big data and cognitive computing

MDPI

Article

An Approach for Opening Doors with a Mobile Robot Using Machine Learning Methods

Lesia Mochurad [1,*], Yaroslav Hladun [1], Yevgen Zasoba [1] and Michal Gregus [2]

[1] Department of Artificial Intelligence, Lviv Polytechnic National University, 79905 Lviv, Ukraine
[2] Faculty of Management, Comenius University in Bratislava, 820 05 Bratislava, Slovakia
* Correspondence: lesia.i.mochurad@lpnu.ua; Tel.: +380-97-868-30-14

Abstract: One of the tasks of robotics is to develop a robot's ability to perform specific actions for as long as possible without human assistance. One such step is to open different types of doors. This task is essential for any operation that involves moving a robot from one room to another. This paper proposes a versatile and computationally efficient algorithm for an autonomous mobile robot opening different types of doors, using machine learning methods. The latter include the YOLOv5 object detection model, the RANSAC iterative method for estimating the mathematical model parameters, and the DBSCAN clustering algorithm. Alternative clustering methods are also compared. The proposed algorithm was explored and tested in simulation and on a real robot manufactured by SOMATIC version Dalek. The percentage of successful doors opened out of the total number of attempts was used as an accuracy metric. The proposed algorithm reached an accuracy of 95% in 100 attempts. The result of testing the door-handle detection algorithm on simulated data was an error of 1.98 mm in 10,000 samples. That is, the average distance from the door handle found by the detector to the real one was 1.98 mm. The proposed algorithm has shown high accuracy and the ability to be applied in real time for opening different types of doors.

Keywords: door-handle detection; robotics door operation; real-time object detection; RGBD stereo camera; kinematic model learning; robotic arm

Citation: Mochurad, L.; Hladun, Y.; Zasoba, Y.; Gregus, M. An Approach for Opening Doors with a Mobile Robot Using Machine Learning Methods. *Big Data Cogn. Comput.* **2023**, *7*, 69. https://doi.org/10.3390/bdcc7020069

Academic Editor: Moulay A. Akhloufi

Received: 11 February 2023
Revised: 30 March 2023
Accepted: 3 April 2023
Published: 6 April 2023

1. Introduction

An autonomous mobile robot (AMR) is a machine capable of moving and performing specific tasks without regular human assistance [1]. Each task requires a different degree of autonomy. A high degree of autonomy is required for tasks such as cleaning a building [2], delivering food [3], etc. For an AMR, it is important to perform human-like actions in typical indoor environments in order to develop an extensive range of tasks [4,5]. One of these actions is opening various types of doors. This task is important for any operation that involves the robot moving from one room to another.

In most cases, LiDAR sensors are used to localize a robot in a building [6]. Usually, the data from LiDAR sensors are sent to the input of either the Particle Filter (PF) algorithm [7] or the Kalman Filter (KF) algorithm [8]. The problem with low-cost LiDAR sensors is that they need to be more accurate to ensure that the robot hand hits the door handle, as an error of just 2–3 cm can cause the detector to miss the door, and therefore, the door will not be opened. This leads to the obvious conclusion that machine learning (ML) algorithms are required in this task [9–11] since most AMRs for such operations have only a camera as a sensor to see the door handle, which makes the direction of object recognition in the camera image relevant in this task. This means that the position of the door handle relative to the detector needs to be constantly updated in order to hit it and successfully open the door.

Wang et al. [12] investigated the control of mobile robotic arms for opening doors. To accomplish this task, they first used an image-recognition method based on YOLOv5 to obtain the position of the door handle. Next, they employed the simulation platform CoppeliaSim to facilitate the interaction between the robotic arm and the environment. Third,

they developed a control strategy based on a reward function to train the robotic arm for the door-opening task in real-world environments. The experimental results demonstrated that their proposed method could accelerate the training process's convergence, reduce the robotic arm's jitter, and enhance control stability.

Palacin et al. [13] proposed the non-parametric calibration of the inverse kinematic matrix of a three-wheeled omnidirectional mobile robot based on genetic algorithms to minimize the positioning error registered. This approach provided an average improvement of 82% in estimating the final position and orientation of the mobile robot.

This research aims to develop and implement a universal algorithm for constructing a door-opening trajectory by an autonomous mobile robot with a seven-stage robotic arm, an RGBD camera, and four LiDAR sensors responsible for localization. First, the algorithm must be universal, which means it must work on many types of doors used in everyday life. Secondly, the algorithm should be computationally efficient and have a relatively low computational complexity, so it can be executed on computers commonly used in such projects, such as Nvidia Jetson or Raspberry Pi.

The object of this research is a cleaning robot with a robotic arm, a unique robotic arm tip, a wheeled base, an RGBD stereo camera, and four LiDAR sensors.

The subject of this research is a system for opening any door using a robotic arm, a wheeled base on which the robotic arm is set, and localization devices.

At first, the research will consist of building a basic door-opening trajectory without using any ML methods. The next step will be to integrate ML and AI methods to improve accuracy, such as neural networks to detect objects in the image: YOLOv5 [14]; clustering methods [15]: DBSCAN, K-Means, OPTICS, EM-algorithm; and methods for finding a plane in an image from an RGBD camera: RANSAC [16].

The practical importance of this study is that a universal door-opening algorithm has been proposed that can be used in many projects, including driving AMRs in a room where it is necessary to move from one room to another.

Robotics uses a considerable number of algorithms from various directions of mathematics and computer science. Here, we outline the main ones used in this work, but they are also standards usually used in similar experiments.

- Proportional-integral-differential (PID) law [17,18]. An example of the usage of this algorithm in robotics is the regulation of the wheel speed to achieve a given linear and angular acceleration of an AMR. At first look, such a task may seem quite simple: we calculate the rotational speeds of each of the four wheels and then apply a certain starting voltage to each wheel, and if any wheel does not rotate at the frequency we need, we reduce or increase the voltage in a certain step until the frequency is equal to the one set. In fact, this is a particular case of a PID controller. The equation of the PID law is as Equation (1).

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\mathrm{d}\tau + K_d \frac{\mathrm{d}e(t)}{\mathrm{d}t}, \quad (1)$$

where K_p, K_i, K_d represent the coefficients of the proportional, integral, and differential components, respectively; $e(t)$ indicates the measured value deviation from the set value at a given time; $u(t)$ indicates the control function. The PID control law helps to move the AMP on the floor quite accurately, which plays an important role in the door-opening process.

- Inverse kinematics using the Jacobian inverse. In computer animation and robotics, inverse kinematics [19] is the mathematical process of calculating the variable joint parameters required to place the end of a kinematic chain, such as a robot arm endeavor, in a given position and orientation relative to the chain beginning. Given the joint parameters, the position and direction of the end of a chain, such as a character arm or robot arm, can be calculated directly through a few uses of trigonometric formulas. This is a process known as forward kinematics. However, in general, inverse kinematics is much more complicated. There are approaches to solving the

inverse kinematics task analytically [20], but iterative methods are used for available cases with many joints. One such method is inverse kinematics using the Jacobian inverse. This is a simple but effective way to solve the inverse kinematics task used in this research.

- Particle filters, or sequential Monte Carlo methods [21,22], are used to solve filtering problems in signal processing and Bayesian statistical conclusion. The task of filtering is to estimate internal states in dynamic systems when partial observations are made using sensors that contain a measurement error. In this research, a particle filter is used to estimate the position and orientation of a robot. The sensor data used are the results of a Lidar scan.

One example of a mobile robot capable of opening a door is the SpotMini robot from Boston Dynamics [23]: a small robot called SpotMini opens the door (on itself) with the help of a unique gripper placed on top, and it enters the room, supporting the door. Unfortunately, there is no detailed description of this algorithm and only some videos are available, but the robot has a built-in RGBD stereo camera, and it is possible that data from it were used to open the door. The design of the SpotMini robot with a gripper fixed on top is perfect for the task of opening doors. It is quite small and uses legs, which provide improved maneuverability when compared with the use of conventional wheels.

A similar analogy to our work is the algorithm for opening doors with a robot manufactured by Toyota–Human Support [24]. This robot is similar in size and has an RGBD stereo camera, which it uses to find the door handle. This mobile robot uses an algorithm for opening doors that is comparable to that proposed in this paper. This robot is different in some respects. For example, its arm has five degrees of flexibility.

As mentioned above, this work aims to develop a universal algorithm for building a door-opening trajectory using an autonomous mobile robot with a seven-stage robotic arm, an RGBD camera, and four LiDAR sensors responsible for localization. The focus should be on the development and realization of the part of the algorithm responsible for estimating the position and direction of the door handle, as this is the universal part of the whole algorithm and can be used for many types of AMRs because the main requirement is the availability of an RGBD stereo camera and a mechanism for finding its position and direction relative to the robot.

In concept, the realization of the door-opening algorithm can be divided into two isolated stages. The first stage is the construction of the door-opening trajectory based on a priori data of the door handle position and orientation. It is not considered in detail in this research since the door-opening trajectory for two different robots will be radically different, since one robot can use a robotic arm with seven degrees of flexibility and the other has six, and this already introduces major changes to the concept of trajectory construction. Usually, robots differ in much more than the number of joints in the arm, so the realization of this logic will be described in abstract steps. The second stage is finding the position and orientation of the handle using machine learning methods. This is a key part of the work, as a completely similar approach can be used for a robot that is completely different from the one used in this project. All that such a robot needs is an RGBD stereo camera and a mechanism for finding its exact position relative to the robot's coordinates. This second stage can be called the algorithm for clarifying the position and orientation of the door and door handle, as it uses a priori and sensor data (in our case, only the RGBD stereo camera).

We must have some a priori data about the position and orientation of the door and door handle on the map. That is, there will be a map of the room onto which the door and the position of the handle can be added, and this information will be read out when the algorithm is initialized. Such information is not accurate in the first place, and neither is the robotic arm [25] nor the localization [26]. To enable the robot to open a door, we must continually update the position of both the door and the door handle in relation to the robot, using information obtained from the RGBD stereo camera. To accomplish this, we will utilize an algorithm designed to refine the position and orientation of the door and its handle. The data obtained from this algorithm will then be fed into a separate algorithm

that is responsible for constructing a trajectory for the robot's arm to follow while opening the door. The resulting output of this process will be the specific trajectory that the robot will use to successfully open the given door, using its initial position and arm state as additional input parameters.

In this article, we will present the following sections: Section 2 will introduce input and output data, as well as a description of the proposed algorithm. Section 3 will present the results of numerical experiments conducted to test the efficacy of the proposed algorithm. Finally, in Section 4, we will draw conclusions from our findings and discuss potential avenues for future research.

2. Materials and Methods

2.1. Input Data

Let us suppose that we have a certain map of the room, with doors defined by an affine transformation D with a rotation matrix $R^{(D)}$ and a translational width $T^{(D)}$:

$$D = \begin{pmatrix} r_{11}^{(D)} & r_{12}^{(D)} & r_{13}^{(D)} & t_1^{(D)} \\ r_{21}^{(D)} & r_{22}^{(D)} & r_{23}^{(D)} & t_2^{(D)} \\ r_{31}^{(D)} & r_{32}^{(D)} & r_{33}^{(D)} & t_3^{(D)} \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} R^{(D)} & T^{(D)} \\ \overline{0} & 1 \end{pmatrix}. \quad (2)$$

Note that the orientation of the door along the axes x, y matches the point of attachment of the door to the wall, and along the axis it matches the floor. The handle of the door will be set by a transformation H_D relative to the transformation of the door. The position of the robot will be set by a transformation M in the global coordinate system, as in the case of the door (see Figure 1).

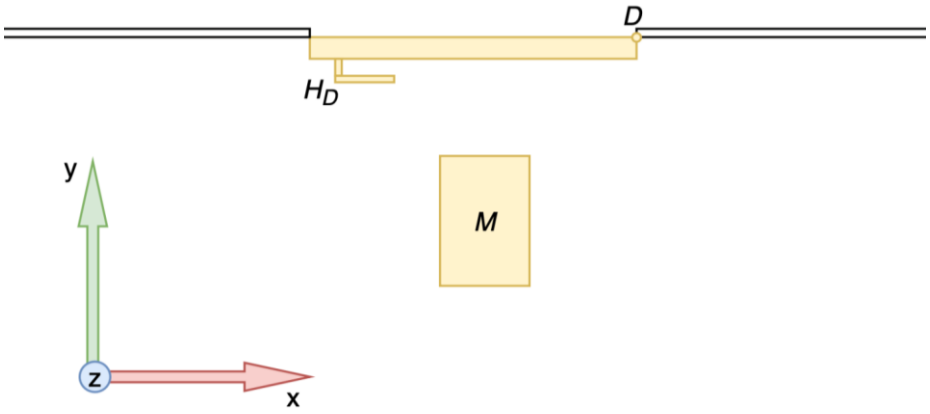

Figure 1. Schematic view of the door, door handle and robot.

The starting position of the robotic arm will be set by a vector $\overline{a} = (a_1, a_2, \ldots, a_7)$ of seven elements, since the robotic arm has seven joints. Each element of the vector represents the corner at which the corresponding joint is rotated.

The last thing that needs to be added is the image from the RGBD camera. Let us assume that the camera returns an image and a depth map in n by m pixels. Since the camera will be looking at the door handle for some time, we will obtain l images from it. So, the resulting data object from the camera will be $I \in \mathbb{R}^{l \times 4 \times m \times n}$. It can also implement additional logic, such as filtering out blurry [27] or outdated images. It is also possible to filter out similar images [28], so that only a few of them are left.

Thus, the input to the door-opening algorithm is $D, H_D, \overline{a}, M, I$.

2.2. Output Data

The output for this task will be a set of N pairs $T = \{(M_1, \bar{a}_1), (M_2, \bar{a}_2), \ldots, (M_N, \bar{a}_N)\}$, where M_i is the robot's position and $\bar{a}_i$ is the state of the arm joints. In other words, it is the trajectory that the robot should follow, given some physical constraints [29,30], to open the door.

2.3. Proposed Algorithm

Before the algorithm starts working, the robot has to perform certain actions, namely, drive as close to the door as possible and point the camera first at the door itself and then at the door handle, in order to fill in the object with images I.

Then, its position and orientation M are recorded, as well as the hand's position $\bar{a}$, and together with the images I and the a priori position of the door D and door handle H_D, they are sent to the door-opening algorithm.

It is also important to note that in the process of opening the door, we can receive new images from the camera and additionally estimate the position of the door handle, and if the estimate is very different from the one we made before the algorithm started, we can rebuild the opening trajectory in real time. For many robots, this is not possible due to the specifics of the design because they cannot use the manipulator and point the camera at the same time. However, in general, the architecture of the developed system will provide such functionality, so let us introduce a metric for the distance between two affine transformations $d(x, y)$ and a threshold ξ. That is, if $d\left(D^{(t)}H_D^{(t)}, D^{(t+1)}H_D^{(t+1)}\right) < \xi$, then we stop the algorithm and rebuild the trajectory with new $D, H_D, \bar{a}, M, I$.

During the research, many combinations of different methods were tested to find the door handle. Finally, our proposed algorithm consisted of the following steps:

1. RANSAC (Random Sample Consensus) is a statistical algorithm commonly used in computer vision and image processing to estimate parameters of a mathematical model from a set of observed data points that may contain outliers or errors. The goal of RANSAC is to identify the inliers (data points that fit the model well) and reject the outliers (data points that do not fit the model well). The algorithm works by randomly selecting a subset of data points, fitting a model to them, and then testing the remaining points to see how well they fit the model. If a sufficient number of data points fit the model well, it is considered a good fit and the algorithm terminates. Otherwise, another subset of data points is selected, and the process is repeated.

RANSAC is often used in computer vision tasks such as image registration, stereo vision, and structure from motion, where it is necessary to estimate a mathematical model from noisy data with outliers. RANSAC is popular because it is relatively fast and simple to implement and can provide robust estimates even in the presence of a large number of outliers. In the context of RANSAC, outliers are data points that do not conform to the underlying mathematical model being estimated. Outliers can arise due to measurement errors, noise, or other factors that cause the data to deviate from the expected pattern. In our case, all points further than a fixed distance from the door plane are considered outliers.

RANSAC is designed to handle datasets that contain outliers by iteratively fitting models to subsets of the data and rejecting points that do not fit the model well. In each iteration, the algorithm selects a random subset of points and fits a model to them. It then uses the model to predict the values of the remaining points and computes a measure of how well each point fits the model. Points that are too far from the model (i.e., outliers) are then discarded, and the algorithm repeats the process with a new subset of points.

By repeating this process many times, RANSAC is able to estimate the parameters of the mathematical model in a way that is robust to outliers. The final model is obtained by using all of the inlier points that were selected in the iterations and can be used for further processing or analysis.

In our case, it will be used to find the coefficients of the door plane, which are actually equal to the orientation of the door handle as it is attached to it (the door). The door plane will be located in the pixel coordinate space. That is, we have a depth map:

$$D = \begin{pmatrix} d_{11} & d_{12} & \dots & d_{1n} \\ d_{21} & d_{22} & \dots & d_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ d_{m1} & d_{m2} & \dots & d_{mn} \end{pmatrix}, \tag{3}$$

where d_{ij} is the pixel depth i, j. So, we have a set on which we will execute the RANSAC algorithm:

$$\widetilde{D} = \left\{ (i, j, D_{ij}) \middle| (i, j) \in \overline{1n} \times \overline{1m} \right\} \tag{4}$$

By performing RANSAC on the set $\widetilde{D}$, we get the plane $ax + by + cz + d = 0$.

2. All points of the point cloud (PC) that are close to the plane will be deleted. Additionally, all points that lie outside the BB that the YOLOv5 ML model finds will be deleted.

So, we need to find a rectangle (Bounding box (BB)) with a door handle on it. That is, we are given an image as input $i \in \mathbb{R}^{4 \times m \times n}$, and the output is a set $B = \left\{ (x_j, y_j, h_j, w_j) \middle| j = \overline{1, k} \right\}$ of elements k, each of which describes the coordinate, height, and width of the BB. The YOLOv5 architecture is used for this purpose [31–33]. Object detection is aimed at creating objects from the input images and then passing these objects through a prediction system to draw frames around the objects and predict their classes.

Arduengo et al. [24] proposed a framework to adaptively operate common doors using a manipulator embedded in a mobile robot. The researchers utilized their own dataset to train YOLOv5 for the purpose of door-handle detection. For this study, the aforementioned model was utilized. However, it should be noted that the model typically generates a BB that does not entirely encompass the visual aspect of the door handle (see Figure 2). Therefore, the BB is generally expanded by a fixed number of pixels.

Figure 2. Door-handle detection result using YOLOv5 for one of the test cases.

3. The resulting PC is then fed into the DBSCAN clustering model. We take the largest cluster from it and consider its centroid to be the position of the door handle. We consider the normal vector of the door plane to be the orientation.

In computer vision, DBSCAN can be used for various tasks, such as object detection, image segmentation, and feature extraction. For instance, DBSCAN can be applied to segment an image by clustering pixels based on their similarity in color or texture. Addi-

tionally, DBSCAN can also be utilized to detect and group together objects in an image by clustering features such as key points or descriptors. By leveraging the density-based nature of DBSCAN, it can be effective in handling complex and varying object shapes, and it is robust to noise and outliers.

DBSCAN can also be applied to point clouds obtained from depth images, which are commonly used in 3D computer vision tasks. By treating the 3D point cloud as a set of spatial coordinates, DBSCAN can cluster points that are close together in 3D space while considering points that are more sparsely distributed as noise or outliers. This can be particularly useful in tasks such as object recognition and segmentation, where the 3D point cloud of an object can be clustered to identify its boundaries and separate it from the background.

One benefit of using DBSCAN for point cloud segmentation is its ability to handle non-uniformly distributed points, making it suitable for noisy or sparse point clouds. Additionally, DBSCAN can handle point clouds with varying densities, making it an appropriate choice for point clouds that may have missing data or varying resolution.

Note that the DBSCAN algorithm takes as input the proximity matrix and two parameters: the radius of the ϵ circle and the number of neighbors. Given some symmetric distance, function $\rho(x, y) = \rho(y, x)$ and constants ϵ, m are given. We call the area $E(x)$, for which $\forall y : \rho(x, y) \leq \epsilon$, the ϵ-vicinity of the object. The root object is an object whose surrounding contains at least m elements: $|E(x)| \geq m$. An object p is directly accessible from the object q if $p \in E(q)$ and q is the root object. The object p is reachable from the object if $\exists p_1, p_2, \ldots, p_n, p_1 = q, p_n = p$, such that $\forall i \in 1 \ldots n - 1 : p_{i+1}$ is directly reachable from p_i. The algorithm is as follows: first, select any root object from the dataset. Then, we mark this object and select all its directly reachable neighbors and add them to the traversal list. Then, for each of the vertices, if they are also the root, we perform similar operations. If we have traversed all the points in a given cluster, we can start with any other root vertex.

DBSCAN only requires a few computing resources, which is important in this task since we will cluster points thirty times per second, corresponding to the camera frame rate. Next, we will consider other clustering methods and compare their complexity.

A relatively simple and popular clustering method is K-Means [34], which has low computational complexity, which is essential for our task since the robot is usually not allowed to stand in one place for a long time while calculating the trajectory of the door opening.

This algorithm can be implemented with asymptotic complexity, where is the number of dimensional vectors, is the number of clusters, and is the number of iterations required to achieve accuracy. However, as can be seen from this description of the complexity estimate, we have a hyperparameter responsible for the number of clusters. We need to know the number of clusters in advance for our task. If we set any number that is not very large, it will significantly affect the accuracy, since this parameter will be different for each door, which is related to its size and the form of the door handle.

As in the case of DBSCAN, another density-based data clustering algorithm is OPTICS (ordering points to identify the clustering structure) [35,36]. In general, the algorithm is very similar to DBSCAN, which was actually used in the final implementation. Still, it solves one of the main weaknesses of DBSCAN: the problem of identifying significant clusters in datasets of different densities. The database objects must be ordered (in linear time) so that objects close to each other will be neighbors in the final ordering. In addition, a particular distance is stored for each point, representing the density acceptable for a cluster to have both neighboring points belong to the same group. OPTICS can be used instead of DBSCAC. However, DBSCAN was used because we do not need the additional improvement provided by the OPTICS algorithm since we usually just need to find the most significant cluster, and whether all the others are also considered clusters or outliers does not play a strong role anymore. That is why we chose DBSCAN for this work.

Another alternative to the DBSCAN algorithm could be the EM algorithm [37,38]. This method estimates the maximum similarity of parameters of probable models when the model depends on some hidden parameters. All iterations of this method consist of two

steps: expectation and maximization. The E-step (expectation) calculates the expected value of the probability function. During this step, hidden variables are treated as observables. The M-step (maximization) calculates the maximum likelihood estimate. As a result, the expected similarity that was calculated in the previous step increases. This value is then used for the E-step in the next iteration. The algorithm is convergent and runs until a certain accuracy criterion is reached. It can also be used as a clustering method since the hidden parameters, in this case, are the mean and variance of normal distributions. This means that the algorithm searches for the parameters of the distribution that consist of the sum of normal distributions and from which the data were most likely generated. The reason why this algorithm was not used is, again, the presence of the hyperparameter, as in the case of K-Means. Additionally, this algorithm takes a long time to execute, as it calculates complex mathematical formulas at each iteration, which usually takes significant time at the program level. Making the algorithm run in fewer iterations is possible, but this requires sacrificing accuracy.

This work generally tested many other methods that did not show the necessary efficiency. There were attempts to use image segmentation [39] instead of object detection, but the results did not show the required accuracy. This is primarily due to the need for sufficiently large datasets for training neural networks, a consequence of datasets for segmentation that are quite difficult to create (much more complex than for object detection).

Efforts have been made to substitute the RANSAC algorithm with linear regression, but such attempts were not able to deliver the desired accuracy. The reason for the failure of linear regression in identifying the plane of the door is that it optimizes the mean square error function, which is not well-suited for identifying the plane with the largest number of points around it. Hence, for this task, the RANSAC algorithm is better suited.

There were also attempts to not use the clustering algorithm afterward and instead try to connect the points with a three-dimensional model of the door handle [40]. Such an algorithm is entirely accurate but requires a lot of computing resources that are usually unavailable on an AMR of this class. Therefore, in the end, it was decided to find the centroid of the central cluster.

3. Results

This section describes the simulation and experimental results obtained in this paper. The omnidirectional mobile robot Dalek is based on the use of four mecanum wheels. This type of wheel has rollers evenly distributed around the rim by which it interacts with the surface of the movement. This wheel allows the vehicle to independently move its wheels, to provide controlled movement in any direction. This design greatly simplifies movement for a robot that moves indoors. For example, it does not need to turn 90 degrees to drive to the left. In tight spaces, the robot often cannot physically turn at all, and this problem is solved by the mecanum wheel design. At a height of up to 20 cm, four LiDAR sensors help localize the robot. These are LiDAR sensors from the Chinese manufacturer SLAMTEC-RPLIDAR A3. They allow you to find the distance to an object no further than 25 m from the robot, with an error of up to 5 cm. These are two-dimensional LiDAR sensors, and in this robot design, they are installed horizontally, so the output is cut in the form of points parallel to the floor plane. As for the robot arm, this robot has a UFactory farm 7. This arm has seven degrees of freedom and weighs 13.7 kg. Its repeatability is one-tenth of a millimeter. A repeatability of one-tenth of a millimeter means that if you put the arm in any position that the limits of the joints allow, record the coordinates in the space of these joints, put the arm in any other position, and tell it to come to the recorded coordinates, it will do so with an accuracy of one-tenth of a millimeter. This is an important nuance since experiments have shown that the distance from the endeavor position calculated by direct kinematics to the actual endeavor position is much greater than the repeatability and is close to 1 cm. This is one of the reasons for using the MN to dynamically find the position of the handle relative to the camera during the trajectory execution since an error of 1 cm can cause a missed hit on the door handle. Note that the maximum speed of the hand

detector is 1 m/s. An RGBD stereo camera manufactured by Intel, the RealSense D435i, is mounted on the sensor. This camera captures depth images at a frequency of 30 frames per second with a resolution of 1280 by 720 pixels. The camera can calculate image depth for objects no closer than 10 cm and no further than 10 m away. The error of the depth image increases with distance. For example, if the thing is 30 cm away from the sensor, it is plus or minus 1 cm. If the object is 5 m away, the error will be several tens of centimeters.

All software was executed on an NVIDIA Jetson TX2 minicomputer. The proposed door-handle detection algorithm can be used for different types of hands and mechanisms, which depends more on engineering solutions. Still, the robot must be physically able to do so and have an RGBD stereo camera to recognize the door plane and the door handle. In this case, we used an Intel RealSense D435i (see Figure 3).

Figure 3. RGBD stereo camera Intel RealSense D435i.

The developed algorithm was tested on a SOMATIC robot of the Dalek version (see Figure 4). The robotic system under consideration is equipped with a robotic arm that possesses seven degrees of freedom, as well as the RGBD stereo camera that was previously described.

Figure 4. Robot manufactured by SOMATIC, Dalek version.

It is important to note that, prior to initiating the door-opening task, a map of the room needs to be generated and the door parameters need to be set. This includes determining the door's width, height, thickness, the height at which the door handle is situated, and other relevant parameters of the door handle. Then, when the map is created, you can

create a basic door-opening plan in the corresponding plan creation menu (see Figure 5) by selecting the appropriate door. In Figure 6, a door is selected to build a trajectory, which can be more complex as the room map can be more difficult. For example, you can recreate an approximate trajectory to see if the robot will not hit the walls. First, it will use the appropriate endeavor to grab the door handle (see Figure 7). Then, in Figure 8, you can see the process of opening the door.

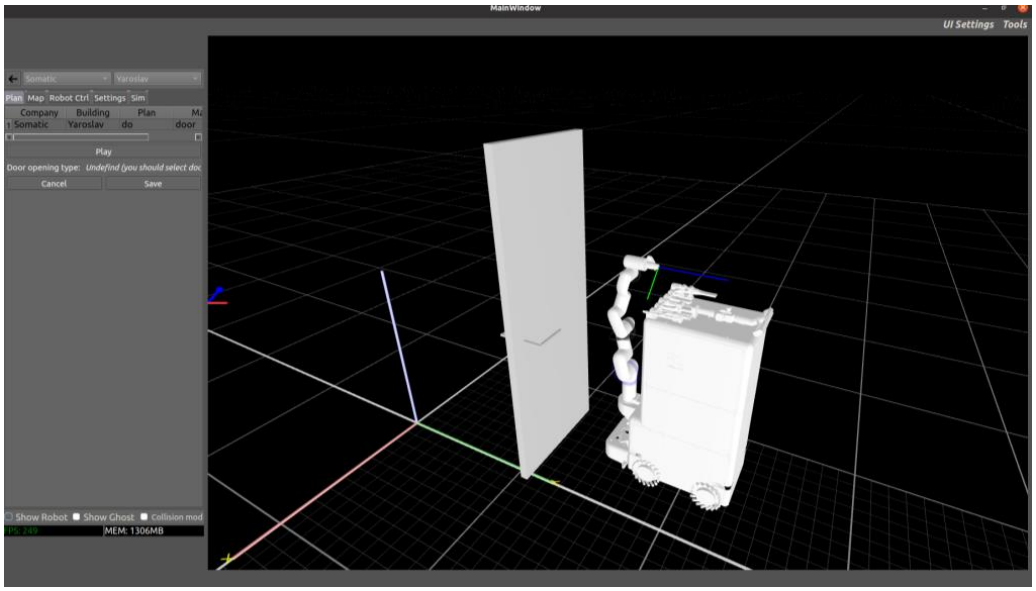

Figure 5. The menu for creating a plan.

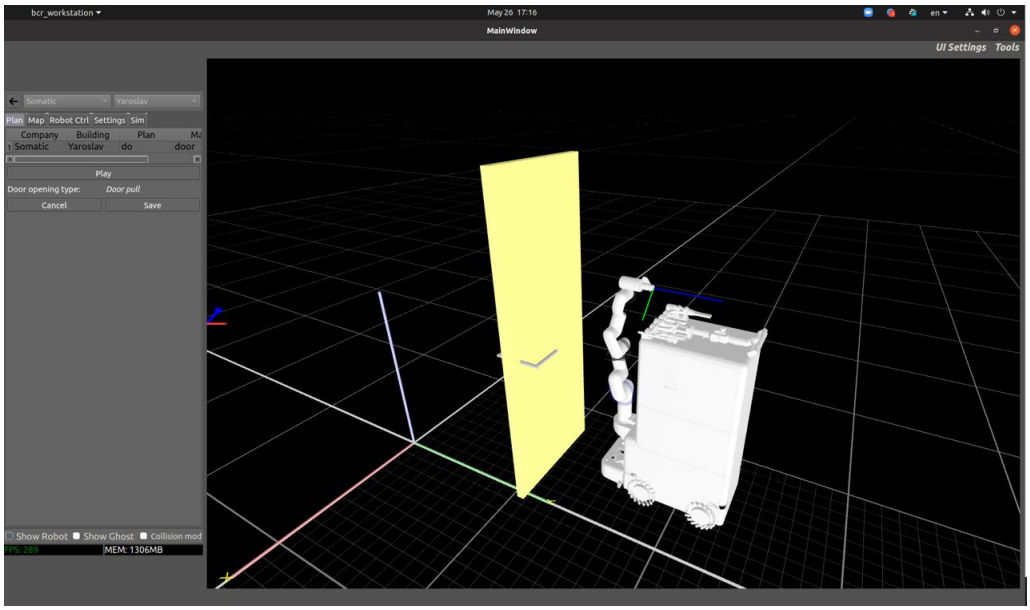

Figure 6. Selected doors for building a door-opening algorithm.

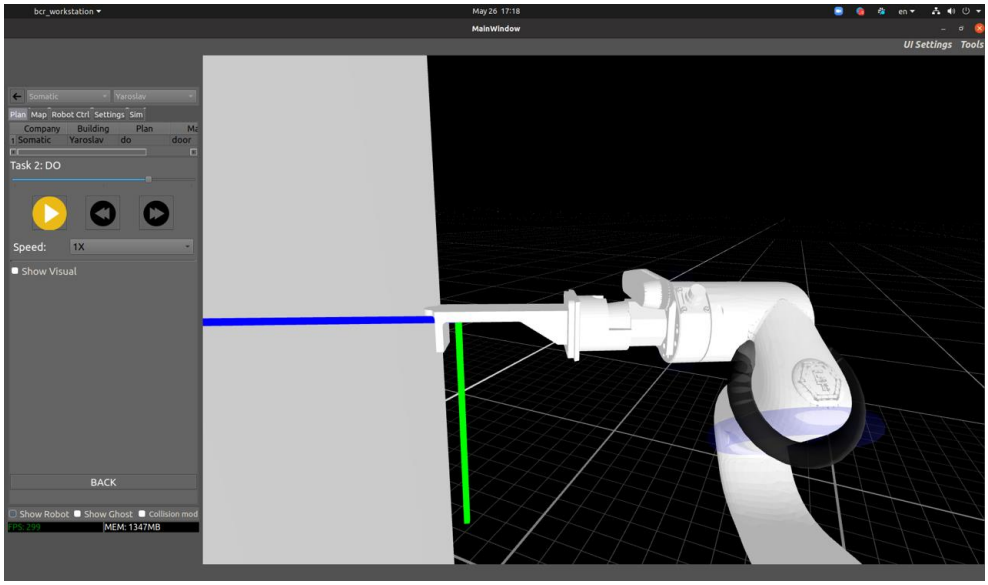

Figure 7. Suitable end effector for gripping door handles.

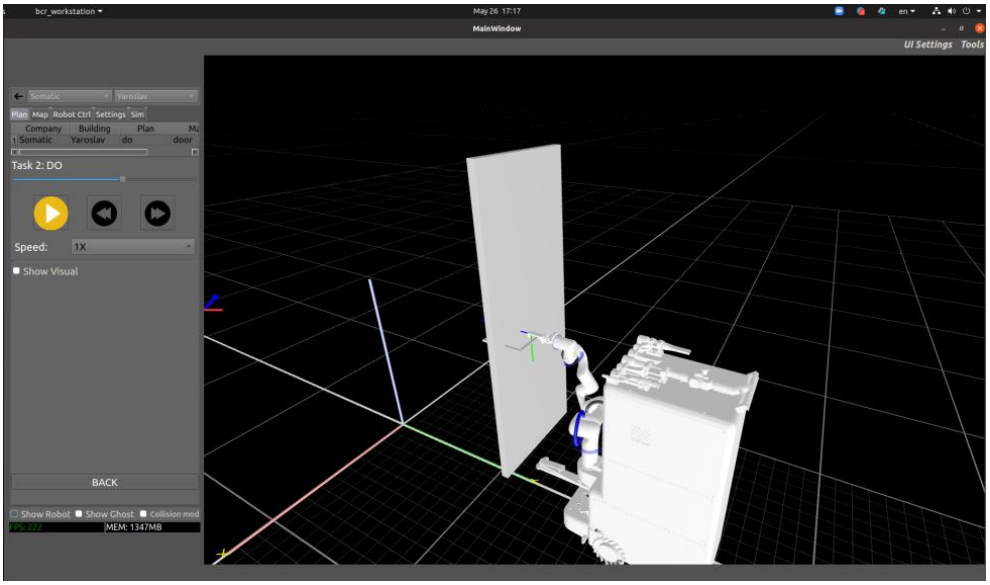

Figure 8. The first stage of the built trajectory execution.

Having tested the proposed algorithm on the newly developed software solution in practice, we present the accuracy marks received.

So, the algorithm was tested in two ways: in simulation and on a real robot. Simulation testing allows us to understand whether the algorithm is valid for different combinations of environmental parameters, such as door width, handle length and whether the door is opened clockwise or anti-clockwise. Testing on a real robot allows you to understand whether the result in the simulation correlates with the natural world for several exceptional cases of environmental parameters. Furthermore, the algorithm was tested in two variations on a real robot: with and without ML. In the second case, the a priori data of the position and orientation of the door and the door handle were used.

3.1. Evaluating Accuracy in Simulation

Testing was performed on 10,000 combinations of door width, door handle position on the door, handle radius, handle length, opening variation (clockwise or counterclockwise), and some other parameters. The proposed algorithm worked successfully in all cases,

but this result is quite expected in the simulation, and it hardly matches the real world. However, we estimate some values that are likely similar to those in the real world. This is the distance from the door handle's found position to the door handle's actual position. The average length to the actual position is 1.98 mm. Figure 9 shows the distribution of this value.

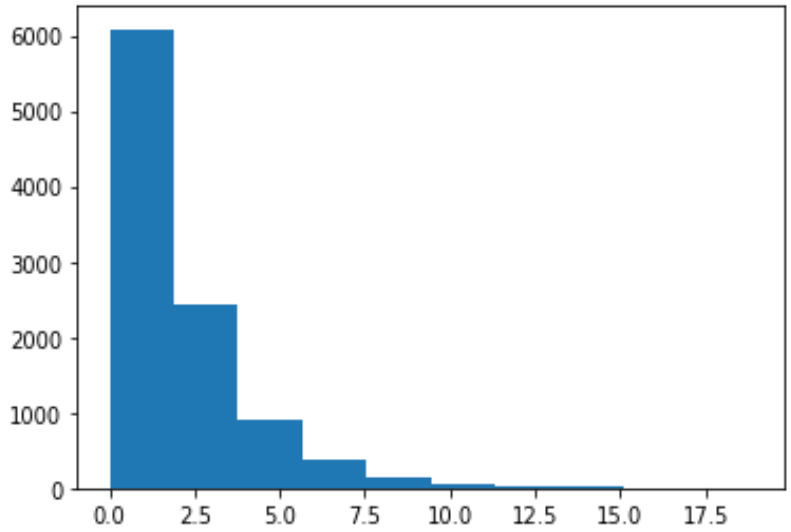

Figure 9. The distribution of the average distance from the real position of the door handle to the one found (the OX axis is the distance in mm, and the OY axis is the number of cases).

3.2. Evaluation of Accuracy on a Real Robot

The number of successful attempts to the total number of shots was used to estimate the accuracy in a natural environment. We tested two algorithm variations: without and with the use of ML.

As we can see from Table 1, the algorithm for detecting a door handle from an RGBD camera image significantly increases the number of successful attempts.

Table 1. Test results on a real robot.

	Successful Attempts	Number of Attempts
Without the use of ML	30	100
With the use of ML	95	100

It was also established based on numerical experiments that the average operating time of the algorithm for finding the door handle is 315 ms for the robot and 254 ms in the simulation. The average running time of the collision-free trajectory construction algorithm for the robot is 8.5 s.

In Figure 10, the robot executes the door-opening trajectory for a test door for which accuracy has been estimated. The video [41] shows a short demonstration of this application.

Thus, as shown in the numerical experiments, the algorithm for detecting door handle wear from RGBD stereo camera data significantly increases the number of successful attempts. It has also been shown in the simulation that the average error for 10,000 test cases is 1.98 mm. That is, on average, the distance between the actual position of the door handle and the detected one is 1.98 mm.

Figure 10. Robot executes trajectory in the test room.

4. Conclusions

In this research, we propose a universal and computationally efficient algorithm for building a trajectory with an autonomous mobile robot with a seven-stage robotic arm, an RGBD camera, and four LiDAR sensors responsible for localization. The versatility of the algorithm lies in the fact that it works on many types of doors encountered in everyday life. The algorithm's effectiveness is that it has relatively low computational complexity and can be run on computers such as Nvidia Jetson or Raspberry Pi.

In this research, we developed a software product for conducting numerical experiments. As a result, the proposed algorithm was tested in two ways: in simulation and on a real robot. Simulation results have demonstrated that the algorithm is valid for different combinations of environmental parameters. In contrast, testing on a real robot told us whether the result in the simulation correlates with the natural world for several exceptional cases of environmental parameters. Additionally, the algorithm was tested on a real robot in two variations: with and without ML. As a result, using the proposed algorithm allowed us to obtain 95% successful attempts, which is 65% more compared to the second method (without using ANN). The percentage of successful door openings out of the total number of attempts was used as an accuracy metric. As for the resulting error, in the simulation, the average error for 10,000 test cases is 1.98 mm.

A promising avenue for future research in this area is optimizing the proposed algorithm by parallelizing the latter based on a modern parallel computing technology such as OpenMP [42], or using GPUs and CUDA technology [43].

Author Contributions: Conceptualization, L.M. and Y.H.; methodology, L.M. and Y.H.; software, Y.Z. and Y.H.; validation, L.M. and Y.Z.; formal analysis, M.G.; investigation, L.M.; resources, Y.H.; data curation, Y.H.; writing—original draft preparation, L.M.; writing—review and editing L.M. and Y.H.; visualization, M.G.; supervision, L.M.; 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: All data are fully available without restriction. They may be found at https://github.com/MiguelARD/DoorDetect-Dataset (accessed on 2 April 2023).

Acknowledgments: The authors would like to thank the Armed Forces of Ukraine for providing security to perform this work. This work has become possible only because of the resilience and courage of the Ukrainian Army.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Alatise, M.B.; Hancke, G.P. A Review on Challenges of Autonomous Mobile Robot and Sensor Fusion Methods. *IEEE Access* **2020**, *8*, 39830–39846. [CrossRef]
2. Joon, A.; Kowalczyk, W. Design of Autonomous Mobile Robot for Cleaning in the Environment with Obstacles. *Appl. Sci.* **2021**, *11*, 8076. [CrossRef]
3. Sun, Y.; Guan, L.; Chang, Z.; Li, C.; Gao, Y. Design of a Low-Cost Indoor Navigation System for Food Delivery Robot Based on Multi-Sensor Information Fusion. *Sensors* **2019**, *19*, 4980. [CrossRef]
4. Bogue, R. Domestic robots: Has their time finally come? *Ind. Robot* **2017**, *44*, 129–136. [CrossRef]
5. Palacín, J.; Rubies, E.; Clotet, E.; Martínez, D. Evaluation of the Path-Tracking Accuracy of a Three-Wheeled Omnidirectional Mobile Robot Designed as a Personal Assistant. *Sensors* **2021**, *21*, 7216. [CrossRef] [PubMed]
6. Mochurad, L.; Kryvinska, N. Parallelization of Finding the Current Coordinates of the Lidar Based on the Genetic Algorithm and OpenMP Technology. *Symmetry* **2021**, *13*, 666. [CrossRef]
7. Elfring, J.; Torta, E.; van de Molengraft, R. Particle Filters: A Hands-On Tutorial. *Sensors* **2021**, *21*, 438. [CrossRef]
8. Urrea, C.; Agramonte, R. Kalman Filter: Historical Overview and Review of Its Use in Robotics 60 Years after Its Creation. *J. Sens.* **2021**, *2021*, 21. [CrossRef]
9. Mochurad, L.; Panto, R. A Parallel Algorithm for the Detection of Eye Disease. In *Advances in Intelligent Systems, Computer Science and Digital Economics IV*; CSDEIS 2022; Springer Nature: Cham, Switzerland, 2023; Volume 158, pp. 111–125. [CrossRef]
10. Izonin, I.; Tkachenko, R.; Shakhovska, N.; Lotoshynska, N. The Additive Input-Doubling Method Based on the SVR with Nonlinear Kernels: Small Data Approach. *Symmetry* **2021**, *13*, 612. [CrossRef]
11. Izonin, I.; Tkachenko, R.; Dronyuk, I.; Tkachenko, P.; Gregus, M.; Rashkevych, M. Predictive modeling based on small data in clinical medicine: RBF-based additive input-doubling method. *Math. Biosci. Eng.* **2021**, *18*, 2599–2613. [CrossRef] [PubMed]
12. Wang, Y.; Wang, L.; Zhao, Y. Research on Door Opening Operation of Mobile Robotic Arm Based on Reinforcement Learning. *Appl. Sci.* **2022**, *12*, 5204. [CrossRef]
13. Palacín, J.; Rubies, E.; Bitrià, R.; Clotet, E. Non-Parametric Calibration of the Inverse Kinematic Matrix of a Three-Wheeled Omnidirectional Mobile Robot Based on Genetic Algorithms. *Appl. Sci.* **2023**, *13*, 1053. [CrossRef]
14. Zhu, X.; Lyu, S.; Wang, X.; Zhao, Q. TPH-YOLOv5: Improved YOLOv5 Based on Transformer Prediction Head for Object Detection on Drone-captured Scenarios. *arXiv* **2021**. [CrossRef]
15. Ahmed, M.A.; Baharin, H.; Nohuddin, P.N.E. Analysis of K-means, DBSCAN and OPTICS Cluster Algorithms on Al-Quran Verses. *Int. J. Adv. Comput. Sci. Appl.* **2020**, *11*, 248–254. [CrossRef]
16. Riu, C.; Nozick, V.; Monasse, P. Automatic RANSAC by Likelihood Maximization. *Image Process. Line* **2022**, *12*, 27–49. [CrossRef]
17. Malayjerdi, E.; Kalani, H.; Malayjerdi, M. Self-Tuning Fuzzy PID Control of a Four-Mecanum Wheel Omni-Directional Mobile Platform. In Proceedings of the Electrical Engineering (ICEE), Iranian Conference, Mashhad, Iran, 8–10 May 2018; pp. 816–820.
18. Meng, Q.; Liu, T. Study on Immune PID Control Method of an In-Wheel Motor Used in an Electric Car. In Proceedings of the 2017 36th Chinese Control Conference (CCC), Dalian, China, 26–27 July 2017; pp. 9554–9559.
19. Karpińska, J.; Tchoń, K.; Janiak, M. Approximation of Jacobian Inverse Kinematics Algorithms: Differential Geometric vs. Variational Approach. *J. Intell. Robot Syst.* **2012**, *68*, 211–224. [CrossRef]
20. Lin, P.-F.; Huang, M.-B.; Huang, H.-P. Analytical Solution for Inverse Kinematics Using Dual Quaternions. *IEEE Access* **2019**, *7*, 166190–166202. [CrossRef]
21. Vlassis, N.; Terwijn, B.; Krose, B. Auxiliary Particle Filter Robot Localization from High-Dimensional Sensor Observations. In Proceedings of the 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292), Washington, DC, USA, 11–15 May 2002; Volume 1, pp. 7–12.
22. Zhang, Q.; Wang, P.; Chen, Z. An Improved Particle Filter for Mobile Robot Localization Based on Particle Swarm Optimization. *Expert Syst. Appl.* **2019**, *135*, 181–193. [CrossRef]
23. New Dog-like Robot from Boston Dynamics Can Open Doors. Available online: https://www.youtube.com/watch?v=wXxrmussq4E (accessed on 2 April 2023).
24. Arduengo, M.; Torras, C.; Sentis, L. Robust and adaptive door operation with a mobile robot. *Intel. Serv. Robot.* **2021**, *14*, 409–425. [CrossRef]
25. Lu, Z.; Chauhan, A.; Silva, F.; Lopes, L.S. A Brief Survey of Commercial Robotic Arms for Research on Manipulation. In Proceedings of the 2012 IEEE Symposium on Robotics and Applications (ISRA), Kuala Lumpur, Malaysia, 3–5 June 2012; pp. 986–991.
26. Wang, Y.-T.; Peng, C.-C.; Ravankar, A.; Ravankar, A. A Single LiDAR-Based Feature Fusion Indoor Localization Algorithm. *Sensors* **2018**, *18*, 1294. [CrossRef]

27. Khan, A.; Javed, A.; Irtaza, A.; Mahmood, M.T. A Robust Approach for Blur and Sharp Regions' Detection Using Multisequential Deviated Patterns. *Int. J. Opt.* **2021**, *2021*, 2785225. [CrossRef]
28. Kokare, M.; Chatterji, B.N.; Biswas, P.K. Comparison of Similarity Metrics for Texture Image Retrieval. In Proceedings of the TENCON 2003. Conference on Convergent Technologies for Asia-Pacific Region, Bangalore, India, 15–17 October 2003; pp. 571–575.
29. Arimoto, S. Learning Control Theory for Robotic Motion. *Int. J. Adapt. Control Signal Process.* **1990**, *4*, 543–564. [CrossRef]
30. Dogan, K.M.; Tatlicioglu, E.; Zergeroglu, E.; Cetin, K. Learning Control of Robot Manipulators in Task Space: Learning Control of Robot Manipulators in Task Space. *Asian J. Control* **2018**, *20*, 1003–1013. [CrossRef]
31. Raguram, R.; Frahm, J.-M.; Pollefeys, M. A Comparative Analysis of RANSAC Techniques Leading to Adaptive Real-Time Random Sample Consensus. In *Computer Vision—ECCV 2008*; Forsyth, D., Torr, P., Zisserman, A., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2008; Volume 5303, pp. 500–513. ISBN 978-3-540-88685-3.
32. Sun, L.; Hu, G.; Chen, C.; Cai, H.; Li, C.; Zhang, S.; Chen, J. Lightweight Apple Detection in Complex Orchards Using YOLOV5-PRE. *Horticulturae* **2022**, *8*, 1169. [CrossRef]
33. Kuznetsova, A.; Maleva, T.; Soloviev, V. Detecting Apples in Orchards Using YOLOv3. In *Computational Science and Its Applications—ICCSA 2020—20th International Conference, Cagliari, Italy, 1–4 July 2020, Proceedings, Part I*; Gervasi, O., Murgante, B., Misra, S., Garau, C., Blečić, I., Taniar, D., Apduhan, B.O., Rocha, A.M.A.C., Tarantino, E., Torre, C.M., et al., Eds.; Lecture Notes in Computer Science; Springer International Publishing: Cham, Switzerland, 2020; Volume 12249, pp. 923–934. ISBN 978-3-030-58798-7.
34. Ahmed, M.; Seraj, R.; Islam, S.M.S. The K-Means Algorithm: A Comprehensive Survey and Performance Evaluation. *Electronics* **2020**, *9*, 1295. [CrossRef]
35. Kamil, I.S.; Al-Mamory, S.O. Enhancement of OPTICS' Time Complexity by Using Fuzzy Clusters. *Mater. Today Proc.* **2021**. [CrossRef]
36. Tang, C.; Wang, H.; Wang, Z.; Zeng, X.; Yan, H.; Xiao, Y. An Improved OPTICS Clustering Algorithm for Discovering Clusters with Uneven Densities. *IDA* **2021**, *25*, 1453–1471. [CrossRef]
37. Do, C.B.; Batzoglou, S. What Is the Expectation Maximization Algorithm? *Nat. Biotechnol.* **2008**, *26*, 897–899. [CrossRef]
38. Moon, T.K. The Expectation-Maximization Algorithm. *IEEE Signal Process. Mag.* **1996**, *13*, 47–60. [CrossRef]
39. Zhang, F.; Hao, M.; Liu, M.; Yang, J. Localize Car Door Handles with Image Segmentation and Saliency Detection. In Proceedings of the 2017 IEEE International Conference on Imaging Systems and Techniques (IST), Beijing, China, 18–20 October 2017; pp. 1–6.
40. Cui, H.; Liao, W.; Cheng, X.; Dai, N.; Guo, C. Flexible Point Cloud Matching Method Based on Three-Dimensional Image Feature Points. *Adv. Mech. Eng.* **2018**, *10*. [CrossRef]
41. Demonstrative the Robot Executes Door Opening Trajectory. Available online: https://www.youtube.com/shorts/fC2-NDin6Tc (accessed on 2 April 2023).
42. Mochurad, L. Optimization of Regression Analysis by Conducting Parallel Calculations. In Proceedings of the COLINS-2021: 5th International Conference on Computational Linguistics and Intelligent Systems, Kharkiv, Ukraine, 22–23 April 2021; pp. 982–996.
43. Mochurad, L.I. Canny Edge Detection Analysis Based on Parallel Algorithm, Constructed Complexity Scale and CUDA. *Comput. Inf.* **2022**, *41*, 957–980. [CrossRef]