad Distrital. The authors thank the ARMOS research group for the given technical support and resources.

References

[1] R. Manzoor and N. Kumar, "Mobile robot path planning approaches: Recent developments," in *Innovations in Information and Communication Technologies (IICT-2020)*, P. K. Singh, Z. Polkowski, S. Tanwar, S. K. Pandey, G. Matei, and D. Pirvu, Eds. Cham: Springer International Publishing, 2021, pp. 301–308.

[2] M. N. Zafar and J. Mohanta, "Methodology for path planning and optimization of mobile robots: A review," *Procedia computer science*, vol. 133, pp. 141–152, 2018.

[3] L. Blasi, E. D'Amato, M. Mattei, and I. Notaro, "Path planning and real-time collision avoidance based on the essential visibility graph," *Applied Sciences*, vol. 10, no. 16, p. 5613, 2020.

[4] J. Krejsa and S. Vechet, "Determination of optimal local path for mobile robot," in *Mechatronics 2017*, T. Březina and R. Jabłoński, Eds. Cham: Springer International Publishing, 2018, pp. 637–643.

[5] T. T. Mac, C. Copot, D. T. Tran, and R. De Keyser, "A hierarchical global path planning approach for mobile robots based on multi-objective particle swarm optimization," *Applied Soft Computing*, vol. 59, pp. 68–76, 2017.

[6] F. M. Santa, S. O. Rivera, and M. A. Saavedra, "Enfoque de navegación global para un robot asistente," *Tecnura*, vol. 21, no. 51, p. 105, 2017.

[7] B. Patle, A. Pandey, D. Parhi, A. Jagadeesh *et al.*, "A review: On path planning strategies for navigation of mobile robot," *Defence Technology*, vol. 15, no. 4, pp. 582–606, 2019.

[8] R. M. C. Santiago, A. L. De Ocampo, A. T. Ubando, A. A. Bandala, and E. P. Dadios, "Path planning for mobile robots using genetic algorithm and probabilistic roadmap," in *2017IEEE 9th international conference on humanoid, nanotechnology, information technology, communication and control, environment and management (HNICEM)*. IEEE, 2017, pp. 1–5.

[9] X. Liang, P. Jiang, and H. Zhu, "Path planning for unmanned surface vehicle with dubins curve based on ga," in *2020 Chinese Automation Congress (CAC)*. IEEE, 2020, pp. 5149–5154.

[10] M. Brand, M. Masuda, N. Wehner, and X.-H. Yu, "Ant colony optimization algorithm for robot path planning," in *2010 international conference on computer design and applications*, vol. 3. IEEE, 2010, pp. V3–436.

[11] K. Akka and F. Khaber, "Mobile robot path planning using an improved ant colony optimization," *International Journal of Advanced Robotic Systems*, vol. 15, no. 3, p. 1729881418774673, 2018.

[12] K. Su, Y. Wang, and X. Hu, "Robot path planning based on random coding particle swarm optimization," *International journal of advanced computer science and applications*, vol. 6, no. 4, pp. 58–64, 2015.

[13] H. Mo and L. Xu, "Research of biogeography particle swarm optimization for robot path planning," *Neurocomputing*, vol. 148, pp. 91–99, 2015.

[14] X. Li, D. Wu, J. He, M. Bashir, and M. Liping, "An improved method of particle swarm optimization for path planning of mobile robot," *Journal of Control Science and Engineering*, vol. 2020, 2020.

[15] A. Tharwat, M. Elhoseny, A. E. Hassanien, T. Gabel, and A. Kumar, "Intelligent bézier curve-based path planning model using chaotic particle swarm optimization algorithm," *Cluster Computing*, vol. 22, no. 2, pp. 4745–4766, 2019.

[16] H.-y. Zhang, W.-m. Lin, and A.-x. Chen, "Path planning for the mobile robot: A review," *Symmetry*, vol. 10, no. 10, p. 450, 2018.

[17] A. Pandey and D. R. Parhi, "Optimum path planning of mobile robot in unknown static and dynamic environments using fuzzy-wind driven optimization algorithm," *Defence Technology*, vol. 13, no. 1, pp. 47–58, 2017.

[18] S. A. Zanlongo, L. Bobadilla, and Y. T. Tan, "Path-planning of miniature rovers for inspection of the hanford high-level waste double shell tanks," in *Florida Conference on Recent Advances in Robotics (FCRAR)*, 2017.

[19] J. D. Contreras, F. Martínez *et al.*, "Path planning for mobile robots based on visibility graphs and a* algorithm," in *Seventh International Conference on Digital Image Processing (ICDIP 2015)*, vol. 9631. SPIE, 2015, pp. 345–350.

[20] S. B. Mane and S. Vhanale, "Genetic algorithm approach for obstacle avoidance and path optimization of mobile robot," in *Computing, Communication and Signal Processing*, B. Iyer, S. Nalbalwar, and N. P. Pathak, Eds. Singapore: Springer Singapore, 2019, pp. 649–659.

[21] N. Adzhar, Y. Yusof, and M. A. Ahmad, "A Review on Autonomous Mobile Robot Path Planning Algorithms," *Advances in Science, Technology and Engineering Systems Journal*, vol. 5, no. 3, pp. 236–240, 2020.

[22] C. Lamini, S. Benhlima, and A. Elbekri, "Genetic algorithm based approach for autonomous mobile robot path planning," *Procedia Computer Science*, vol. 127, pp. 180–189, 2018.

[23] N. Roy, R. Chattopadhay, A. Mukherjee, and A. Bhuiya, "Implementation of image processing and reinforcement learning in path planning of mobile robots," *International Journal of Engineering Science*, vol. 15211, 2017.

[24] S. Luo, Y. Singh, H. Yang, J. H. Bae, J. E. Dietz, X. Diao, and B.-C. Min, "Image processing and model-based spill coverage path planning for unmanned surface vehicles," in *OCEANS 2019 MTS/IEEE SEATTLE*. IEEE, 2019, pp. 1–9.

[25] D. H. Ko, A. U. Hassan, S. Majeed, and J. Choi, "Skelgan: A font image skeletonization method," *Journal of Information Processing Systems*, vol. 17, no. 1, pp. 1–13, 2021.

[26] X. Bai, L. Ye, J. Zhu, L. Zhu, and T. Komura, "Skeleton filter: A self-symmetric filter for skeletonization in noisy text images," *IEEE Transactions on Image Processing*, vol. 29, pp. 1815–1826, 2019.

[27] M. Nazarkevych, S. Dmytruk, V. Hrytsyk, O. Vozna, A. Kuza, O. Shevchuk, Y. Voznyi, I. Maslanych, and V. Sheketa, "Evaluation of the effectiveness of different image skeletonization methods in biometric security systems," *International Journal of Sensors Wireless Communications and Control*, vol. 11, no. 5, pp. 542–552, 2021.

[28] T. Y. Zhang and C. Y. Suen, "A fast parallel algorithm for thinning digital patterns," *Communications of the ACM*, vol. 27, no. 3, pp. 236–239, 1984.

[29] T.-C. Lee, R. L. Kashyap, and C.-N. Chu, "Building skeleton models via 3-d medial surface axis thinning algorithms," *CVGIP: Graphical Models and Image Processing*, vol. 56, no. 6, pp. 462–478, 1994.

[30] F. M. Santa, E. J. Gomez, and H. M. Ariza, "Global path planning for mobile robots using image skeletonization," *Indian J. Sci. Technol*, vol. 10, p. 14, 2017.