




Article

The Method for Storing a Seabed Photo Map of the During Surveys Conducted by an Autonomous Underwater Vehicle

Chang Liu ^{1,*}, Vladimir Filaretov ², Eduard Mursalimov ^{3,4}, Alexander Timoshenko ³ and Alexander Zuev ²

¹ St. Petersburg School of Shipbuilding and Ocean Technology, Guangdong Ocean University, Zhanjiang 524088, China

² Department of Computer Science and Control in Technical Systems, Sevastopol State University, Sevastopol 299053, Russia; filaretov@inbox.ru (V.F.); alvzuev@yandex.ru (A.Z.)

³ Laboratory of Intelligent Information Systems for Marine Robots, M.D. Ageev Institute of Marine Technology Problems of Far Eastern Branch of the Russian Academy of Sciences, Vladivostok 690091, Russia; murs@dvo.ru (E.M.); timoshenko.aal@mail.ru (A.T.)

⁴ Department of Automation and Robotics, Far Eastern Federal University, Vladivostok 690090, Russia

* Correspondence: liuchang@gdou.edu.cn

Abstract: The paper introduces a novel method for creating a photographic map of the seabed using images captured by the on-board photo and video systems of autonomous underwater vehicles (AUVs) during various missions, while incorporating navigation parameters. Additionally, it presents a new approach for storing this photo map on the on-board device in a mosaic format (tiles), which significantly accelerates operational visual inspection by enabling the automatic search and recognition of underwater objects that may exceed the coverage area of a single photograph. This capability is achieved by organizing the photo map into layers with varying zoom levels. Semi-natural experiments were conducted with data from actual missions using the real underwater vehicle demonstrate the high efficiency of the proposed method and algorithm. Unlike existing methods that form photo maps after the underwater vehicle has taken pictures of the bottom using special high-performance computers, the developed method forms a photo map directly during the movement of the vehicle, using only the computing power of the on-board computer. In addition, in the event of accidents, when it is necessary to detect objects of interest on the seabed as quickly as possible, it is necessary to provide a quick visual inspection of the generated photo map. For this purpose, we have developed an algorithm for saving a photo map in the form of a mosaic, which is widely used in interactive geographic maps, such as Google Maps. This algorithm differs from existing methods in that it selectively saves data to the on-board storage device to reduce the number of read and write operations, thus ensuring the timely operation of the entire process of creating a photo map at a given frequency of photography. After the generated map has been stored as a mosaic and a high-speed connection with the vehicle has appeared, the operator can immediately view the entire generated map using a regular web browser.

Keywords: autonomous underwater vehicle; photo map; tile; on-board photo and video system; navigation data; photo image processing



Academic Editors: Asiya Khan, Pablo Borja, Dena Bazazian, Daxiong Ji and Mohd Hisham Bin Nordin

Received: 28 October 2024

Revised: 26 January 2025

Accepted: 27 January 2025

Published: 4 February 2025

Citation: Liu, C.; Filaretov, V.; Mursalimov, E.; Timoshenko, A.; Zuev, A. The Method for Storing a Seabed Photo Map of the During Surveys Conducted by an Autonomous Underwater Vehicle. *Drones* **2025**, *9*, 114. <https://doi.org/10.3390/drones9020114>

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Currently, autonomous underwater vehicles (AUVs) are primarily used for surveying, searching and inspecting operations, especially at great depths. During these operations, it is often necessary to find objects sunk as a result of accidents, places where underwater pipelines are covered with soil, etc.

For a detailed inspection of a given section of the seabed, AUVs equipped with on-board photo and video systems are used. Performing preassigned missions, these AUVs move along the formed trajectories at the required velocities at specified distances from the bottom or depths, providing photographing objects getting into the frames with a set frequency. All digital photo images are stored on the on-board storage device of the AUV. Usually, search missions take several tens of hours. This leads to the accumulation of a very large number of photographic images, and it requires a lot of time to transfer them to the control post after the completion of the mission and the return of the AUV to its base. Usually, the analysis of each received photo image is performed manually and is time-consuming. Three approaches can be used to reduce the decision-making time based on search results:

- Ensuring the ascent of the AUVs to the surface for data transmission;
- The implementation of data transmission under water immediately during the operation of the AUVs;
- Creation of special software products for fast and accurate processing of all data read from the AUVs in a stationary controlled point using powerful computing tools.

Since most modern AUVs have wireless interfaces on its board for data transmission [1–4], then it is possible to communicate with the carrier vessel or ground station after the vehicle's ascent using radio frequency interfaces such as radio modems and Wi-Fi. However, due to the long hours of operation under water, either a continuous transmission of a large amount of data will be necessary, or frequent surfacing of the vehicle, which would be highly undesirable from an energy perspective. However, due to the many hours of underwater operation, either a lengthy transfer of a significant amount of data would be required, which would take time, or the vehicle would need to frequently surface, which is highly undesirable from an energy perspective.

Currently, the transmission of large amounts of data through sonar channels without delays is impossible due to the limited bandwidth of these channels. However, efforts are already underway to develop effective underwater communication systems for transmitting data from on-board vision systems (VS) of the AUVs using specially developed image compression algorithms [5]. In [6], the authors propose to transmit the part of the photo image that contains the interest objects via the sonar communication interface. However, it is difficult to identify these objects on-board the vehicle. In recent years, there have been attempts to create optical communication interfaces that can transfer data at high speeds. However, their capabilities are still not well understood and are severely limited (especially in turbid waters and at long distances for data transmission) [7–9]. Due to the above-mentioned reasons, the majority of data received from AUVs are only processed and analyzed by the operator on shore after the completion of each mission and the return of the vehicle to the base.

The difficulties described above can be partially solved using machine learning methods [10], which make it possible to identify and classify interesting objects in images, significantly reducing the analysis time. This significantly reduces the time needed for analysis. However, it has not yet been possible to successfully identify arbitrary objects using these techniques. This is especially true when objects in photographic images are represented by fragments, considering their large real sizes, or when they are partially obscured at the edges of the image due to poor lighting conditions. In addition, when objects are combined in different images, the same object can still be recognized but may be incorrectly counted multiple times.

These disadvantages can be overcome by a joint set of overlapping photo images into an entire photographic map [11–13] of the study area using already developed methods and algorithms for processing photographic images [11,12,14,15]. They can combine and

reconstruct images by finding and matching similar elements in the images or by matching their spectral characteristics. However, due to the fact that errors accumulate during the operation of these algorithms, they are demanding of computing resources, and on-board computers have limited computing power, their use on board underwater vehicles is difficult [16]. To improve the accuracy of photographic mapping, the important water areas are sometimes photographed at different depths [17]. However, this significantly increases the operation time of the AUV.

Sometimes, towed systems with multiple cameras are used to search for and identify objects, as well as create 3D models of the underwater environment [18]. However, the creation of high-quality models of the studied areas and photographic images occurs after shooting on powerful computers [19,20]. Therewith, the waves of the underwater environment and the currents are not permitted, as they significantly reduce the accuracy of the positioning of cameras in space.

Considering the circumstances mentioned above, the issue of fast and high-quality image processing obtained using AUV remains significant and relevant. At the same time, in order to increase the efficiency of analyzing the obtained data, it would be advisable to create a complete photographic map of the sea bottom directly during the underwater vehicle's movement on board. In addition, such a map should be accessible for viewing and analysis after the ascent of AUV to the surface of the ocean, when the radio frequency communication systems are activated.

2. Materials and Methods

This section describes the developed method for creating and algorithm for storing a photomap during the AUV movement. First, Section 2.1 describes the composition of the AUV on-board equipment and the conditions under which the bottom photo survey is performed. Based on this, the problem statement is formulated. Section 2.2 outlines a general algorithm for creating a scalable photo map and storing it on the on-board computer during bottom survey. Section 2.3 describes the processing of each photo image received from the on-board photo-video system, considering the navigation data of the vehicle, its height above the bottom, and the camera parameters. Section 2.4 describes the developed algorithm for saving all layers of a scalable photo map to an on-board storage device, which provides the necessary performance of the entire system during the movement of the vehicle. Section 2.5 describes the software implementation of the developed software component of the on-board information and control system that generates and saves a photo map during the movement of the AUV.

2.1. Detailed Description of the Task to Be Solved

It is assumed that the AUV has the following on-board equipment: a navigation system to determine its geographical coordinates and orientation in space, a photo-video system with a known focal length f and optical axis directed vertically downward, lamps for illuminating objects for photography, and a typical on-board computer of average performance. The on-board control system of the AUV must ensure its movement at a given velocity and distance from the bottom (or objects located on it) during photo and video shooting at a given frequency and with storing each photo image on the on-board data storage device. The frequency of taking photographs depends on the speed of the AUV movement and its distance from the bottom surface. It increases with increasing movement velocity and becomes such that the new photo has a predetermined overlap area with the previous photo to eliminate gaps between the sequentially taken photos. The frequency decreases proportionally with increasing distance between the AUV and the surface being photographed.

In the presence of the above equipment, it is necessary to create on-board software that will, during the movement of the AUV, continuously build a high-resolution photographic image (photo map) of the observed surface. This photo map will be created from a variety of overlapping photographs collected in real time. The number of photographs that can be taken per hour at the maximum photography frequency can reach several thousand. A continuously generated photographic image of the observed surface should be stored on an on-board storage device. This allows for the detection of objects of interest using special recognition algorithms. These algorithms can then automatically take more detailed pictures of these objects from different angles, without the need for an operator. This is because these objects may not all fit into one photograph.

When additional examination of the objects found and adding their detailed photographic images to the already constructed photo map is completed, it is necessary to take into account full navigation information about the current position and orientation of the AUV, and its height above the photographed surface. This is necessary for accurate placement of the new image in the designated location on the photo map and for accurate scaling of each added photograph. The entire photo map, which is under construction, should be stored on the on-board AUV storage device in a format that is convenient for quick viewing at the control post, at various scales.

2.2. The Algorithm for Generation an Entire Photo Map of the Sea Bottom While the AUV Is Moving

To solve this problem, we propose combining photographs obtained from the on-board photo-video system into an entire photo map of the seabed while the AUV is moving, using an algorithm that ensures the required accuracy for comparing photographs during long-term missions. The algorithm takes into account the navigation parameters of the AUV movement and the parameters of on-board photo and video system. It can be run on standard on-board computers. At the same time, we propose storing an entire photo map in a mosaic format on the on-board storage device. This means that the image will be divided into small frames (tiles) with the same resolution for each zoom level. This storage method is used in tile maps, which are commonly used in Geographic Information Systems (GIS) [21,22], such as in the well-known mapping services like Google Maps and Yandex Maps. It is also used when creating 3D applications and games [23]. The map created using this method can be saved as separate images. This allows users to quickly analyze the map (zooming and visual viewing) using an interactive web interface.

To combine the photographs that were received from the AUV on-board photo and video system, data on the motion parameters of the vehicle itself should be used. These parameters include: firstly, the coordinates of the underwater vehicle in the horizontal plane; secondly, the height of vehicle above the seabed; and thirdly, the angle of vehicle yaw. In addition, the parameters of the on-board video camera should be known, since they affect the properties of the images received from it. After all the data has been collected, the photo images are combined. This includes linear displacement, rotation, and scaling of each image received from the video system. Next, each image is split into tiles, which together form an entire photo map. A simplified representation of the map is shown in Figure 1.

Ultimately, each photo map will be composed of square tiles with the same resolution. This allows the operator to use existing GIS tools, such as Leaflet JS [24], to view the map visually using standard browsers. At the same time, the presentation of a photo map as separate images allows the use of existing machine learning techniques to automatically detect interesting objects during the movement of AUV, with their precise geographical location. Also, neural networks such as YOLO [25] accept only square images as input. Thus, created tiles can be used for the recognition purposes as is without additional

modifications. This approach will significantly reduce the time it takes to search for the necessary objects, as the AUV will be able to quickly send a short message about the detected object, even over a limited hydroacoustic communication channel.

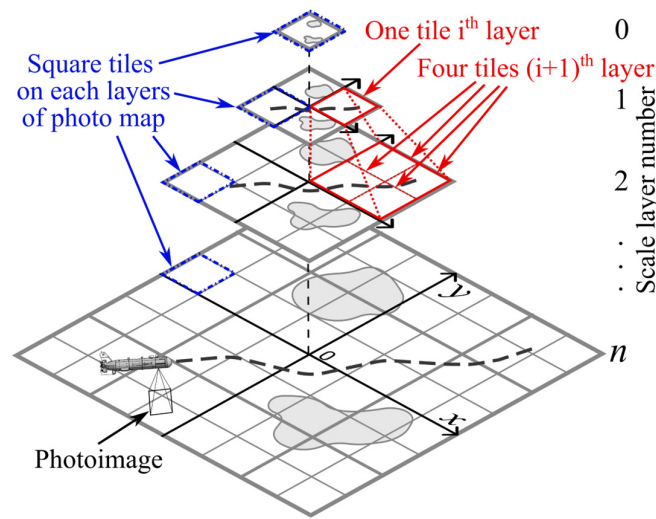


Figure 1. A general overview of the created photo map.

Several layers are used to zoom in on the map using the browser (see Figure 1). Each subsequent layer, starting from the lowest and moving to the top, is a four-fold reduced copy of the previous one. This allows the use of machine learning methods to automatically detect large objects that only fit on the highest layers.

To ensure maximum performance, the proposed algorithm and software product are implemented using the C++ programming language on typical on-board computers. All image processing operations are performed using the OpenCV library [26]. The general structure of the developed algorithm is presented in Figure 2.

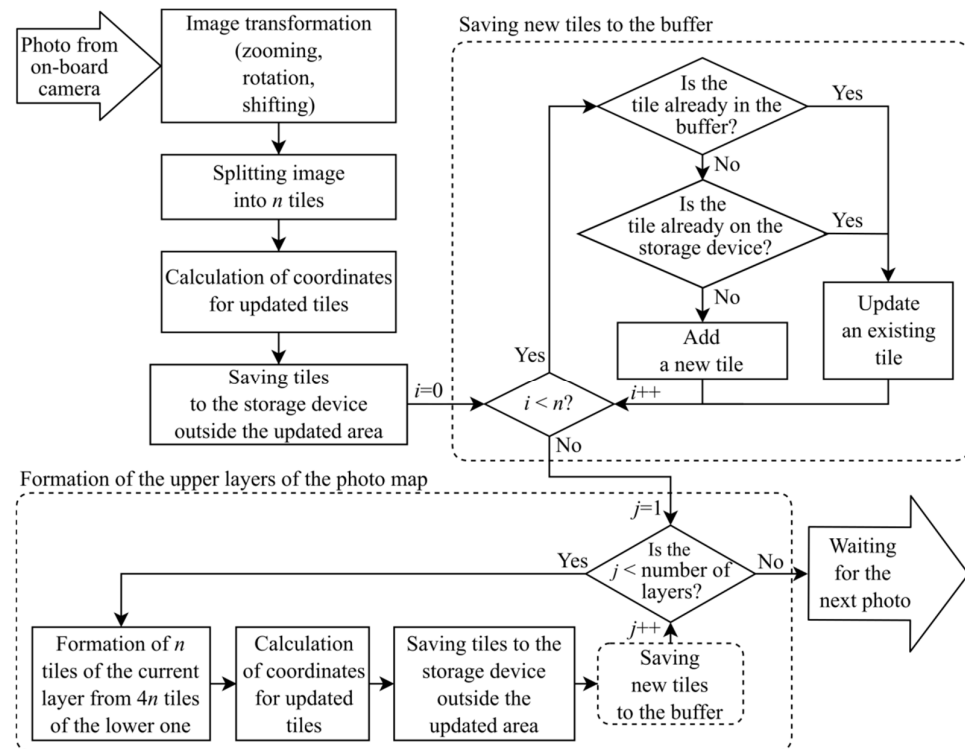


Figure 2. The general structure of the developed algorithm.

The main difference between the proposed algorithm and existing ones is that it combines two adjacent images using the navigation data from the vehicle. And other algorithms described in [11–15] require a large number of additional computational operations to combine two adjacent images, which are used to search for certain correspondences in these images. Therefore, the proposed algorithm, having less complexity, allows using the algorithm during the movement of the underwater vehicle creating the photo map of the seabed in real-time.

Furthermore, a detailed description of the individual steps of the algorithm is provided.

2.3. Photo Image Transformation

Before cutting the photo images received from the on-board photo and video system into separate tiles, they must be rotated and scaled. This is done so that the objects that appear in different images have the same size, and their orientation should also be the same (see Figure 3). The following describes an algorithm that pre-increases the size of the image so that the edges of the original image are not lost during the transformation.

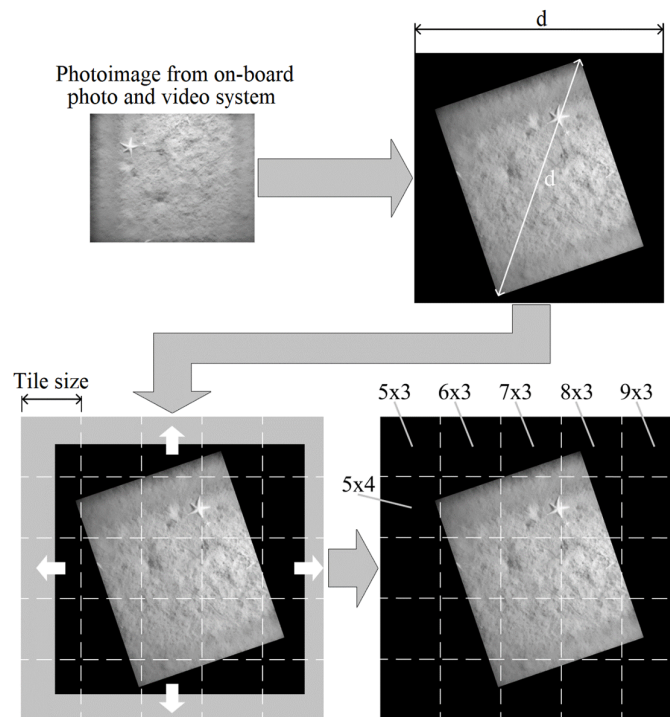


Figure 3. Photo image transformation to split into tiles.

Each image is a matrix $C = R^{[k \times n]}$ consisting of pixels, where $k, n \in \mathbb{N}$ are the height and width of the image, respectively. Each pixel has coordinates x and y . Denoted by a vector, $r = [x \ y]^T$. When rotating and zooming the image using OpenCV tools, the rotation matrix is used:

$$A^{[2 \times 3]} = \begin{bmatrix} \alpha & \beta & x_c(1 - \alpha) - y_c\beta \\ -\beta & \alpha & x_c\beta - y_c(1 - \alpha) \end{bmatrix}, \tag{1}$$

where $0 \leq x_c \leq n - 1, 0 \leq y_c \leq k - 1$ are coordinates of the image rotation point; $\alpha = \cos \theta$; $\beta = \sin \theta$; θ is the image rotation angle that is equal to the opposite value angle of the underwater vehicle yaw; s is zoom factor. These transformations use linear interpolation, which slightly reduces the quality of the final image.

The zoom factor of an image is calculated using the following formula [27]:

$$s = \frac{H_n}{H} = \frac{fm}{H}, \quad (2)$$

where H is the height of the AUV above the seabed in the moment of taking the picture; $H_n = fm$ is nominal photographing height; f is the focus of the camera lens; m is the defined map scale.

When rotating images using OpenCV tools, the image corners may exceed its frame. This is because the rotation matrix (1) is applied to the pixels of the image, but the frame itself doesn't rotate. To avoid it, the original image is put into the center of the larger square $\tilde{C} = R^{\lfloor sd \rfloor \times \lfloor sd \rfloor}$, where d is a length of the original image diagonal in pixels, $\lfloor \cdot \rfloor$ is an operation of rounding down to an integer. In this case, the image C pixel coordinates in the coordinate system of the image \tilde{C} can be obtained as $\tilde{r} = \begin{bmatrix} \tilde{x} & \tilde{y} \end{bmatrix}^T = \begin{bmatrix} x - x_c + \tilde{x}_c & y - y_c + \tilde{y}_c \end{bmatrix}^T$, where \tilde{x}_c, \tilde{y}_c are the image \tilde{C} center coordinates. When rotating and zooming the image, a "1" is added to the vector \tilde{r} and the rotation matrix (1) is entered:

$$\hat{r} = \begin{bmatrix} \hat{x} \\ \hat{y} \\ 1 \end{bmatrix} = A^{[2 \times 3]} \begin{bmatrix} \tilde{r} \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & \beta & \tilde{x}_c(1 - \alpha) - \tilde{y}_c\beta \\ -\beta & \alpha & \tilde{x}_c\beta - \tilde{y}_c(1 - \alpha) \end{bmatrix} \begin{bmatrix} \tilde{x} \\ \tilde{y} \\ 1 \end{bmatrix}, \quad (3)$$

After image transformation, it is necessary to determine its position on the photo map. To do this, the coordinates of the underwater vehicle are converted from meters to pixels. The number of the image pixels per meter on the map can be defined as $P = 1000 \frac{p}{m}$, where p is the number of pixels per millimeter of the photo camera imager, at the nominal scale of m . The coordinates of the image center r_C are calculated using the formula:

$$r_C = \begin{bmatrix} x_C & y_C \end{bmatrix}^T = \begin{bmatrix} \lfloor Px_\mu \rfloor & \lfloor -Py_\mu \rfloor \end{bmatrix}^T, \quad (4)$$

where x_μ, y_μ are metric coordinates of the image center.

Position of the upper-left tile corner is determined as:

$$r_{tl} = \begin{bmatrix} x_{tl} & y_{tl} \end{bmatrix}^T = \begin{bmatrix} \lfloor \frac{x_C - \frac{d}{2}}{l} \rfloor & \lfloor \frac{y_C - \frac{d}{2}}{l} \rfloor \end{bmatrix}^T, \quad (5)$$

where l is the length of the one tile side in pixels. The image is then expanded so that it consists of an integer number of tiles. At the last step, the coordinates of all image tiles are calculated using the coordinates of the upper-left tile r_{tl} .

2.4. Saving Tiles to the On-Board Storage Device

The resulting tiles are saved to the on-board physical storage device of the AUV. However, tile saving process can be slow, so an alternative approach was proposed instead of direct recording.

Since each new input photo image may have intersection with the previous ones, when saving directly to the on-board storage device, it is necessary not only to save new tiles, but also to update and overwrite already saved ones. To avoid unnecessary overwriting of tiles and increase the speed of image processing, it is suggested to save only the tiles that are not updated with a new image (Figure 4). To do this, a buffer is created in the RAM of the on-board computer of the underwater vehicle, which temporarily stores unfinished tiles. If these tiles are not included in the area of the new image, they are saved to the on-board storage.

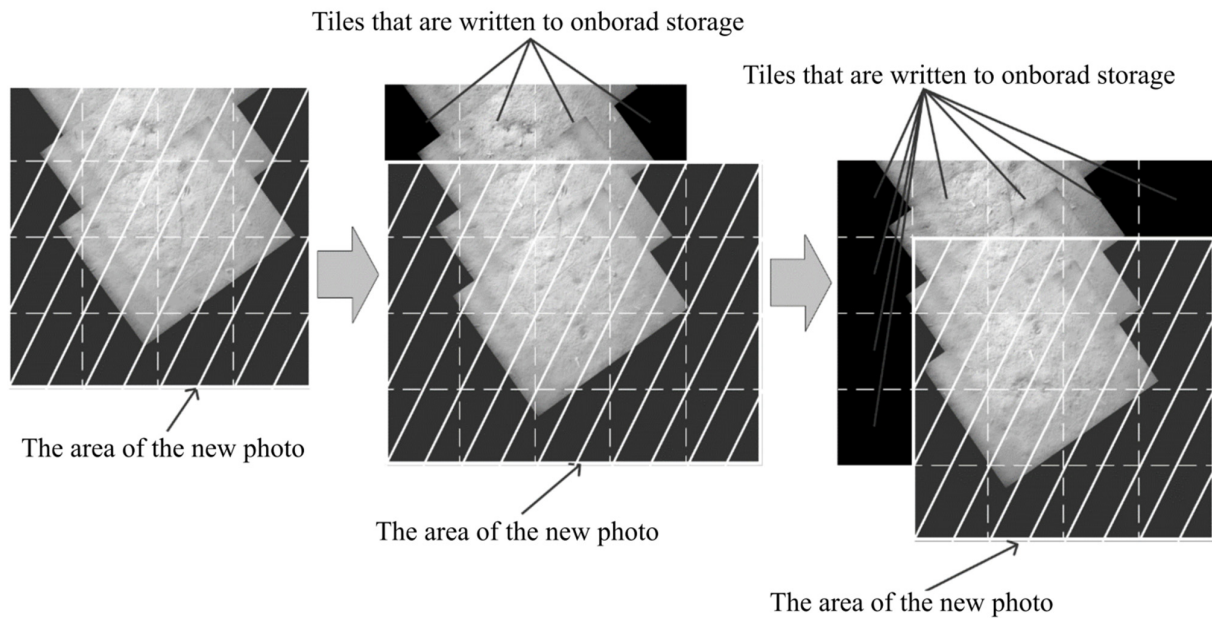


Figure 4. The principle of storing tiles on the on-board storage of AUV.

After this process, tiles are formed for the upper layers of the map (see Figure 5). For each scale layer, its own buffer is created, since tiles with the same coordinates may appear on different layers. The tiles forming the photo map layer with a larger scale are used to form tiles of a smaller scale layer. To do this, the corresponding tiles of a larger scale are grouped into groups of four, then this image is reduced to the size of one tile. Coordinates are assigned to these tiles and then they are saved to the storage device, according to the principle described above. Saving is using the Portable Network Graphics (PNG) format, which implies “lossless compression” using the DEFLATE algorithm [28].

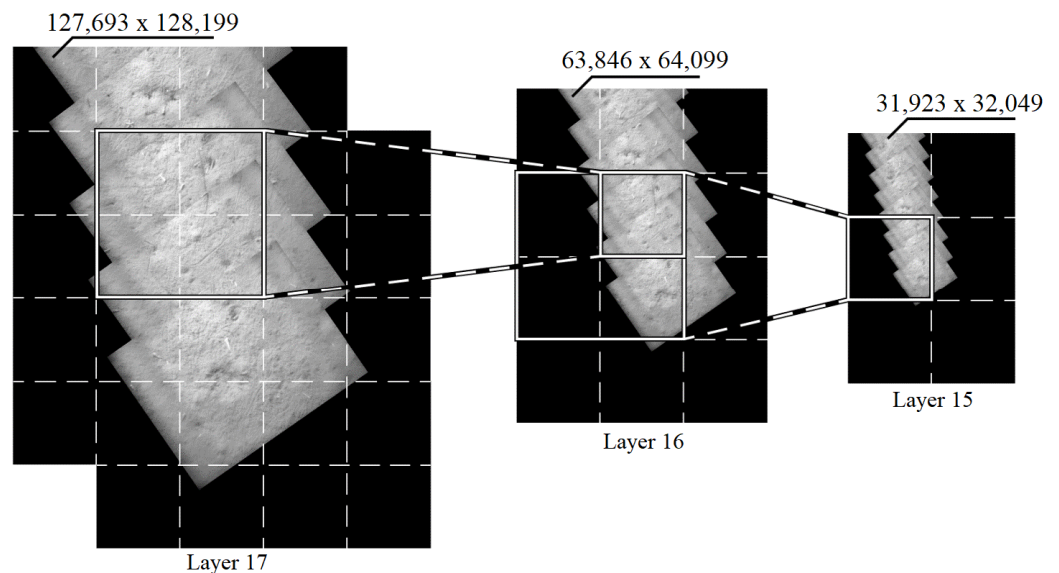


Figure 5. Formation of tiles on the upper layers.

After processing a new photo, all tiles that are stored in the buffer of the initial map layer and are not included in the area of this image are considered completed, saved to the storage device, and deleted from the buffer. Then, the new set of tiles is compared with those stored in the buffer. If tiles with the same coordinates are present in the buffer, tile overlays are performed. Otherwise, the presence of such tiles on the storage device is checked. If they are detected, an overlay operation is performed using tiles on storage

device. Otherwise, new tiles are just stored in the buffer. A similar procedure is performed for tiles on the higher map layers. Next, a detailed description of the implementation of this algorithm will be given on the standard on-board computers.

2.5. Software Implementation of the Algorithm

To test the effectiveness of the proposed algorithm, a program was developed. The scheme of this program is shown in Figure 6. This program is a component of the on-board information control system (ICS) of the AUV, presented in [29]. Its operation requires data from several components of the ICS, including:

- NavigCMP, a component that processes data from on-board navigation sensors and generates motion parameters for the AUV. This includes data on the position, orientation and height of the AUV above the seabed;
- CameraCMP, a component that interacts with the on-board photo-video system. This component takes pictures of the seabed at a calculated frequency, saving each image on the on-board data storage device without changing it. The data on the path to the file containing the last saved image is also published;
- MissManagerCMP, a component that monitors the execution of commands given by the mission operator and notifies other components about their further actions through special commands.

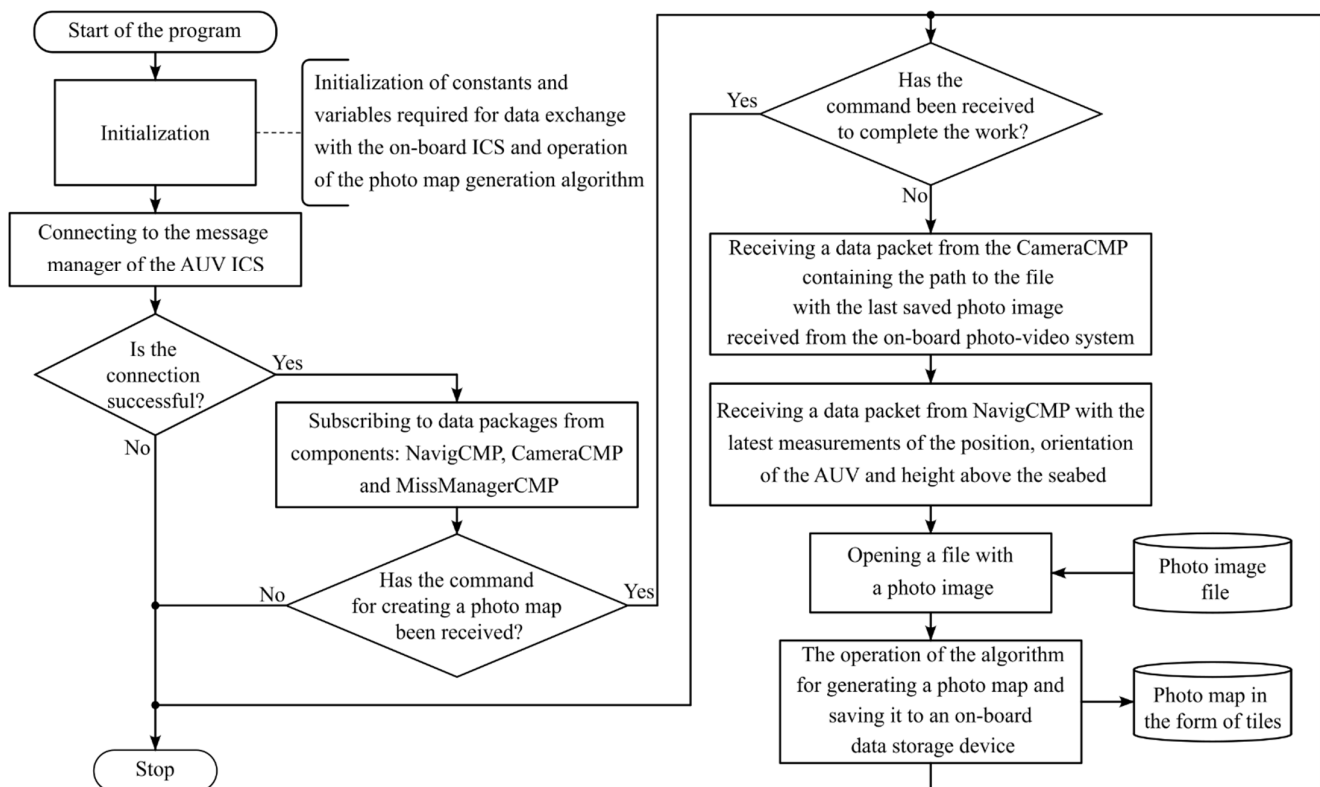


Figure 6. The scheme of operation of the developed program.

The proposed algorithm is implemented in C++ using OpenCV libraries (the source code is available at the link: <https://github.com/Fufiko/DUM-repository>, accessed on 1 December 2024). It follows the principles of object-oriented programming and makes use of the *Tiles* and *SBuffer* classes. The *Tiles* class represents a set of tiles extracted from a single image, storing the necessary information for processing and storing them. The *SBuffer* class provides a buffer for temporarily storing these tiles. To implement the buffer, we use an associative *map* container from the C++ standard library. This container stores an

array of tiles in RAM and allows us to interact with them using a key that represents the proper coordinates of each tile. The class diagram of the development program is shown in Figure 7.

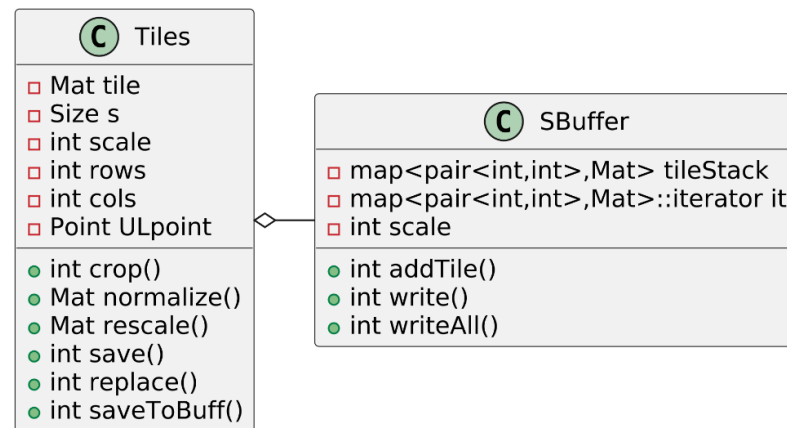


Figure 7. Program class diagram.

The *Tiles* class contains the following fields:

- *s* is the size of one tile; it is an instance of the *Size* class from the OpenCV library. It contains integer values of the tile width and height.
- *tile** is a pointer to a dynamic array of tiles described by the *Mat* class from the OpenCV library. The size of the array is determined after dividing the image into tiles.
- *scale* is the scale layer number to which the tiles belong.
- *rows* is number of tile rows.
- *cols* is number of tile columns.
- *ULpoint* refers to the coordinates of the upper-left tile described by the *Point* class from the OpenCV library.
- The *Tiles* class uses the following functions to implement the algorithm:
- *crop()* is a function that takes an image *A* as input, splits it into tiles and returns the number of tiles. The resulting tiles are stored in an array *tile*.
- *normalize()* is a function that normalizes the input image *A* based on navigation data and camera parameters and returns a normalized output image.
- *rescale()* is an auxiliary function that scales the input image *A* up or down by a specified factor.
- *save()* is a function that saves the contained tiles to the AUV storage device with overlapping tiles.
- *replace()* is a function that saves the contained tiles to the data storage with replacement, without an overlay operation.
- *saveToBuff()* is a function that saves tiles stored in a tile array of the *Tiles* class into a buffer. This function takes a pointer to the buffer where tiles should be saved as input, as well as a flag indicating whether the tiles are being saved with an overlay or simply replaced.
- The *SBuffer* class contains the following fields:
- *tileStack*—a dictionary of the *map* type, the elements of which are tiles of the *Mat* class from the OpenCV library, and the keys are the coordinates of the tiles described by the *pair* structure from the C++ standard library. The pair structure is a pair of values with any data type.
- *it* is the dictionary iterator, which is used to iterate through the dictionary elements.
- *scale* is the scale layer number to which the buffer belongs.
- The *SBuffer* class implements the algorithm using the following main functions:

- *addTile()* is a function that adds a tile to the dictionary based on the specified key (tile coordinates). If there is already a tile with those coordinates in the dictionary, the function returns zero. If the tile is new, it will be added to the dictionary and the function will return one.
- *write()* is a function that takes the boundaries of a given area as the input (*min_x, max_x, min_y, max_y*) and writes all tiles from the *tileStack* dictionary outside of this area to storage device. The function then returns the number of tiles that were recorded.
- *writeAll()* is a function that writes all the tiles stored in the *tileStack* dictionary to storage device, used to save the tiles at the end of the algorithm, and returns the number of tiles written.
- In addition to the designated classes, two additional features are implemented:
- *getValue()* is a function that returns a specific number from a given character string based on a keyword (specifically, it returns the heading angle of the AUV data by the word “Heading (deg):”). The assumption is that the data received from AUV is organized and saved in a text file with the same number of characters for each image.
- *gammaCorrection()* is a function that implements gamma correction to change the illumination of an image. The new pixel value is calculated using a formula $V'_i = V_{max} \left(\frac{V_i}{V_{max}} \right)^\gamma$, where V'_i is the output pixel value, V_i is the input pixel value, V_{max} is the maximum pixel value (for OpenCV it is 255), and γ is the gamma coefficient.

3. Results

For the experiments, a series of real images were taken by AUV, presented in [4]. The depth at which the shooting was performed was approximately 35 m. The underwater vehicle was equipped with a Videoscan-285 video camera (VIDEOSCAN LLC, Moscow, Russia). The sea bottom was photographed at a rate of 5 frames per second. An ICX285 sensor (Sony Corporation, Minato, Tokyo, Japan) with a resolution of 1392×1040 was installed in the video camera, and the focal length was 8 mm. The sensor’s physical size is $8.8 \text{ mm} \times 6.6 \text{ mm}$. The number of pixels per millimeter of the matrix is approximately 158.

Black-and-white photo images with a resolution of 696×520 pixels were used. The tiles had a width and height of 256 pixels each. The construction of the photo map started with the 17th layer and ended with the zeroth layer. During the acquisition of the photo, the onboard ICS stored data on the position, orientation and velocity of the AUV.

LattePanda V1 (LattePanda Company, Shanghai, China) was used as an on-board computer. It has an Intel Atom Z8350 processor (Intel Corporation, Santa Clara, CA, USA) with a clock frequency of 1.4 GHz. A solid-state drive (SSD) was used as a data storage device.

Table 1 shows a sample of the results from the algorithm. The average time taken to process an image was 0.173 s, which is less than the time it takes to receive each subsequent image from the photo-video system. Figure 8 shows two screenshots of a web browser window when viewing fragments of the photo map generated as a result of an experiment.

Table 1. The results of the algorithm on real data.

No.	Photo Image File Name	The Formation Time of the Initial Layer, s	The Formation Time of the Upper Layers, s	Total Time Spent, s	Number of Saved Tiles
1	09204051.jpg	0.139	0.110	0.249	12
2	09204055.jpg	0.042	0.013	0.055	0
3	09204059.jpg	0.188	0.086	0.274	6
4	09204103.jpg	0.091	0.012	0.103	4

Table 1. Cont.

No.	Photo Image File Name	The Formation Time of the Initial Layer, s	The Formation Time of the Upper Layers, s	Total Time Spent, s	Number of Saved Tiles
5	09204107.jpg	0.156	0.022	0.178	5
6	09204111.jpg	0.125	0.025	0.150	4
7	09204115.jpg	0.093	0.053	0.146	7
8	09204119.jpg	0.127	0.056	0.183	7
9	09204123.jpg	0.111	0.061	0.172	6
10	09204127.jpg	0.089	0.052	0.141	7

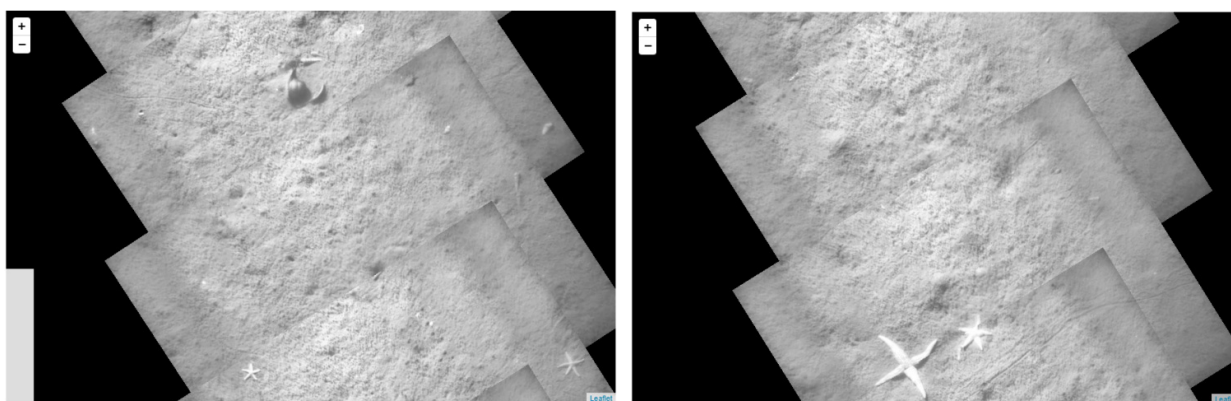


Figure 8. The fragments of the generated photo map.

4. Conclusions and Discussion

In this paper, a new method for creating a photo map from a sequence of individual photos or video frames received from on-board photo-video systems on the AUV, directly during the performance of specified survey, search, and inspection missions is proposed. This photo map is created based on data from the on-board navigation system and is stored in the form of tiles, according to a proposed algorithm that ensures the functioning of this method on a typical on-board computer. Due to the use of square tiles, it is possible to conveniently and quickly view the entire photo map using the web interface, as well as the fact that this data format is preferable for the on-board object recognition system. Additionally, the use of AUV navigation data allows the user to quickly insert images in their appropriate places on the photo map. When used in conjunction with recognition algorithms, the photo map obtained using the proposed method also allows for accurate positioning of detected objects, tying them to the world coordinate frame. This increases the accuracy of the system operation and allows for more efficient surveying and inspection tasks.

From the experimental results, it can be seen that the method and algorithm developed using the C++ programming language make it possible to implement them on standard on-board AUV computers. As a result, it became possible to quickly analyze the data obtained and make decisions directly during continuous underwater photography. Due to the use of tiles of different scales, it became possible to automatically recognize large or prolonged objects and precisely determine their geographical location during the movement of an AUV.

Author Contributions: C.L.: Conceptualization, Methodology, Formal analysis, Supervision; V.F.: Project administration, Funding acquisition, Writing—original draft, Methodology; E.M.: Writing—review & editing, Investigation, Resources, Storage tiles algorithm, Software; A.T.: Visualization, Validation, Writing—review & editing, Image transformation algorithm, Software; A.Z.: Data curation, Software, Writing—review & editing. All authors have read and agreed to the published version of the manuscript.

Funding: The project was supported by a grant from Sevastopol State University 42-01-09/253/2023-1, a grant from Far Eastern Federal University FEFU-24-04-1.02-0020, the Ministry of Science and Technology High-end Foreign Experts Recruitment Program (G2023030082L) and Guangdong Provincial Science and Technology Program—Special Project for Regional Innovation Capacity and Support System Development.

Data Availability Statement: The datasets used and/or analyzed during the current study are available from the corresponding author on reasonable request.

Conflicts of Interest: The authors declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

References

1. Martínez de Alegría, I.; Rozas Holgado, I.; Ibarra, E.; Robles, E.; Martín, J.L. Wireless Power Transfer for Unmanned Underwater Vehicles: Technologies, Challenges and Applications. *Energies* **2024**, *17*, 2305. [CrossRef]
2. Pierre-Jean, B.; Philippe, F.; Beatrice, T.; Yves, A.; Yassine, A.; Gilles, F.; Maëlle, D.; Pauline, S. Contactless data transfer for autonomous underwater vehicle docking station. In Proceedings of the OCEANS 2021: San Diego—Porto, San Diego, CA, USA, 20–23 September 2021; pp. 1–5. [CrossRef]
3. Teledyne Marine: Gavia AUV Software and Data Handling. Available online: <https://www.teledynemarine.com/brands/gavia/gavia-auv-software-and-data-handling> (accessed on 1 May 2024).
4. Gornak, V.E.; Inzartsev, A.V.; Lvov, O.Y.; Matvienko, Y.V.; Scherbatyuk, A.P. MMT 3000—Small AUV of New Series of IMTP FEB RAS. In Proceedings of the OCEANS 2006, Boston, MA, USA, 18–21 September 2006; pp. 1–6. [CrossRef]
5. Danckaers, A.; Seto, M.L. Transmission of images by unmanned underwater vehicles. *Auton. Robot.* **2020**, *44*, 3–24. [CrossRef]
6. Ahn, J.; Yasukawa, S.; Sonoda, T.; Nishida, Y.; Ishii, K.; Ura, T. An optical image transmission system for deep sea creature sampling missions using autonomous underwater vehicle. *IEEE J. Ocean. Eng.* **2018**, *45*, 350–361. [CrossRef]
7. Al-Zhrani, S.A.; Bedaiwi, N.M.; El-Ramli, I.F.E.; Barasheed, A.Z.; Abduldaem, A.; Al-Hadeethi, Y.A.; Umar, A. Underwater Optical Communications: A Brief Overview and Recent Developments. *Eng. Sci.* **2021**, *16*, 146–186. [CrossRef]
8. Fang, C.; Li, S.; Wang, Y.; Wang, K. High-Speed Underwater Optical Wireless Communication with Advanced Signal Processing Methods Survey. *Photonics* **2023**, *10*, 811. [CrossRef]
9. Zhang, M.; Zhou, H. Real-Time Underwater Wireless Optical Communication System Based on LEDs and Estimation of Maximum Communication Distance. *Sensors* **2023**, *23*, 7649. [CrossRef] [PubMed]
10. Venkatesh Alla, D.N.; Bala Naga Jyothi, V.; Venkataraman, H.; Ramadass, G.A. Vision-based Deep Learning algorithm for Underwater Object Detection and Tracking. In Proceedings of the OCEANS 2022—Chennai, Chennai, India, 21–24 February 2022; pp. 1–6. [CrossRef]
11. Bülow, H.; Birk, A. Fast and robust photomapping with an Unmanned Aerial Vehicle (UAV). In Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, St. Louis, MO, USA, 10–15 October 2009; pp. 3368–3373. [CrossRef]
12. Bulow, H.; Birk, A.; Unnithan, V. Online generation of an underwater photo map with improved fourier mellin based registration. In Proceedings of the OCEANS 2009-EUROPE, Bremen, Germany, 11–14 May 2009; pp. 1–6. [CrossRef]
13. Ait-Aoudia, S.; Mahiou, R.; Djebli, H.; Guerrou, E. Satellite and aerial image mosaicking—A comparative insight. In Proceedings of the 2012 16th International Conference on Information Visualisation, Montpellier, France, 11–13 July 2012; pp. 652–657. [CrossRef]
14. Fischler, M.A.; Bolles, R.C. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM* **1981**, *24*, 381–395. [CrossRef]
15. Lowe, D.G. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis.* **2004**, *60*, 91–110. [CrossRef]
16. Pathak, K.; Pflingsthor, M.; Bülow, H.; Birk, A. Robust estimation of camera-tilt for iFMI based underwater photo-mapping using a calibrated monocular camera. In Proceedings of the 2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, 6–10 May 2013; pp. 5111–5116. [CrossRef]
17. Gracias, X.; Negahdaripour, S. Underwater mosaic creation using video sequences from different altitudes. In Proceedings of the OCEANS 2005 MTS/IEEE, Washington, DC, USA, 18–23 September 2005; pp. 1295–1300. [CrossRef]
18. Mizuno, K.; Tabet, S.; Matsumoto, Y.; Sakamoto, S.; Sugimoto, Y.; Ogawa, T.; Sugimoto, K.; A Jimenez, L.; Terayama, K.; Fukami, H.; et al. Development of a towed optical camera array system (SSS: Speedy Sea Scanner) for sea environmental monitoring. In Proceedings of the 2018 OCEANS—MTS/IEEE Kobe Techno-Oceans (OTO), Kobe, Japan, 28–31 May 2018; pp. 1–5. [CrossRef]
19. Abbadi NK, E.; Hassani SA, A.; Abdulkhaleq, A.H. A Review Over Panoramic Image Stitching Techniques. *J. Phys. Conf. Ser.* **2021**, *1999*, 012115. [CrossRef]

20. Zhang, H.; Zheng, R.; Zhang, W.; Shao, J.; Miao, J. An Improved SIFT Underwater Image Stitching Method. *Appl. Sci.* **2023**, *13*, 12251. [CrossRef]
21. Garca, R.; de Castro, J.P.; Verd, E.; Jess, M.; Mara, L. Web Map Tile Services for Spatial Data Infrastructures: Management and Optimization. In *Cartography—A Tool for Spatial Analysis*; InTech: Rijeka, Croatia, 2012. [CrossRef]
22. Kobakian, S.; Cook, D.; Duncan, E. A hexagon tile map algorithm for displaying spatial data. *R J.* **2023**, *15*, 6–16. [CrossRef]
23. Statham, N.; Jacob, J.; Fridenfalk, M. Photogrammetry for Game Environments 2014-2019: What Happened Since The Vanishing of Ethan Carter. In *Proceedings of the 2020 DiGRA International Conference: Play Everywhere, DiGRA'20, Tampere, Finland, 2–6 June 2020*; pp. 1–20.
24. Leaflet: Overview: Leaflet—A JavaScript Library for Interactive Maps. Available online: <https://leafletjs.com/> (accessed on 1 May 2024).
25. Reis, D.; Kupec, J.; Hong, J.; Daoudi, A. Real-time flying object detection with YOLOv8. *arXiv* **2023**, arXiv:2305.09972. [CrossRef]
26. OpenCV: Introduction: Open-Source Computer Vision. Available online: <https://docs.opencv.org/3.4/d1/dfb/intro.html> (accessed on 1 May 2024).
27. Linder, W. *Digital Photogrammetry: A Practical Course*, 4th ed.; Springer: Berlin-Heidelberg, Germany, 2016. [CrossRef]
28. Deutsch, L.P. DEFLATE Compressed Data Format Specification version 1.3. *IETF* **1996**, *1951*, 1–17. [CrossRef]
29. Filaretov, V.; Yukhimets, D.; Mursalimov, E. The Universal Onboard Information-Control System for Mobile Robots. *Procedia Eng.* **2015**, *100*, 737–745. [CrossRef]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.