Article

The Deep Convolutional Neural Network Role in the Autonomous Navigation of Mobile Robots (SROBO)

Shabnam Sadeghi Esfahlani [1,*], Alireza Sanaei [2], Mohammad Ghorabian [2] and Hassan Shirvani [1]

1 Medical Technology Research Centre (MTRC), School of Engineering and the Built Environment, Anglia Ruskin University, Essex CM1 1SQ, UK; hassan.shirvani@aru.ac.uk
2 School of Engineering and the Built Environment, Anglia Ruskin University, Essex CM1 1SQ, UK; alireza.sanaei@aru.ac.uk (A.S.); mhg103@pgr.aru.ac.uk (M.G.)
* Correspondence: shabnam.sadeghi-esfahlani@aru.ac.uk

Abstract: The ability to navigate unstructured environments is an essential task for intelligent systems. Autonomous navigation by ground vehicles requires developing an internal representation of space, trained by recognizable landmarks, robust visual processing, computer vision and image processing. A mobile robot needs a platform enabling it to operate in an environment autonomously, recognize the objects, and avoid obstacles in its path. In this study, an open-source ground robot called SROBO was designed to accurately identify its position and navigate certain areas using a deep convolutional neural network and transfer learning. The framework uses an RGB-D MYNTEYE camera, a 2D laser scanner and inertial measurement units (IMU) operating through an embedded system capable of deep learning. The real-time decision-making process and experiments were conducted while the onboard signal processing and image capturing system enabled continuous information analysis. State-of-the-art Real-Time Graph-Based SLAM (RTAB-Map) was adopted to create a map of indoor environments while benefiting from deep convolutional neural network (Deep-CNN) capability. Enforcing Deep-CNN improved the performance quality of the RTAB-Map SLAM. The proposed setting equipped the robot with more insight into its surroundings. The robustness of the SROBO increased by 35% using the proposed system compared to the conventional RTAB-Map SLAM.

Keywords: semantic segmentation; object detection; autonomous navigation; convolutional networks; transfer learning; deep learning

check for updates

Citation: Sadeghi Esfahlani, S.; Sanaei, A.; Ghorabian, M.; Shirvani, H. The Deep Convolutional Neural Network Role in the Autonomous Navigation of Mobile Robots (SROBO). *Remote Sens.* **2022**, *14*, 3324. https://doi.org/10.3390/rs14143324

Academic Editors: Donato Amitrano and Luca Cicala

Received: 19 May 2022
Accepted: 8 July 2022
Published: 10 July 2022

1. Introduction

A robot's autonomous navigation is a complicated task that relies on composing a representation of an area, grounded by robust visual processing and computer vision [1]. Robots need to simultaneously maintain a steady self-localization and representation of objectives. Autonomous ground robots have various applications, i.e., surveillance, healthcare, and transporting care facilities. With enhanced development, they have the potential to automate a vast collection of labor-intensive chores and execute well in unstructured, challenging environments with extreme pressures and temperatures.

An autonomous mobile robot constructs a robust model of the environment (mapping), locates itself on the map (localization), governs the movement from one location to the other (navigation) and accomplishes assigned tasks. Cognitive robotics [2,3] is a compelling topic among researchers, computer vision and robotics communities. It adopts artificial intelligence (AI) to enable agents to make decisions independently. Cognitive robotics offers more robust solutions in more critical situations, i.e., heterogeneous environments, concerning accuracy, integrity, and potentially enhancing execution.

There has been a significant improvement in the accuracy and robustness of visual simultaneous localization and mapping (VSLAM) in the last two decades [4–7]. However, there is a gap in investigating the role of AI in VSLAM systems and fusing them. It has the potential to improve the robustness of the system and optimize its performance. It also provides robots with a better understanding of the environment they function.

Biologic neurons inspire an artificial neural network (ANN) in that inputs are weighted and added. The neuron outputs a scale value following a transfer function. Based on the artificial neuron, a different collection of the neurons creates different types of ANNs, i.e., Convolutional Neural Network (CNN) [8], Long Short-Term Memory [9], Auto Encoder [10], Recursive Neural Network [11], or Restricted Boltzmann Machine [12].

CNNs are learning algorithms with outstanding image classification, segmentation, and detection results due to their remarkable configuration [13]. CNN manipulates spatial or temporal correlation in data with the topology of multiple learning sets comprised of convolutional layers, nonlinear processing elements, and sub-sampling layers [14]. In the feedforward multilayered ranked CNNs, each layer employs convolutional kernels to execute numerous conversions [15]. CNN has automatic feature extraction capacity, diminishing the demand for a distinct feature extractor [16], thus facilitating suitable internal explanation of raw pixels. During training, CNN learns through the backpropagation algorithm by restraining the weight change according to the target. Deep-CNN's multilayered, hierarchical structure allows it to pull varied level features [17]. Semantic image segmentation (pixel-level prediction) is a method of clustering parts of an image and classifying each pixel into its classification [18]. Numerous public datasets are available for image segmentation. These benchmark datasets consist of various images with different aspects, including brightness deviation, intra-class divergence, and background sophistication [13].

RGB-D SLAM is a VSLAM that employs RGB-D sensors for localization, and mapping [19]. It generates a dense 3D environment replication while keeping track of the camera pose. Each pixel corresponds to a depth value that provides a dense and colored point cloud. Metric information is rendered, addressing the scale obscurity issue in image-based SLAM systems. Depth data allows the system to use depth information in poor texture environments, while in limited structure environments, RGB report is adopted for matching and registration [20].

In this study, we propose fusing artificial neural networks (ANN) known as deep convolutional neural networks (Deep-CNN) with the conventional RTAB-Map visual SLAM (VSLAM). The proposed system demands the acquaintance of the current camera pose when observing a scene from a SLAM system running in the background. VSLAM uses RGB and depth info for sparse tracking. The point clouds rendered from the depth pipeline are projected to the corresponding sensor location to obtain a map of the environment. The output of VSLAM is used to execute semantic labeling and object detection in the scene. The open-source ground robot known as SROBO is developed and tested the performance of the presented system. The results indicated the satisfactory performance of the robot regarding robustness and decision-making.

The remainder of the manuscript is organized as follows. In Section 2, the mechanical and electronic hardware of SROBO is introduced. It describes the RTAB-Map and Deep-CNN (semantic segmentation and object detection) architectures adopted to generate the system. Section 3 demonstrates the results of semantic segmentation and object-detection algorithms and their fusion with RTAB-Map SLAM. The proposed system performance was assessed by comparing the conventional RTAB-Map with the proposed system (fusion of RTAB-Map with Deep-CNN). Finally, we outline the conclusion in Section 4.

2. Design and Methodology

Estimating the pose of an agent (translation and orientation) over time is the process of odometry [5,21]. An agent (e.g., robot, vehicle, and human) uses exteroceptive (external data) and proprioceptive (internal to the system) sensors to estimate its position relative to an initial location. Visual Odometry (VO) uses a stream of images acquired from a camera attached to the robot to localize it. Wheel odometry (WO) provides data on the translation and rotation of wheels by measuring the number and speed of wheel rotations.

Odometry of an inertial navigation system (INSO) is calculated by employing an inertial measurement unit (IMU) consisting of accelerometers, gyroscopes, and magnetometers (9-DOF). The gyroscope's angular rate measurements are integrated for a high-update rate attitude explanation. At the same time, the magnetometer provides a heading reference using a state-of-the-art Kalman filter algorithm to determine the attitude, velocity and position. The attitude is calculated by integrating angular rate as estimated by the gyros over a period. The accelerometer measures the system's linear acceleration due to motion and the pseudo-acceleration caused by gravity. WO constructs noisy data with high error rates due to sliding, uneven surfaces, and weight shifts. On the other hand, VO can afford precise trajectory estimates, with a relative position error of [0.1–2.0]% [22] with progressive frames and scene overlap to extract probable movement.

The fusion of proprioceptive (inertial measurement unit, encoders) and exteroceptive sensors (camera, LiDAR) can improve the robustness and accuracy of odometry estimation [23]. The map is generally represented as landmarks within a three-dimensional space, while sensors traverse the environment and generate a trajectory [24,25]. The landmarks are observed from the sensor poses, and the most feasible landmark positions and sensor poses are estimated. Information regarding the visual landscape of where the robot is navigating is as essential as creating a map for autonomous navigation and path planning.

Dense (pixel-wise) semantic segmentation was employed using the residual learning deep neural networks (ResNet) framework [26]. The result is a training network made of layers of learning residual functions concerning the layer inputs instead of learning unreferenced functions [27]. Dense per-pixel semantic segmentation classification was conducted by overlaying masks on the image.

Single-Shot Multi-Box Detector (SSD) is one of the fastest object-detection algorithms [28]. A single convolutional neural network discretizes the resulting space of bounding boxes in images into further default boxes over different scales and aspect ratios per feature map location. The network generates the scores for each object in the scene for classification and adjusts to better approximate the object's shape. In this study, SSD-MobileNet is generated by combining the SSD network as a meta-structure with the MobileNet [29] for surface defects. SSD-Inception [30,31] is another object-detection algorithm used in this research, which is designed based on SSD, with high performance and accuracy. It uses a single convolutional neural network to detect the object in an image in that the inception block replaces the extra layers in SSD.

2.1. Mechanical and Electronic Hardware

The primary control system in SROBO is Jetson Nano, which possesses its sensor information such as RPLidat A1, MYNTEYE-D camera, an IMU and two 12V DC motors with a rotary quadrature encoder. The integrated control system has 8GB RAM and a 128-core Maxwell GPU powered using a power bank with 5 V and 3 A. A TB67H420FTG dual-motor drive carrier handles the speed and direction of motors via PWM pulse computation. We incorporated the PID (Proportional, Integral, Derivative) regulator to facilitate the control loop feedback mechanism processed on a Teensy 4.1 microcontroller.

The LiDAR (2D light detection and ranging) rotates 360° clockwise and has a range of [1.5–12,000] mm. It collects 1450 sampling points at a frequency of 5.5 Hz, in that eight thousand sample points were collected per second at 10 Hz. The LiDAR emits modulated infrared laser signals and receives echoed signs from detected objects. It is powered by 5 V USB protocol via the primary control system.

We used an MYNTEYE-D-1000 camera with 120 FOV (field of view) and depth map resolution of 1280×720 at 60 FPS (frame per second). MYNTEYE's FOV is D:121° H:105° and V:58° and has six-axis IMU.

A nine-axis mems motion tracking device—MPU9250—is implemented and combined with MYNTEYE's IMU. The sensors and peripherals were calibrated and integrated to facilitate high performance.

The visual data from the MYNTEYE-D camera is blended with inertial measurement units (IMU), wheel encoders, the RPLiDAR laser scanner, and an extended Kalman filter (EKF) [32] to achieve state estimation in real time. Data are assembled into a unit with the high-frequency pose estimation from EKF, odometry and point clouds corresponding to the camera. Frequently received information is combined to build a consistent global 3D map via RTAB-Map [33]. Robust and accurate pose estimation based on sensors combination are transformed into a fixed coordinate frame in real time to build a global map. The sensors calibrated information is transformed to the robot's base frame using the unified robot description format (URDF) (http://wiki.ros.org/urdf; accessed on 18 May 2022) in the ROS (https://www.ros.org/; accessed on 18 May 2022) Melodic platform on Ubuntu 18.04. Although inferring properties from vision is a complex task and requires human perception [34]. This study aims to integrate automated image analysis semantic segmentation and object-detection algorithms combined with VSLAM to provide the robot with a richer perception of the environment. TensorRT SDK (https://developer.nvidia.com/tensorrt; accessed on 18 May 2022) was installed on the embedded device in that the NVIDIA CUDA parallel programming model was used for deep learning and optimization inference. The PyTorch framework was used for training models. The open neural network exchange (ONNX) system enabled interoperability and format exchange in the algorithm. Images from PyTorch were converted to TensorTR for training using training steps (epochs).

SROBO has 3DOF and is controlled across the x, y, and yaw axis. The communication through the I2C (Inter-Integrated Circuit) and general-purpose input-output (GPIO) protocols enables robust and consistent communication between the system and subsystems. SROBO's kinematics is determined as it receives information on position and speed through the respective algorithm and determines the PWM references of the motors.

Autocad Inventor 2022 is used to create the 3D model of the ground robot body parts, designed for 3D printing and laser cutting. The mechanical structure is a modular assembly, where body parts can be manufactured and upgraded individually, allowing easy parts construction and replacement as required. Figure 1 illustrates the mechanical parts created in Autocad. Table 1 lists the hardware specification and prices for components used in devising SROBO. SROBO's wheels are taken from a mobile car made by Makeblock (https://www.makeblock.com/; accessed on 18 May 2022). Figure 2 displays the electronic parts involved in the configuration of the ground robot.

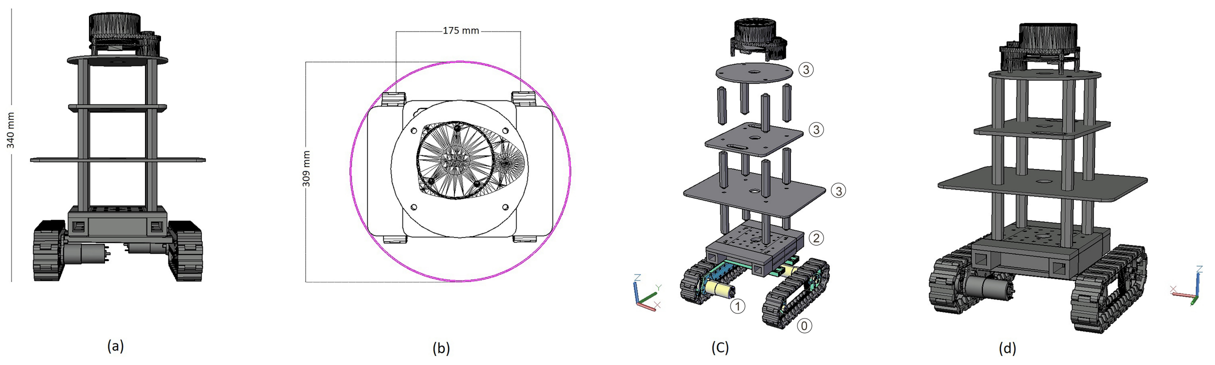

Figure 1. Mechanical hardware of SROBO. (**a**) 2D view of the robot. (**b**) Top view of the robot. (**c**) Assembly of the robot parts; ⓪ Caster wheel of the robot (two caster wheels), ① The motor connected to a wheel, ② 3D printed base, ③ Laser cut plates. (**d**) 3D view of the robot.

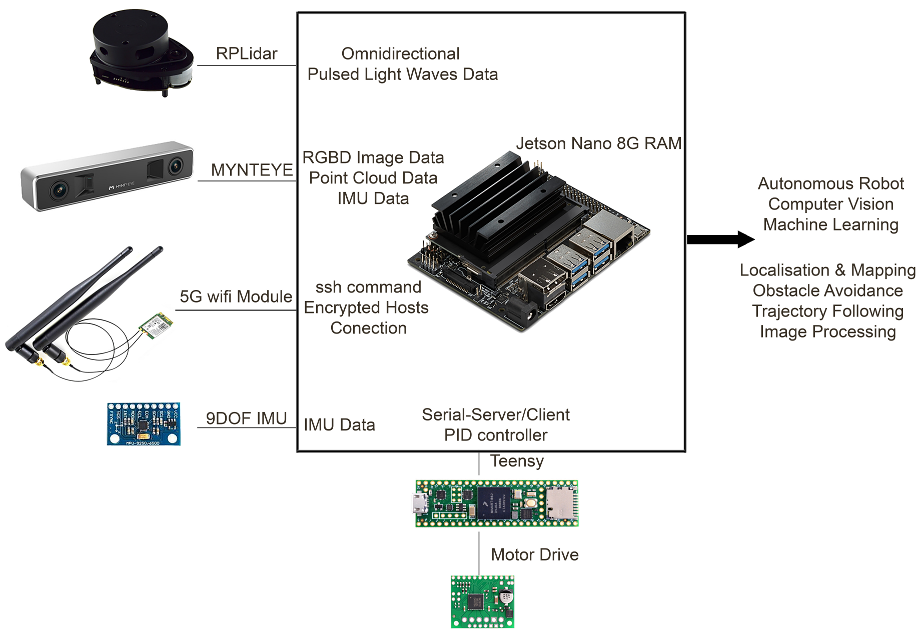

Figure 2. Electronic hardware used in the design of SROBO mobile car.

Table 1. Components and price range of SROBO.

Quantity	Device	Description	Cost Estimate	Weight
1	Jetson Nano	Quad-core ARM A57 @ 1.43 GHz of CPU	£299.99	250.00 g
1	TB67H420FTG	Dual-Motor Driver Carrier	£13.00	3.50 g
1	MPU-9250 module	9-axis Motion Tracking device	£8.74	2.72 g
2	12V DC Motor	251 revolution per minute (RPM) w/Encoder	£26.00	205.0 g
1	Development Board	Teensy 4.1 (ARM Cortex-M7 @ 600 MHz)	£26.00	4.5 g
1	MYNTEYE D1000-120	Stereo and Depth Camera	£289.97	44.0 g
1	RPLiDAR A1M8	2D Laser Scanner 360°	£87.35	170.0 g

2.2. Graph-Based SLAM

Solving SLAM problems concern evaluating an agent's trajectory and the map of the environment. Due to the intrinsic noise in the sensor measurements, a SLAM problem is usually expressed via probabilistic mechanisms. We describe the agent's movement in an unknown environment along a trajectory described by the sequence of random variables; $P_{1:S} = \{P_1, \ldots, P_S\}$. When moving, it formulates a sequence of odometry measurements $u_{1:S} = \{u_1, \ldots, u_S\}$ and perceptions of the environment $z_{1:S} = \{z_1, \ldots, z_S\}$. It required estimating the former probability of the robot's trajectory $P_{1:S}$ and the map M of the environment given all the measurements plus an initial position P_0:

$$p(P_{1:S}, M \mid z_{1:S}, u_{1:S}, P_0). \tag{1}$$

Estimating for Equation (1) involves operating in high-dimensional state spaces, which is only manageable if a dynamic Bayesian network configuration or, in other words, the graph-based formulation is used. A Bayesian network is a graphical representation defining a stochastic method as a directed graph. The graph has one node (x_s) per random variable in the procedure, and a supervised edge (z_s or u_s) between two nodes with a particular constraint and dependency [35]. A single front-end observation z_s might result in multiple conceivable edges connecting different poses in the graph. A graph-based mapping algorithm computes a Gaussian approximation of the previous location over the robot trajectory when the observations are influenced by Gaussian noise, and the data relationship is comprehended. The Gaussian mean is determined as the arrangement of the nodes that maximizes the likelihood of the observations. Once it is selected, the input matrix of the Gaussian is obtained. For more detailed information, the readers are referred to [35].

In a real-time graph-based SLAM approach such as RTAB-Map [33], a graph is constructed where nodes represent a robot's poses or landmarks. A sensor measurement data constrains the nodes created by connected poses. Noisy observations of the information are filtered through finding a configuration of the nodes that is maximally consistent with the measurements and solving a significant error minimization problem [36] while receiving information via various sensors. Loop closure is an essential step in graph-based SLAMs that helps optimize the graph, generate a map and correct drifts. A robust localization is needed to identify the revisiting past locations used for loop closure and map update.

Real-world dynamic environments cause challenges in autonomous navigation, mainly when dealing with localization on the generated map. It is due to the need to explore new areas and a lack of biased environmental features. This problem is addressed using multi-session mapping that enables a robot to recognize its relative position to a previously generated map, plan a path to a previously visited location, and localize itself. It allows the SLAM approach to initialize a new map based on its allusions and transform it into the previously created map while a previously seen area is discovered. It avoids remapping the whole environment when only a new location should be explored and incorporated. This approach is considered in this study when dealing with semantic segmentation in images described in the following section.

Graph-based SLAM systems are constructed of three parts: front-end (graph construction), back-end (graph optimization), and final map [20]. The front-end processes the sensor data to extract geometric connections, i.e., between the robot and landmarks at different points in time. The system's back end constructs a graph representing the geometric relations and uncertainties followed by final map generation. The graph structure is optimized to obtain a maximum likelihood explanation for the designated robot trajectory. The sensor data are projected into a standard coordinate frame with the known trajectory.

A short-term memory (STM) module in RTAB-Map, is another advantage of this SLAM. It spawns a node by determining the odometry pose and sensor data in visual words used for loop closure (LC), proximity detection (PD) and local occupancy grid of global map assembly (GMA).

Our local occupancy grid (LOG) is generated from the camera depth images (RGB-D), point clouds and the LiDAR. The sensor's range and FOV determine the size of a local occupancy grid. LOG generates a global occupancy grid by converting it into map referential using the poses of the map's graph. LC is conducted by identifying features kept in the node and corresponding nodes in the memory for validation.

Nodes are created at a fixed rate in milliseconds (ms), and a link retains a rigid transformation between the two nodes. Neighbor links are joined in the STM within constant nodes with odometry transformation data, and later, LC and PD links are added as detected. All the links are used as constraints for graph optimization. When a new LC or proximity link is added, graph optimization propagates the computed error to the whole graph to reduce any drifts [33,37].

RTAB-Map's memory is divided into two; working memory (WM) and long-term memory (LTM). When a new node is transferred to LTM, it is not available anymore for modules inside the WM to manage memory load. A weighting agent recognizes more significant areas than others to decide which nodes to transfer to LTM. It is accomplished using heuristics such as the longer a location is observed, the more critical it is; therefore, it should be maintained in the WM.

When adding a new node, STM assigns zero weight to the node and compares it visually (deriving a percentage of corresponding visual words) with the last node in the graph. When an LC occurs with a location in the WM, neighbor nodes can be brought back from LTM to WM for more LC and PD. As the robot moves in a previously visited area, it recognizes the ex-visited locations, incrementally develops the created map further and localizes itself [37].

Input data in RTAB-Map are used to perform feature detection and matching, motion prediction, motion estimation, local bundle adjustment (LBA), and pose update [38,39]. Feature detection and matching execute neighbor distance ratio (NNDR) [40]. NNDR test is a ratio of the first and second nearest neighbors to the feature descriptor. It uses a fast feature descriptor method known as binary robust independent elementary features (BRIEF) descriptors of the extracted features [37]. In motion estimation, the Perspective-n-Point iterative random sample consensus (RANSAC) method [41] is used to compute the transformation of the current frame according to features. The resulting transformation is refined using LBA on features of keyframes. In pose update, the output odometry is updated using transformation estimation. The covariance is computed using the median absolute deviation approach between feature correspondences.

2.3. Deep Convolutional Neural Network (Deep-CNN)

Deep networks inherently incorporate different features (low to high) and classify in an end-to-end multilayer style in that the depth enriches the features level or the number of stacked layers [42]. Convolution is a process of two functions with real-valued arguments that are defined for any functions in the following format;

$$s(t) = (x \star \omega)(t) \tag{2}$$

Considering that x and ω are defined only on integer t, we can define the discrete convolution as follows:

$$s(t) = \sum_{a=-\infty}^{\infty} x(a)\omega(t-a) \tag{3}$$

where $\omega(a)$ is a weighted function (kernel), a is the measurement period, $x(a)$ is the input and $s(t)$ referred to as the feature map [43]. The time index t is the data collected at regular intervals.

In machine learning, the input is generally a multi-dimensional data tensor, and the kernel is a multi-dimensional tensor of parameters the learning algorithm adjusts. In a CNN, each kernel member is used at every input position. For a 2D input I and 2D kernel K, the cross-correlation or convolution function is written as:

$$S(i,j) = (K \star I)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n). \tag{4}$$

Each kernel component is used in a CNN at every input position. It makes it more efficient in terms of memory requirements and statistical efficiency.

Compared to simple NNs, which involve several hidden layers, CNN consists of many layers. This characteristic qualifies it to represent highly nonlinear functions compactly. CNN involves many connections, and the architecture generally comprises diverse classes of layers, including convolution, pooling with thoroughly connected layers, and regularisation.

In the first phase of a CNN, the layers achieve several convolutions in parallel to create a set of linear activations. It is followed by manipulating linear activations by nonlinear functions known as detector phase that rectifies linear activation functions. Pooling functions are used to modify the output of the layer. Furthermore, it substitutes the net result at a specific location with an outline statistic of the adjacent results.

To learn complicated features and functions that can represent high-level generalizations, CNNs require deep architectures. Deep learning methods are used to train a convolution neural network based on large-scale datasets [44]. Deep-CNNs are a specialized kind of ANNs that use convolution in place of matrix multiplication in at least one of its layers [34,43].

Deep-CNNs, consist of a large number of neurons and multiple levels of calculating non-linearity. Divisions in the deeper layers may indirectly interact with a more significant portion of the input in a Deep-CNN. This allows the network to efficiently describe complex relationships between numerous variables by constructing interactions from simple construction unions that only describe sparse relations.

In this study, the CNN comprises a small number of reasonably complex layers, each with different stages. There is a one-to-one mapping between kernel tensors and network layers. The pooling function enables making the model approximately invariant to small input translations.

Deep learning architectures tackle semantic segmentation and object detection, a high-level task toward complete scene understanding [45]. It marks the object in an image and reserves a classified index. In the subsequent phase, localization or detection is performed for fine-grained inference, providing the classes and additional information regarding the classes' spatial location. Performing scene-related navigation via trajectory and floor detection is a complicated task for embedded devices as it demands a lot of onboard computation and power. To tackle the issue transfer learning method is employed [45,46]. Pre-trained inferences and image recognition backbones were used to convert the model into a deep-CNN for per-pixel labeling, transformation and object detection.

2.4. SROBO's Autonomous Navigation Using Deep-CNN and RTAB-Map

Object detection and semantic segmentation frameworks on frame images with confidence scores and per-pixel colored labels are considered to verify the local target map. The confidence scores higher than 85% and brown mapped areas in Figure 3 are used in Deep-CNN for identifying the target and the areas SROBO is allowed to track to achieve the target. The results from RTAB-Map with and without the 2D LiDAR scanner indicate the following frame estimation according to recommendations from the earlier structure. The confidence scores and colored maps in each frame are updated to optimize the inter-frame spatial continuity. Deep-CNN results of the current frame and estimation results from VSLAM are projected into a local map and updated to the global target map by a pose modification matrix between the new frame and former frames acquired by the point clouds for more promising performance. There has been a great progress in base convolution neural network (CNN) training algorithms, including AlexNet subset [18], GoogLeNet [47], Inception [31], MobileNet [48] and ResNet [27]. ResNet has the highest accuracy compared to the other algorithms, 96.4% [45].

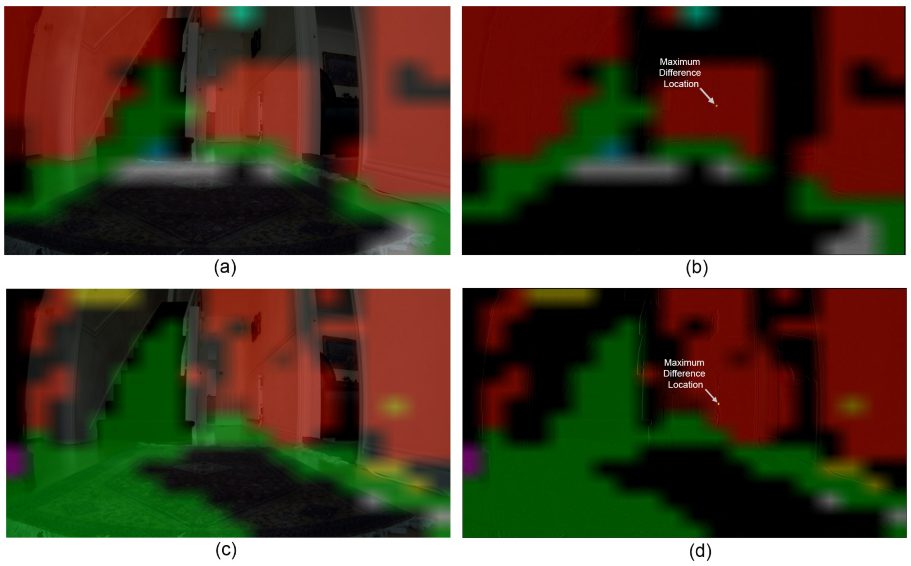

Figure 3. Real-time video segmentation with output spatial maps (**a**) ResNet trained on SUN RGB-D dataset (640 × 512). (**b**) The residual image of ResNet SUN RGB-D (640 × 512). (**c**) ResNet trained on SUN RGB-D dataset (512 × 400). (**d**) The residual image of ResNet SUN RGB-D (512 × 400).

These base CNNs are high-level training features mostly used for classification and detection with Deep-CNN architectures. The advancement in the architectures resulted in more optimal and complex system such as network pruning [49], reinforcement learning [50] and hyper-parameter optimization [51], structure connectivity [52] and optimization [53], to name a few. It makes it more challenging to implement Deep-CNN on mobile robots due to the required amount of power and processing time. The advancement of micro/single board computers and embedded systems provided low-cost solutions that allowed robots to integrate Deep-CNN and data-fusion algorithms. We will use semantic segmentation and object-detection algorithms to facilitate more robust navigation by SROBO.

3. Data Analysis and Results

MYNT EYE camera's intrinsic parameters were calibrated to transform the depth and color images using the standard calibration tool [54]. SROBO mobile robot is displayed in Figure 4. Figure 4a illustrates the conventional graph-based RTAB-Map SLAM generated in an indoor environment. The transformations between different sets of coordinate frames [55] were performed via the URDF model of SROBO. Sensor data are transmitted in their original frame across the network and stored in the original frame. Figure 4b shows a dense 3D map of an indoor scenario with SROBO's trajectory.

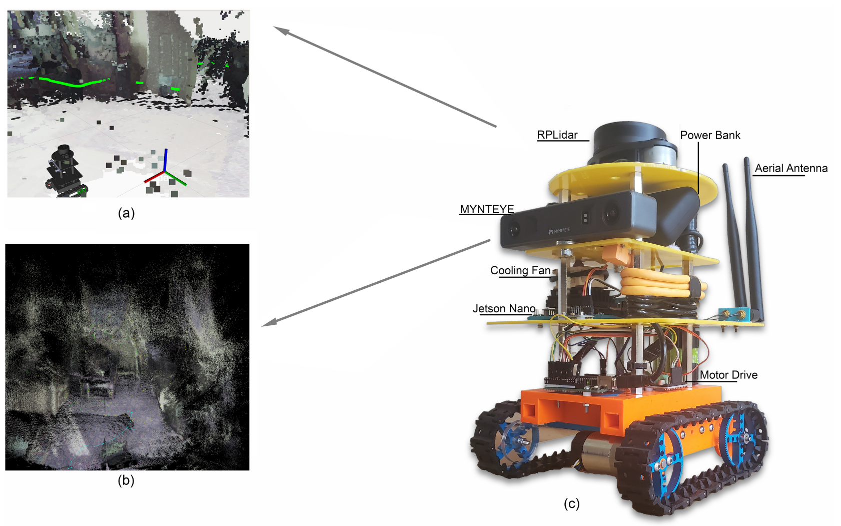

Figure 4. (**a**) 3D point clouds generated using RTAB-Map with the robot's URDF model in the scene. (**b**) 3D map generated through RGB-D camera in an indoor environment. (**c**) The ground robot (SROBO) with the mechanical and electronic parts assembled.

Semantic segmentation aims to create dense projections extrapolating labels for each pixel, labeled with its enclosing object or area class. Figure 3a,c show semantic segmented images created using SUN RGB-D (http://rgbd.cs.princeton.edu/challenge.html; accessed on 18 May 2022) database in real time. The scene labeling of per-pixel class segmentation is created for each image pixel using ResNet RGB-D dataset (512 × 400) and (640 × 512) input images. RGB-D Sun dataset with 10,335 labeled RGB-D dense images of indoor objects with 146,617 2D polygons and 58,657 3D bounding boxes [56] was used. The deep learning was performed on SROBO's primary control system with an active power supply and consumption of 1.25 W.

It is combined with ResNet-18 architecture [27] of 640 × 512 and 512 × 400 for the object segmentation. It offers the possibility of learning hierarchies of feature representations for indoor scenarios. The resNet-18 network has 152 layers of deep training architecture in that each layer's information is transferred to the next layer to be used to extract further information.

The solid pixel segmentation in Figure 3a,c were created using ResNet of 640 × 512 and 512 × 400, respectively. The images corresponding with the classified images of the standard PASCAL Visual Object Class (VOC) dataset [57], shown in Figure 3b,d. Figure 3b illustrates the residual image created by subtracting ResNet SUN RGB-D (640 × 512) from ResNet Pascal VOC (320 × 320) input images of pixel-level labeling. Figure 3c shows the residual image of subtracting ResNet SUN RGB-D (512 × 400) from ResNet Pascal VOC (320 × 320). Matlab Image processing Toolbox was used to create the residuals images and compare the performance. The location of the maximum difference is identified by a yellow star in Matlab shown in Figure 3b,d. Both classifiers showed good results for this study. However, ResNet SUN RGB-D (640 × 512) per-pixel classifier was chosen. It was used to identify the places SROBO is allowed to enter for autonomous navigation.

For object detection in this study, we used Model Zoo (https://modelzoo.co/; accessed on 18 May 2022) machine learning deployment platform to train our algorithm using Single-shot detector MobileNet and Inception (SSD-Mobilenet and SSD-Inception). SSD-Mobilenets [58] has a NN architecture tailored for mobile platforms with a significantly decreased required number of operations and memory [17]. It has a good level of accuracy and takes low-dimensional compressed inputs. It is initially expanded to the high dimension, followed by filtering with a lightweight depth-wise convolution [58]. Features are subsequently projected back to a low-dimensional representation with a linear convolution. It leaves less memory footprint as it never fully sets large intermediate tensors.

The object detection of 91 classes of pre-trained SSD-Mobilenet-v2 model was retrained on SROBO's control system using 100,000 images of indoor objects. In that, nine more categories were added to the pre-trained algorithm. The training was performed by adding five more batches and 40 training steps, which took around a hundred and 92 h to be completed. A total of 95,256 images were used for the retraining, from which 3098 used for validation and 3586 for testing, listed in Table 2. Google's open images (https://github.com/openimages/dataset; accessed on 18 May 2022) and Zoo Model dataset were employed for training.

Table 2. SSD-Mobilenet object-detection model using PyTorch, and image datasets.

Images for Training	Training	Validation	Testing
Window	0.7	0.52	0.53
Chair	0.18	0.16	0.16
Furniture	0.05	0.20	0.20
Door	0.03	0.05	0.05
Desk	0.02	0.02	0.01
Stairs	0.01	0.01	0.01
Curtain	0.01	0.02	0.02
Clock	0.00	0.01	0.01
Door handle	0.00	0.01	0.01

Semantic segmentation and object-detection algorithms were combined with RTAB-Map SLAM to facilitate autonomous navigation by SROBO. The framework enabled the robot to perform real-time image processing, pathfinding and collision avoidance. It is conducted by identifying the objects in the room and setting targets based on the objects detected in the scene. The system uses keyframes to label objects and later imports them to the 3D point cloud for optimization. Pixel-wise maps are created via semantic segmentation based on the 2D input images from the camera.

RTAB-MAP creates dense local point clouds using keyframes and the camera's depth information. It constructs the global point cloud by calculating the camera poses. Depth information is employed for precise localization. SROBO stops as soon as detecting and recognizing the target object in the scene.

Dynamic objects in the environment are identified based on the collected input images and their pixel values, comparing each image with the previous one. Object removal is conducted based on the pixel classification and boundaries using the nearest neighbor interpolation. The designed framework creates an optimized map where the SROBO can navigate autonomously. It benefits from RTAB-Map's feature-based visual odometry, loop closure detection, pose optimization and mapping in real time.

The proposed idea was compared with the original RTAB-Map using the sensors discussed in previous sections. Three indoor environments were used to validate the framework using the abovementioned algorithms. Table 3 compares the algorithms' mean average precision (Mean-AP) accuracy during real-time processing before and after implementing RTAB-Map.

Table 3. The Mean-AP accuracy of the algorithms used for segmentation and object detection in real-time processing.

Semantic Segmentation Mean-AP	ResNet (640 × 512) 74.2%	ResNet (512 × 400) 73.8%
Object Detection Mean-AP	SSD-Mobilenet 88.2%	SSD-Inception 84.3%

The comparison was conducted using the proposed system (fusion of semantic segmentation, object detection and RTAB-Map) with and without LiDAR sensor, regarding relative pose error (RPE), root mean square error (RMSE) and robustness. Three indoor scenarios were investigated, including a static scene (S_1-no moving object), a dynamic scene (S_2-one person moving) and a multi-dynamic scene (S_3-four people moving) in a house with two corridors, three rooms, a kitchen and household items. Figure 5a represents the 2D map of the environment in which we conducted the experimental analysis (S_1, S_2 and S_2). It shows the indoor layout and SROBO's two trajectory paths in the scenario S_1, with the same start and endpoint. Figure 5b displays the 3D map of the front room with point clouds and the path SROBO was following. Figure 5c,d are the objects SROBO detected in the room with the accuracy percentage of 82.2%, 88.4% and 75.8%, respectively. Figure 5e depicts the trajectory of SROBO in S_2 scenario in which the test was conducted twice sequentially. It shows the path trajectory of SROBO and the path it follows in detecting the specified pixels following semantic segmentation. It noticed the first object (the couch) with 88.4% confidence. The figure also shows that the frame in which the 3rd couch had a confidence score of 75.8%, which is lower than the accepted confidence value set (higher than 85%). Thus, SROBO resumes tracking its path (pixels) until it reaches the accepted confidence score for the detected objects in the scene and stops where it began. Figure 5f,g illustrate the 3D map and the trajectory of SROBO in the 3rd scenario (S_3). Figure 5f shows the proposed system while using LiDAR sensor.

The RPE determines the local accuracy of the trajectory over a fixed time interval that corresponds to the drift of the robot's trajectory from a sequence of camera poses. The RMSE is calculated over all time indices of the translation component of the RPE to evaluate the drift per frame. The conventional RTAB-Map and proposed system's RMSE, mean and RPE values are illustrated in Table 4. The results showed that the proposed approach is 35% more robust (less average error) than the conventional RTAB-Map in all three scenarios. The RMSE values for the traditional RTAB-Map and the proposed system in S_3 scenario using LiDAR were 13% and 9%, and without LiDAR, it was 15% and 10%, respectively.

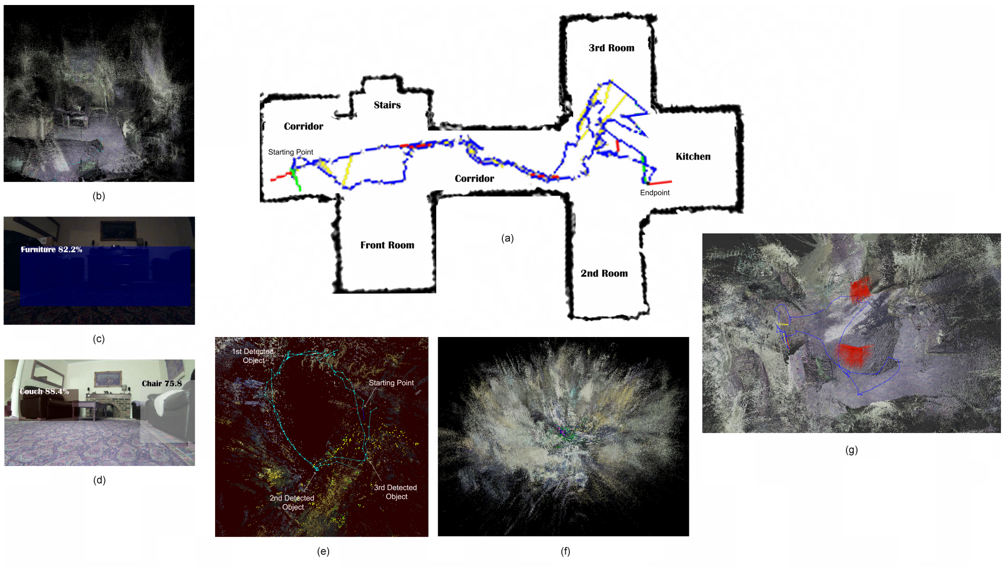

Figure 5. The proposed system's 2D and 3D maps and SROBO's path tracking.

Table 4. Error estimation values of conventional RTAB-Map and the proposed system.

Error Estimation	**Conventional RTAB-Map**			**Proposed System**		
	S_3	**S_2**	**S_1**	**S_3**	**S_2**	**S_1**
RMSE	0.13	0.09	0.08	0.09	0.07	0.04
Mean	0.11	0.08	0.08	0.09	0.06	0.05
RPE Translation	0.05	0.04	0.05	0.04	0.04	0.03
RPE Rotation	0.08	0.06	0.07	0.06	0.04	0.04

The results collected from SROBO while navigating autonomously illustrated that the robot could safely navigate the environment—avoiding dynamic and static obstacles. It also reached the targets set for it. Although occasionally faced system disconnections problems due to the required computational power for real-time image processing. The other issue was that the MYNTEYE camera requires high power to function.

4. Conclusions

Our results demonstrated that adapting contemporary classification networks into fully convolutional networks was practical. It transfers the learned representations by fine-tuning to the segmentation task. Shirked architecture merges semantic information from a deep, rough layer with appearance data from a fine surface layer to produce accurate and detailed segmentations [59]. This study developed a mapping system that enables SROBO to recognize the locations it is permitted to perceive. It detects the objects and follows the trail of the specified entities in the scene. It plans routes to a previously visited site, localizes itself on the map, and stops as soon as it detects an object confined in the location. It allows the SLAM to initialize a new map based on its allusions and transform it into the previously created map while a once-seen area is discovered. It avoids remapping the whole environment when only a new area should be explored and incorporated. This approach is considered in this study when dealing with semantic segmentation in images described in the following section. Furthermore, the position estimation error could be reduced by around 68% when combining vision-based position estimates with other sensor fusions [60]. RTAB-Map's localization and mapping improved by enforcing Deep-CNN, which furnished the robot with more awareness about the environment in which it functioned. The results showed that SROBO gained more insight into the environment into which it is exposed. It uses semantic segmentation to recognize the path it is allowed to track and follow the set constraint. Retraining the object-detection algorithm improved its recognition capability by 62%.

The experimental comparisons regarding pose accuracy, and swiftness, are conducted using RMSE. The results showed that blending RTAB-Map with Deep-CNN has improved performance by 20% to 48% compared to the conventional RTAB-Map. SROBO's target detection gradually enhanced as the algorithm learned and noticed the target object in its frame. The RMSE value increased in the S_3 scenario compared to S_1 was due to the existence of dynamic obstacles. However, there was no significant change in RPE values in different scenarios. During the experimental study, we faced some issues related to the robots' wheels. Friction affected the robot turning to its sides, where wheels came out of the rubber tracks. In future study, we will replace the wheels with an alternative solution for better performance. Future studies will also assess robot motion performance in a closed-loop setting using a VICON motion capture system to validate the proposed framework further.

Author Contributions: Conceptualization, S.S.E.; methodology, S.S.E. and H.S.; software, S.S.E.; validation, S.S.E., H.S., A.S. and M.G.; formal analysis, S.S.E. and H.S.; investigation, S.S.E., H.S., A.S. and M.G.; resources, S.S.E.; data curation, S.S.E. and M.G.; writing—original draft preparation, S.S.E., H.S., A.S. and M.G.; writing—review and editing, S.S.E., H.S., A.S. and M.G.; visualization, S.S.E.; supervision, S.S.E. and H.S.; project administration, S.S.E., A.S. and M.G.; funding acquisition, S.S.E. and H.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

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

References

1. Crook, C. *Computers and the Collaborative Experience of Learning*; Routledge: London, UK, 2018.
2. Levesque, H.; Lakemeyer, G. Cognitive robotics. *Found. Artif. Intell.* **2008**, *3*, 869–886.
3. Samani, H. *Cognitive Robotics*; CRC Press: Boca Raton, FL, USA, 2015.
4. Mur-Artal, R.; Montiel, J.M.M.; Tardos, J.D. ORB-SLAM: A versatile and accurate monocular SLAM system. *IEEE Trans. Robot.* **2015**, *31*, 1147–1163. [CrossRef]
5. Mur-Artal, R.; Tardós, J.D. ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras. *IEEE Trans. Robot.* **2017**, *33*, 1255–1262. [CrossRef]
6. Qin, T.; Li, P.; Shen, S. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Trans. Robot.* **2018**, *34*, 1004–1020. [CrossRef]
7. Fontan, A.; Civera, J.; Triebel, R. Information-Driven Direct RGB-D Odometry. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 4929–4937.
8. LeCun, Y.; Bengio, Y. Convolutional networks for images, speech, and time series. *Handb. Brain Theory Neural Netw.* **1995**, *3361*, 1995.
9. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
10. Bengio, Y. Learning deep architectures for AI. *Found. Trends® Mach. Learn.* **2009**, *2*, 1–127. [CrossRef]
11. Irsoy, O.; Cardie, C. Deep recursive neural networks for compositionality in language. *Adv. Neural Inf. Process. Syst.* **2014**, *27*. [CrossRef]
12. Varsamou, M.; Antonakopoulos, T. Classification using Discriminative Restricted Boltzmann Machines on Spark. In Proceedings of the 2019 International Conference on Software, Telecommunications and Computer Networks (SoftCOM), Split, Croatia, 19–21 September 2019; pp. 1–6.
13. Liu, X.; Deng, Z.; Yang, Y. Recent progress in semantic image segmentation. *Artif. Intell. Rev.* **2019**, *52*, 1089–1106. [CrossRef]
14. Jarrett, K.; Kavukcuoglu, K.; Ranzato, M.; LeCun, Y. What is the best multi-stage architecture for object recognition? In Proceedings of the 2009 IEEE 12th International Conference on Computer Vision, Kyoto, Japan, 29 September–2 October 2009; pp. 2146–2153.
15. LeCun, Y.; Kavukcuoglu, K.; Farabet, C. Convolutional networks and applications in vision. In Proceedings of the 2010 IEEE International Symposium on Circuits and Systems, Paris, France, 30 May–2 June 2010; pp. 253–256.
16. Najafabadi, M.M.; Villanustre, F.; Khoshgoftaar, T.M.; Seliya, N.; Wald, R.; Muharemagic, E. Deep learning applications and challenges in big data analytics. *J. Big Data* **2015**, *2*, 1–21. [CrossRef]
17. Khan, A.; Sohail, A.; Zahoora, U.; Qureshi, A.S. A survey of the recent architectures of deep convolutional neural networks. *Artif. Intell. Rev.* **2020**, *53*, 5455–5516. [CrossRef]
18. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. *Adv. Neural Inf. Process. Syst.* **2012**, *25*. [CrossRef]

19. Yousif, K.; Bab-Hadiashar, A.; Hoseinnezhad, R. An overview to visual odometry and visual SLAM: Applications to mobile robotics. *Intell. Ind. Syst.* **2015**, *1*, 289–311. [CrossRef]
20. Endres, F.; Hess, J.; Sturm, J.; Cremers, D.; Burgard, W. 3-D mapping with an RGB-D camera. *IEEE Trans. Robot.* **2013**, *30*, 177–187. [CrossRef]
21. Kohlbrecher, S.; Von Stryk, O.; Meyer, J.; Klingauf, U. A flexible and scalable SLAM system with full 3D motion estimation. In Proceedings of the 2011 IEEE International Symposium on Safety, Security, and Rescue Robotics, Kyoto, Japan, 1–5 November 2011; pp. 155–160.
22. Scaramuzza, D.; Fraundorfer, F. Visual odometry [tutorial]. *IEEE Robot. Autom. Mag.* **2011**, *18*, 80–92. [CrossRef]
23. Hackett, J.K.; Shah, M. Multi-sensor fusion: A perspective. In Proceedings of the IEEE International Conference on Robotics and Automation, Cincinnati, OH, USA, 13–18 May 1990; pp. 1324–1330.
24. Taketomi, T.; Uchiyama, H.; Ikeda, S. Visual SLAM algorithms: A survey from 2010 to 2016. *IPSJ Trans. Comput. Vis. Appl.* **2017**, *9*, 16. [CrossRef]
25. Chekhlov, D.; Gee, A.P.; Calway, A.; Mayol-Cuevas, W. Ninja on a plane: Automatic discovery of physical planes for augmented reality using visual slam. In Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, Nara, Japan, 13–16 November 2007; pp. 153–156.
26. Lin, G.; Milan, A.; Shen, C.; Reid, I. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 1925–1934.
27. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
28. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. Ssd: Single shot multibox detector. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 11–14 October 2016; pp. 21–37.
29. Li, Y.; Huang, H.; Xie, Q.; Yao, L.; Chen, Q. Research on a surface defect detection algorithm based on MobileNet-SSD. *Appl. Sci.* **2018**, *8*, 1678. [CrossRef]
30. Ning, C.; Zhou, H.; Song, Y.; Tang, J. Inception single shot multibox detector for object detection. In Proceedings of the 2017 IEEE International Conference on Multimedia & Expo Workshops (ICMEW), Hong Kong, China, 10–14 July 2017; pp. 549–554.
31. Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; Wojna, Z. Rethinking the inception architecture for computer vision. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 2818–2826.
32. Ribeiro, M.I. Kalman and extended kalman filters: Concept, derivation and properties. *Inst. Syst. Robot.* **2004**, *43*, 46.
33. 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]
34. Habib, G.; Qureshi, S. Optimization and acceleration of convolutional neural networks: A survey. *J. King Saud Univ.-Comput. Inf. Sci.* **2020**, *34*, 4244–4268. [CrossRef]
35. Grisetti, G.; Kümmerle, R.; Stachniss, C.; Burgard, W. A tutorial on graph-based SLAM. *IEEE Intell. Transp. Syst. Mag.* **2010**, *2*, 31–43. [CrossRef]
36. Lu, F.; Milios, E. Globally consistent range scan alignment for environment mapping. *Auton. Robot.* **1997**, *4*, 333–349. [CrossRef]
37. Labbé, M.; Michaud, F. Long-term online multi-session graph-based SPLAM with memory management. *Auton. Robot.* **2018**, *42*, 1133–1150. [CrossRef]
38. Shi, J. Good features to track. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 21–23 June 1994; pp. 593–600.
39. Zhang, G.; Vela, P.A. Good features to track for visual slam. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 1373–1382.
40. Muja, M.; Lowe, D.G. Fast approximate nearest neighbors with automatic algorithm configuration. *VISAPP* **2009**, *2*, 2.
41. 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]
42. Zeiler, M.D.; Fergus, R. Visualizing and understanding convolutional networks. In Proceedings of the European Conference on Computer Vision, Zurich, Switzerland, 6–12 September 2014; pp. 818–833.
43. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
44. Wofk, D.; Ma, F.; Yang, T.J.; Karaman, S.; Sze, V. Fastdepth: Fast monocular depth estimation on embedded systems. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 6101–6108.
45. Garcia-Garcia, A.; Orts-Escolano, S.; Oprea, S.; Villena-Martinez, V.; Garcia-Rodriguez, J. A review on deep learning techniques applied to semantic segmentation. *arXiv* **2017**, arXiv:1704.06857.
46. Tan, C.; Sun, F.; Kong, T.; Zhang, W.; Yang, C.; Liu, C. A survey on deep transfer learning. In Proceedings of the International Conference on Artificial Neural Networks, Rhodes, Greece, 4–7 October 2018; pp. 270–279.
47. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 1–9.
48. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv* **2017**, arXiv:1704.04861.

49. Guo, Y.; Yao, A.; Chen, Y. Dynamic network surgery for efficient dnns. *Adv. Neural Inf. Process. Syst.* **2016**, *29*. [CrossRef]
50. Zoph, B.; Le, Q.V. Neural architecture search with reinforcement learning. *arXiv* **2016**, arXiv:1611.01578.
51. Bergstra, J.; Bengio, Y. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.* **2012**, *13*, 281–305.
52. Zhang, X.; Zhou, X.; Lin, M.; Sun, J. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 6848–6856.
53. Real, E.; Moore, S.; Selle, A.; Saxena, S.; Suematsu, Y.L.; Tan, J.; Le, Q.V.; Kurakin, A. Large-scale evolution of image classifiers. In Proceedings of the International Conference on Machine Learning, PMLR, Sydney, NSW, Australia, 6–11 August 2017; pp. 2902–2911.
54. Bouguet, J.Y. Camera Calibration Toolbox for Matlab. 2004. Available online: http://www.vision.caltech.edu/bouguetj/calib_doc/index.html (accessed on 18 May 2022).
55. Foote, T. tf: The transform library. In Proceedings of the 2013 IEEE International Conference on Technologies for Practical Robot Applications (TePRA), Woburn, MA, USA, 22–23 April 2013; pp. 1–6. doi: 10.1109/TePRA.2013.6556373. [CrossRef]
56. Song, S.; Lichtenberg, S.P.; Xiao, J. Sun rgb-d: A rgb-d scene understanding benchmark suite. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 567–576.
57. Everingham, M.; Eslami, S.; Van Gool, L.; Williams, C.K.; Winn, J.; Zisserman, A. The pascal visual object classes challenge: A retrospective. *Int. J. Comput. Vis.* **2015**, *111*, 98–136. [CrossRef]
58. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.C. Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 4510–4520.
59. Long, J.; Shelhamer, E.; Darrell, T. Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 3431–3440.
60. Montijano, E.; Cristofalo, E.; Zhou, D.; Schwager, M.; Saguees, C. Vision-based distributed formation control without an external positioning system. *IEEE Trans. Robot.* **2016**, *32*, 339–351. [CrossRef]