


## Article

# Painting Path Planning for a Painting Robot with a RealSense Depth Sensor

Vladimir Tadic <sup>1,\*</sup>, Akos Odry <sup>1</sup> , Ervin Burkus <sup>1</sup>, Istvan Kecskes <sup>1</sup>, Zoltan Kiraly <sup>1</sup>, Mihaly Klincsik <sup>2</sup>, Zoltan Sari <sup>2</sup>, Zoltan Vizvari <sup>3</sup>, Attila Toth <sup>4</sup> and Peter Odry <sup>1</sup>

<sup>1</sup> Institute of Information Technology, University of Dunaujvaros, Tancsics Mihaly u. 1 / A Pf.: 152, 2401 Dunaujvaros, Hungary; odrya@uniduna.hu (A.O.); ervinbur@appl-dsp.com (E.B.); kecskes.istvan@gmail.com (I.K.); kiru@uniduna.hu (Z.K.); podry@uniduna.hu (P.O.)

<sup>2</sup> Department of Technical Informatics, Faculty of Engineering and Information Technology, University of Pecs, Boszorkany str. 2, H-7624 Pecs, Hungary; klincsik@mik.pte.hu (M.K.); sari.zoltan@mik.pte.hu (Z.S.)

<sup>3</sup> Department of Environmental Engineering, Faculty of Engineering and Information Technology, University of Pecs, Boszorkany str. 2, H-7624 Pecs, Hungary; vizvari.zoltan@mik.pte.hu

<sup>4</sup> Institute of Physiology, Medical School, University of Pecs, Szigeti str 12, H-7624 Pecs, Hungary; attila.toth@aok.pte.hu

\* Correspondence: laslo.tadic@gmail.com

**Abstract:** The utilization of stereo cameras in robotic applications is presented in this paper. The use of a stereo depth sensor is a principal step in robotics applications, since it is the first step in sequences of robotic actions where the intent is to detect and extract windows and obstacles that are not meant to be painted from the surrounding wall. A RealSense D435 stereo camera was used for surface recording via a real-time, appearance-based (RTAB) mapping procedure, as well as to navigate the painting robot. Later, wall detection and the obstacle avoidance processes were performed using statistical filtering and a random sample consensus model (RANSAC) algorithm.

**Keywords:** depth image; RTAB mapping; statistical filter; RANSAC; obstacle avoidance



**Citation:** Tadic, V.; Odry, A.; Burkus, E.; Kecskes, I.; Kiraly, Z.; Klincsik, M.; Sari, Z.; Vizvari, Z.; Toth, A.; Odry, P. Painting Path Planning for a Painting Robot with a RealSense Depth Sensor. *Appl. Sci.* **2021**, *11*, 1467. <https://doi.org/10.3390/app11041467>

Academic Editor: Oscar Reinoso Garcia

Received: 8 January 2021

Accepted: 1 February 2021

Published: 5 February 2021

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2021 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

This paper presents a specific robotic application based on the processing of depth images captured using a low-cost Intel RealSense D435 stereo camera [1–14]. This work is part of an industrial-research project aiming to build a robot that would automatically—based on data obtained from the depth camera—be able to paint a building's facades or interior walls with wall paint. The goal was to develop a robust and simple computer vision algorithm to detect and extract windows and coarse obstacles on walls using depth images recorded with a stereo camera, which is also able to notify the control system [15–22]. The acquired depth images and point clouds of the environment serve as resources for the robot to determine which surfaces to paint.

This paper introduces a procedure for window and obstacle detection and for extracting these objects from walls using captured depth images.

This project is a part of the KFI\_16-1-2017-00485 project, which involves the development of a robot for painting and thermal insulation of facades of monument buildings. The client's instruction and the main goal of the project is to use already proven and reliable algorithms to ensure smooth and reliable operation of a painting robot. Almost all wall painting robots on the market are not automated, meaning the operator fully controls the painting arm and decides what should be painted. In this project, the client requires automation using a computer vision system. This approach is new in the design of commercial wall painting robots. The basic hardware in the vision system is the Intel RealSense depth camera.

The wall extraction algorithm is based on real-time, appearance-based mapping (RTAB-Map) [23–27] and a random sample consensus model (RANSAC) [28,29]. The

input point cloud is formed via an RTAB-Map algorithm using zig-zag robot movements along with a built-in RealSense depth camera by merging recorded point clouds during movement [30–32]. Further, before extracting the RANSAC algorithm, the initial step is to apply a statistical filter [30] to remove most of the outliers from the captured point clouds. Finally, the RANSAC algorithm combined with the clustering procedure is performed for the wall extraction process. RANSAC is a powerful iterative method used to assess the parameters of a mathematical model from a set of data containing outliers.

## 2. Related Work

Plane detection and extraction is a common problem in robotic vision. This operation can be crucial in certain applications where it is necessary to separate certain objects from the background. There are various ways to make this distinction, but one of the most common methods is to use the RANSAC algorithm [28,29].

Carfagni et al. [1] compared devices through four types of experiments: with the error measured using a calibrated sphere at a very close distance; with the errors measured with the Association of German Engineers (VDI)/Association for Electrical, Electronic, and Information Technologies (VDE) 2634 part 2 standard; with the systematic depth errors produced through the acquisition of a planar surface at increasing ranges; and with the 3D reconstruction of an object. Experiments have shown that the RealSense D415 model is fully comparable to its predecessor model in terms of errors assessed through the VDI/VDE standard. The D415 camera is also in line with results obtained from other cameras in the scientific literature and surpasses their performance when considering filtered data. El-Sayed et al. [32] proposed a new method for plane detection from 3D point clouds. The method is contingent on two concepts to achieve a balance between high-accuracy and fast performance. The first concept is to utilize a fast new octree-based, balanced density down-sampling technique to reduce the number of points. The next concept is the fact that the number of planes in any dataset is much lower than the number of the points. Random points are inspected to find the 3D planes. It has been demonstrated experimentally that the suggested algorithm yields precise results and has expeditious performance, is robust to noise, and has a high potential to detect planes with small angle variations.

Gallo et al. [33] introduced a new algorithm, the connected components RANSAC (CC-RANSAC), which uses only the largest connected components of inliers to estimate the suitability of a candidate plane. They stated that this allegedly minor modification could yield appreciably better fits than RANSAC. Mufti et al. [34] considered the task of ground plane estimation in range image data acquired from a time-of-flight camera. They extended the 3D spatial RANSAC for ground plane estimation to a 4D spatiotemporal RANSAC by including a time axis. The ground plane models were obtained from spatiotemporal random data points, thereby making the proposed method robust against short-term transitory effects such as passing cars and pedestrians. They noted that the computationally fast and robust assessment of ground planes led to the reliable identification of obstacles and pedestrians using statistically obtained spatial thresholds. Nurunnabi et al. [35] introduced two robust statistical methods for outlier detection and robust saliency properties, such as surface normal and curvature assessments in laser scanning 3D point cloud data. The first was formed using a robust z-score, while the second technique uses a Mahalanobis-type robust distance. They stated that the algorithms are fast, accurate, and robust.

Li et al. [36] presented an ameliorative RANSAC method, relying on normal distribution transformation (NDT) cells in their research to avoid false planes for 3D point cloud plane segmentation. They selected a planar NDT cell as a minimal sample in each iteration to ensure the correctness of sampling on the same plane surface. Basically, the 3D NDT presents the point cloud with a set of NDT cells and models the observed data with a normal distribution within each cell. The geometric representations of NDT cells were used to separate them into planar and non-planar cells. The experiments showed that the correctness exceeded 88.5% and the completeness exceeded 85.0%. These results indicate that the introduced method is more dependable and accurate than the standard RANSAC.

Li et al. [37] proposed an accurate plane-fitting procedure for a structured light-based red–green–blue–depth (RGB-D) sensor. In the first step, they obtained an error model of point cloud data from the structured light-based RGB-D camera through error propagation from the raw measurement to the point coordinates. Then, the new cost function based on minimizing the radial distances with the obtained rigorous error model was proposed for the RANSAC-based plane fitting procedure. Schwarze and Lauer [38] presented two approaches to assess the local geometric structures of urban street canyons captured from a head-mounted depth camera. The dense disparity assessment was the only input used in both approaches. The first approach showed how left and right building facades were obtained using planar segmentation based on RANSAC. In the other approach, they transformed the disparity into an elevation map, from which they extracted the main building orientation. Xu and Lu [39] presented a new type of RANSAC, named the distributed RANSAC (D-RANSAC), to reduce the computation time and increase accuracy. The authors compared their outcomes with the classical RANSAC and randomized RANSAC (R-RANSAC) results. The tests showed that D-RANSAC was superior to RANSAC and R-RANSAC in computational complexity and accuracy in most cases, primarily when the inlier proportion was below 65%. Xu et al. [40] developed a new weighted RANSAC method for roof point cloud segmentation, whereby the hard threshold voting function using both the point-plane distance and the normal vector consistence was converted into a soft threshold voting function based on two weight functions. Zhou et al. [41] introduced a handheld 3D scanning device developed alongside light detection and ranging (LiDAR), inertial measurement units, and other auxiliary equipment, based on which a 3D map of a forest environment was generated. The ground point cloud was removed using the RANSAC algorithm. The trees in the experimental area were segmented using a Euclidian clustering algorithm. They noted that the experimental results were good and met the requirements for forestry mapping. Deschaud and Goulette [42] proposed a rapid and accurate method to detect planes in unorganized point clouds utilizing filtered normal and voxel growth. They showed the efficiency of the method with different kinds of data, as well as its speed on large data sets.

### 3. Industrial Robots and Depth Cameras

Industrial robots are multifunctional, reprogrammable, manipulative, and automatically controlled machines with multiple degrees of freedom (DOFs), which can be fixed or mobile, as well as used for automated industrial applications. Industrial robots are designed to work over a long period of time in challenging working conditions in industrial environments. Robots are very complex devices that involve automatic control theory, machine theory, computer engineering, artificial intelligence, and sensor technology.

People now view robots as machines that allow for further and more flexible automation. Robots substitute humans primarily in hazardous, monotonous, and difficult jobs. As a result, robotic systems contribute concurrently to increasing productivity and humanizing work.

In the application introduced in this paper, the paint spraying unit was moved by a 5 DOF robotic arm. The first three joints (base, shoulder, and elbow) are responsible for moving the end effector to the desired position, while the last two joints (wrist pitch and yaw) are responsible for correcting the orientation. Regarding the orientation, the requirement is that the axis of the industrial spray gun has to be perpendicular to the wall, hence using a robot with 5 degrees of freedom was sufficient.

In the first three joints, servo motors and harmonic drives were implemented. The robot's wrist uses a specially designed differential drive driven by two servo motors equipped with planetary gearboxes. The torques of the motors and the lengths of the segments were determined based on previously performed dynamic simulations. The first three segments are made of aluminum alloy. The parts of the differential joint are made of ABS plastic using 3D printing. The robot's reach is 1820 mm, and its total weight is about

50 kg. During the tests, the robot was mounted on a wheeled structure, which simulated the movement of the crane.

The behavior of the robot was tested first in the Gazebo open-source dynamics simulator. This simulator uses physics engines, and therefore it enabled us to observe the dynamics of the designed robot arm. The Gazebo environment spawned the robot model based on the Unified Robot Description Format (URDF) file, which contained the inertia and mass properties of the robot. Moreover, this file contained the limits of joints and the actuators that drove the robot joints. The joints were controlled in closed loop with proportional–integral–derivative (PID) position controllers. The *ros\_controllers* meta-package was employed to implement these controllers. Based on iterative tuning, the PID parameters ( $K_p$ ,  $K_i$ , and  $K_d$ ) were set up heuristically in Gazebo. The true states of the robot model were sampled using the joint state controller. The RGB-D measurements were provided by the Gazebo Depth Camera Plugin [43], which supplied the instantaneous point cloud measurements. Similarly to the real robot, this Gazebo depth camera was attached to the end effector of the robot model, therefore an identical structure was achieved in the simulation environment. Based on the simulated robot behavior and measurements, the processing of camera data was achieved with our software and the functionality of the whole painting process was validated.

Lately, image processing applications based on two-dimensional (2D) image processing have been extensively constrained due to a lack of information in the third dimension, i.e., depth. Unlike 2D computer vision, three-dimensional (3D) computer vision makes it feasible for computers and other machines to distinguish different shapes, shape dimensions, distances, and control accurately and with high precision in the real 3D world [1–4].

In 2014, Intel presented a family of depth sensors, i.e., Intel RealSense cameras. These systems are composed of a microprocessor for image processing, a component for forming depth images, a component for tracking movements, and depth cameras. These stereo depth cameras rely on deep scanning technology, which allows computers to see objects in the same way as humans.

RealSense cameras use stereovision to determinate depth information. A stereovision system consists of left- and right-side cameras, i.e., sensors, and an infrared (IR) projector. The IR projector projects invisible infrared rays that increase the accuracy of the depth information in scenes with poor texture. The left-side and right-side cameras capture the scene and send information about the image to the signal processor, which based on the received information, calculates the depth values for each pixel of the depth image, thereby correlating the points received with the left-side camera to the image received with the right-side camera [1–14]. The value of a depth pixel depicting the depth of an object is calculated in relation to a parallel plane from the sensor doing the recording and not in relation to the actual range of the object from the camera [7].

It should be noted that the RealSense depth cameras are already factory pre-calibrated and tuned for optimal use by the manufacturer, and that calibrating these cameras is not recommended. Calibration is necessary only if there are serious problems during the capture process [7]. Further, for development and testing, the open-source RealSense official (ROS) software package was used. This software contains all procedures and algorithms needed for depth image recording and embedding in a robot's operating system [7–10].

The performance and affordable cost are the advantages that make Intel RealSense cameras a popular choice in many devices that require depth cameras with high-speed data processing [1–14].

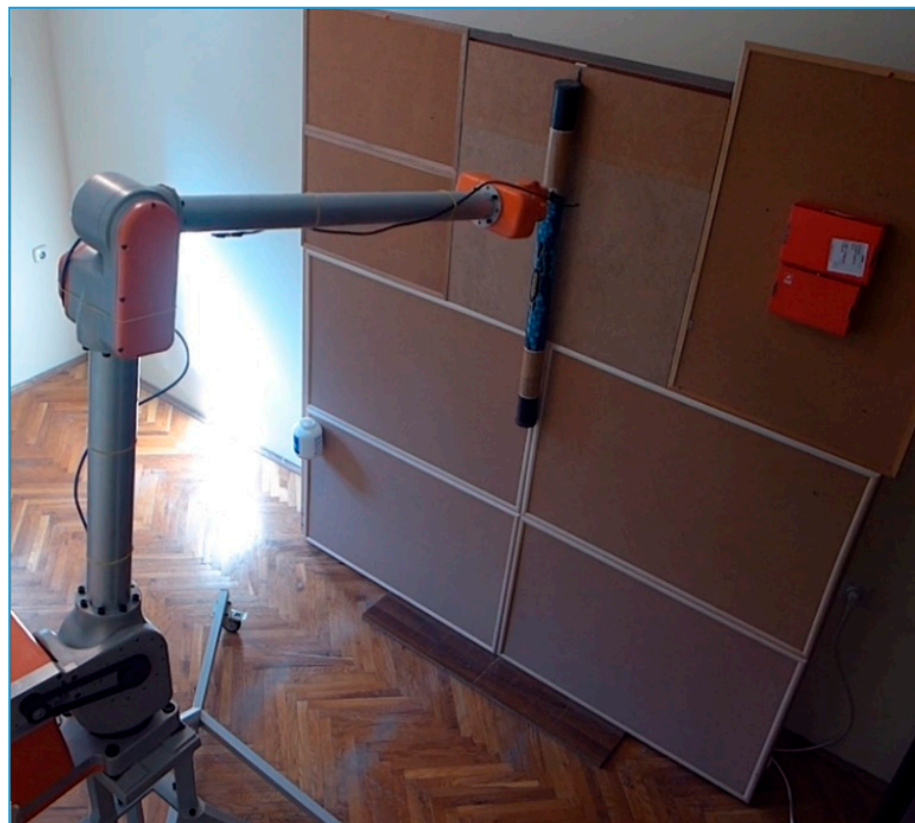
As will be discussed in the next sections, a depth cameras will be used for depth estimation of the obstacles on a wall. The depth camera detects objects at different distances and separates them. This information is contained by Z coordinate values. Later, the wall will be separated with the segmentation algorithm based on the depth difference from the obstacles and the background. Hence, the separation of windows from walls will be achieved using depth measurement and visualization.

#### 4. Wall Extraction Algorithm

The task related to the painting robot and the depth camera can be divided into the following steps:

1. Capturing images for part of the building;
2. Detecting the window and obstacles in the wall that must not be painted;
3. Forwarding the information about the coordinates of the wall to the robot for painting;
4. Painting the wall by bypassing the detected window and the obstacles.

The corresponding painting robot is shown in Figure 1. Since the process of developing and testing the painting equipment is still in its building phase, the experiments were operated with the embedded robot operating system (ROS) simulator.

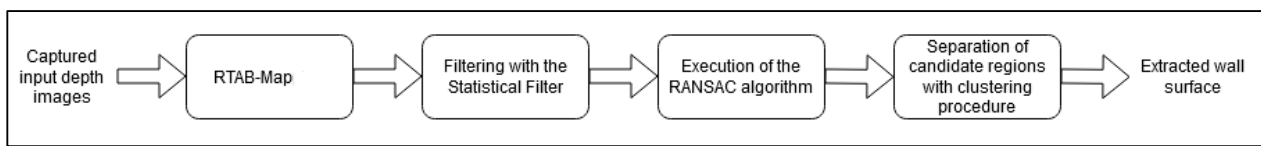


**Figure 1.** The painting robot in the testing process.

In the simulation phase, a laser pointer was placed in the robot's tool center point. Later, a spray gun was attached here using a tool holder.

Further, because in robotics a high degree of accuracy and precision is required, the depth cameras needed to be calibrated in the appropriate manner, so that the obstacle was clearly visible and detected in a complex image. There were numerous parameters and conditions that affected the quality of the depth image, such as noise and light [1–14]. Finally, when the camera and the robot parameters were tuned appropriately, the experiments began.

Next, the principles and methods for the four main steps of the wall extraction procedure are briefly described. The first sub-section introduces the RTAB-Map, the second describes the statistical filter, the third sub-section depicts the RANSAC algorithm, and the fourth sub-section describes the segmentation algorithm with the corresponding clustering procedure. Figure 2 shows a block diagram of the wall extraction algorithm.



**Figure 2.** A block diagram of the wall extraction algorithm.

#### 4.1. Real-Time Appearance-Based Mapping (RTAB-Map)

RTAB-Map is an open-source library implementing loop closure detection with a memory management approach [23–27]. This method limits the size of the map so that loop closure detection is always performed under a fixed time limit, hence satisfying online requirements for long-term and large-scale environment mapping [23]. RTAB-Map includes a complete graph-based simultaneous localization and mapping (SLAM) approach, and it is, therefore, possible to use it in various setups and applications [23]. RTAB-Map has the following options [23–27]:

- Online processing;
- Robust and low-drift odometry;
- Robust localization;
- Practical map generation and exploitation;
- Multisession mapping;
- Integration in ROS.

RTAB-Map can accept a variety of inputs, such as depth color images (RGB-D), stereo images, odometry, 2D laser scans, 3D point clouds, and other user data. All input data can be used, depending on the available sensors. Further, RTAB-Map supports the use of multiple RGB-D cameras, as long as they have the same image size [23]. The loop closure detection method is implemented using the bag-of-words approach described by Labbé and Michaud [27]. Proximity detection is used to localize nodes close to the current position with laser scans when they are available [24]. Using proximity detection, the localization can be done when the robot traverses back through the same corridor in a different direction, during which the camera cannot be used to find loop closures. After loop closure or proximity detection are performed, or when certain nodes are retrieved or transferred because of memory management, a graph optimization approach is executed to minimize errors in the map [23]. If loop closure occurs, the global map should be reassembled according to all new optimized positions for nodes in the map’s graph [23–27]. This process is necessary so that obstacles that were wrongly cleared before the loop closure process may later be reincluded. The point cloud outputs are composed of all points on the local maps, presented in the standard ROS format. Finally, voxel grid filtering can be applied to merge overlapping surfaces in point clouds. The resulted merged point cloud is then suitable for further processing [23].

#### 4.2. Statistical Filter

The first operation for the given point cloud before the wall extraction step was to apply a statistical filter to remove unnecessary, noisy gross outlier data. Namely, stereo cameras generated point clouds of varying point densities and the resulting point clouds had sparse outliers that corrupted the results. This complicated the estimation of local point cloud characteristics, such as surface normal or curvature changes, leading to incorrect values, which in turn could cause point cloud processing failures.

The applied filter used point neighborhood statistics to filter outlier data [30]. The sparse outlier removal step corrected these irregularities by computing the mean and standard deviation of nearest  $k$  neighbor distances and pruning the points that fell outside the calculated threshold  $t$ :

$$t = \mu \pm \alpha\sigma \quad (1)$$

where  $\mu$  is the mean,  $\sigma$  is the standard deviation, and  $\alpha$  is the multiplier. The value of  $\alpha$  depends on the size of the analyzed neighborhood [30]. The values of  $k$  and  $\alpha$  are set experimentally. For instance, if 1% of point cloud points are considered as noise outliers, one example would be  $\alpha = 1$  and  $k = 30$ , because experiments with multiple datasets have confirmed the applicability of the  $\mu \pm \sigma$  threshold as a good practice [30].

Basically, the filtering algorithm iterates through the entire input point cloud twice. During the first iteration, it computes the average distance that each point has to its nearest  $k$  neighbors [29]. Further, the mean and standard deviation of all these distances are computed in order to determine a distance threshold. During the second iteration, the points can be classified as either inliers or outliers if their average neighbor distances are below or above the threshold, respectively [30].

#### 4.3. Random Sample Consensus Model: RANSAC Algorithm

Since the statistical filter made huge quality improvements [30], the next step was the obstacle and window detection, which resulted in a wall extraction being painted by the robot.

Since the wall mainly differed in depth from the obstacles and the window, it was essentially located in a separate parallel plane from the obstacles and the window. Therefore, the goal was to separate the wall plane from the other planes that were parallel to the wall plane in point cloud data. The most popular planar segmentation algorithms are based on the RANSAC algorithm.

RANSAC is a general parameter estimation approach designed to cope with a large proportion of outliers in input data. This is a resampling technique that generates candidate solutions by using the minimum number of observations and data points required to estimate underlying model parameters. Unlike conventional sampling techniques that use as much of the data as possible to obtain an initial solution and then proceed to prune outliers, RANSAC uses the smallest set possible and proceeds to enlarge this set with consistent data points [28,29].

The RANSAC algorithm process can be described as follows:

1. Randomly select the minimum number of points required to determine the model parameters, i.e., all free parameters of the model reconstructed from the inliers;
2. Test all data for the parameters of the determined model;
3. Determine how many points from the set fit with a predefined tolerance (this is defined through the probability);
4. If the fraction of the number of inliers over the total number of points in the set exceeds a predefined threshold  $T$ , re-estimate the model parameters using the identified inliers and terminate the algorithm;
5. Otherwise, repeat steps 1–4 a maximum of  $N$  times [28,29].

The number of iterations  $N$  should be high enough to ensure that the probability  $p$  at least has one set of random samples that does not include an outlier [28,29]. The value for the probability is usually set to 0.99, but this is not mandatory. Let  $a$  represent the probability that any selected datum point is an inlier and  $b = 1 - a$  the probability of observing an outlier. Then, the  $N$  iterations of the minimum number of points denoted as  $m$  are required, where [28,29]:

$$1 - p = (1 - a^m)^N. \quad (2)$$

After the reorganization of the previous expression, one can obtain the following [28,29]:

$$N = \frac{\log(1 - p)}{\log(1 - (1 - v)^m)} \quad (3)$$

The steps of the algorithm are repeated a fixed number of times, each time producing either a model that is rejected because too few points are classified as inliers or a refined model together with a corresponding error measure. In the latter case, the refined model should be kept if its error is lower than the last saved model [28].

An advantage of RANSAC is its ability to perform robust estimation of the model parameters, because it can estimate the parameters with a high degree of accuracy even when a significant number of outliers are present in the data set. A disadvantage of RANSAC is that there is no upper limit on the time it takes to compute the model parameters. When the number of iterations computed is limited, the solution obtained may not be optimal and it may not fit the data well. In this way, RANSAC offers a trade-off; by computing a greater number of iterations, the probability of a reasonable model being produced is increased. Another disadvantage of RANSAC is that it requires the setting of problem-specific thresholds. In addition, RANSAC can only estimate one model for a particular data set. For any single-model approach, when two or more models exist, RANSAC may fail to find either one. This problem can be solved by adding particular conditions; for example, if the largest object of the given geometry is sought in the point cloud data, the assumption is that the wall represents the largest plane in the image. This optimization will overcome the problem related to estimating incorrect model parameters [28,29].

In this project, the plane detection was based on 3D coordinates. This means that  $N$  sets of a point triplets were formed by random distribution within the point cloud and that for each of the triplets a corresponding plane was defined based on geometric equations. In the second step, the quality of each of these  $N$  models was evaluated based on the whole sample. In this case, for each pixel in the image, we estimated whether it belonged to a defined plane or not. Finally, the detected plane represented the wall to be painted by the painting robot.

#### 4.4. Detailed Explanation of the Segmentation Algorithm Steps

In this section, an explanation of the steps in the developed segmentation algorithm are summarized. The segmentation algorithm uses a 3D point cloud (with  $X, Y, Z$  dimensions) and performs the following steps:

- (1) Apply a statistical filter [30] on the 3D point cloud to remove outlier points, based on the thresholded standard deviation calculated on neighboring points. The number of neighbors used to analyze for each point is set to 50 and the standard deviation multiplier is set to 3.5. This means that all points with a distance larger than 3.5 standard deviations of the mean distance to the query point will be marked as outliers and removed;
- (2) Perform the plane clustering step using the iterative RANSAC algorithm, with the “distance threshold” set to 0.03 m, which determines how close a point the model must be in order to be considered an inlier. The RANSAC [28,29] algorithm selects the greatest plane from the remaining point cloud in each iteration, where the points of selected planes are removed before the next iteration. Each plane receives a new cluster id or label (using integers). The cycles are stopped when the remaining cloud is smaller than 0.5% of the original or the maximum iteration number of 20 is reached;
- (3) Resample the 3D cluster label cloud to a 2D cluster label with a predefined resolution on the plane of the wall. The output is a  $2 \times 2$  m rectangle in the plane of the wall, gridded equally in both directions ( $GY$  = height by direction  $Y$ ,  $GX$  = width by direction  $X$ ) at 0.01 m, i.e., 1 cm. This produces a  $200 \times 200$  cluster label image ( $C2d$ ) along with the depth in the  $Z$  direction ( $Z2d$ ). The algorithm first collects all available points  $(X, Y, Z)$  belonging to each  $1 \times 1$  cm pixel  $(x, y)$ ; the set of these pixels is denoted  $S_{x,y}$  (4). Secondly, it selects the closest point to its average  $Z$  distance ( $\bar{Z}$ ) from this small set and inherits the cluster label and  $Z$  axis for the selected point (5).

$$S_{x,y} = \left\{ \forall s : \left| \sqrt{X(s) + Y(s)} - \sqrt{GX(x) + GY(y)} \right| \leq 0.01\sqrt{2}/2 \right\} \quad (4)$$

$$C2d(x, y) = C(i^*), Z2d(x, y) = Z(i^*), i^* = \underset{i \in S_{x,y}}{\operatorname{argmin}} \left( \left| Z(i) - \bar{Z}_{S_{x,y}} \right| \right) \quad (5)$$

- (4) Disconnect regions within clusters based on two criteria:



- a. Separate regions based on Euclidean distance clustering with a distance tolerance of 0.08 m [44]. This step uses the cluster method and k-dimensional tree search method to separate clusters [45–48]. K-d trees are very useful for range and nearest neighbor searches, which are the best for separating possible regions. This process also takes into account the depth information (z axis);
  - b. Separate regions based on adjacent criteria in the 2D plane (if the cluster consists of more than one region or when the region is a subset containing only adjacent pixels). The algorithm checks the adjacency of pixels within one cluster, and if more than one is region found, these are separated and labeled differently. This process does not take into account the depth information (z axis);
- (5) Erase small-area clusters that are highly likely to be noise or outliers. The algorithm first calculates the area for each cluster and selects “tiny” clusters smaller than  $0.005 \text{ m}^2$  ( $50 \text{ cm}^2$ ). The pixels of tiny clusters inherit cluster labels from the nearest neighbor pixels belonging to non-tiny clusters. The nearest neighbor criteria are checked on the 2D plane of the wall (X/Y plane) independently from the depth (z axis). This process filters out any remaining bad pixels, i.e., any clusters with only one pixel;
  - (6) Sort the cluster labels by area, whereby each of the remaining clusters get a new sorted label, with the first belonging to the largest area. Typically, the largest cluster is the wall, but this is checked using the expected wall distance (in the Z direction) too. If there is a window, then typically the second biggest area is the plane of the window, the depth of which is greater than the wall distance.

Finally, after finishing the segmentation process, the appropriate clusters containing the information about the wall surface will be specified for the painting operation.

All the procedures described in Sections 4.2–4.4 are included in the open-source Point Cloud Library [49–56].

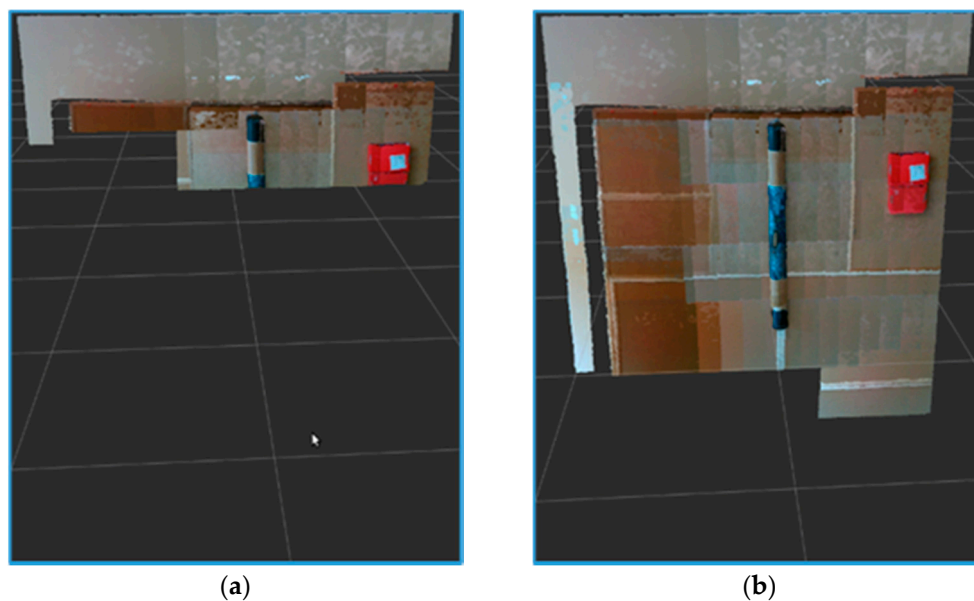
## 5. Experiments and Results

In this part, the experiments and their outcomes are explained in detail. The experiments were conducted with a RealSense D435 depth camera. We chose the D435 because this camera has a wider field of view [7,8,46,47]. This is an important point because the goal is to cover a wide surface during the painting process. The camera and the robot’s tool center point were about 70 cm from the wall. This distance is reported as being the best for the recording the depth via D415 and D435 RealSense cameras in the literature [1,7,8,46,47].

ROS-based robot navigation is the robotic basis for this experiment and will be explained only in brief in the next paragraph, since a robotics explanation is not within the scope of this paper. The communication between the motor drivers and the control software is achieved via controller area network open (CANopen) protocol. A controller area network (CAN) is a robust vehicle bus standard developed to allow devices and microcontrollers to communicate with each other’s applications without a host computer. The communication initialization process is composed of two parts. First, the socketCAN driver is executed, which loads the CAN drivers and networking stacks. Then, the CANopen control package is executed, which establishes the connection between the ROS and motor drivers via the CANopen interface. This package is based on the CiA402 standard (CAN in automation), uses the robot description file (URDF file), and defines the high-level controllers that drive the motors in a closed loop (joint trajectory controller and joint state controller). The motion planning of the robot is executed with the MoveIt software package. This package considers the structure of the robot and generates both the achievable poses and collision configurations. Moreover, MoveIt is used to (i) define the group of joints to be used in the painting process, (ii) save different joint combinations (e.g., home position), and (iii) incorporate the stream of the RGB-D camera in the motion planning process. A custom-made kinematics package was defined for forward and inverse kinematics-based manipulation, which is the base of the aforementioned MoveIt package. The RGB-D camera is started with the official realsense2\_camera ROS package, which loads the driver and

makes the real-time point cloud measurements available in the ROS environment. A static transformation is applied between the camera and robot coordinate systems. The RTAB-Map package is executed to merge the real-time camera stream. This algorithm is executed with default parameters and produces the concatenated point cloud data structure, which describes the environment in front of the robot. Our custom-made graphical user interface is used to control the whole painting process.

The first step was to form a point cloud using the RTAB mapping procedure [23–27]. During the mapping process, the robot arm tracked the surface with the camera and slowly formed the point cloud. Then, voxel grid filtering was applied to merge overlapping surfaces in point clouds [23–27]. Figure 3a,b shows the procedure used to merge the point clouds.

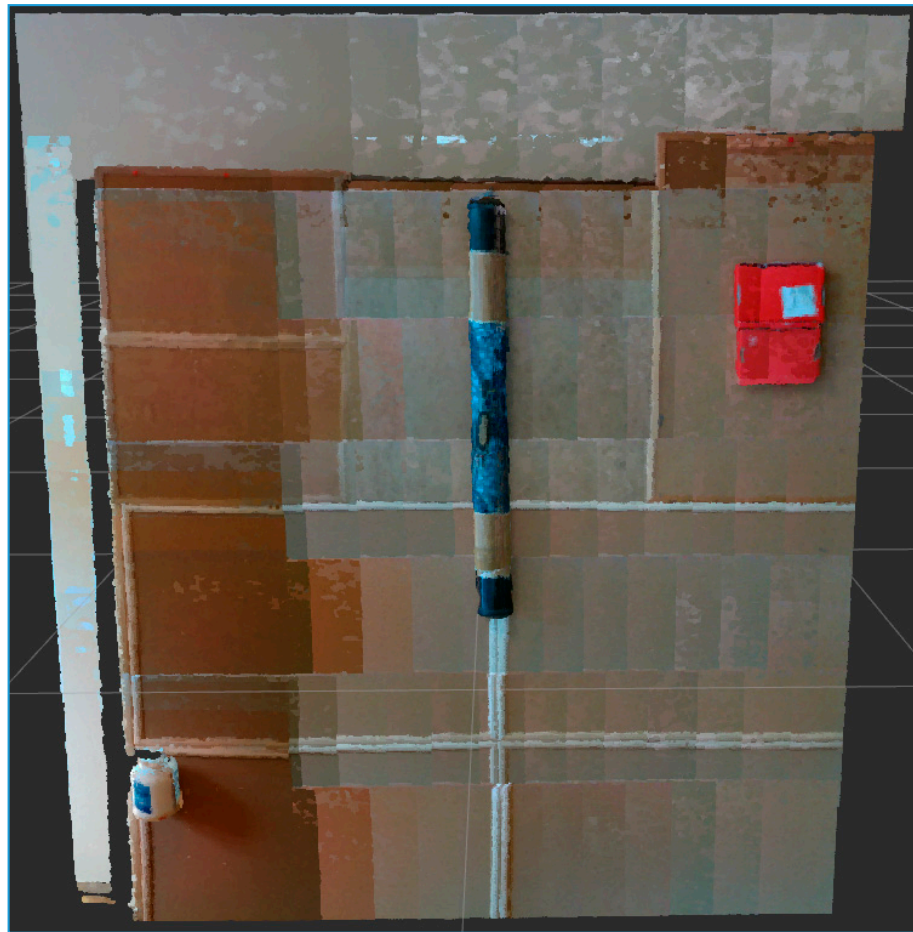


**Figure 3.** (a,b) The painting robot during the RTAB mapping process in order to form the point cloud of the wall surface.

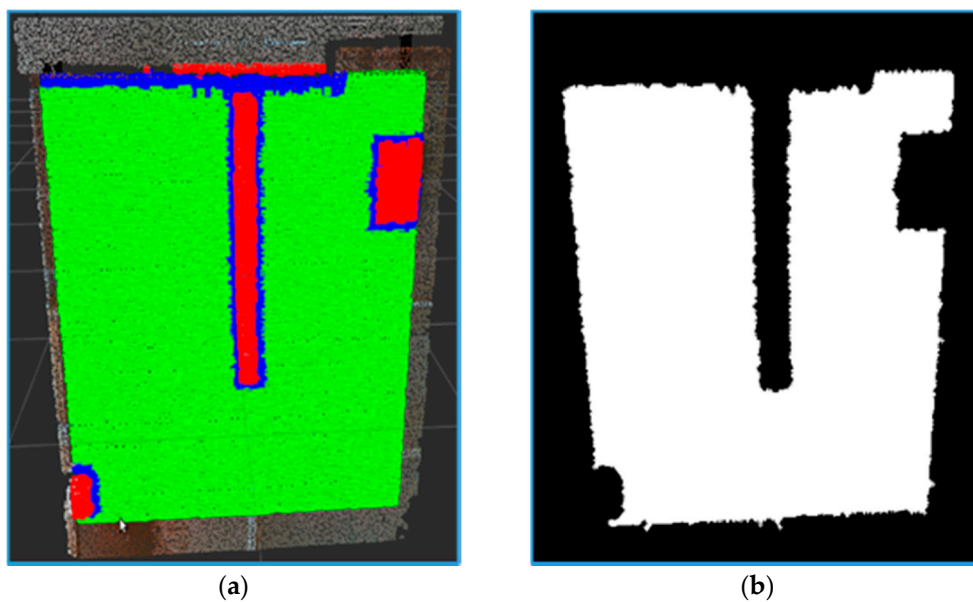
Finally, the resulting point cloud was observed to be suitable for further processing. Figure 4 shows the final merged point cloud of the wall surface with the obstacles (the blue colored stick, the white bottle, and the red colored boxes).

The next step was plane detection, specifically the detection and extraction of the wall area to be painted. The goal of the wall detection process was to avoid the barriers on the wall that should not be colored. The plane extraction process was performed by executing the RANSAC algorithm, preceded by statistical filtering. The statistical filtering [30] removed unnecessary, noisy gross outlier data from the point cloud. This operation is important because the point cloud is cleaned of non-essential outliers, meaning it is more efficient and able to execute the RANSAC algorithm quicker [28,29]. As was mentioned earlier, the main precedence of the RANSAC algorithm is its capability for robust assessment of the model parameters, because it can estimate the parameters with a high degree of accuracy, even when a great number of outliers are present in the data set. This means that in the case of lower quality depth information and depth images, the RANSAC algorithm detects the desired plane of the wall. This is an important feature, since a low-cost D435 depth camera is used for surface depth mapping [45,46].

The RANSAC plane detection results are presented in Figure 5a. The wall area was successfully extracted and is shown as green in the image. The red parts denote the detected obstacles. Since the obstacles were closer to the camera, they were set to be red in the camera's built-in software. The blue parts are the shadows of the obstacles, which the D435 camera recognized as very distant objects. As a result, they were not significant in the later painting process.



**Figure 4.** The resulting merged point cloud of the wall surface with obstacles included.

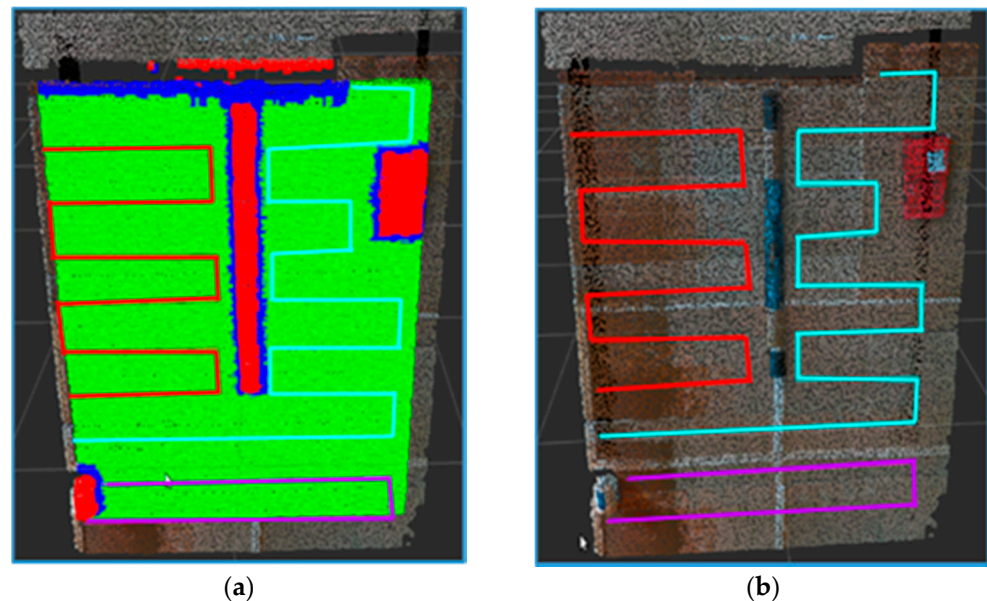


**Figure 5.** (a) The detected plane without obstacles representing the wall, as detected by the RANSAC algorithm. (b) Binary mask of the detected plane without obstacles, which represents the wall to be painted by the painting robot.

Figure 5b shows the binary mask of the plane-detected image. The white area represents the surface that should be painted. This binary image was connected with the

coordinates of the point cloud of the surface and this information was forwarded to the robot's control system. Taking the obtained data and using the ROS, the control software began the painting process. Since the development and equipping of the painting equipment was still in progress, the painting process was simulated using the robot's arm and ROS software.

Figure 6a,b shows the simulation process for the wall painting. It should be noted that the red, blue, and purple lines show the movement of the painting head. It is easy to notice that the painting head avoided the obstacles during the painting process.

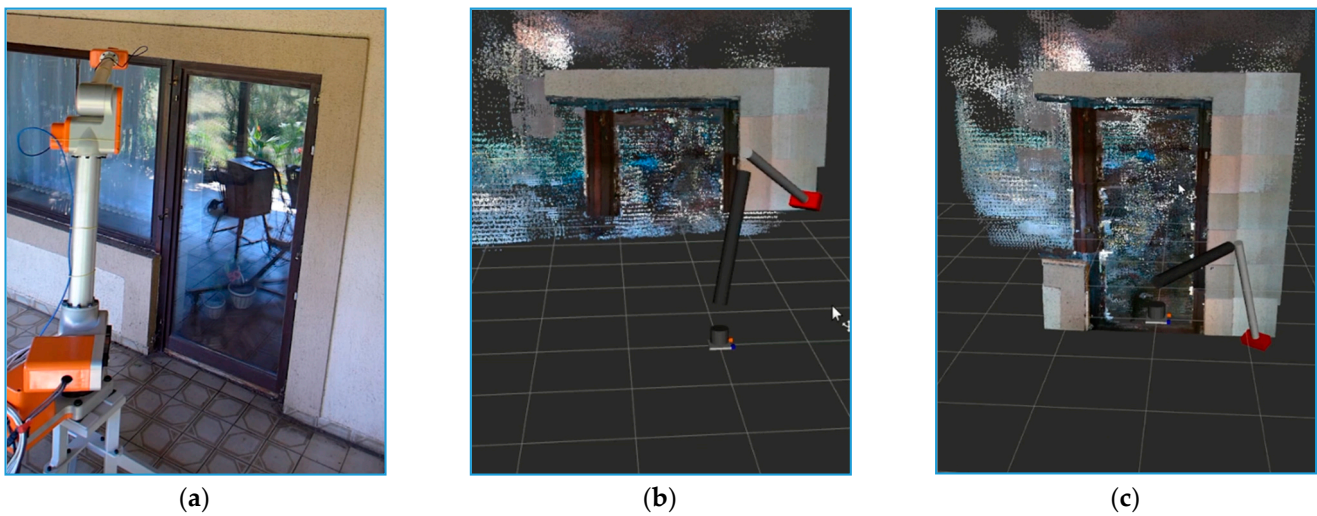


**Figure 6.** (a) The movement of the painting robot during the painting simulating process for the detected wall surface. (b) The movement of the painting robot in the painting simulating process on the detected wall surface for the real point cloud.

The complete procedure for wall capturing, point cloud merging, plane detection, and wall painting lasted about 4–5 min on the corresponding hardware. The hardware configuration used in the tests included an Intel Core i7 8700 K CPU, Asus GeForce GTX 1050 Dual 2 GB GDDR5 128 bit graphic card, 32 GB 3000 MHz DDR4 RAM memory, Samsung 850 EVO 250 GB SATA3 SSD, and Asus PRIME Z370-A desktop motherboard. This is important for commercial purposes, since the whole procedure is quite fast—certainly faster than the time it would take a person to paint the same surface.

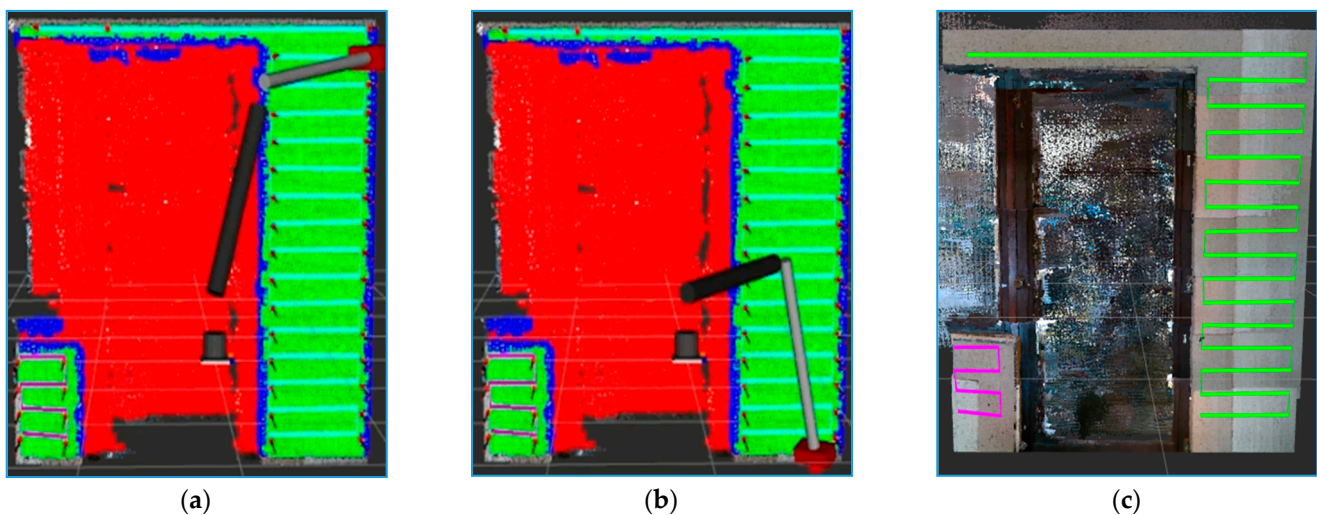
The next example will introduce the segmentation process and the painting simulation process for an outdoor wall on a terrace.

Figure 7a shows a door and the window surrounded by the wall to be painted. Figure 7b,c displays the painting robot during the RTAB mapping process used to form the point cloud of the complex surface. The red rectangle-based object represents the depth camera used in simulation process. As can be seen, the point cloud is formed successfully using the previously explained procedure.



**Figure 7.** (a) The experiment setup. (b) The painting robot during RTAB mapping process used to form the point cloud of the wall surface. (c) The resulting merged point cloud of the wall surface with a door and window.

The next step in the process was plane detection, again with the goal of detecting and extracting the wall area to be painted. The goal of the wall detection process was to avoid the door and the window, which were not to be painted. The plane extraction process was performed again by executing the RANSAC-based segmentation algorithm, preceded by statistical filtering. In this example, the wall area was successfully extracted and is colored green in the image. The red parts denote the detected obstacles, notably the door and the window, as can be seen in Figure 8a,b. The blue areas represent the frames of the door and window in the image. Since the depth camera sees through the glass, the door and the window areas are detected as distant objects in the depth image and are successfully separated from the wall area.

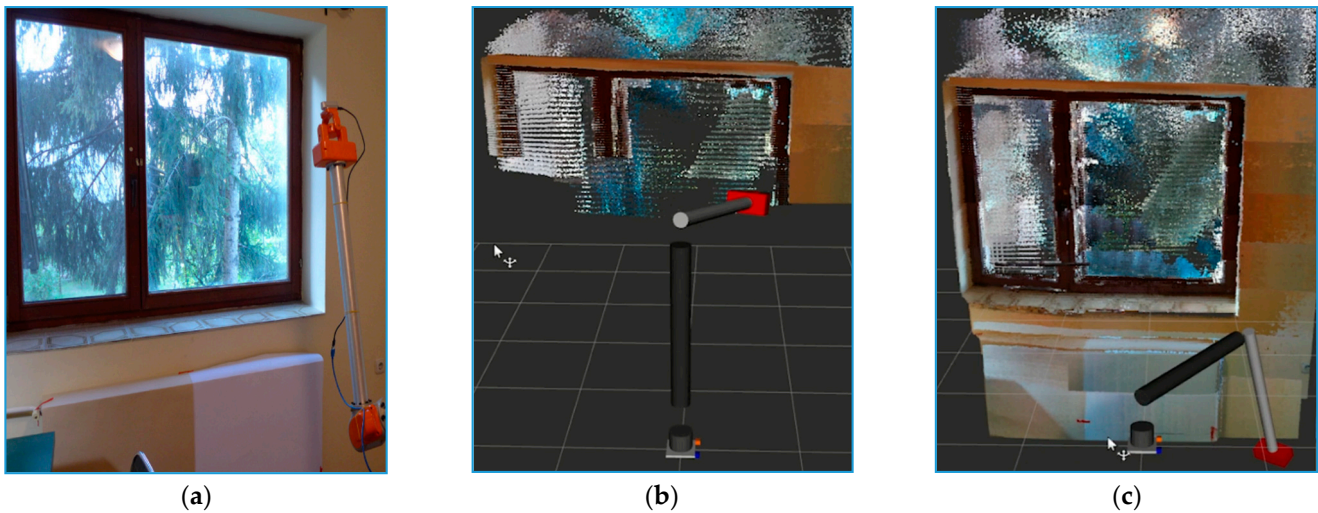


**Figure 8.** (a,b) The painting robot during the painting simulating process on the detected wall surface. (c) The painting robot in the painting simulating process on the detected wall surface based on the real point cloud.

Finally, Figure 8a–c shows the painting process simulation. Again, it should be noted that the red, blue, and purple lines show the movement of the painting head. It is easy to notice that the painting head avoids the door and window during the painting process.

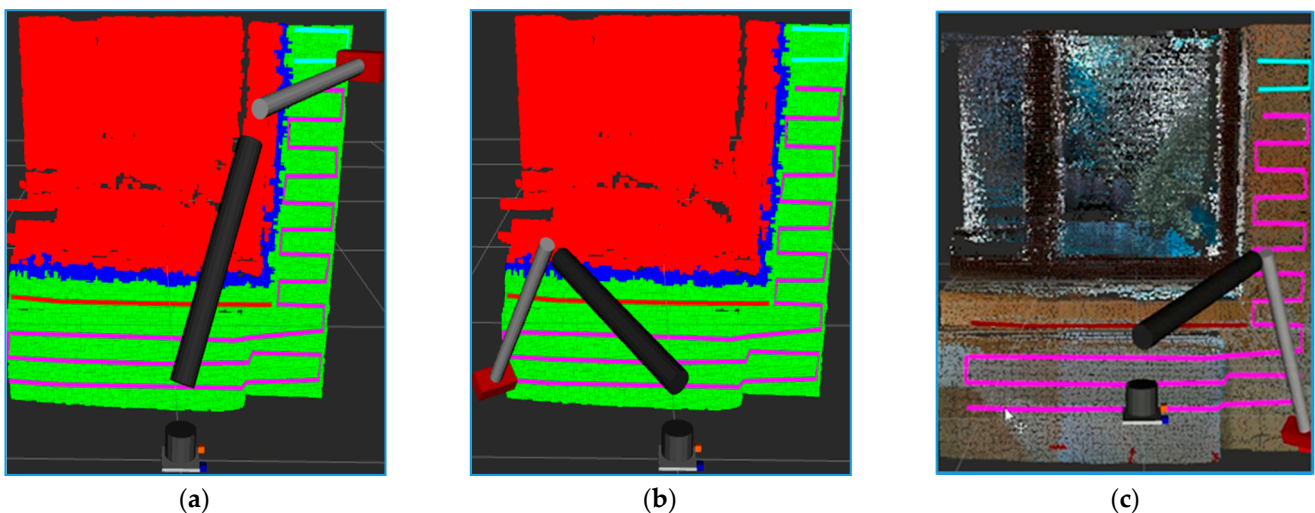
The next example will show the segmentation process and the painting simulation process for an indoor wall in a room.

Figure 9a shows the window surrounded by the wall to be painted. Figure 9b,c shows the painting robot during the RTAB mapping process used to form the point cloud of the complex surface. Again, the point cloud was formed successfully using the previously explained procedure.



**Figure 9.** (a) The experiment setup. (b) The painting robot during the RTAB mapping process used to form the point cloud of the wall surface. (c) The resulting merged point cloud of the wall surface with the windows included.

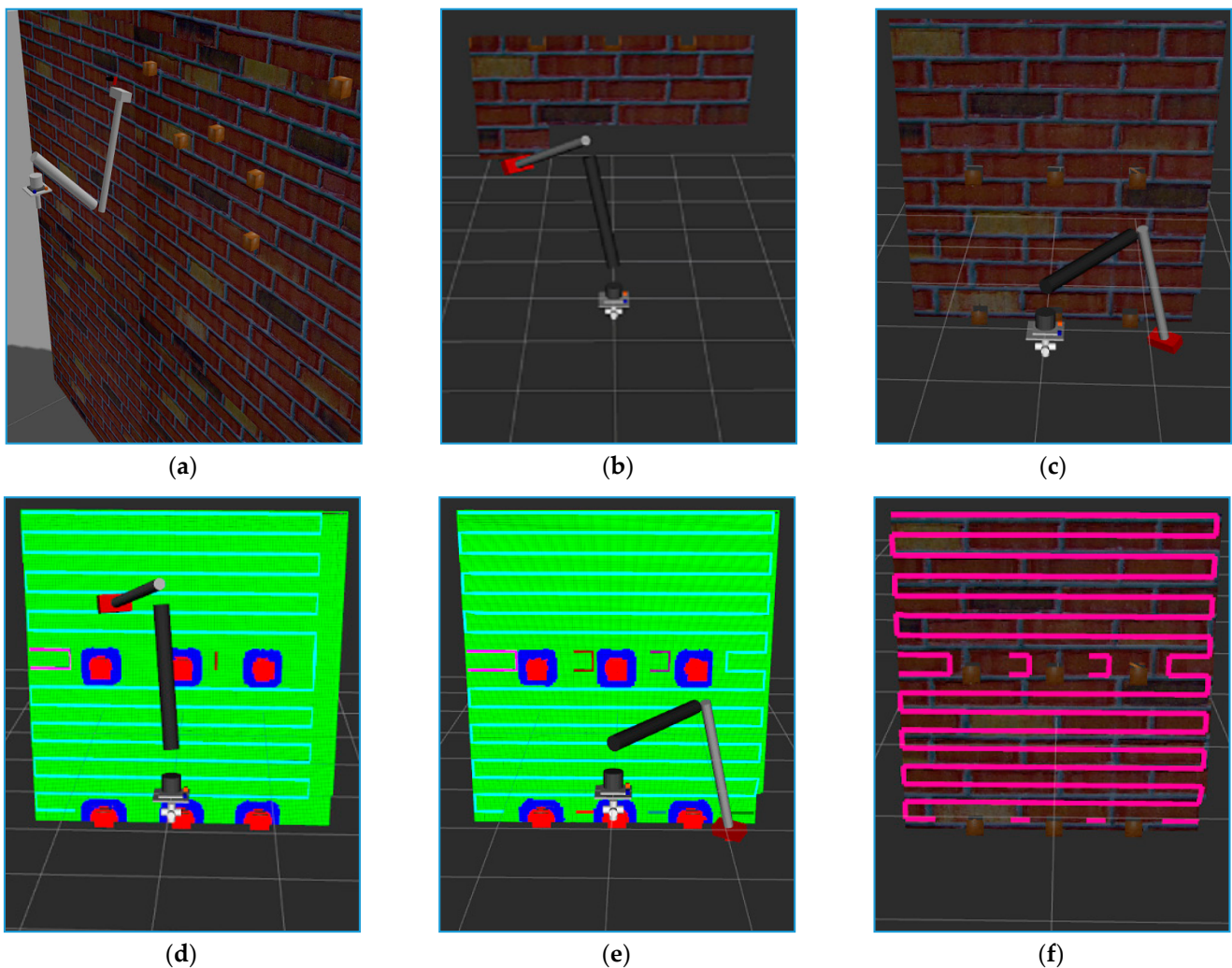
The next step involved detecting the wall area to be painted. The goal of the wall detection step was to avoid the window, which was not to be painted. The plane extraction process was again performed successfully. In this example, the wall area was extracted and is colored green in the image. The red parts denote the detected obstacles, notably the window, as can be seen in Figure 10a,b. In this example also, the depth camera was able to see through the glass and the window area was detected as a distant object in the scene of the depth image, and was successfully separated from the wall area.



**Figure 10.** (a,b) The painting robot during the painting simulating process on the detected wall surface. (c) The painting robot during the painting simulating process on the detected wall surface based on the real point cloud.

Finally, Figure 10a–c shows the painting process simulation. The red, blue, and purple lines show the movement of the painting head. It is easy to notice again that the painting head avoided the window during the painting process.

Finally, in order to verify the experiments, the simulation results are summarized in brief. Figure 11a–f shows the simulation process performed with Gazebo. During the simulation, the movement of the 5 DOF robot can be seen. In this example, a wall was simulated, whereby small cubes were placed as obstacles. Figure 11a shows the simulation setup, i.e., the wall with the obstacles. Figure 11b,c depicts the wall scanning process, which is analogous to the RTAB mapping process. Figure 11d,e illustrates the point cloud of the wall with the detected obstacles and the path planning process. Finally, Figure 11f shows the wall painting process in the simulation environment. As can be seen in the verification example in Figure 11, the testing results for the actual scenarios are consistent with the simulation results.



**Figure 11.** (a–f) The simulation process for the painting robot application.

Since the client's requirement, and the main goal of the project, was the separation of the wall and windows, this goal was fulfilled. The windows would be covered with a protective foil in real life, as in any other painting job. The reason for covering the window is that even the best spray guns are not precise enough to avoid painting some parts of the edges of windows, while light winds and dripping paint can also damage the window. The experiments proved that the RealSense depth camera can reliably separate depth differences from about 4 cm. Usually, the depth differences between the walls and windows are greater than 4 cm, which is suitable for the RealSense camera. In this manner, the depth camera performed satisfactorily in terms of the client's satisfaction and instructions.

## 6. Conclusions

Here, we introduced the working principles and properties of a painting robot. The main steps of the wall extraction process were presented, i.e., RTAB mapping, statistical filtering, the RANSAC algorithm, and the clustering procedure. The point cloud of the surface was formed using an RTAB-Map procedure with a RealSense depth camera. Next, a statistical filter was applied to remove the outliers from the merged point cloud. In the end, the RANSAC algorithm together with the clustering procedure was used for the wall extraction process. RANSAC is a powerful iterative method used to assess parameters of mathematical models from sets of data containing outliers. The goal of this industrial research project was to build a robot vision system with known and reliable procedures in order to ensure the reliability of the robot's working process. Appropriate experiments were conducted according to the client's instructions on certain wall–window surfaces. All experiments were performed successfully. During the development of the robot and vision system, all of the client's requirements and instructions were followed. As a result, the client was satisfied and the project's goal was fully achieved.

## 7. Future Works

In the future, special painting equipment with a spray gun should be embedded in the robot with the depth camera and the robot should be tested in real painting applications. Further, there is a possibility for refinement of the visual mechanism using a depth camera with better performance or by adapting the robot control system to the technological process during practical use. We also plan to start series production of the painting robot. Finally, all improvements were client-dependent, and future research and development will be done depending on the requirements of each client.

**Author Contributions:** A.O., E.B., and P.O. conceived and performed the experiments. I.K. and V.T. checked the test results and suggested the corrections. A.O. and E.B. developed the hardware and software components and performed the complete implementation process in the ROS. A.T., M.K., Z.S., and Z.V. were consulted for some mathematical subtasks. Z.K. and P.O. supervised the research and contributed to the organization of article. V.T. drafted the manuscript and all authors revised and approved the final version of the manuscript. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by projects KFI\_16-1-2017-00485, EFOP-3.6.1-16-2016-00004, 2020-4.1.1-TKP2020, and 2020-1.1.2-PIACI-KFI-2020-00173, co-financed by the European Union.

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

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Acknowledgments:** The authors would like to thank the editors and the anonymous reviewers for their valuable comments, which significantly improved the quality of this paper.

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

## References

1. Carfagni, M.; Furferi, R.; Governi, L.; Santarelli, C.; Servi, M.; Uccheddu, F.; Volpe, Y. Metrological and Critical Characterization of the Intel D415 Stereo Depth Camera. *Sensors* **2019**, *19*, 489. [[CrossRef](#)] [[PubMed](#)]
2. Hu, J.; Niu, Y.; Wang, Z. Obstacle avoidance methods for rotor UAVs using RealSense camera. In Proceedings of the 2017 Chinese Automation Congress (CAC), Jinan, China, 20–22 October 2017; pp. 7151–7155.
3. Giancola, S.; Valenti, M.; Sala, R. *A Survey on 3D Cameras: Metrological Comparison of Time of Flight, Structured-Light and Active Stereoscopic Technologies*; Springer Nature: Berlin/Heidelberg, Germany, 2018.
4. Keselman, L.; Woodfill, J.L.; Grunnet-Jepsen, A.; Bhowmik, A. Intel RealSense Stereoscopic Depth Cameras. *arXiv* **2017**, arXiv:1705.05548, 1–10.
5. Legendijk, R.L.; Ruggero, E.H.F.; Hendriks, E.A. The work was supported in part by the Euro-pean Union under the RACE-II project DISTIMA and the ACTS project PANORAMA. In *Stereoscopic Image Processing*; Delft University of Technology, Electrical Engineering: Delft, The Netherlands, 2002.



6. Siena, F.L.; Byrom, B.; Watts, P.; Breedon, P. Utilising the Intel RealSense Camera for Measuring Health Outcomes in Clinical Research. *J. Med. Syst.* **2018**, *42*, 1–10. [[CrossRef](#)] [[PubMed](#)]
7. New Technologies Group, Intel Corporation. *Intel RealSense D400 Series Product Family Datasheet*; Document Number: 337029-005; New Technologies Group, Intel Corporation: Santa Clara, CA, USA, 2019.
8. Grunnet-Jepsen, A.; Tong, D. *Depth Post-Processing for Intel®RealSense™ D400 Depth Cameras*; Revision 1.0.2; New Technologies Group, Intel Corporation: Santa Clara, CA, USA, 2018.
9. Berkeley Design Technology, Inc. *Evaluating Intel's RealSense SDK 2.0 for 3D Computer Vision Using the RealSense D415/D435 Depth Cameras*; Berkeley Design Technology, Inc.: Walnut Creek, CA, USA, 2018.
10. New Technologies Group, Intel Corporation. *Intel®RealSense™ Camera Depth Testing Methodology*; Revision 1.0; New Technologies Group, Intel Corporation: Santa Clara, CA, USA, 2018.
11. Grunnet-Jepsen, A.; Sweetser, J.N.; Woodfill, J. *Best-Known-Methods for Tuning Intel®RealSense™ D400 Depth Cameras for Best Performance*; Revision 1.9; New Technologies Group, Intel Corporation: Santa Clara, CA, USA, 2018.
12. Gastal, E.S.L.; Oliveira, M.M. Domain transform for edge-aware image and video processing. *ACM Trans. Graph.* **2011**, *30*, 1–12. [[CrossRef](#)]
13. Grunnet-Jepsen, A.; Winer, P.; Takagi, A.; Sweetser, J.; Zhao, K.; Khuong, T.; Nie, D.; Woodfill, J. *Using the Intel®RealSense™ Depth Cameras D4xx in Multi-Camera Configurations*; Revision 1.1; New Technologies Group, Intel Corporation: Santa Clara, CA, USA, 2018.
14. New Technologies Group, Intel Corporation. *Intel RealSense Depth Module D400 Series Custom Calibration*; Revision 1.5.0; New Technologies Group, Intel Corporation: Santa Clara, CA, USA, 2019.
15. Grunnet-Jepsen, A.; Sweetser, J.N. *Intel RealSense Depth Cameras for Mobile Phones*; New Technologies Group, Intel Corporation: Santa Clara, CA, USA, 2019.
16. Krejov, P.; Grunnet-Jepsen, A. *Intel RealSense Depth Camera Over Ethernet*; New Technologies Group, Intel Corporation: Santa Clara, CA, USA, 2019.
17. Cunha, J.; Pedrosa, E.; Cruz, C.; Neves, A.J.R.; Lau, N. Using a Depth Camera for Indoor Robot Localization and Navigation. In Proceedings of the Conference: RGB-D: Advanced Reasoning with Depth Cameras Workshop, Robotics Science and Systems conference (RSS), Los Angeles, CA, USA, 27 June 2011.
18. Hemmat, H.J.; Bondarev, E.; De With, P.H.N. *Real-Time Planar Segmentation of Depth Images: From 3D Edges to Segmented Planes*; Eindhoven University of Technology, Department of Electrical Engineering: Eindhoven, The Netherlands, 2015.
19. Flacco, F.; Kröger, T.; De Luca, A.; Khatib, O. A depth space approach to human-Robot collision avoidance. In Proceedings of the 2012 IEEE International Conference on Robotics and Automation, Sain Paul, MN, USA, 14–18 May 2012; pp. 338–345.
20. Saxena, A.; Chung, S.H.; Ng, A.Y. 3-D Depth Reconstruction from a Single Still Image. *Int. J. Comput. Vis.* **2007**, *76*, 53–69. [[CrossRef](#)]
21. Sterzentsenko, V.; Karakottas, A.; Papachristou, A.; Zioulis, N.; Doumanoglou, A.; Zarpalas, D.; Daras, P. A Low-Cost, Flexible and Portable Volumetric Capturing System. In Proceedings of the 2018 14th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS), La Palmas de Gran Canaria, Spain, 26–29 November 2018; pp. 200–207.
22. Carey, N.E.; Nagpal, R.; Werfel, J. Fast, accurate, small-scale 3D scene capture using a low-cost depth sensor. In Proceedings of the 2018 IEEE Winter Conference on Applications of Computer Vision (WACV), Santa Rosa, CA, USA, 24–31 March 2017; pp. 1268–1276.
23. Labbé, M.; Michaud, F. RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for Large-Scale and Long-Term online operation. *J. Field Robot.* **2019**, *36*, 416–446. [[CrossRef](#)]
24. Labbé, M.; Michaud, F. Long-term online multi-session Graph-Based SPLAM with memory management. *Auton. Robot.* **2018**, *42*, 1133–1150. [[CrossRef](#)]
25. Labbé, M.; Michaud, F. Memory management for Real-Time Appearance-Based loop closure detection. In Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, San Francisco, CA, USA, 25–30 September 2011. [[CrossRef](#)]
26. Labbe, M.; Michaud, F. Online global loop closure detection for Large-Scale Multi-Session Graph-Based SLAM. In Proceedings of the 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, 14–18 September 2014; pp. 2661–2666.
27. Labbe, M.; Michaud, F. Appearance-Based Loop Closure Detection for Online Large-Scale and Long-Term Operation. *IEEE Trans. Robot.* **2013**, *29*, 734–745. [[CrossRef](#)]
28. 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](#)]
29. Derpanis, K.G. Overview of the RANSAC Algorithm. *Image Rochester N. Y.* **2010**, *4*, 2–3.
30. Rusu, R.B.; Marton, Z.C.; Blodow, N.; Dolha, M.; Beetz, M. Towards 3D Point cloud based object maps for household environments. *Robot. Auton. Syst.* **2008**, *56*, 927–941. [[CrossRef](#)]
31. Li, X.; Guo, W.; Li, M.; Sun, L. Combining two point clouds generated from depth camera. In Proceedings of the 2013 IEEE International Conference on Robotics and Biomimetics (ROBIO), Shenzhen, China, 12–14 December 2013; pp. 2620–2625.
32. El-Sayed, E.; Abdel-Kader, R.F.; Nashaat, H.; Marei, M. Plane detection in 3D point cloud using octree-balanced density down-sampling and iterative adaptive plane extraction. *IET Image Process.* **2018**, *12*, 1595–1605. [[CrossRef](#)]
33. Gallo, O.; Manduchi, R.; Rafii, A. CC-RANSAC: Fitting planes in the presence of multiple surfaces in range data. *Pattern Recognit. Lett.* **2011**, *32*, 403–410. [[CrossRef](#)]

34. Mufti, F.; Mahony, R.; Heinzmann, J. Spatio-Temporal RANSAC for Robust Estimation of Ground Plane in Video Range Images for Automotive Applications. In Proceedings of the 11th International IEEE Conference on Intelligent Transportation Systems, Beijing, China, 12–15 October 2008.
35. Nurunnabi, A.; West, G.; Belton, D. Outlier detection and robust normal-curvature estimation in mobile laser scanning 3D point cloud data. *Pattern Recognit.* **2015**, *48*, 1404–1419. [[CrossRef](#)]
36. Li, L.; Yang, F.; Zhu, H.; Li, D.; Li, Y.; Tang, L. An Improved RANSAC for 3D Point Cloud Plane Segmentation Based on Normal Distribution Transformation Cells. *Remote. Sens.* **2017**, *9*, 433. [[CrossRef](#)]
37. Li, Y.; Li, W.; Darwish, W.; Tang, S.; Hu, Y.; Chen, W. Improving Plane Fitting Accuracy with Rigorous Error Models of Structured Light-Based RGB-D Sensors. *Remote. Sens.* **2020**, *12*, 320. [[CrossRef](#)]
38. Schwarze, T.; Lauer, M. Wall Estimation from Stereo Vision in Urban Street Canyons. In Proceedings of the 10th International Conference on Informatics in Control, Automation and Robotics, Reykjavik, Iceland, 29–31 July 2013; Springer: Berlin/Heidelberg, Germany, 2014; pp. 83–90.
39. Xu, M.; Lu, J. Distributed RANSAC for the robust estimation of three-dimensional reconstruction. *IET Comput. Vis.* **2012**, *6*, 324–333. [[CrossRef](#)]
40. Xu, B.; Jiang, W.; Shan, J.; Zhang, J.; Li, L. Investigation on the Weighted RANSAC Approaches for Building Roof Plane Segmentation from LiDAR Point Clouds. *Remote. Sens.* **2015**, *8*, 5. [[CrossRef](#)]
41. Zhou, S.; Kang, F.; Li, W.; Kan, J.; Zheng, Y.; He, G. Extracting Diameter at Breast Height with a Handheld Mobile LIDAR System in an Outdoor Environment. *Sensors* **2019**, *19*, 3212. [[CrossRef](#)] [[PubMed](#)]
42. Deschaud, J.E.; Goulette, F. A Fast and Accurate Plane Detection Algorithm for Large Noisy Point Clouds Using Filtered Normals and Voxel Growing. In *3DPVT*; Hal Archives-Ouvertes: Paris, France, 2010.
43. ROS Depth Camera Integration. Available online: [http://gazebosim.org/tutorials/?tut=ros\\_depth\\_camera](http://gazebosim.org/tutorials/?tut=ros_depth_camera) (accessed on 25 January 2021).
44. Najdataei, H.; Nikolakopoulos, Y.; Gulisano, V.; Papatriantafidou, M. Continuous and Parallel LiDAR Point-Cloud Clustering. In Proceedings of the 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS), Vienna, Austria, 2–6 July 2018; pp. 671–684.
45. Sproull, R.F. Refinements to nearest-neighbor searching in k-dimensional trees. *Algorithmica* **1991**, *6*, 579–589. [[CrossRef](#)]
46. Tadic, V.; Odry, A.; Kecskes, I.; Burkus, E.; Kiraly, Z.; Odry, P. Application of Intel RealSense Cameras for Depth Image Generation in Robotics. *WSEAS Transac. Comput.* **2019**, *18*, 2224–2872.
47. Tadic, V.; Burkus, E.; Odry, A.; Kecskes, I.; Kiraly, Z.; Odry, P. Effects of the post-processing on depth value accuracy of the images captured by RealSense cameras. *Contemp. Eng. Sci.* **2020**, *13*, 149–156. [[CrossRef](#)]
48. Yow, K.-C.; Kim, I. General Moving Object Localization from a Single Flying Camera. *Appl. Sci.* **2020**, *10*, 6945. [[CrossRef](#)]
49. Point Cloud Library. Available online: <https://pointclouds.org> (accessed on 16 October 2020).
50. Qi, X.; Wang, W.; Liao, Z.; Zhang, X.; Yang, D.; Wei, R. Object Semantic Grid Mapping with 2D LiDAR and RGB-D Camera for Domestic Robot Navigation. *Appl. Sci.* **2020**, *10*, 5782. [[CrossRef](#)]
51. Kang, X.; Li, J.; Fan, X.; Wan, W. Real-Time RGB-D Simultaneous Localization and Mapping Guided by Terrestrial LiDAR Point Cloud for Indoor 3-D Reconstruction and Camera Pose Estimation. *Appl. Sci.* **2019**, *9*, 3264. [[CrossRef](#)]
52. Tadic, V.; Odry, A.; Toth, A.; Vizvari, Z.; Odry, P. Fuzzified Circular Gabor Filter for Circular and Near-Circular Object Detection. *IEEE Access* **2020**, *8*, 96706–96713. [[CrossRef](#)]
53. Tadic, V.; Odry, A.; Burkus, E.; Kecskes, I.; Kiraly, Z.; Odry, P. Edge-Preserving Filtering and Fuzzy Image Enhancement in Depth Images captured by RealSense Cameras in Robotic Applications. *Adv. Electr. Comput. Eng.* **2020**, *20*, 83–92. [[CrossRef](#)]
54. Odry, A.; Kecskés, I.; Šarčević, P.; Vizvari, Z.; Toth, A.; Odry, P. A Novel Fuzzy-Adaptive Extended Kalman Filter for Real-Time Attitude Estimation of Mobile Robots. *Sensors* **2020**, *20*, 803. [[CrossRef](#)]
55. Chen, Y.; Zhou, W. Hybrid-Attention Network for RGB-D Salient Object Detection. *Appl. Sci.* **2020**, *10*, 5806. [[CrossRef](#)]
56. Shang, D.; Wang, Y.; Yang, Z.; Wang, J.; Liu, Y. Study on Comprehensive Calibration and Image Sieving for Coal-Gangue Separation Parallel Robot. *Appl. Sci.* **2020**, *10*, 7059. [[CrossRef](#)]