*applied sciences*

MDPI

*Article*

# A-Star (A*) with Map Processing for the Global Path Planning of Autonomous Underwater and Surface Vehicles Operating in Large Areas

**Rafał Kot [1], Piotr Szymak [1], Paweł Piskur [1,*] and Krzysztof Naus [2]**

1 Faculty of Mechanical and Electrical Engineering, Polish Naval Academy, 81-127 Gdynia, Poland; r.kot@amw.gdynia.pl (R.K.); p.szymak@amw.gdynia.pl (P.S.)

2 Faculty of Navigation and Naval Weapon, Polish Naval Academy, 81-127 Gdynia, Poland; k.naus@amw.gdynia.pl

* Correspondence: p.piskur@amw.gdynia.pl

**Abstract:** The global path planning system is one of the basic systems ensuring the autonomous operation of unmanned underwater vehicles (UUVs) and unmanned surface vehicles (USVs) in a complex aquatic environment. The A* path planning algorithm is one of the most well-known algorithms used to obtain an almost optimal path, avoiding obstacles even in a complex environment containing objects with specific shapes and non-uniform arrangements. The main disadvantage of this algorithm is the computational cost of path calculation. This article presents a new approach based on the image processing of the map before determining the path using A*. The results of numerical research based on a large-sized map expressing the port area confirm the proposed method's effectiveness, which reduces the calculation time by over 500 times with a slight increase in the path length compared to the basic version of the A* algorithm. Based on the obtained results, the proposed approach also increases the path's safety by designating narrow and risky areas as closed to vehicle movement. For this reason, the method seems suitable for use in global path planning for autonomous underwater vehicles (AUVs) and autonomous surface vehicles (ASVs) operating in large areas.

**Keywords:** path planning; collision avoidance; autonomous underwater vehicle; autonomous surface vehicle; image processing

check for updates

**Citation:** Kot, R.; Szymak, P.; Piskur, P.; Naus, K. A-Star (A*) with Map Processing for the Global Path Planning of Autonomous Underwater and Surface Vehicles Operating in Large Areas. *Appl. Sci.* **2024**, *14*, 8015. https://doi.org/10.3390/app14178015

Academic Editor: Zaopeng Dong

Received: 1 August 2024
Revised: 30 August 2024
Accepted: 5 September 2024
Published: 7 September 2024

## 1. Introduction

The global path planning system is one of the basic high-level control systems for autonomous underwater vehicles (AUVs) and autonomous surface vehicles (ASVs), allowing for the autonomous operation of the vehicle in a known underwater environment and the optimisation of the tracked trajectory in terms of energy. At the mission planning stage for an AUV/ASV, it is necessary to consider all information about the area where the vehicle will operate. Having a map of the environment in which the vehicle will be moving, the safe path can be designed between the entered coordinates. AUV/ASV path planning based on a known environment is usually performed by specifying waypoints to prevent the vehicle from collisions, assuming straight-line movement between subsequent coordinates. In a complex environment, determining multiple coordinates to achieve mission objectives, including a safe AUV/ASV route, may be time-consuming, and the path may be far from optimal in terms of length or power energy consumption. Therefore, the path planning problem requires additional analysis to automate the mission planning system. Another challenge is planning the optimal path regarding length and safety in large areas. Over the years, many path planning methods have been developed, the effectiveness of which has been verified by simulations and experiments [1]. It is worth noting that path planning can be carried out globally or locally. A smaller area is usually analysed for local trajectory determination rather than for global planning. Therefore, some methods that work effectively in local collision avoidance systems may be ineffective in applications of global path planning systems.

This article presents a new approach focused on calculating the path using the standard A* method with map preprocessing and postprocessing. The algorithm is designed to overcome the shortcomings resulting from the large dimensions of the area in which the path is calculated. The proposed method involves reducing the map's dimensions by a known factor and then performing an image processing operation, determining the path based on the received map, and multiplying the obtained coordinates by the same factor. The map preparation and postprocessing procedure used in the proposed algorithm is straightforward to implement and significantly reduces the computational cost while maintaining satisfactory path length and safety levels from the view of mission planning for AUVs/ASVs.

The specific research objectives of this study are as follows:

- To develop a novel approach for global path planning by integrating the A* algorithm with image processing techniques aimed at reducing computation time;
- To validate the effectiveness of the proposed method in large-scale environments through numerical simulations;
- To compare the performance of the proposed method with the standard A* algorithm in terms of path safety, length, and computation time.

The article is organised as follows: Section 2 presents the analysis of solutions related to improving the efficiency of the A* algorithm. Section 3 describes the standard and proposed approach to global path planning using the A* method and specifies the algorithm for creating an environment map. Section 4 discusses the research results based on a large-dimensional map of the underwater environment located in the port area. The final conclusions are discussed in Section 5.

## 2. Related Works

Recently, a significant increase in the use of artificial intelligence methods for path planning has been observed [2]. This is related to current technological development. An example of such methods can be artificial neural network-based methods such as deep reinforcement learning [3–7]. Popular path planning methods also include those based on animal behaviour and evolution, such as ant colony (ACO) [8], particle swarm optimisation (PSO) [9,10], and genetic algorithms (GAs) [11,12], which are still being modified to increase their effectiveness. Another group of intelligent path planning methods includes heuristic methods. The most popular example of such a method is the A* algorithm. It allows for finding the shortest path to a destination in the presence of static obstacles, assuming that both the environment and the location of the obstacles are known. For this purpose, a heuristic function is used to calculate the path cost, assuming passing through individual map cells. In this method, an increased amount of computation is needed for large areas of analysis or areas with many obstacles. This significantly increases the path planning time.

The efficiency of the A* method can be improved by combining it with other efficient algorithms, such as GA for pre-determining the effective path [13], or in combination with the visibility graph-based space representation [14] instead of occupancy grid representation, which significantly reduces the number of nodes to search and substantially speeds up the algorithm. Since the visibility graph connects obstacles' vertices that can see each other, the path determined based on these vertices will be near-optimal in terms of the path length. On the other hand, the successive waypoints border on an obstacle. From a practical point of view, calculating a path based on a visibility graph for AUVs/ASVs is very risky due to the error of navigation positioning systems and the possible occurrence of sea currents or unfavourable weather conditions. The proposed method utilises occupancy grid representation. Mathematical morphology-based processing allows for the regulation of the safety zone around an obstacle and the closure of very narrow passages or trap obstacles. It determines a path that will be more practical and feasible for AUVs/ASVs to use than the one calculated based on the visibility graph. Additionally, a visibility graph is hard to implement in 3D space due to the difficulties in representing and parameterising lines in 3D space and the loss of separability [15–17]. In the case of space representation as an occupancy grid, this problem does not occur, and the cells are clearly described. For this reason, implementing map processing using mathematical morphology, e.g., a cube-shaped structural element, is a relatively straightforward process. Therefore, the map preprocessing approach presented in this article could increase the relative gain in path planning efficiency.

The disadvantage of the proposed approach compared to the visibility graph is the loss of information about the exact shape of obstacles, which in some situations may result in difficult access to cells located very close to obstacles. Excessive resolution reduction may cause significant map distortions, making it impossible to determine the path or leading to an excessive increase in the path length (depending on the set mathematical morphology parameters). Additionally, in the case of places where the target point is located inside an almost closed obstacle, it may result in fully closing the obstacle and making it impossible to determine the path. For this reason, the algorithm is unsuitable for calculating paths, e.g., in narrow mazes. However, the algorithm's main purpose is to compute a path based on maps representing a large area of the real environment.

Over the years, many modifications have been proposed to improve the efficiency of the standard A* solution, such as JPS (A*) [18], variable-step based A* [19], geometric A* [20], EBHSA* algorithm [21], IMOA* [22], RJA* [23], ORMBA* [24], APF-A* [25], and other improvements [26–35]. The comprehensive review of the latest literature on the methods aimed at optimising the A* algorithm is presented in Table 1. Each approach modifies the basic A* algorithm to improve the computational efficiency, obtain a shorter or smoother path, or improve path safety. A significant improvement in one of these parameters does not allow for a significant improvement in the other (JPS-A*, Geometric A*). Therefore, each algorithm optimisation is associated with a compromise between computational performance and the calculated path length. In most of the studies presented in Table 1, tests were carried out on relatively small maps. In most cases, computational performance was not improved significantly, or the algorithm did not consider the vehicle's safety. The A* algorithm with the map processing approach presented in this article primarily focuses on reducing the computation time and providing a safety path feasible for AUVs/ASVs.

**Table 1.** A comparison of modifications of the A* algorithm available in the literature.

| Source | Method | Brief Description | Map Size [pixels] | Path Length (Proposed/Basic A*)—The Worst Cases) | Max. Calculation Time Ratio (Basic A*/Proposed)—The Best Cases) |
|---|---|---|---|---|---|
| [18] | JPS (A*) | JPS A* speeds up pathfinding by expanding only pivotal nodes called "jump points", which are directly reachable in a straight line or after encountering forced neighbours, bypassing redundant intermediate nodes. | From 30 × 21 To 1104 × 1260 | 100% | 215 |
| [19] | Variable Step A* | The Variable Step A* algorithm dynamically adjusts its search step size based on obstacle distribution to optimise pathfinding efficiency and accuracy for autonomous land vehicles. | NA | 97.8% | 7.8 (based on the difference in expanded points) |

**Table 1.** *Cont.*

| Source | Method | Brief Description | Map Size [pixels] | Path Length (Proposed/Basic A*)—The Worst Cases) | Max. Calculation Time Ratio (Basic A*/Proposed)—The Best Cases) |
|---|---|---|---|---|---|
| [20] | Geometric A* | The Geometric A* algorithm generates a grid map for initial pathfinding with A*, optimises by eliminating irregular nodes for smoother turns, and then applies cubic B-spline interpolation to smooth the path for realistic navigation. | 20 × 20<br>50 × 50<br>100 × 100 | 58.9% | 1.6 |
| [21] | EBHSA* | The EBHSA* method enhances the A* algorithm by integrating expansion distance, using bidirectional search, optimising heuristic function, and smoothing to improve path robustness and efficiency while reducing collision risks and unnecessary turns for autonomous land vehicles. | 50 × 50<br>100 × 100<br>200 × 200 | N/A | 30 |
| [22] | IMOA* | IMOA-star generates and stores the obstacle map as a pickle file, eliminating the need to recreate it for future path planning in the same workspace and thus significantly reducing computation time. The algorithm also incorporates a path-problem-aware executor to refine the path, reducing its length and improving smoothness before the final output. | 7120 × 9490 | 98.3% | 1 (first use)<br>8440 (next use) |
| [23] | RJA* | The RJA-star algorithm improves UAV path planning by detecting obstacles, selecting the closest one as a coercive neighbour, and generating jump points at key vertices to optimise the path. The algorithm iteratively evaluates these jump points, computes the cost for each potential path, and guides the UAV along the most efficient route until the target is reached. | 15 × 30 × 15<br>to<br>100 × 100 × 15 | 97.7% | 42.5 |
| [30] | Adaptive A* algorithm | The adaptive A-star algorithm introduces adaptive weights in the heuristic function to optimise path quality, while the improved dynamic window approach (DWA) adds a trajectory point estimation function, enhancing obstacle avoidance and path smoothness, enabling effective global path planning and dynamic obstacle avoidance. | 30 × 30 | 80.1% | 1.8 |
| [31] | Optimised A* algorithm | The optimised A* algorithm involves using dynamic weight coefficients, allowing for the adjustment of search priorities based on conditions, and integrating Bezier curves to smooth the generated path, enhancing its fluidity and efficiency. | 10 × 10 | 71.1% | 1.8 |

**Table 1.** *Cont.*

| Source | Method | Brief Description | Map Size [pixels] | Path Length (Proposed/Basic A*)—The Worst Cases) | Max. Calculation Time Ratio (Basic A*/Proposed)—The Best Cases) |
|---|---|---|---|---|---|
| [32] | Improved A* algorithm | The algorithm was optimised by implementing a hybrid data structure combining a minimum heap with a 2D array, reducing the time complexity of data processing. Additionally, the search strategy was refined by deferring the end-point check until later in the process, significantly improving execution efficiency. | 13,139 × 13,245 | 100% | 551 |
| [33] | Improved A* algorithm | The algorithm was optimised by introducing a method to merge adjacent small segments of the path and using the shortest line segment between two points to guide path planning. | 20 × 20 | 96.7% | 0.3 |
| [34] | Improved A* algorithm | The algorithm's improvements include extending the search neighbourhood in the traditional A-star algorithm to increase path smoothness and reduce redundant turning points, as well as integrating an enhanced DWA algorithm for better dynamic obstacle avoidance. | 30 × 30 | 97.9% | 2.3 |
| [24] | ORMBA* | The algorithm incorporates two key optimisations: an adaptive cost function that dynamically adjusts the movement cost based on the goal node and an optimised robot motion block (RMB) that reduces unnecessary and redundant node searches. | 261 × 261<br>462 × 261<br>462 × 462 | 102.7% | 140 |
| [26] | Improved A* algorithm | The algorithm includes an improved evaluation function that considers both distance and angular deviation from the optimal path and a bidirectional search strategy that reduces the number of search nodes. Additionally, the algorithm eliminates redundant nodes and smooths the resulting path using cubic uniform B-spline curves. | 30 × 30<br>50 × 50 | 96.5% | 2.8 |
| [27] | Improved A* algorithm | The algorithm features a segmented evaluation function with dynamic heuristic adjustments to balance convergence speed and path quality, a steering cost heuristic that minimises sharp turns, a strategy for removing redundant turning points, and a smoothing process using quasi-uniform cubic B-splines to ensure a smooth, efficient path. | 246 × 200 | 95.2% | 1.7 |

**Table 1.** *Cont.*

| Source | Method | Brief Description | Map Size [pixels] | Path Length (Proposed/Basic A*)—The Worst Cases) | Max. Calculation Time Ratio (Basic A*/Proposed)—The Best Cases) |
|---|---|---|---|---|---|
| [35] | Improved A* algorithm | The improved algorithm includes dynamic weighting in the evaluation function to adapt the search priorities based on the node's location and directional screening of the search neighbourhood using azimuth angles to enhance efficiency. Additionally, the algorithm incorporates a safety radius to avoid obstacles. It employs Bezier curves to smooth the path, resulting in fewer inflexion points and a safer path. | 70 × 60 | 87.7% | 1.2 |
| [25] | APF-A* | APF-A* uses dynamic weight adjustments in the cost function to minimise unnecessary turns by penalising sharp turning angles. It integrates the artificial potential field method to incorporate obstacle information, ensuring that turning points are strategically placed away from obstacles. These optimisations lead to smoother paths with reduced turning points near obstacles, enhancing path stability and reducing the risk of collisions. | 20 × 20<br>40 × 40<br>200 × 150 | 100% | 1.2 |
| **This article** | A* with map processing | The algorithm uses resizing and morphological image processing to reduce the computation time of the A* algorithm by map optimisation. | 3600 × 4025 | 102.8% | 719 |

## 3. Methods

This section presents a general description of the simulation environment, as well as the description of the A* algorithm in its basic form and its modification, which consists of using image processing techniques for the appropriate preparation of the occupancy grid to reduce the computational cost of path calculation.

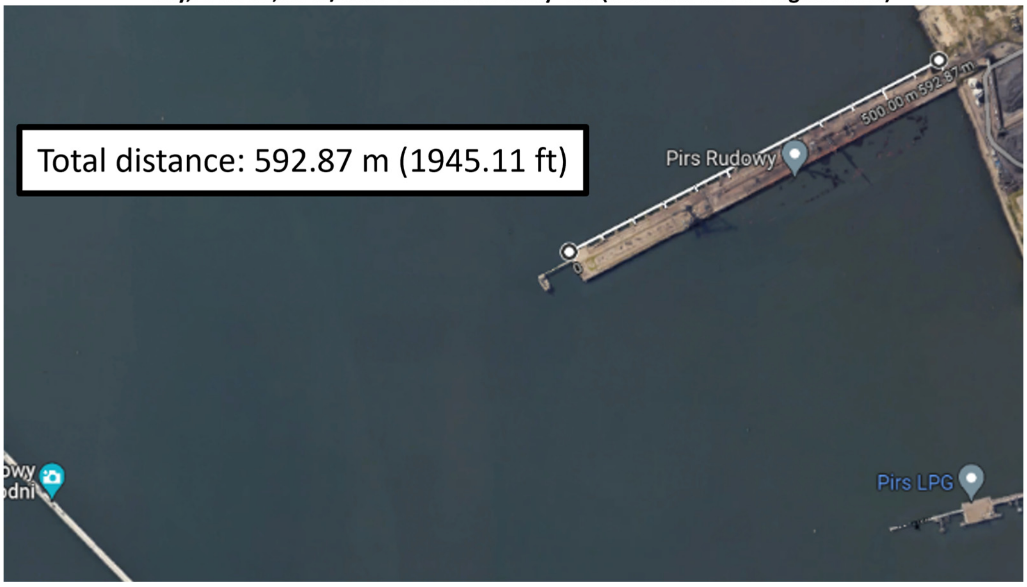
### *3.1. Simulation Environment*

A simulation environment covering the extensive port area of 3600 m × 4025 m was used to test the proposed method regarding the computation time, path length, and path safety compared to the basic version of the A* algorithm. As a reference map, a satellite map of the Northern Port in Gdańsk was used as an RGB image obtained based on the Mapping Toolbox in the MATLAB environment [36].

Based on the actual distance measured from the geographic coordinates of the satellite image vertices, the map was resized to show a 1 m × 1 m natural area as one pixel. The geodetic vertical and horizontal distance between satellite image vertices was calculated taking into account the WGS84 Earth ellipsoid model.

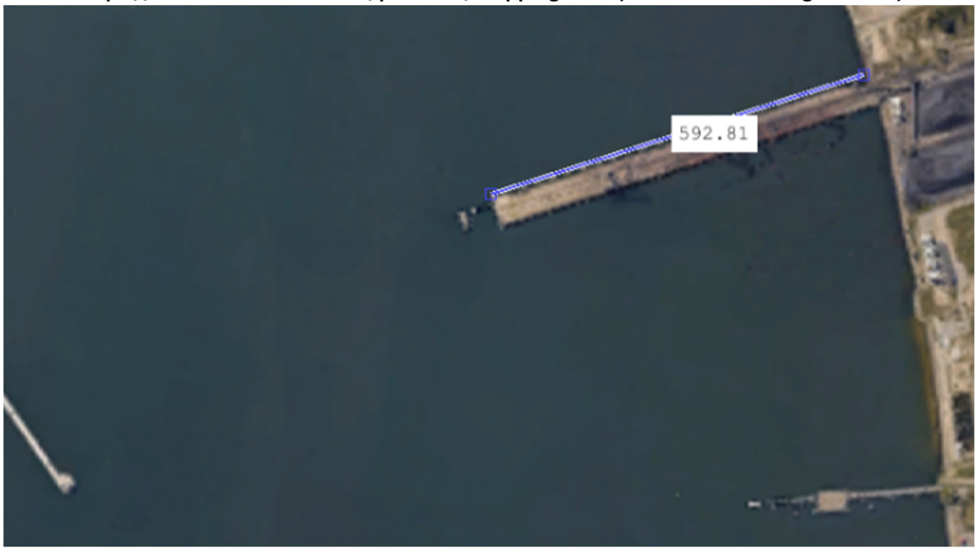
Figure 1 shows the same pier measured in a satellite image and in the base image for the occupancy grid. It can be observed that the length of the pier calculated using Google tools (expressed in meters) is almost the same as that calculated based on Euclidean distance (expressed in pixels).

**Source: Google, CNES/Airbus, MGGP Aero, Mexar Technologies**
**Obtained using Google Maps 2023: https://www.google.pl/maps/@54.3985359,18.7207342,7518a,35y,133.79h,0.72t/data=!3m1!1e3?entry=ttu (accessed on 29 August 2023)**



(**a**)

**Source: Esri, Maxar, Earthstar Geographics and the GIS User Community**
**Obtained using: Matlab software, ver. 9.13.0 (R2022b)**
**https://www.mathworks.com/products/mapping.html (accessed on 29 August 2023)**



(**b**)

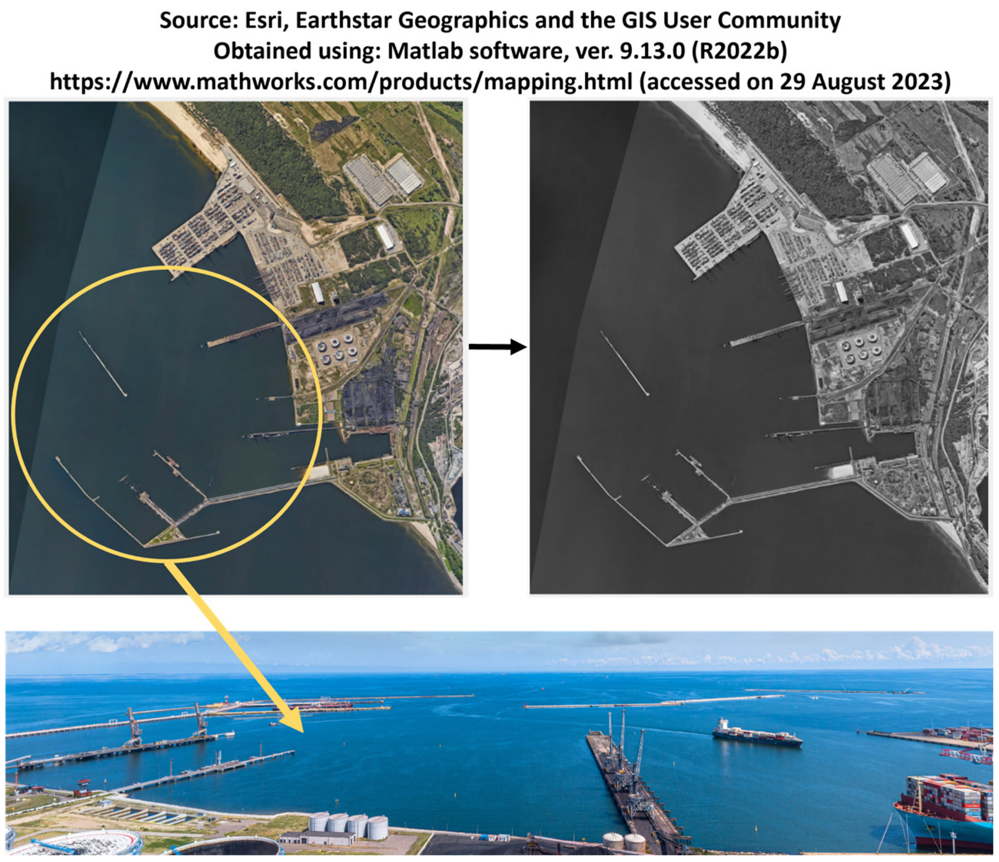
**Figure 1.** An example of the measurement of the same pier: (**a**) in a satellite image obtained from Google Maps [37]; (**b**) input image for occupancy grid with a resolution selected in such a way that 1 pixel corresponds to an area of 1 m$^2$ (distance in pixels).

The image was converted from the RGB system to grayscale (Figure 2) by the following weight relationship [36]:

$$G = 0.2989 \times R + 0.5870 \times G + 0.1140 \times B \quad (1)$$

where R, G, and B are red, green, and blue colour components, respectively, expressed in the range of 0–255.

In the next step of creating the environment for the basic A*, a grayscale image was converted to a binary map representing the occupied and free space based on threshold segmentation. For the proposed modification of the A* algorithm with map processing, the threshold segmentation operation was preceded by a sequence of image processing techniques that will be described in the next subsection. In both approaches, the segmentation threshold was set at 75/255 to obtain an occupancy grid.



**Source: Esri, Earthstar Geographics and the GIS User Community**
**Obtained using: Matlab software, ver. 9.13.0 (R2022b)**
**https://www.mathworks.com/products/mapping.html (accessed on 29 August 2023)**

**Figure 2.** A satellite photo of the Northern Port in Gdańsk and its grayscale representation used as a test environment with an indication of the port's real part in the panorama available in [38]. The area marked with an orange circle on the satellite image corresponds to the real port area indicated by the arrow in the panorama.

Depending on the entered coordinates of the starting and destination points, the path planning algorithm determines the path in the form of subsequent pixel coordinates, considering obstacles in the environment.

## 3.2. Basic A* Algorithm

The A* algorithm concerns finding the shortest path between the start and destination point by calculating the total cost of reaching the target with the following:

$$F(n) = G(n) + H(n) \tag{2}$$

where

n—the next node on the path;
F(n)—the total path cost;
G(n)—the cost of reaching the analysed cell from the starting point;
H(n)—the cost of reaching the target from the analysed cell.

The heuristic function H(n) is usually calculated using the Euclidean distance as follows:

$$H(n) = \sqrt{\left(x_{goal} - x_n\right)^2 + \left(y_{goal} - y_n\right)^2} \tag{3}$$

where ($x_{goal}$, $y_{goal}$) represents the coordinates of a target point, and ($x_n$, $y_n$) represents the coordinates of node n.

A simplified workflow of the basic A* algorithm is shown in Figure 3. First, based on the occupancy grid, depending on the presence of obstacles, environmental map coordinates are saved to an open and closed list. In the main loop, the algorithm selects the node with the lowest total cost F(n) from the open list and evaluates its neighbours, updating their costs if a more optimal path is found. This process repeats until the goal node is selected, indicating that the shortest path has been found, or the open list is empty, signifying that no path exists.

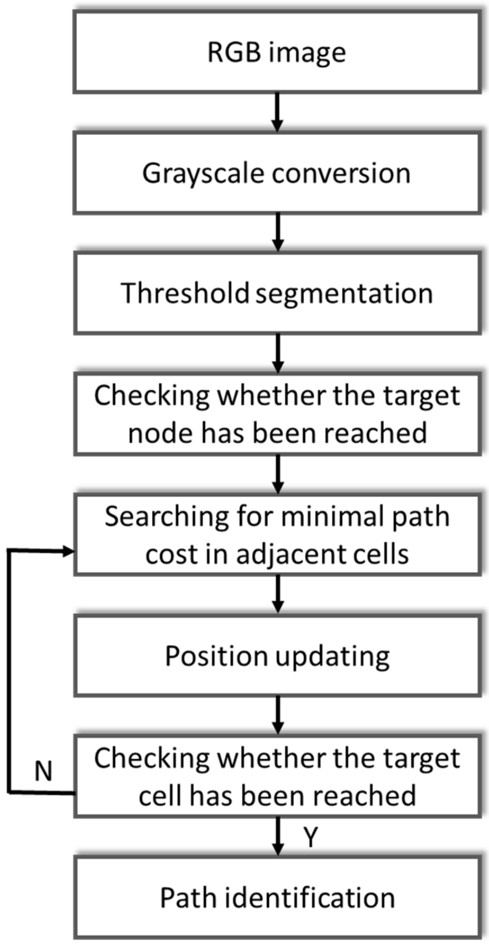

RGB image
Grayscale conversion
Threshold segmentation
Checking whether the target node has been reached
Searching for minimal path cost in adjacent cells
Position updating
Checking whether the target cell has been reached
N
Y
Path identification

**Figure 3.** Workflow of the basic A* algorithm.

The disadvantage of this approach Is the number of calculations performed, which increases significantly for large areas of analysis or a large number of obstacles. Additionally, the algorithm ensures the determination of the shortest path, assuming the presence of known static obstacles, which makes it impossible to use it in its basic form in relation to a dynamic environment.

## *3.3. A* with Map Processing*

In the modified method, the main assumption was to reduce the size of the map, which reduces the number of operations necessary to perform by the A* algorithm. In order to compensate for the loss of data about obstacles resulting from the interpolation of the values of each pixel, image processing methods based on mathematical morphology were used. As a result, the obstacle was artificially enlarged so that, after rescaling, a collision-free path was achieved with an insignificant change in the path length. The general workflow of the improved A* algorithm is shown in Figure 4.

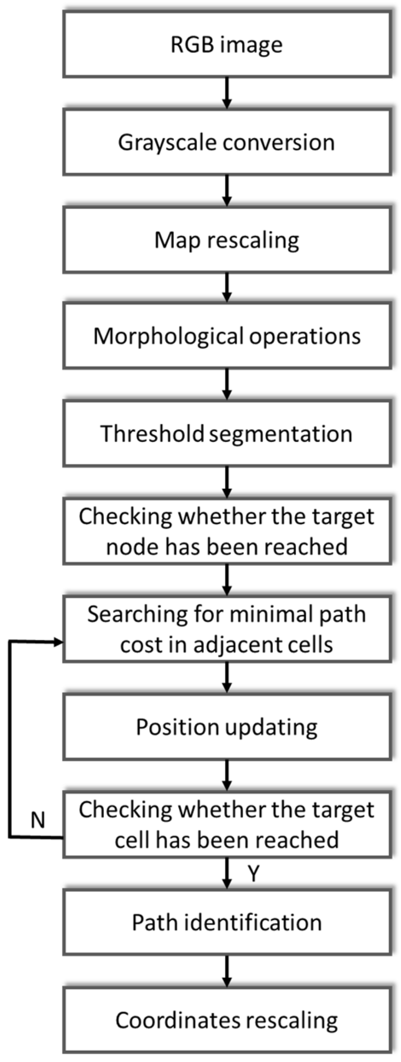

RGB image
Grayscale conversion
Map rescaling
Morphological operations
Threshold segmentation
Checking whether the target node has been reached
Searching for minimal path cost in adjacent cells
Position updating
Checking whether the target cell has been reached
N
Y
Path identification
Coordinates rescaling

**Figure 4.** Workflow of the modified A* algorithm.

In the first step, depending on the entered value of the scaling factor, the map size was reduced using the bicubic interpolation method according to the following formula [36,39]:

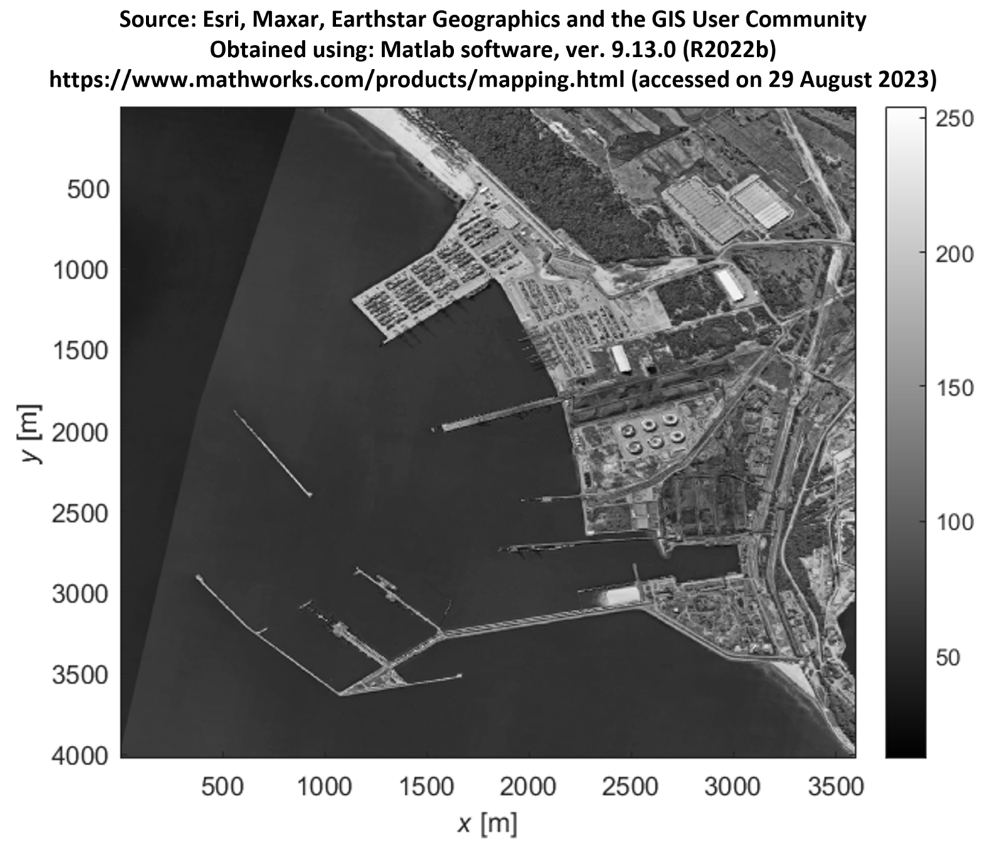
$$I(x,y) = \sum_{i=-1}^{2} \sum_{j=-1}^{2} p(i,j) \times w(x-i) \times w(y-j) \tag{4}$$

where

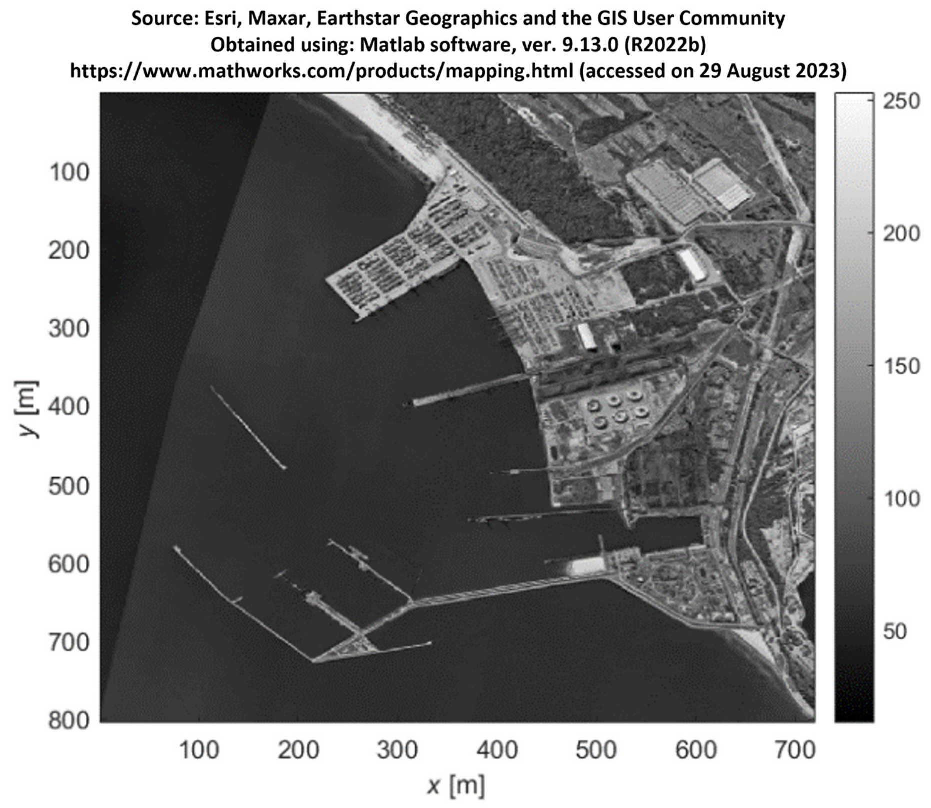
$I(x,y)$—the interpolated pixel value at the point $(x,y)$;
$p(I,j)$—the pixel value in the neighbourhood of the point $(x,y)$ before scaling;
$w(d)$—the weighting function (bicubic interpolation kernel) that defines the influence of neighbouring pixels on the value of $I(x,y)$, calculated as follows:

$$\mathrm{w(d)} = \begin{cases} 1.5|d|^3 - 2.5|d|^2 + 1 & \text{for } |d| \le 1, \\ -0.5|d|^3 + 2.5|d|^2 + 4|d| - 2 & \text{for } 1 < |d| \le 2, \\ 0 & \text{for } |d| > 2. \end{cases} \tag{5}$$

In the analysed case shown in Figure 5, the scaling factor was set to 0.2, which means a five-fold reduction in the map resolution. No significant differences can be observed between the two images.


Source: Esri, Maxar, Earthstar Geographics and the GIS User Community
Obtained using: Matlab software, ver. 9.13.0 (R2022b)
https://www.mathworks.com/products/mapping.html (accessed on 29 August 2023)
y [m]
x [m]
500
1000
1500
2000
2500
3000
3500
4000
250
200
150
100
50

(**a**)


Source: Esri, Maxar, Earthstar Geographics and the GIS User Community
Obtained using: Matlab software, ver. 9.13.0 (R2022b)
https://www.mathworks.com/products/mapping.html (accessed on 29 August 2023)
y [m]
x [m]
100
200
300
400
500
600
700
800
250
200
150
100
50

(**b**)

**Figure 5.** Grayscale image used as a map for comparison of methods: (**a**) at original resolution; (**b**) after rescaling.

Then, morphological operations such as dilation and closing were performed with a square-shaped structural element with dimensions of 4 × 4 pixels (Figure 6). The value of the structural element was selected empirically to ensure a safe collision-free path.

| | | | |
|---|---|---|---|
| 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 |

**Figure 6.** Structural element used for morphological operations.

Dilation involves adding pixels to the boundaries of obstacles in an image. For grayscale images, it can be described by the following formula:

$$(\mathrm{A} \oplus \mathrm{B})(x,y) = \max(i,j) \in \mathrm{B}\ \{\mathrm{A}(x - i, y - i)\} \tag{6}$$

where

A—dilated image;
B—structuring element which defines the neighbourhood of the pixel (x,y) being processed;
(x,y)—coordinates of the pixel of a dilated image;
(i,j)—the coordinates of the pixels within the structuring element.

The above operation expanded the obstacle to prevent the calculation of route coordinates, which, after being scaled to the original size, was performed through the area occupied by the obstacle. Then, a closing operation was performed consisting of dilation followed by morphological erosion, expressed by the following formula:

$$(\mathrm{A} \cdot \mathrm{B})(\mathrm{x},\mathrm{y}) = (\mathrm{A} \oplus \mathrm{B}) \ominus \mathrm{B} \tag{7}$$

The morphological erosion operation is expressed by the following formula:

$$(\mathrm{I} \ominus \mathrm{B})(\mathrm{x},\mathrm{y}) = \min(\mathrm{i},\mathrm{j}) \in \mathrm{B}\ \{\mathrm{I}(\mathrm{x} + \mathrm{i}, \mathrm{y} + \mathrm{j})\} \tag{8}$$

The closing operation is useful for filling small holes in an image and removing image noise such as single high-intensity pixels while preserving the shape and size of large gaps and obstacles in the image. A comparison of the original grayscale image with the image obtained after image processing using mathematical morphology operations is shown in Figure 7. After performing the above processing, the image was segmented with a threshold of 75/255 to obtain an occupancy grid.

The type of morphological operation, size, and shape of the structural element significantly impact the result of map processing. Reducing the map causes the effect known as the "blurring" of pixels, which, after applying threshold segmentation, may increase the loss of information about obstacles in the occupancy grid. Therefore, in order to protect the A* algorithm from determining the path through the area occupied by the obstacles on the original map, the obstacle is artificially expanded as a result of dilation. Additionally, erosion is used to filter random noise in the image. If, for example, a disc-shaped structural element is used, some information about obstacles will be lost in places where the values of the structural element are equal to 0 (Figure 8—black pixels). On the other hand, a larger structural element size will result in an excessive safety zone, significantly extending the route planned by the A* algorithm. A similar situation will occur in the case of a diamond-shaped structural element (Figure 8). Other examples of structural elements may be a line, a rectangle, or an octagon. However, in this case, the obstacle must be stretched equally in x and y coordinates, so a square was chosen from among the various shapes.

The size of the structural element must be appropriately selected to compensate for losses resulting from averaging pixel values in the process of reducing resolution. If the map is reduced five times, 1 pixel represents an area of 5 × 5 pixels relative to the original map size. Therefore, the structural element must have a size at least equal to the number of pixels that 1 pixel represents in the original map, i.e., 5 × 5 pixels. In the proposed approach, a structural element in the shape of a square 4 × 4 pixels was chosen in order to use the same structural element in all morphological operations, which are dilation and closing. Since a double dilatation was performed (the closing operation consisted of dilation as well), the above condition was met.

**Source: Esri, Maxar, Earthstar Geographics and the GIS User Community**
**Obtained using: Matlab software, ver. 9.13.0 (R2022b) https://www.mathworks.com/products/mapping.html (accessed on 29 August 2023)**

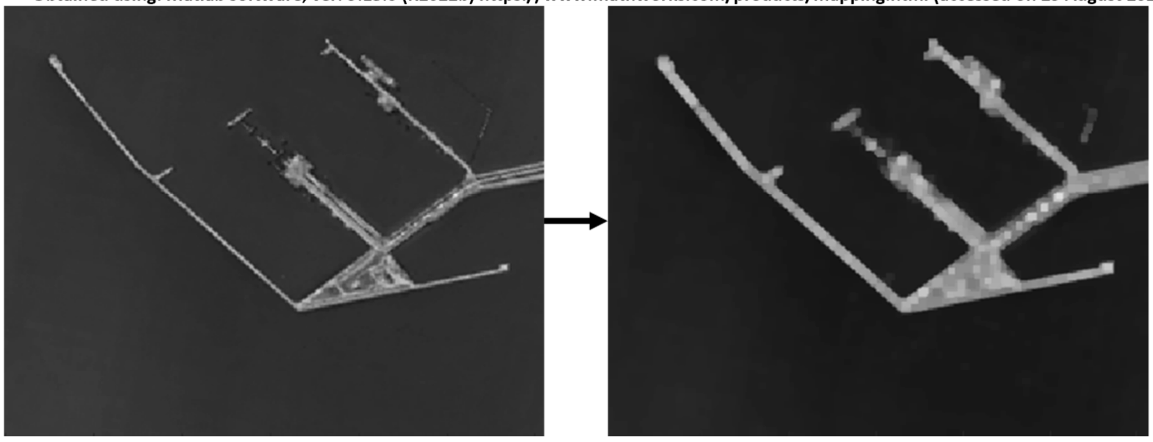


**Figure 7.** An example of map processing using mathematical morphology operations.

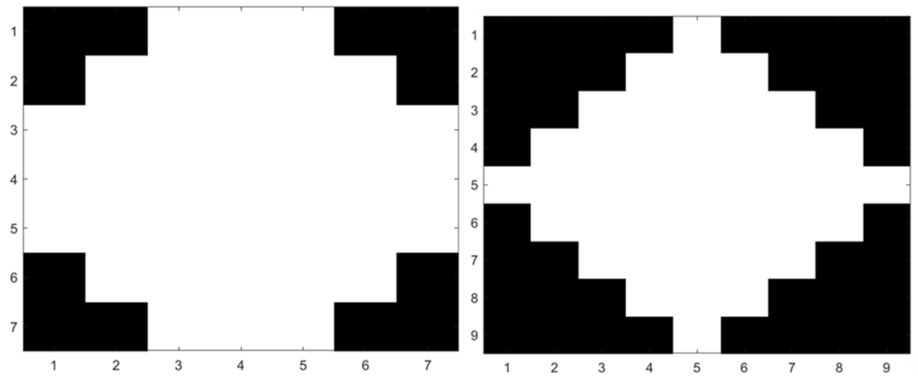


**Figure 8.** Examples of disc- and diamond-shaped structural elements.

Examples of the same map fragment using different scaling factors and different structural elements for the same mathematical morphology operations are presented in Table 2.

For a disc-shaped structural element with a radius of 4 pixels, the matrix used in mathematical morphology-based processing had dimensions of 7 × 7 pixels, while for a diamond-shaped with the distance from the origin to the point of diamond r = 4, the matrix had dimensions of 9 × 9 pixels. It can be seen that a greater reduction in map resolution for the same size of structural elements leads to greater deformations of the original shape and excessive enlargement of obstacles. Additionally, the use of structural elements such as a disc or a diamond of the sizes shown above excessively stretches the obstacles in relation to square-shaped structural elements. On the other hand Table 3 shows a comparison of the same structural elements with the dimensions selected in such a way that the matrix used in the mathematical morphology operation was 5 × 5 pixels in relation to the processing results obtained for a square-shaped structural element with dimensions of 4 × 4 pixels.

Due to the fact that for r = 2, the disc-shaped and diamond-shaped structural element matrices look the same, the results are presented in one column. Based on the processing results, it can be seen that a disc-shaped structural element with a radius of 2 provides a smaller safety zone than a square-shaped one (for both sizes). It is worth noting that for r = 3, the disc-shaped element gives the same matrix as the square-shaped element with dimensions of 5 × 5 pixels. Moreover, for r = 3, a diamond-shaped element gives a matrix with dimensions of 7 × 7 pixels. Therefore, a square-shaped element with dimensions of 4 × 4 pixels was selected for further processing.

**Table 2.** A comparison of the same map fragments obtained as a result of processing with different parameters.

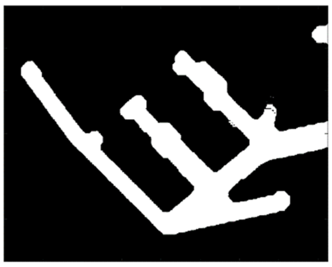
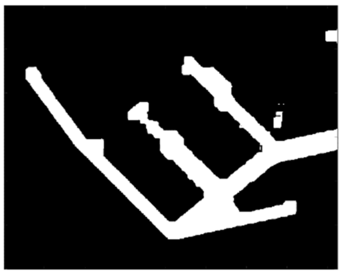
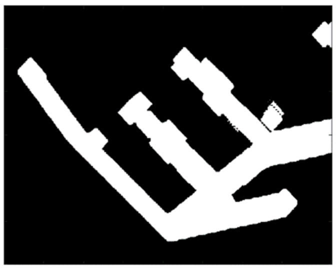
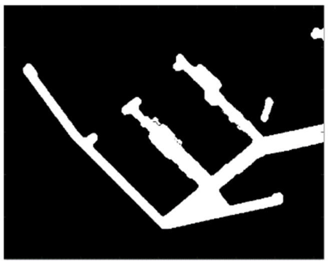
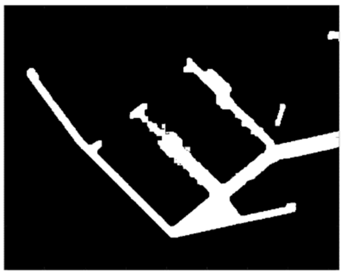
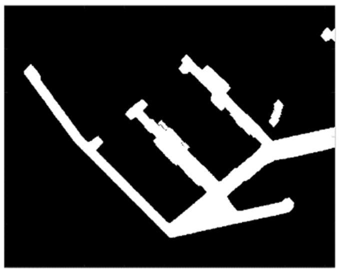
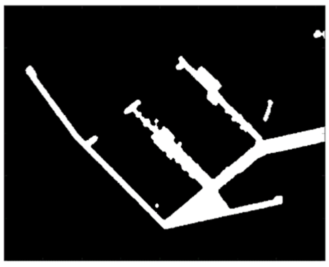
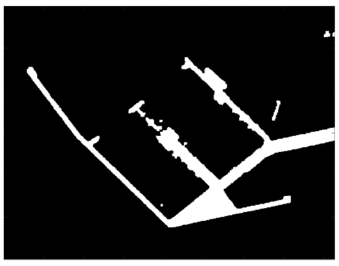
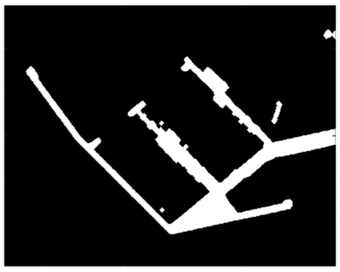
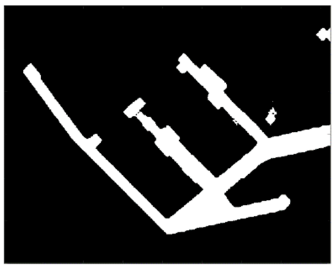
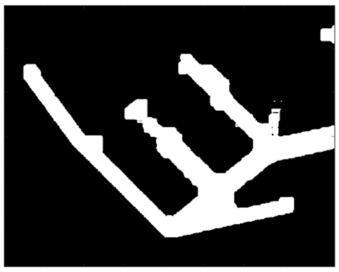
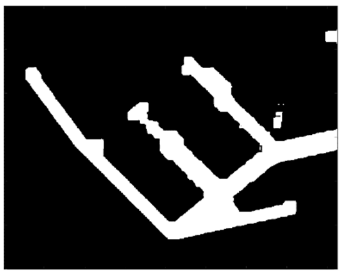
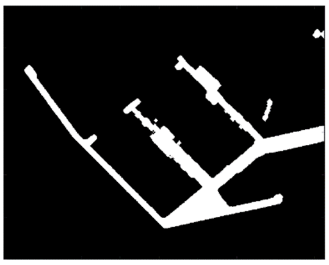
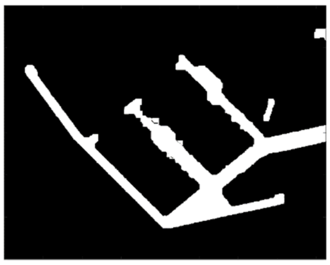
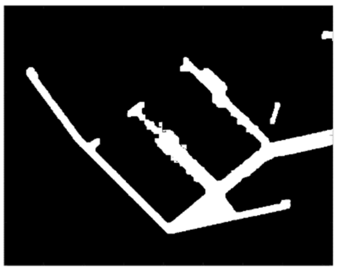
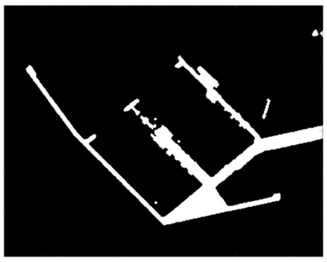
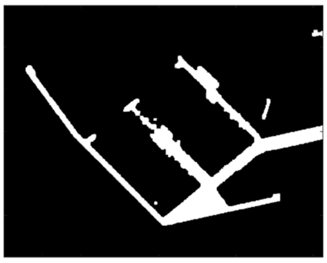
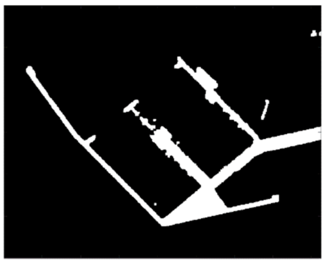
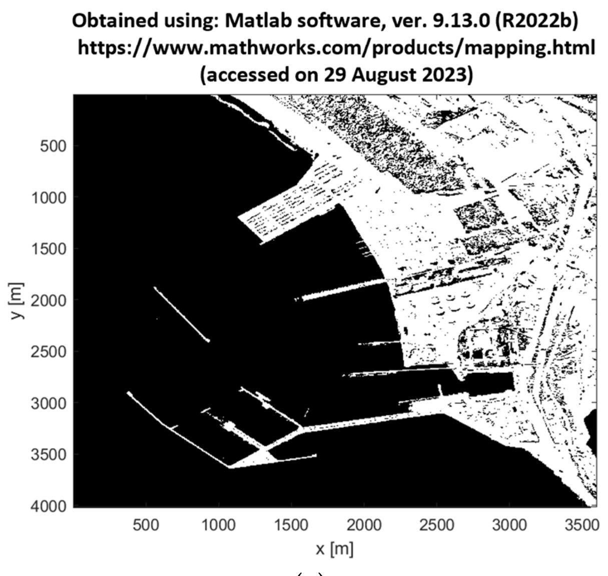
| Morphological Operations: Dilation + Closing | | | |
|---|---|---|---|
| **Scaling Factor** | **Disc: Radius r = 4 Pixels** | **Square: 4 × 4 Pixels** | **Diamond: Distance between the Origin and the Points of the Diamond r = 4 Pixels** |
| 0.1 |  |  |  |
| 0.2 |  |  |  |
| 0.4 |  |  |  |

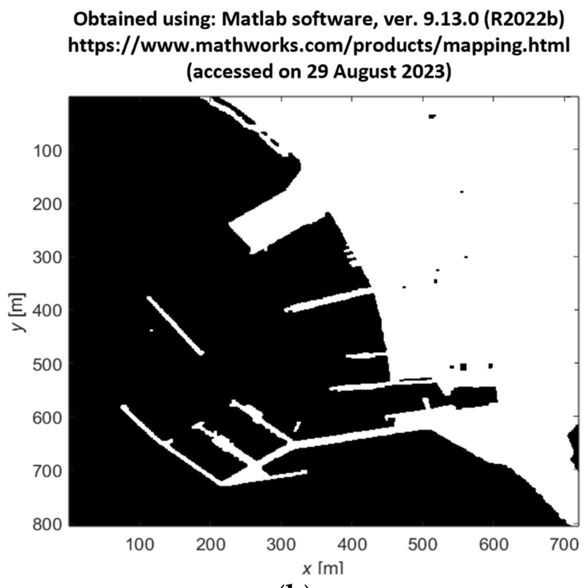
Figure 9 shows a comparison of the map obtained with threshold segmentation (75/255) and the map processed according to the proposed approach. All the image processing parameters used in this study are summarised in Table 4. Since this article considers a simple method of segmenting occupied and unoccupied areas based on an experimentally selected segmentation threshold, additional fine-tuning of the segmentation threshold may be necessary for different maps, depending on the lighting. Based on a rescaled and preprocessed map, the standard A* algorithm was used to calculate a collision-free path.

**Table 3.** A comparison of the same map fragments obtained as a result of processing with different parameters.

| | Morphological Operations: Dilation + Closing | | |
|---|---|---|---|
| **Scaling Factor** | **Disc, Diamond r = 2** | **Square (5 × 5 Pixels)** | **Square (4 × 4 Pixels)** |
| 0.1 |  |  |  |
| 0.2 |  |  |  |
| 0.4 |  |  |  |


Obtained using: Matlab software, ver. 9.13.0 (R2022b)
https://www.mathworks.com/products/mapping.html
(accessed on 29 August 2023)
500
1000
1500
2000
2500
3000
3500
4000
y [m]
x [m]

(**a**)


Obtained using: Matlab software, ver. 9.13.0 (R2022b)
https://www.mathworks.com/products/mapping.html
(accessed on 29 August 2023)
100
200
300
400
500
600
700
800
y [m]
x [m]

(**b**)

**Figure 9.** The occupancy grid created based on (**a**) threshold segmentation and (**b**) resizing, morphological processing, and threshold segmentation from Table 4.

**Table 4.** Key parameters of used map processing approach.

| Parameter | Value |
|---|---|
| Scaling factor | 0.2 |
| Structural element shape | Square |
| Structural element size | 4 × 4 pixels |
| Morphological operations | Dilation + closing |
| Threshold segmentation | 75/255 |

The resulting route is expressed in coordinates appropriate for the reduced image. For this reason, during postprocessing, all coordinates of the received path were scaled according to the value of the scaling factor.

## 4. Simulation Results and Analysis

The test map based on the image of the port area is an example of a complex environment from the point of view of route calculation using the A* method due to the following factors:

- A large map size of 14,490,000 pixels;
- A large number of occupied pixels (obstacles);
- Non-regular and trap-shaped obstacles (e.g., U-shaped or V-shaped).

Due to the map dimensions' adjustment to the area's actual dimensions in a ratio corresponding to 1 pixel–1 m$^2$, the path length obtained in the simulation was directly determined in meters. This allowed for comparing the proposed approach with the basic one in conditions close to real ones.

Since the map was created based on surface obstacles, it was assumed that in the case of AUVs, the vehicle moved at a constant depth, and the elements protruding above the water surface had the same shape at the depth of the AUV. In the simulation, a path was determined between the seven waypoints. The subsequent waypoints' locations were selected to achieve a complicated path. The path length for both methods was calculated based on the determined path coordinates between waypoints. The computation time for both methods was also measured during the simulation based on the in-build MATLAB function. In this case, in order to fairly compare the effectiveness of the proposed method with the basic A*, and in particular, the computation time, the raw path determined by both algorithms was used. The path length determined for individual cases is presented in Table 5, the calculation times are included in Table 6, and the minimal distance to the obstacle and number of calculated waypoints are in Table 7. Visualisations of the paths are shown in Figure 10a–f.

**Table 5.** Results of path calculations in terms of path length.

| | Path Length | | |
|---|---|---|---|
| **Case** | **A* with Map Processing [m]** | **A* [m]** | **Difference in Path Length [m]** |
| **1** | 1932 | 1932 | 0 |
| **2** | 3081.5 | 3010.4 | 71.1 |
| **3** | 1092.5 | 695.5 | 397 |
| **4** | 2385.6 | 2378.5 | 7.1 |
| **5** | 3019.2 | 2937 | 82.2 |
| **6** | 5673.5 | 5635.7 | 37.8 |
| **Total** [1] | 16,091.8 | 15,893.6 | 198.2 |

[1] Without taking into account case 3.

**Table 6.** Results of path calculations in terms of computation time.

| Calculation Time | | | |
|---|---|---|---|
| **Case** | **A* with Map Processing [s]** | **A* [s]** | **Ratio (A*/A* with Map Processing)** |
| **1** | 89 (1 m 29 s) | 59,542 (2 days 10 h 22 m) | 669 |
| **2** | 356 (5 m 56 s) | 210,122 (2 days 10 h 22 m) | 590 |
| **3** | 33 | 24,083 (6 h 41 m) | 730 |
| **4** | 56 | 40,250 (11 h 10 m) | 719 |
| **5** | 207 (3 m 27 s) | 120,262 (1 day 9 h 24 m) | 584 |
| **6** | 1399 (23 m 19 s) | 728,163 (8 days 10 h 16 m) | 521 |
| **Total** [1] | 2098 (34 m) | 1,156,797 (13 days 9 h 19 m) | 551 |

[1] Without taking into account case 3.

**Table 7.** Results of path calculations in terms of minimal distance to the obstacle and number of waypoints.

| **Case** | **Minimal Distance to the Obstacle** | | **Number of Waypoints** | |
|---|---|---|---|---|
| | **A* with Map Processing [m]** | **A* [m]** | **A* with Map Processing [s]** | **A* [s]** |
| **1** | 12.37 | 6.08 | 370 | 1846 |
| **2** | 4.47 | 1.00 | 492 | 2388 |
| **3** | 3.16 | 1.00 | 174 | 590 |
| **4** | 4.00 | 1.00 | 459 | 2288 |
| **5** | 4.00 | 1.00 | 543 | 2651 |
| **6** | 4.47 | 1.00 | 966 | 4778 |

Case 3 (marked in red in Tables 5 and 6) was not taken into account during the comparative analysis of the route length and calculation time due to the high risk of collision in the case of the route determined by the basic A* algorithm (see Figure 10c). Even though the occupancy grid created as a result of threshold segmentation did not show an obstacle at the route, from a practical point of view, the path was risky because it led to the pier.

Figure 10 shows the routes calculated for each case by the traditional A* and modified A* by zooming in on the most important parts of the routes. All the resulting figures (.png) and their MATLAB source figures (.fig) are available in the Supplementary Materials (S1). It can be seen that the route planned by the basic A* operated very close to the obstacles, which made the route shorter than in the proposed algorithm. However, from the point of view of path planning for ASVs and AUVs, a safe distance must be maintained in order to ensure the collision-free execution of the designed mission. Based on Figure 10, it is clearly visible that the proposed approach provides safe path planning with a larger distance from obstacles than the basic A*. Moreover, based on Figure 10c,e, it can be seen that the proposed approach tends to avoid the narrow paths between closely located obstacles.

Based on the results presented in Table 5, the path determined using the modified A* algorithm was longer than that determined using the basic A* algorithm. However, the differences were not significant (except for case 3), and the largest difference of 82.2 m in case 5 was 2.8% of the path length determined by the basic A*. However, the difference in the sum of the path length for both algorithms for cases 1–2 and 4–5 was 198.2 m, which was 1.25% of the value obtained as a result of the basic A* calculation.

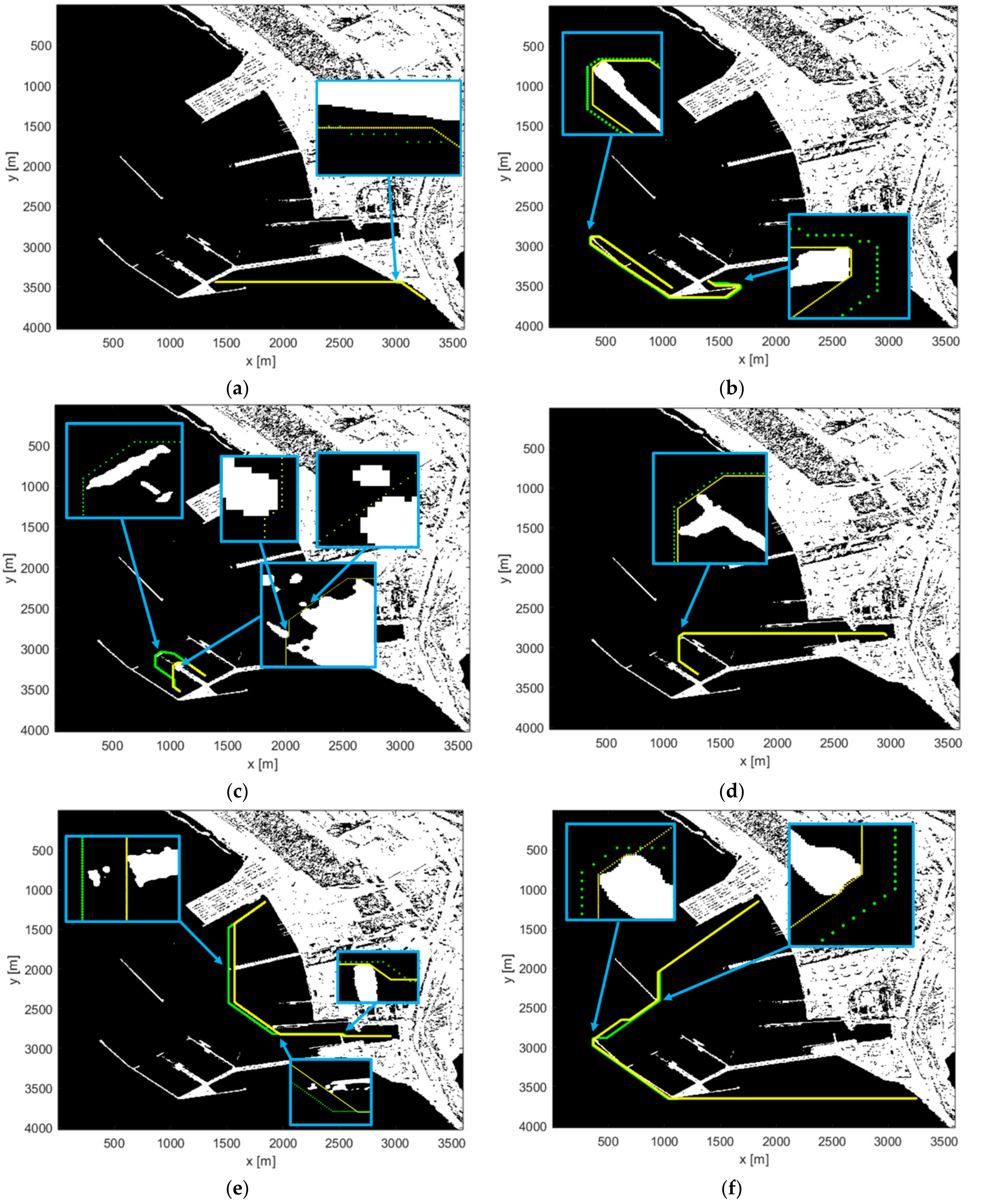


**Figure 10.** Visualisation of determining the path between subsequent waypoints: (**a**) case 1, (**b**) case 2, (**c**) case 3, (**d**) case 4, (**e**) case 5, (**f**) case 6. The path calculated using the basic version of the A* algorithm is marked in yellow, while the route calculated using A* with map image processing is marked in green. The blue frames highlight zoomed-in areas of the map.

Table 6 shows the calculation time for path determination using both methods. It can be observed that the longest calculation time for A* with map image processing was 23 m 19 s, while for the basic A*, it was 8 days, 10 h, and 16 s, indicating that the calculation time using the modified method was reduced by a factor of 521. Based on the results, it can also be concluded that usually, the shorter the time for calculating the path, the greater the ratio of the calculation time of both methods. This is due to the long-term creation of the list of obstacles by the A* method, which essentially determines the total path calculation time. In case 4, A* with map processing achieved a result 719 times faster than the basic algorithm with a difference in path length of 7.1 m (0.29%). In the worst case (not counting case 3) in terms of the calculation time ratio, in case 6, the ratio was 521, and the difference in path length was 37.8 m (0.67%). The total calculation time (excluding case 3) was 34 m and 13 days 9 h 19 m for the modified and basic methods, respectively.

The main reason for such a large difference in calculation time between the basic and proposed approaches is reducing the map's resolution, which means that 1 pixel after processing represents an area of $5 \times 5$ m. In order to compensate for losses in the area occupied by obstacles resulting from averaging the values of neighbouring pixels, mathematical morphology was used to enlarge areas occupied by obstacles artificially. Applying a greater dimension reduction results in a greater gain in computation time. However, excessive reduction in dimensions may lead to limited access to some areas of the map due to merging obstacles into one large obstacle.

Table 7 shows the distance to the nearest obstacle for both routes and the number of waypoints determined by both algorithms. It can be seen that the proposed algorithm ensured that each waypoint was at least 4 m away from obstacles (excluding case 3). In comparison, for the basic A* approach, this parameter was 1 m in almost all cases. It is worth noting that due to the map resolution, a distance of 1 m means that the pixel to which a specific waypoint is assigned is a direct neighbour of the obstacle. Analysing the number of waypoints determined by both algorithms, it can be seen that the proposed method provides a smaller number of waypoints, which is beneficial from the point of view of processing a set of waypoints in order to smooth the route and using the determined path to generate trajectories for real ASVs or AUVs.

The above images and results prove that the proposed method, including map image processing, reduces computation time and increases the path's safety. This is particularly important from the point of view of the vehicle's operation in an underwater environment, where finding a lost vehicle, e.g., due to a collision, requires a time-consuming search with the appropriate equipment, and in a port environment, also the special coordination of ship traffic. The benefit from shortening the calculation time significantly outweighs the losses associated with the difference in path length, which does not exceed 3%. However, from a practical point of view, to accelerate the entire trajectory planning process, more efficient computing hardware and parallel processing should be used, which would significantly shorten the trajectory calculation time.

## 5. Conclusions

In numerical research carried out based on a high-dimensional map expressing the port environment, the effectiveness of the modified A* method was compared with its basic version. The simulation results confirm the proposed method's effectiveness, which reduces the calculation time by over 500 times with a slight increase in the route length compared to the basic version of the A* algorithm. Additionally, as a result of the analysis, it can be concluded that the map image processing used in the proposed approach increases the path's safety by treating narrow and risky areas as prohibited for vehicle movement.

Based on the above results, it can be concluded that the proposed method is suitable for use in AUVs and ASVs. In the case of AUVs, this study assumes that the path is planned only in the horizontal plane and that the vehicle has the ability to maintain constant depth. However, due to the easy implementation, the method is also promising in 3D map processing. For this reason, further work is planned to test the algorithm's operation based on an artificially generated three-dimensional map of the underwater environment.

In this article, the planned paths are presented in their raw form to compare the effectiveness of the proposed modification of the A* method and its basic version. The use of the above method in a real mission planner requires knowledge of the physical parameters and manoeuvring limitations of the vehicle intended to be used in the mission. Based on this knowledge, the path can be smoothed to avoid sharp turns and reduce energy consumption. Additionally, in relation to vehicles with a trajectory tracking control system, specific arrival times should be added to the smoothed path depending on the dynamic parameters of the vehicle in order to obtain a feasible trajectory. For AUVs, depth changing is usually ensured by adding specific depth values that the vehicle should achieve at a given waypoint. However, for large changes in depth, it is also necessary to consider the manoeuvring limitations to calculate arrival times for each waypoint in a set trajectory. Additional aspects to be taken into account when planning a mission include information on weather conditions, sea currents, and other vessels operating in the mission area, as well as specific areas related to maritime traffic (e.g., anchorages, traffic separation zones, etc.).

While the proposed method demonstrates significant improvements in computation time for global path planning, its application in real-time local collision avoidance scenarios requires additional modifications and the tuning of image processing parameters such as the scaling factor and detection threshold, as well as the shape and size of structural elements of mathematical morphology depending on the resolution of data from the environmental sensor. Further improvements may include planning a path to a virtual point located in the direction of the destination point in an artificial map window around the vehicle, which, in combination with the proposed image processing, should provide close to real-time decision-making for standard computers currently used in ASVs and AUVs. Since the method was primarily designed for global path planning where the environment is known and static, strong surface waves and sea currents will reduce the algorithm's efficiency in local collision avoidance systems. However, the applied image processing methods allow for compensating this issue by closing narrow passages where the vehicle's path may be risky and adding a buffer zone as a result of the artificial enlargement of obstacles. Additionally, in a complex and rapidly changing environment, it is necessary to use replanning to ensure vehicle safety.

**Supplementary Materials:** The following supporting information can be downloaded at: https://doi.org/10.6084/m9.figshare.26827198.

**Author Contributions:** Conceptualisation: R.K. and P.S.; methodology: R.K., P.S. and K.N.; software: R.K.; data curation: R.K.; formal analysis: P.S. and K.N.; writing—original draft preparation: R.K. and P.S.; final manuscript writing: R.K., P.S. and K.N.; visualisation: K.N.; investigation and validation: R.K. and P.P.; supervision: P.S. and P.P.; writing—reviewing and editing: R.K., P.S. and K.N. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** The original contributions presented in the study are included in the article; further inquiries can be directed to the corresponding authors.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References

1. Kot, R. Review of Obstacle Detection Systems for Collision Avoidance of Autonomous Underwater Vehicles Tested in a Real Environment. *Electronics* **2022**, *11*, 3615. [CrossRef]
2. Liu, L.; Wang, X.; Yang, X.; Liu, H.; Li, J.; Wang, P. Path planning techniques for mobile robots: Review and prospect. *Expert Syst. Appl.* **2023**, *227*, 120254. [CrossRef]

3. Cheng, C.; Zhang, H.; Sun, Y.; Tao, H.; Chen, Y. A cross-platform deep reinforcement learning model for autonomous navigation without global information in different scenes. *Control Eng. Pract.* **2024**, *150*, 105991. [CrossRef]
4. Pan, C.; Zhang, Z.; Chen, Y.; Lin, D.; Huang, J. Improved Reinforcement Learning Task Supervisor for Path Planning of Logistics Autonomous System. *IFAC-Pap.* **2023**, *56*, 10010–10015. [CrossRef]
5. Tang, Z.; Cao, X.; Zhou, Z.; Zhang, Z.; Xu, C.; Dou, J. Path planning of autonomous underwater vehicle in unknown environment based on improved deep reinforcement learning. *Ocean Eng.* **2024**, *301*, 117547. [CrossRef]
6. Haoran, Z.; Hang, F.; Fan, Y.; Che, Q.; Yaoming, Z. Data-driven offline reinforcement learning approach for quadrotor's motion and path planning. *Chin. J. Aeronaut.* **2024**, *in press*. [CrossRef]
7. Lan, W.; Jin, X.; Chang, X.; Zhou, H. Based on Deep Reinforcement Learning to path planning in uncertain ocean currents for Underwater Gliders. *Ocean Eng.* **2024**, *301*, 117501. [CrossRef]
8. Ronghua, M.; Xinhao, C.; Zhengjia, W.; Du, X. Improved ant colony optimization for safe path planning of AUV. *Heliyon* **2024**, *10*, e27753. [CrossRef]
9. Das, P.; Jena, P.K. Multi-robot path planning using improved particle swarm optimization algorithm through novel evolutionary operators. *Appl. Soft Comput.* **2020**, *92*, 106312. [CrossRef]
10. Kot, R.; Szymak, P.; Piskur, P.; Naus, K. A Comparative Study of Different Collision Avoidance Systems with Local Path Planning for Autonomous Underwater Vehicles. *IEEE Access* **2024**, *12*, 61443–61466. [CrossRef]
11. Ab Wahab, M.N.; Nazir, A.; Khalil, A.; Ho, W.J.; Akbar, M.F.; Noor, M.H.M.; Mohamed, A.S.A. Improved genetic algorithm for mobile robot path planning in static environments. *Expert Syst. Appl.* **2024**, *249*, 123762. [CrossRef]
12. Hao, K.; Zhao, J.; Li, Z.; Liu, Y.; Zhao, L. Dynamic path planning of a three-dimensional underwater AUV based on an adaptive genetic algorithm. *Ocean Eng.* **2022**, *263*, 112421. [CrossRef]
13. Mehmood, D.; Ali, A.; Ali, S.; Kulsoom, F.; Chaudhry, H.N.; Haider, A.Z.U. A Novel Hybrid Genetic and A-star Algorithm for UAV Path Optimization. In Proceedings of the 2024 IEEE 1st Karachi Section Humanitarian Technology Conference (KHI-HTC), Tandojam, Pakistan, 8–9 January 2024; pp. 1–5.
14. Majeed, A.; Lee, S. A fast global flight path planning algorithm based on space circumscription and sparse visibility graph for unmanned aerial vehicle. *Electronics* **2018**, *7*, 375. [CrossRef]
15. Bygi, M.N.; Ghodsi, M. 3D visibility graph. In Proceedings of the Computational Science and its Applications, Kuala Lumpur, Malaysia, 26–29 August 2007.
16. You, Y.; Cai, C.; Wu, Y. 3d visibility graph based motion planning and control. In Proceedings of the 5th International Conference on Robotics and Artificial Intelligence, Singapore, 22–24 November 2019; pp. 48–53.
17. Babič, M.; Hluchy, L.; Krammer, P.; Matovič, B.; Kumar, R.; Kovač, P. New method for constructing a visibility graph-network in 3D space and a new hybrid system of modeling. *Comput. Inform.* **2017**, *36*, 1107–1126. [CrossRef]
18. Harabor, D.; Grastien, A. Online graph pruning for pathfinding on grid maps. In Proceedings of the AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 7–11 August 2011; Volume 25, pp. 1114–1119.
19. Erke, S.; Bin, D.; Yiming, N.; Qi, Z.; Liang, X.; Dawei, Z. An improved A-Star based path planning algorithm for autonomous land vehicles. *Int. J. Adv. Robot. Syst.* **2020**, *17*, 1729881420962263. [CrossRef]
20. Tang, G.; Tang, C.; Claramunt, C.; Hu, X.; Zhou, P. Geometric A-star algorithm: An improved A-star algorithm for AGV path planning in a port environment. *IEEE Access* **2021**, *9*, 59196–59210. [CrossRef]
21. Wang, H.; Qi, X.; Lou, S.; Jing, J.; He, H.; Liu, W. An Efficient and Robust Improved A* Algorithm for Path Planning. *Symmetry* **2021**, *13*, 2213. [CrossRef]
22. Martins, O.O.; Adekunle, A.A.; Olaniyan, O.M.; Bolaji, B.O. An Improved multi-objective a-star algorithm for path planning in a large workspace: Design, Implementation, and Evaluation. *Sci. Afr.* **2022**, *15*, e01068. [CrossRef]
23. Li, J.; Zhang, W.; Hu, Y.; Fu, S.; Liao, C.; Yu, W. RJA-Star Algorithm for UAV Path Planning Based on Improved R5DOS Model. *Appl. Sci.* **2023**, *13*, 1105. [CrossRef]
24. Kabir, R.; Watanobe, Y.; Islam, M.R.; Naruse, K. Enhanced Robot Motion Block of A-Star Algorithm for Robotic Path Planning. *Sensors* **2024**, *24*, 1422. [CrossRef]
25. Zhang, D.; Chen, C.; Zhang, G. AGV path planning based on improved A-star algorithm. In Proceedings of the 2024 IEEE 7th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), Chongqing, China, 15–17 March 2024; Volume 7, pp. 1590–1595.
26. Yin, C.; Tan, C.; Wang, C.; Shen, F. An Improved A-Star Path Planning Algorithm Based on Mobile Robots in Medical Testing Laboratories. *Sensors* **2024**, *24*, 1784. [CrossRef]
27. Li, J.; Kang, F.; Chen, C.; Tong, S.; Jia, Y.; Zhang, C.; Wang, Y. The Improved A* Algorithm for Quadrotor UAVs under Forest Obstacle Avoidance Path Planning. *Appl. Sci.* **2023**, *13*, 4290. [CrossRef]
28. Wang, H.; Lou, S.; Jing, J.; Wang, Y.; Liu, W.; Liu, T. The EBS-A* algorithm: An improved A* algorithm for path planning. *PLoS ONE* **2022**, *17*, e0263841. [CrossRef] [PubMed]
29. Fu, B.; Chen, L.; Zhou, Y.; Zheng, D.; Wei, Z.; Dai, J.; Pan, H. An improved A* algorithm for the industrial robot path planning with high success rate and short length. *Robot. Auton. Syst.* **2018**, *106*, 26–37. [CrossRef]
30. Liao, T.; Chen, F.; Wu, Y.; Zeng, H.; Ouyang, S.; Guan, J. Research on Path Planning with the Integration of Adaptive A-Star Algorithm and Improved Dynamic Window Approach. *Electronics* **2024**, *13*, 455. [CrossRef]

31. Chatzisavvas, A.; Dossis, M.; Dasygenis, M. Optimizing Mobile Robot Navigation Based on A-Star Algorithm for Obstacle Avoidance in Smart Agriculture. *Electronics* **2024**, *13*, 2057. [CrossRef]
32. Hong, Z.; Sun, P.; Tong, X.; Pan, H.; Zhou, R.; Zhang, Y.; Han, Y.; Wang, J.; Yang, S.; Xu, L. Improved A-Star algorithm for long-distance off-road path planning using terrain data map. *ISPRS Int. J. Geo-Inf.* **2021**, *10*, 785. [CrossRef]
33. Ju, C.; Luo, Q.; Yan, X. Path planning using an improved a-star algorithm. In Proceedings of the 2020 11th International Conference on Prognostics and System Health Management (PHM-2020 Jinan), Jinan, China, 23–25 October 2020; pp. 23–26.
34. Li, X.; Hu, X.; Wang, Z.; Du, Z. Path planning based on combinaion of improved A-STAR algorithm and DWA algorithm. In Proceedings of the 2020 2nd International Conference on Artificial Intelligence and Advanced Manufacture (AIAM), Manchester, UK, 15–17 October 2020; pp. 99–103.
35. Li, J.; Xiong, X.; Yang, Y. A Method of UAV Navigation Planning Based on ROS and Improved A-star Algorithm. In Proceedings of the 2023 CAA Symposium on Fault Detection, Supervision and Safety for Technical Processes (SAFEPROCESS), Yibin, China, 22–24 September 2023; pp. 1–5.
36. The MathWorks Inc. MATLAB Version: 9.13.0 (R2022b). Available online: https://www.mathworks.com (accessed on 12 September 2023).
37. Google.pl. Available online: https://www.google.pl/maps/@54.3985359,18.7207342,7518a,35y,133.79h,0.72t/data=!3m1!1e3?entry=ttu (accessed on 29 August 2023).
38. Panorama 360—Port Północny. Available online: http://www.port-polnocny.pl/Panorama_360/Port_Polnocny.html (accessed on 15 September 2023).
39. Keys, R. Cubic convolution interpolation for digital image processing. *IEEE Trans. Acoust. Speech Signal Process.* **1981**, *29*, 1153–1160. [CrossRef]