Hindawi
Journal of Robotics
Volume 2018, Article ID 4942034, 13 pages
https://doi.org/10.1155/2018/4942034

Research Article

MapFuse: Complete and Realistic 3D Modelling

Michiel Aernouts, Ben Bellekens, and Maarten Weyn

IMEC, IDLab, Faculty of Applied Engineering, University of Antwerp, Groenenborgerlaan 171, 2000 Antwerp, Belgium

Correspondence should be addressed to Michiel Aernouts; michiel.aernouts@uantwerpen.be

Received 18 August 2017; Accepted 20 December 2017; Published 19 February 2018

Academic Editor: L. Fortuna

Copyright © 2018 Michiel Aernouts et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Validating a 3D indoor radio propagation model that simulates the signal strength of a wireless device can be a challenging task due to an incomplete or a faulty environment model. In this paper, we present a novel method to simulate a complete indoor environment that can be used for evaluating a radio propagation model efficiently. In order to obtain a realistic and robust model of the full environment, the OctoMap framework is applied. The system combines the result of a SLAM algorithm and secondly a simple initial model of the same environment in a probabilistic way. Due to this approach, sensor noise and accumulated registration errors are minimised. Furthermore, in this article, we evaluate the merging approach with two SLAM algorithms, three vision sensors, and four datasets, of which one is publicly available. As a result, we have created a complete volumetric model by merging an initial model of the environment with the result of RGB-D SLAM based on real sensor measurements.

1. Introduction

Due to the recently increased demands on location-based services, localisation of wireless devices based on received signal strength has become a significant research topic [1]. Within this research, simulation based approaches are used to evaluate localisation algorithms. These approaches use wave propagation models, which can predict the attenuation of a signal as it propagates through space. Moreover, such a propagation model will improve the localisation accuracy and precision [2].

Many different types of propagation models have been defined [3]. Previous research focussed on the validation of an indoor ray launching propagation model for sub-GHz frequencies [4–7]. In [4], Bellekens et al. utilised a sub-1 GHz measurement device that was programmed with the LPWAN (Low Power Wide Area Network) standard DASH7 [8]. A Pioneer 3DX robot was used to collect RF measurements from 6 stationary transmitters, laser range measurements, and the wheel odometry. Next, a laser based online SLAM algorithm (Gmapping) is being used to obtain a 2D map of the environment as well as the robots' estimated trajectory. By combining map and trajectory information with the RF measurements, the researchers were able to validate their 2D propagation simulation with an accuracy of 2.8 dB and a precision of 7.8 dB [4].

Evaluating a 3D propagation model based on a real environment model makes it possible to evaluate signal strengths at different heights as well as the specific multipath introduced by the environments. This will enable the evaluation of localisation algorithms that are highly influenced by multipath effects, that is, Angle of Arrival localisation. Therefore, a UAV (Unmanned Aerial Vehicle) can be utilised instead of a driving robot to receive first the RF measurements and secondly to capture the environment in 3D. Due to noisy depth image measurements and accumulated registration errors, the result of a 3D-SLAM does not cover the entire environment where ceilings, floor, walls, and objects are included.

In this paper, we aim to create a complete model of an environment of which we have foreknowledge. In order to benefit the completeness of our result, we used a probabilistic, volumetric mapping approach called OctoMap [9]. In contrast to using point clouds, OctoMap allows us to render a model which contains information about occupied spaces, free spaces, and unknown spaces. Also, we benefit from the probabilistic nature of an OctoMap, as it allows us to update

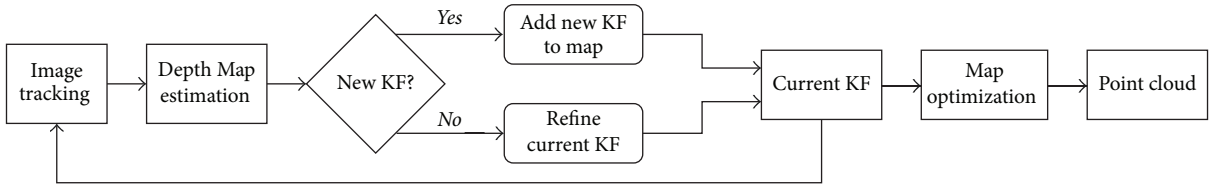


Figure 1: This figure illustrates a simplistic schematic of the LSD SLAM algorithm.

an initial guess model of the environment with real sensor measurements.

Current research about completing an environment model based on the surface of a captured point cloud has rather been limited [11, 12]. Breckon and Fisher presented a method to complete partially observed objects by deriving plausible data from known portions of the object. However, this method is time-consuming and involves complex calculations. Furthermore, the completions are not an accurate reconstruction of reality, as they are only meant to be visually acceptable for the viewer [11]. In [12], laser range data is used to create a 2D floor plan, which can be extruded to a 2.5D model. By aligning this simplified model with a complex octree of the environment, the researchers were able to build a final model which includes previously hidden surfaces. However, automatic generation of a 2.5D model is challenging when a flawed dataset is used as input. Also, this approach assumes floors and ceilings to be horizontal and to have fixed heights. Moreover, model merging is not done in a probabilistic fashion.

The main contributions of MapFuse regarding the state of the art are two different approaches for merging an initial environment model with real measurements. First, the initial model can be merged iteratively with the final SLAM result. With this technique, the accuracy can be regulated by changing the amount and sequence of both models. Secondly, an online merging process, which updates the initial model while SLAM is processing, can be applied. Both methods use OctoMap to probabilistically build a complete volumetric model.

In order to create and evaluate MapFuse, we have compared three different camera types in a simulated environment: a basic monocular camera, a wide field-of-view camera, and a depth-sense camera. With these cameras, datasets were recorded to be used as input for a visual SLAM algorithm such as Large-Scale Direct SLAM (LSD SLAM) or feature-based RGB-D SLAM. Afterwards, the simulated SLAM results were validated in four real environments.

The remainder of the paper is structured as follows: Section 2 lists techniques that were tested and implemented in our system. Section 3 describes the three main blocks of which our method consists. In Section 4, results of our approach are discussed. Finally, Section 5 concludes these results.

2. Related Work

For many years, researchers have come up with solutions to the SLAM problem. This problem occurs when a robot is placed in an unknown environment. Without a map or information about its own location, the robot has to be able to build a map of the environment while determining its own position at the same time [13]. When a SLAM algorithm is implemented, a robots' location can be determined in a probabilistic fashion by combining sensor measurements with odometry information. Bayes filters such as Kalman filters are well suited for this purpose, as they predict the robots' state based on its previous state and received motion commands [14]. Afterwards, the predicted state is corrected with the obtained sensor measurements [15].

Laser range sensors are commonly used for building 2D and 3D maps [4, 16, 17]. However, cameras have become increasingly more popular for building 3D maps. This is due to their low cost compared to laser range sensors and the possibility of obtaining 3D data when using depth-sense cameras or stereo cameras [18]. Also, compared to laser range sensors, depth-sense cameras allow us to record an entire depth image that contains a collection of points that are registered with an RGB image. For this reason, we will research visual SLAM (VSLAM) algorithms to map an indoor environment. VSLAM can be subdivided into two subclasses: *Feature-Based Methods* and *Direct Methods*. For our research, we evaluated the two merging processes with both of these VSLAM methods by comparing LSD SLAM to RGB-D SLAM.

With OctoMap, volumetric models can be created by calculating the occupancy probability of all nodes in the map. As information about occupied spaces, free spaces, and unknown spaces is obtained, this mapping approach can be used for autonomous exploration of various environments.

2.1. Large-Scale Direct SLAM. LSD SLAM is a direct VSLAM algorithm that can be implemented with monocular cameras as well as stereo cameras [19, 20]. Direct VSLAM methods profit from pixel intensity values of the entire image. Due to the fact that the complete image is used as input data, these methods result in a high accuracy and provide a lot of information about the geometry of the environment [19, 21].

Figure 1 illustrates the LSD SLAM workflow in a basic schematic. Firstly, the tracking component estimates the rigid body pose with $se(3)$ [22]. Secondly, the current KF will be refined or a new KF will be created from the most recent tracked image, depending on the estimated transformation between two images. Finally, the map optimisation component inserts keyframes into the global map when they are replaced by a new reference frame.

2.2. RGB-D SLAM. Contrary to LSD SLAM, RGB-D SLAM is a feature-based VSLAM algorithm. Feature-based VSLAM

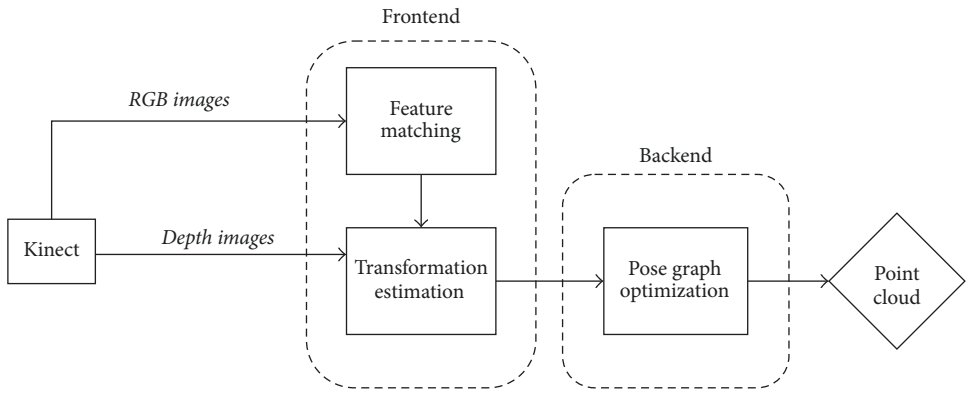


Figure 2: This figure illustrates a simplistic schematic of the RGB-D SLAM algorithm.

collects feature observations from the camera image and then compares these features to the previous camera image. Numerous feature detectors can be implemented for this purpose, for example, Oriented FAST and Rotated BRIEF (ORB), Scale Invariant Feature Transform (SIFT), and Speeded Up Robust Features (SURF) [23, 24].

In the RGB-D SLAM frontend, feature locations are visualised in three dimensions by overlaying the RGB image with its respective depth image. With $se(3)$, a transformation estimation between two subsequent frames can be calculated [22]. However, this estimation cannot be considered accurate due to false positives in feature detection and the fact that RGB images can be inconsistent with depth images. Therefore, a Random Sample Consensus (RANSAC) algorithm is applied to abolish this effect [25].

In the SLAM backend, frames are added as a node to the pose graph. When a frame matches one of the previous frames, it will be connected to the existing pose graph of the matching frame. Otherwise, the new frame is connected to the previous node in the pose graph. After obtaining spatial relations via the SLAM frontend, RGB-D SLAM implements the g^{2}o framework for pose graph optimisation [26]. With pose graph optimisation, a trajectory is estimated using the robots' relative pose measurements, that is, the robots' current and previous poses. Figure 2 provides a simplified overview of the RGB-D SLAM workflow.

2.3. OctoMap. OctoMap is a volumetric mapping framework based on an octree data structure and probabilistic occupancy estimation [9]. When it comes to mapping arbitrary 3D environments, OctoMap has numerous advantages over other mapping approaches. Octrees are highly memory efficient; they consist of an octant which can be divided into eight leaf nodes. Subsequently, these leaf nodes can be seen as new octants, which in their turn can be divided into leaf nodes again. The desired resolution of the 3D model is determined by the depth of the octree. For example, large adjacent volumes can be represented by a single leaf node to save memory. The octree data structure is shown in Figure 3.

Figure 3 also displays that an octant node n can have different states: free, occupied, or unknown. This state can be derived from the calculated probability according to

$$P(n \mid z_{1:t}) = \left[1 + \frac{1 - P(n \mid z_t)}{P(n \mid z_t)} \frac{1 - P(n \mid z_{1:t-1})}{P(n \mid z_{1:t-1})} \frac{P(n)}{1 - P(n)}\right]^{-1}. \quad (1)$$

Equation (1) states that the probability of whether a node is occupied or free is determined by the current sensor measurement z_t, the previous estimate $P(n \mid z_{1:t-1})$, and a prior probability $P(n)$, which is assumed to equal 0.5. In order to rewrite the equation, we will apply the log-odds notation:

$$L(n) = \log\left[\frac{P(n)}{1 - P(n)}\right]. \quad (2)$$

Thus, (1) can be rewritten as

$$L(n \mid z_{1:t}) = L(n \mid z_{1:t-1}) + L(n \mid z_t). \quad (3)$$

With log-odds, probabilities of 0% to 100% are mapped to $-\infty$ dB and $+\infty$ dB. A main advantage of this notation is that small differences at the outer edges of the range have the strongest influence on the probability. For example, 50.00% and 50.01% are mapped to 0 dB and 0.0017 dB, while 99.98% and 99.99% are mapped to 37 dB and 40 dB. As (3) makes use of additions instead of multiplications, the probability of a leaf node can be updated faster than in (1). Faulty measurements due to noise or reflections are cancelled out by the update formula.

Furthermore, the map can be extended at any time when the robot explores new unknown areas. As the OctoMap holds information about unmapped space, the robot knows which areas it has to avoid for safety reasons, or which areas are yet to be explored.

3. System Approach

In pursuance of building a complete model, we propose a system that consists of three main steps. Figure 4 displays a basic schematic that represents the MapFuse workflow. Firstly, visual information from the camera is recorded into a dataset, and an initial guess is modelled. Secondly, the dataset that was gathered in the first step is used as input for a SLAM algorithm. Finally, OctoMap is used to merge the SLAM point cloud with the initial guess.

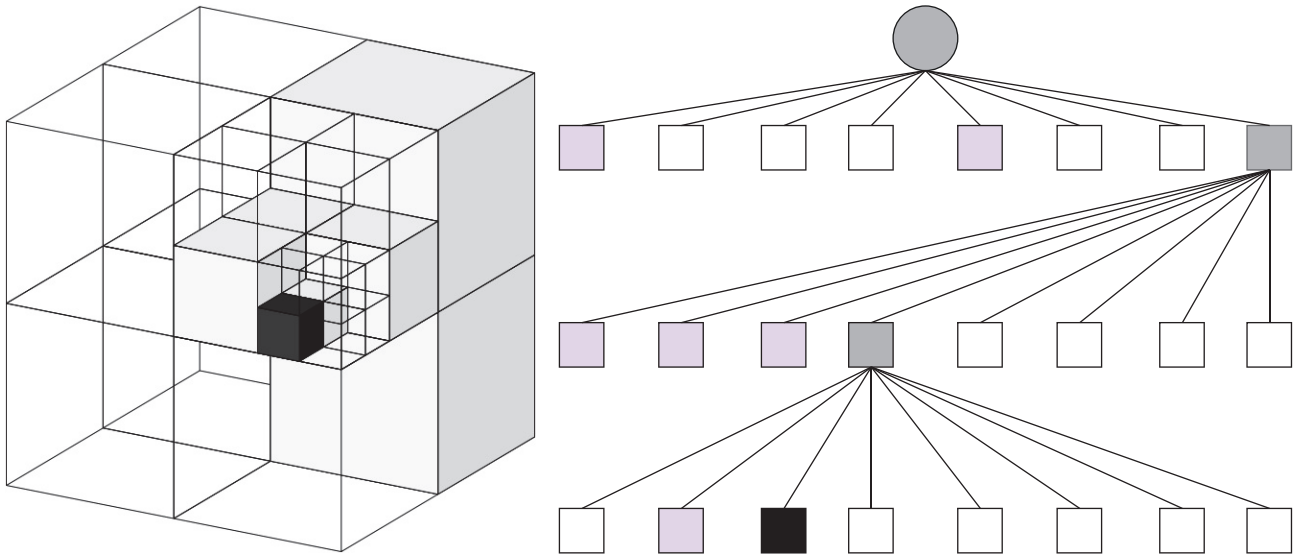


Figure 3: A visual representation of the octree data structure [9]. The black leaf node represents occupied space, whereas grey nodes indicate free space. Unknown space is marked by transparent nodes.

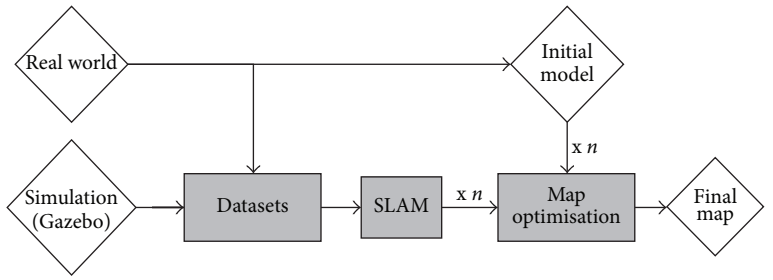
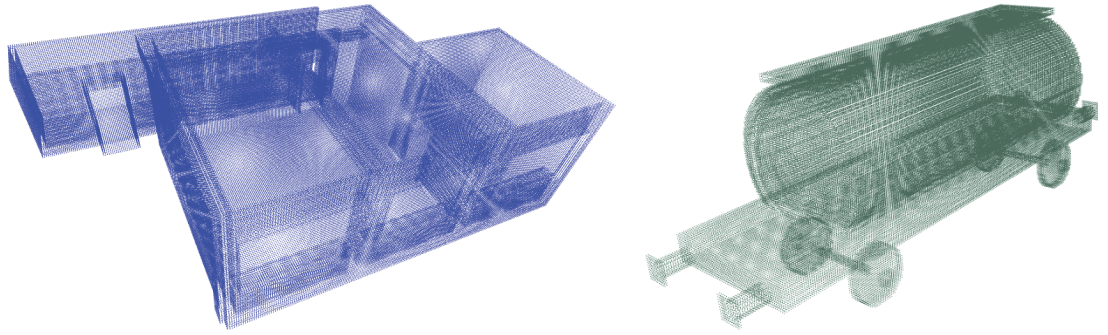


Figure 4: With MapFuse, a dataset that is recorded in a simulated or real environment is used as input for a SLAM algorithm. In the map optimisation component, the resulting SLAM point cloud is merged with an initial model which was modelled based on exact dimensions of the environment. The final MapFuse result is a complete volumetric model of the environment.



(a) (b)

Figure 5: An initial guess point cloud will be used so as to complete the unfinished SLAM point cloud.

3.1. Dataset. Since we have foreknowledge of the environment, an initial guess model can be created. In order to do so, we resort to OpenSCAD. This 3D modelling software allows us to build a model that matches the exact dimensions of the real environment. The most important goal for such a model is to provide an incomplete SLAM map with complementary information about the environment. The amount of detail that has to be included in the initial guess mostly depends on the quality of the dataset. With an admissible dataset, SLAM will provide a lot of details, so that the initial guess can be limited to a bounding box of the environment.

Figure 5 displays two different initial guess models. In Figure 5(a), a bounding box of an indoor environment with doors and windows is shown. For visualisation purposes,

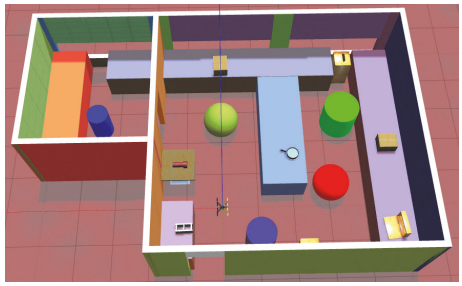


Figure 6: With Gazebo, we are able to simulate quadcopter flight and sensor measurements in order to gather an ideal dataset. This dataset was used to evaluate which SLAM algorithm was most suitable for our approach.

ceilings were not included in this model. In Figure 5(b), we created a simplified model of an industrial train cart.

An additional benefit is that the model can be imported in a simulator, which induces numerous advantages. Above all, simulation is time-saving and abates the risk of crashing the quadcopter. It has allowed us to experiment with multiple camera types and algorithms in order to design an optimal work flow. Therefore, our system was assessed by employing Gazebo. This software allows us to spawn a quadcopter as well as our initial guess model. However, our model needs to be extended with colour and objects, as VSLAM algorithms require visual features to build a map of the environment. In Figure 6, an example of the simulated environment is shown.

As our quadcopter employs a ROS-based operating system, trajectories can be scripted and tested in Gazebo before real world tests are conducted. By doing so, the quadcopter will always follow the same trajectory. Hence, a better comparison between camera types can be made. Figure 7 shows which cameras we have evaluated in our experiments.

However, scripting a trajectory requires some form of ground truth such as GPS. Since we are operating the quadcopter indoors, we cannot rely on GPS communication. An accurate indoor ground truth pose estimation system would have to be implemented in order to use these trajectory scripts in reality, which is an expensive and time-consuming process [27]. Therefore, our real world implementation of the system will control the quadcopter via a remote controller.

After setting up the simulator with an environment, a flying quadcopter, and a camera, datasets can be recorded via ROS topics. Such a topic can hold camera images, odometry, or information about the relationship between all coordinate frames. With the latter, it is possible to deduce the camera pose relative to the quadcopter. Consecutively, we can deduce the initial pose of the quadcopter relative to the map. A VSLAM algorithm combines all this information with visual odometry of the camera, with the purpose of obtaining a more accurate trajectory estimate.

Datasets that were recorded in Gazebo were used as input for several VSLAM algorithms in order to determine which camera and which algorithm are most suitable for our method. In order to validate our simulation results, we implemented the same process to gather datasets in real environments.

3.2. SLAM. The second step adopts the dataset as input for a ROS implementation of a visual SLAM algorithm. By playing back the datasets, camera images will be published to ROS topics required by the SLAM algorithm. The playback speed of the dataset can be slowed down, so that the applied SLAM algorithm has more time to detect and process visual features. We have experimented with LSD SLAM as well as RGB-D SLAM in order to analyse which of these algorithms is most suitable for our method. As discussed in Section 4, parameters for both algorithms were changed empirically until we found an optimal result.

3.3. Map Optimisation. The final step in our system combines the initial guess point cloud of Figure 5 and the SLAM point cloud into a single OctoMap. In order to obtain an accurate OctoMap, these point clouds have to be aligned as well as possible. Point cloud alignment is achieved by empirically transforming the initial guess coordinate frame to the SLAM coordinate frame. After the transformation is regulated correctly, both clouds are sent into an OctoMap server node. This way, the initial guess model will be updated with real measurements from a camera. Because SLAM is not able to map all elements in the environment, for example, ceilings or walls that are blocked by furniture, our initial guess model will provide the OctoMap server with information about these missing elements and updates the occupancy probability accordingly. A basic schematic of our map optimisation component is shown in Figure 8.

Two options can be considered to merge point clouds. On the one hand, the final SLAM result can be merged iteratively with the initial guess. Occupancy estimations can be altered by changing the initial OctoMap occupancy probability or by inserting both point clouds multiple times. When using this method, a balance between map completeness and detail has to be mediated. On the other hand, the merging process can be affected while SLAM is building a point cloud. After sending the initial guess point cloud to the OctoMap server a single time, node probabilities will be updated iteratively as the SLAM algorithm refines its point cloud based on current and previous measurements. Due to the fact that multiple measurements are taken into account, the occupancy probability will be more conclusive. Figure 9 demonstrates the difference between both merging methods.

4. Results

MapFuse was evaluated using four different datasets, three of which we have recorded ourselves. Dataset 4 is publicly available via the RGB-D benchmark dataset [27]. Dataset 1 was recorded with a wide field-of-view camera; all other datasets were recorded with a Microsoft Kinect. For all datasets, we have built an initial guess model with OpenSCAD.

(i) *Dataset 1*: room V329 at the University of Antwerp. This meeting room contains many empty tables and closets

(a) Logitech C615 (b) Genius WideCam (c) Microsoft Kinect

Figure 7: In both simulation and reality, we conducted tests with a common webcamera (a), a wide field-of-view webcamera (b), and a Microsoft Kinect (c).

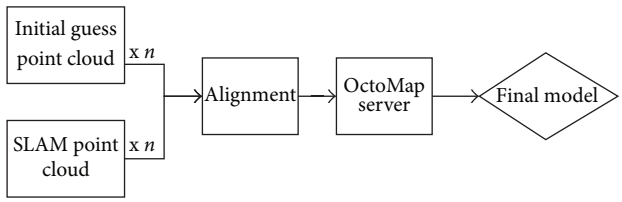


Figure 8: Basic schematic of the optimisation block of our system.

(ii) *Dataset 2*: room V315 at the University of Antwerp (6.67 m × 7.02 m × 3.77 m). The adjacent room V317 was also included in this dataset (4.12 m × 3.42 m × 3.77 m). Both rooms contain desks and closets with a high amount of clutter

(iii) *Dataset 3*: an industrial tank car located in a small hangar at the port of Antwerp (9.2 m × 2.45 m × 3.75)

(iv) *Dataset 4*: the "freiburg1_room" dataset provided by Sturm et al. This dataset is recorded in a small office environment.

These datasets were used as input for the two VSLAM algorithms that were discussed in Section 2: LSD SLAM and RGB-D SLAM.

Finally, the MapFuse optimisation step of Section 3.3 was evaluated by applying iterative merging and online merging on the initial model and the SLAM output.

All tests were performed on a Dell Inspiron 15 5548 laptop, which is provided with an Intel i7 5500U 2.4 GHz, 8 GB RAM, and an AMD Radeon R7 M265 graphics card. Ubuntu 14.04.5 LTS was used as the operating system.

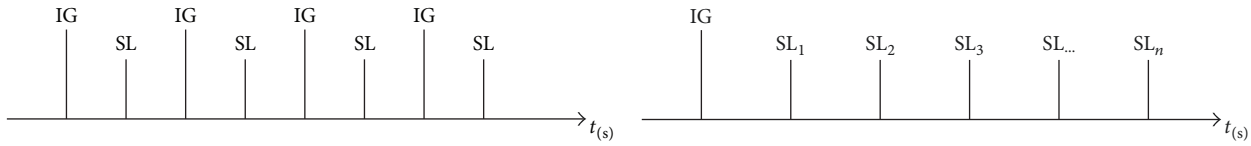
4.1. LSD SLAM. Our first tests were conducted with LSD SLAM. We connected a wide field-of-view web camera (120°) to a laptop and downsampled the image to a 640 × 480 resolution in order to evaluate the algorithm. With this setup, we recorded dataset 1 by walking around the room in a sideways motion. This was necessary to ensure sufficient camera translation, which is required for LSD SLAM. In order to optimise the map with loop closures, the same trajectory was repeated multiple times.

When running LSD SLAM, a few important parameters have to be reckoned with. First, a pixel noise threshold is set to handle faulty sensor measurements. Second, the amount of keyframes to be saved is defined. This amount is based on the image overlap and the distance between two consecutive keyframes. A large number of keyframes will result in an accurate trajectory, but also induces more noise in the map.

Figure 10(b) illustrates the point cloud and trajectory estimate of LSD SLAM. Empirical comparison with the real environment of Figure 10(a) leads us to conclude that LSD SLAM produces an accurate trajectory estimate. However, the point cloud holds a high amount of noise. As our approach requires a dense and detailed SLAM point cloud with little noise, we will not pursue LSD SLAM in our research any further.

4.2. RGB-D SLAM. For our tests with RGB-D SLAM, we mounted a Kinect camera to an Erle-Copter as shown in Figure 11 [10]. The Kinect was slightly tilted downwards to capture as many visual features as possible. The camera was connected to a laptop which ran the camera driver correctly. Contrary to LSD SLAM, RGB-D SLAM can handle camera translation as well as camera rotation. We found that the best trajectory for this algorithm is to rotate the camera 360 degrees at the centre of the room and then apply coastal navigation. This process should be repeated for every new room that is entered.

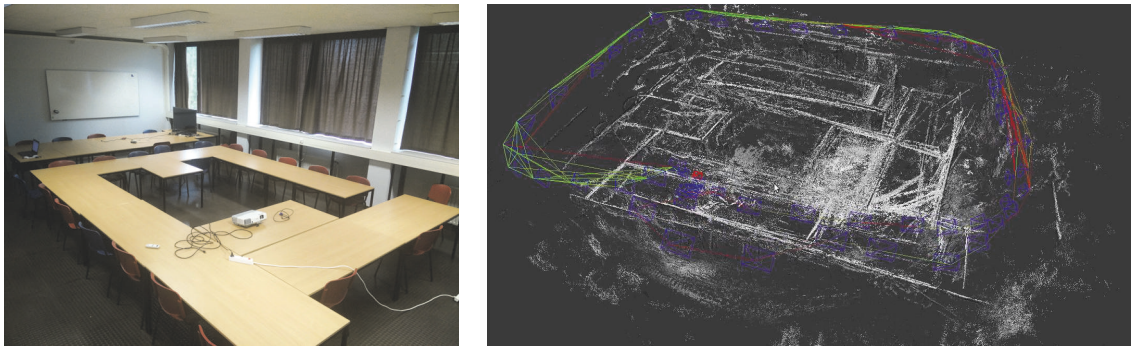
RGB-D SLAM allowed us to configure numerous parameters. First, a feature extractor had to be chosen. Our tests indicated that the SIFTGPU extractor, combined with FLANN feature matching, induces a satisfying result. Second, we filtered the depth image by implementing a minimum and maximum processing depth. As a result, noisy measurements outside the valid range were diminished. The optimal values for these parameters depend on the environment that was recorded. For example, for Figure 12, we have set these parameters to 0.5 metres and 7 metres, respectively. Lastly, the computed point cloud was downsampled, as we noticed that RGB-D SLAM failed to process new visual features when the CPU is overloaded. Downsampling the point cloud with a factor n significantly decreases CPU usage, while maintaining



(a) Iterative merging

(b) Online merging

Figure 9: In (a), the initial guess (IG) is iteratively merged with the complete SLAM point cloud (SL). A balance between map completeness and detail is regulated by the amount of IG or SL point clouds we merge. (b) illustrates another option, where a single IG is merged with partial online SLAM clouds (SL_n). The online merging process is finished when SLAM has completely processed the dataset. The difference between both merging methods is discussed in detail in Section 4.3.



(a) The meeting room that was used to record dataset 1

(b) Resulting point cloud

Figure 10: LSD SLAM result.

Figure 11: For our research, we mounted a Kinect camera to an Erle-Copter [10].

an acceptable point cloud density. Normally, the Kinect outputs a 640×480 array (307200 entries). By downsampling this array, RGB-D SLAM keeps every nth entry in the Kinect array. For example, if n equals 4, only 76800 entries (25%) are kept to be processed by RGB-D SLAM.

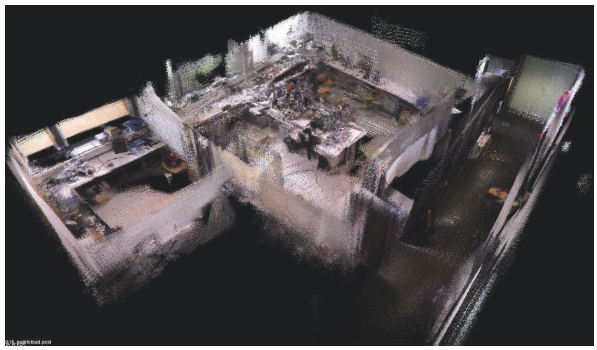
After configuring the RGB-D SLAM parameters, we managed to build the point clouds shown in Figure 12. When we compare these results with the LSD SLAM result in Figure 10, it becomes clear that RGB-D SLAM builds point clouds with higher density and less noise. This is mainly due to the fact that RGB-D SLAM inserts all visual information into the point cloud, whereas LSD SLAM creates a point cloud which merely consists of pixels that were used for depth map estimation. However, the RGB-D SLAM point clouds still contain gaps due to registration errors and limited observations. For example, the ceilings of dataset 2 are not visible in Figure 12(b) and the industrial train car of dataset 3 is incomplete in Figure 12(e).

Inquiring the accuracy of a trajectory requires the implementation of a ground truth estimation system. As mentioned in Section 3.1, implementing such a system does not lie within the scope of our research, so we adopt the accuracy measurements of Endres et al. [25]. In their research, the authors state that the RGB-D SLAM trajectory estimate has an average root mean square error (RMSE) of 9.7 cm and 3.95 degrees if SIFTGPU is used as feature extractor. This number was obtained by testing the SLAM algorithm with datasets which include ground truth information [27].

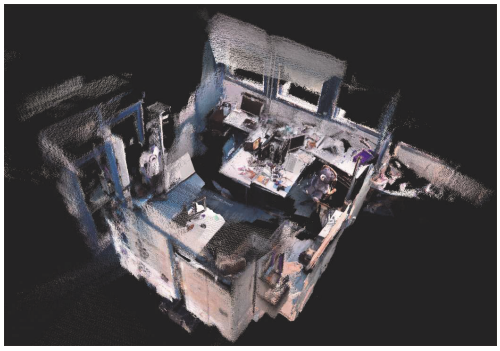
We conducted our own tests in order to determine the xyz precision of the trajectory estimate. RGB-D SLAM was launched several times, each time with the same parameters. One reference trajectory estimate and ten test trajectory estimates were extracted from these tests. For every trajectory, we plotted the error relative to the reference trajectory. Boxplots for the x-, y-, and z-axis error can be found in Figures 13, 14, and 15.

In Figure 13(a), we can observe that the test trajectories correspond well to our reference trajectory, as well as to each other. Errors relative to the reference remain very limited for all trajectories. Nonetheless, we also detect outliers with a difference of up to 80 cm relative to the reference trajectory. Figure 13(b) illustrates when these outliers occur in time. This plot shows high precision until a certain point where the trajectories start to spread out. At this point, RGB-D SLAM was not able to process visual features. Thus, the trajectory

(a) Room V315 at the University of Antwerp, where we recorded dataset 2

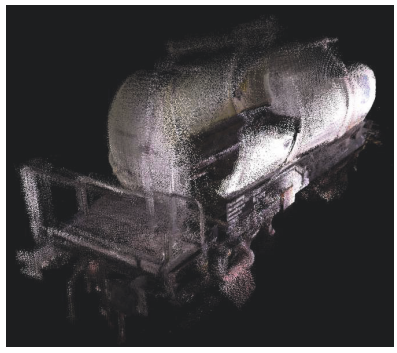


(b) RGB-D SLAM result for dataset 2



(c) RGB-D SLAM result for dataset 4

(d) Dataset 3 was recorded with the purpose of modelling an industrial tank car at the port of Antwerp



(e) RGB-D SLAM result for dataset 3

FIGURE 12: We assessed the RGB-D SLAM algorithm in several environments. First, we tested indoor environments as shown in (a), (b), and (c). Second, we applied the algorithm to map an industrial train cart ((d) and (e)).

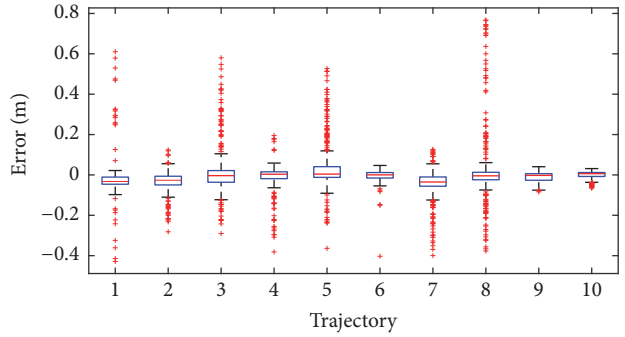
estimate could not be calculated correctly until visual features were tracked again. Compared to the y- and z-axis, the x-axis trajectory accumulated more errors due to the fact that the camera mainly travelled along the x-axis.

Similarly to the x-axis trajectory, Figure 14 demonstrates a high precision on the y-axis. As this axis contains fewer translations than the x-axis, outliers are less distinct.

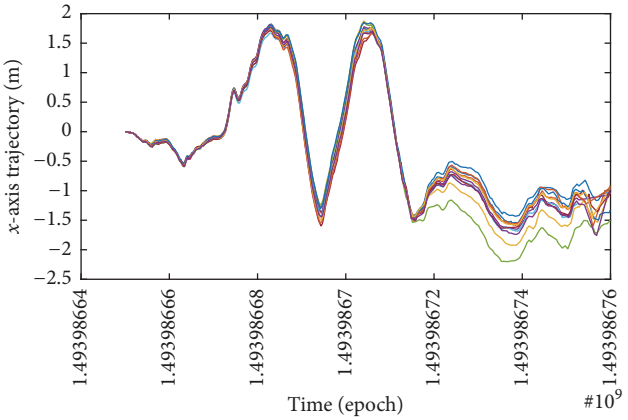
Finally, the z-axis boxplot in Figure 15 exhibits precision results that are comparable with the x and y precision plots.

In general, we can conclude that the RGB-D SLAM algorithm results in precise trajectory estimates, as long as visual features are continuously detected while mapping an environment. In order to ensure continuous feature tracking, the dataset can be played out at a lower speed. By doing so, RGB-D SLAM will have more time to process new images which is beneficial for feature extraction.

4.3. Optimisation Results. Although RGB-D SLAM has provided us with an accurate and dense point cloud, Figure 12 has shown us that the map merely contains information about all environmental elements that were visible in the dataset images. For example, ceilings were not recorded, so they will not be included in the resulting map. When we use this point cloud to render an OctoMap, only a partial volumetric model of the environment is obtained, as seen in Figure 16.



(a) This boxplot demonstrates the x-axis precision error of all trajectories relative to a test trajectory



(b) This plot shows the x-axis precision error over time of all trajectories relative to a test trajectory

Figure 13: RGB-D SLAM x-axis precision.

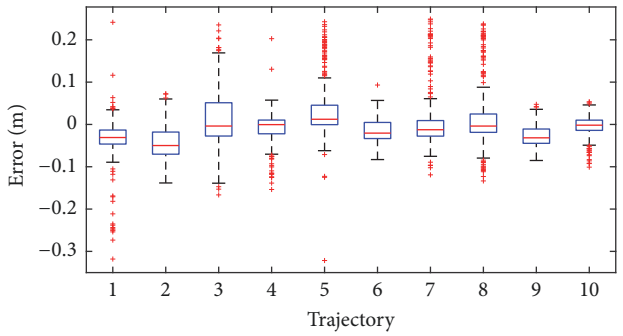


Figure 14: RGB-D SLAM y-axis precision.

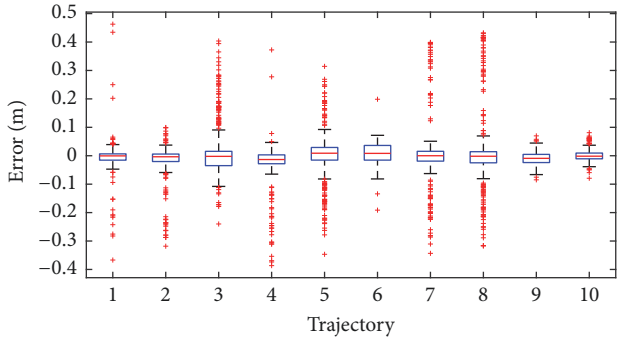


Figure 15: RGB-D SLAM z-axis precision.

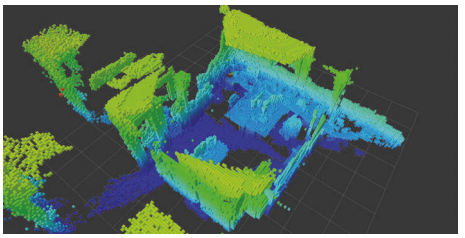


Figure 16: An OctoMap created from our RGB-D SLAM result of Figure 12(c).

Our approach resolves this issue by combining the SLAM point cloud with an initial guess. Merging these clouds can be achieved in two ways: iterative and online. The first method builds an OctoMap using the initial guess and the final SLAM result as input. Occupancy probability is steered by iteratively inserting the point clouds multiple times. However, this form of map completion also updates valid measurements with free space, causing the map to lose some of its detail. Figure 17 demonstrates this problem. In Figure 17(a), both point clouds were inserted once. The initial guess has successfully filled in gaps that were present in the SLAM result of Figure 16, although it has also caused doors and windows to disappear. By adding another instance of the SLAM result (Figure 17(c)), doors and windows started to reappear along with unwanted gaps in the floor. Another factor that has to be taken into account is the order in which point clouds are being merged. As can be seen in Figures 17(e) and 17(g), inverting the merging sequence has a significant effect on the occupancy probability calculation. Concisely, balancing the amount of point clouds and uncovering an appropriate merging sequence are a troublesome task.

A second method to merge point clouds was mentioned in Section 3.3. With this method, point clouds are already being merged while SLAM is running. Instead of using a single SLAM point cloud, RGB-D SLAM constantly pushes its current online point cloud. The main advantage of this method is that OctoMap can now render a volumetric model based on previous and current observations, which leads to a more conclusive probability calculation. Contrary to iterative merging, online merging allows us to obtain an adequate balance between map completeness and detail. This is demonstrated in Figure 18(a): undesirable gaps were completed by the initial guess model, without completely closing up doors and windows.

Both optimisation methods were evaluated using our own datasets as well. First, we take a look in Figure 19(a). An initial

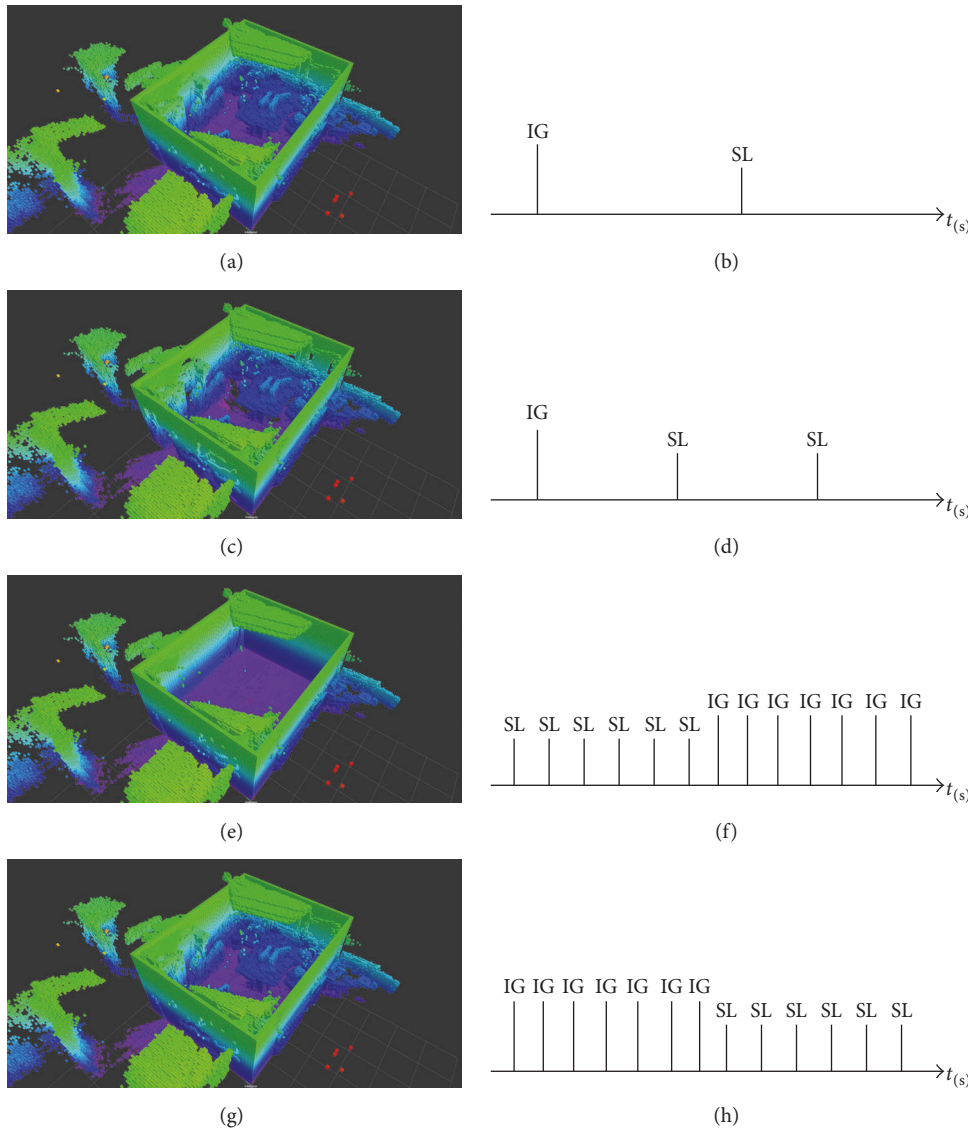


Figure 17: Iterative merging of our initial guess model (IG) with the complete SLAM point cloud (SL).

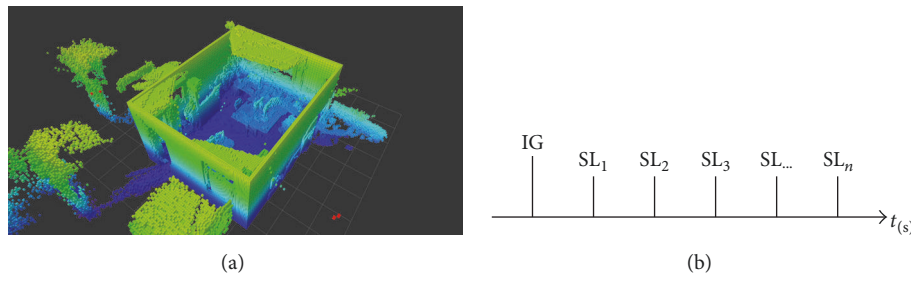
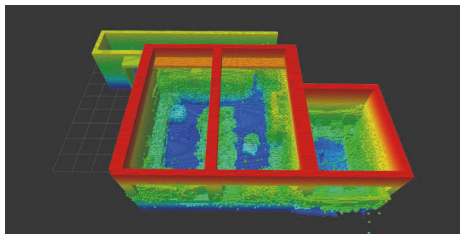


Figure 18: Online merging.

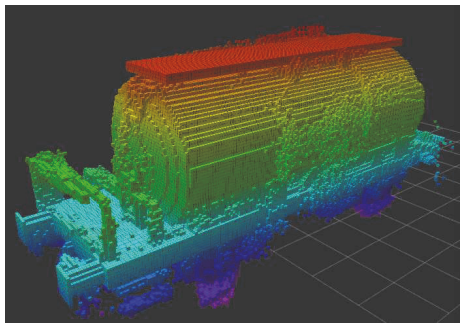
guess model was created with OpenSCAD and converted to a point cloud (Figure 5(a)). Also, we recorded a dataset in our indoor environment (Figure 12(a)) to generate online point clouds via RGB-D SLAM. Through an OctoMap server, these online point clouds were constantly merged with our initial guess until the entire dataset was played.

Secondly, we discuss Figure 19(c). In order to build this model, RGB-D SLAM was used to create a point cloud of an industrial environment, as can be seen in Figures 12(d) and 12(e). Before merging the SLAM point cloud with our initial guess, we removed all unnecessary data, as we only wished to obtain a model of the train cart. Next, this point cloud

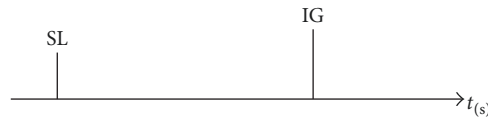


(a) Online merging of an initial guess model (Figure 5(a)) with live RGB-D SLAM output

(b)



(c) Iterative merging of Figure 5(b) with Figure 12(e). The model consists of one initial guess and one SLAM point cloud



(d)

Figure 19: Optimisation results for our own datasets. For (a), online merging was applied. In (c), we conducted iterative merging of 2 point clouds.

was aligned and iteratively merged with the initial guess point cloud of Figure 5(b). In this case, the initial guess and the SLAM result were merged a single time.

5. Conclusion

In this paper, we present an efficient, robust method for completion and optimisation of 3D models using MapFuse. In a simulator as well as in reality, we have evaluated combinations of proven open-source technologies in order to attain a realistic map optimisation technique. Apart from these technologies, our method does not require additional complex calculations for map optimisation.

Several aspects affect the quality of our final result. Firstly, the accuracy of the initial guess model has to be considered. For known environments, the accuracy is assumed to be 100%, as exact measurements can be collected. In other situations, the user has to speculate about dimensions based on visual observations or the result of a SLAM algorithm. Also, the amount of detail that is included in the initial guess, for example, windows, doors, and furniture, will affect the occupancy probability for those elements within the model. In general, outlines of the filtered SLAM environment are sufficient to serve as initial guess, as detail will be provided by SLAM. Future work could involve automatic generation of the initial model from the SLAM output.

Secondly, MapFuse requires an accurate RGB-D SLAM point cloud in order to update the initial guess correctly. For this purpose, the SLAM algorithm has to be provided with a valid dataset that contains a significant amount of visual information about the environment. SLAM accuracy and completeness are directly related to the amount and quality of visual observations in the dataset. Improvements for the SLAM result can be made by altering parameters of the algorithm, or by lowering the playback speed of the dataset. The latter measure allows RGB-D SLAM more time per frame to process visual features.

Finally, we have to choose a method for bringing both point clouds together. Iterative merging fuses complete point clouds by aligning them and sending them to an OctoMap server. A balance between map completeness and detail is set by regulating the amount of point clouds that is being forwarded, as well as implementing an appropriate merging sequence. However, obtaining this balance has proven to be a difficult exercise. A main advantage of the iterative merging method is that the SLAM point cloud can be edited before using it in the merging process. Online merging starts by sending a single initial guess to an OctoMap server and continues with running the RGB-D SLAM algorithm. After an initial map alignment, online SLAM point clouds are continuously merged with the initial guess, leading to an improved balance between map completeness and detail. For both merging methods, OctoMap parameters such as initial probability and resolution can be altered in order to influence the final result. Also, MapFuse could cope with dynamic environments by setting an occupancy probability threshold which cancels out moving objects.

As initially intended, MapFuse is suitable for creating 3D models of various environments for the purpose of validating wireless propagation models. Furthermore, the proposed

approach can be applied in other application domains such as the optimisation of dynamic control algorithms. Researchers would be able to model realistic 3D objects which makes it possible to validate complex control simulations [28]. Due to the realistic nature of our approach, such validations could improve control systems which work with complex 3D objects. Additionally, the accuracy and precision of the validation will be affected by the OctoMap resolution. Hence, a performance trade-off for the control system could be analysed.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this article.

References

[1] F. Xia, L. T. Yang, L. Wang, and A. Vinel, "Internet of things," *International Journal of Communication Systems*, vol. 25, no. 9, pp. 1101-1102, 2012.

[2] M. F. Iskander and Z. Yun, "Propagation prediction models for wireless communication systems," *IEEE Transactions on Microwave Theory and Techniques*, vol. 50, no. 3, pp. 662–673, 2002.

[3] C. Phillips, D. Sicker, and D. Grunwald, "A Survey of wireless path loss prediction and coverage mapping methods," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 1, pp. 255–270, 2013.

[4] B. Bellekens, R. Penne, and M. Weyn, "Validation of an indoor ray launching RF propagation model," in *Proceedings of the 6th IEEE-APS Topical Conference on Antennas and Propagation in Wireless Communications, IEEE APWC 2016*, pp. 74–77, Australia, September 2016.

[5] Z. Yun and M. F. Iskander, "Ray tracing for radio propagation modeling: Principles and applications," *IEEE Access*, vol. 3, pp. 1089–1100, 2015.

[6] Z. Lai, G. De La Roche, N. Bessis et al., "Intelligent ray launching algorithm for indoor scenarios," *Radioengineering*, vol. 20, no. 2, pp. 398–408, 2011.

[7] J. Chan, C. Zheng, and X. Zhou, "3D printing your wireless coverage," in *Proceedings of the 2nd ACM International Workshop on Hot Topics in Wireless, HotWireless 2015*, pp. 1–5, France.

[8] M. Weyn, G. Ergeerts, R. Berkvens, B. Wojciechowski, and Y. Tabakov, "DASH7 alliance protocol 1.0: Low-power, mid-range sensor and actuator communication," in *Proceedings of the IEEE Conference on Standards for Communications and Networking, CSCN 2015*, pp. 54–59, Japan, October 2015.

[9] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, vol. 34, no. 3, pp. 189–206, 2013.

[10] E. Robotics, "Erle-Copter — Erle Robotics," http://erlerobotics.com/blog/erle-copter/.

[11] T. P. Breckon and R. B. Fisher, "Non-parametric 3D surface completion," in *Proceedings of the 5th International Conference on 3-D Digital Imaging and Modeling, 3DIM 2005*, pp. 573–580, Canada, June 2005.

[12] E. Turner and A. Zakhor, "Automatic Indoor 3D Surface Reconstruction with Segmented Building and Object Elements," in *proceedings of the 2015 International Conference on 3D Vision*, vol. 10, Institute of Electrical and Electronics Engineers (IEEE), Lyon, France, 2015.

[13] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part I," *IEEE Robotics & Automation Magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[14] L. Zhang, R. Zapata, and P. Lepinay, "Self-adaptive monte carlo localization for mobile robots using range sensors," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '9)*, pp. 1541–1546, St. Louis, Mo, USA, October 2009.

[15] D. Fox, J. Hightower, L. Liao, D. Schulz, and G. Bordello, "Bayesian filtering for location estimation," *IEEE Pervasive Computing*, vol. 2, no. 3, pp. 24–33, 2003.

[16] D. M. Cole and P. M. Newman, "Using laser range data for 3D SLAM in outdoor environments," in *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, IEEE, Orlando, FL, USA, 2006, http://www.robots.ox.ac.uk/mobile/Papers/3DScanMacthingCole_ICRA2006.pdf.

[17] A. Aghamohammadi, A. H. Tamjidi, and H. D. Taghirad, "SLAM Using Single Laser Range Finder," *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 14657–14662, 2008, http://linkinghub.elsevier.com/retrieve/pii/S1474667016413479.

[18] A. Gil, O. M. Mozos, M. Ballesta, and O. Reinoso, "A comparative evaluation of interest point detectors and local descriptors for visual SLAM," *Machine Vision and Applications*, vol. 21, no. 6, pp. 905–920, 2010.

[19] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-Scale Direct monocular SLAM," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Preface*, vol. 8690, no. 2, pp. 834–849, 2014.

[20] J. Engel, J. Stuckler, and D. Cremers, "Large-scale direct SLAM with stereo cameras," in *proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 9, IEEE, Hamburg, Germany, 2015.

[21] G. Silveira, E. Malis, and P. Rives, "An Efficient Direct Approach to Visual SLAM," in *Proceedings of the IEEE Transactions on Robotics*, vol. 24, IEEE, Roma, Italy, 2007, https://www.researchgate.net/profile/Patrick_Rives/publication/224330808_An_efficient_direct_approach_to_visual_SLAM/links/00b7d52983db5ed842000000/An-efficient-direct-approach-to-visual-SLAM.pdf.

[22] S. Umeyama, "Least-Squares Estimation of Transformation Parameters Between Two Point Patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 4, pp. 376–380, 1991.

[23] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: an efficient alternative to SIFT or SURF," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV '11)*, pp. 2564–2571, Barcelona, Spain, November 2011.

[24] P. M. Panchal, S. R. Panchal, and S. K. Shah, "A comparison of SIFT and SURF," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 1, no. 2, pp. 323–327, 2013.

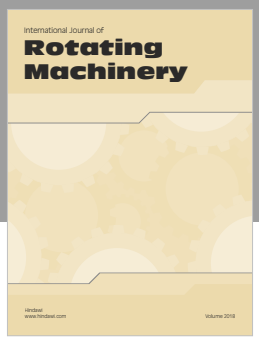
[25] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard, "An evaluation of the RGB-D SLAM system," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '12)*, pp. 1691–1696, 2012.

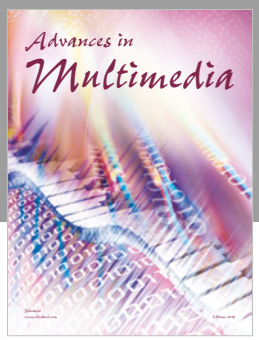
[26] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G^2o: a general framework for graph optimization," in *Proceedings of the IEEE International Conference on Robotics and*

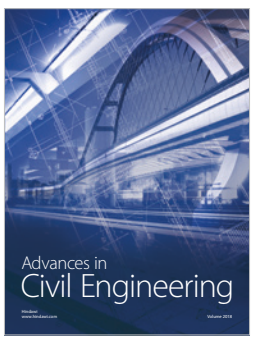
Automation (ICRA '11), pp. 3607–3613, Shanghai, China, May 2011.

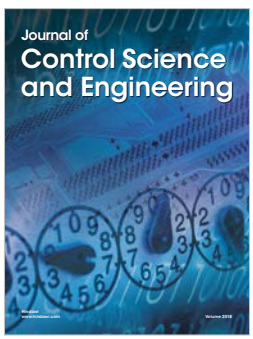
[27] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *Proceedings of the 25th IEEE/RSJ International Conference on Robotics and Intelligent Systems (IROS '12)*, pp. 573–580, October 2012.

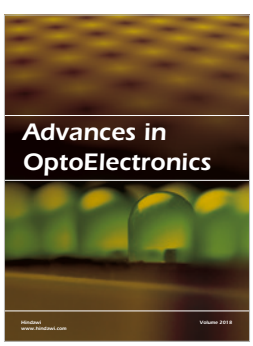
[28] L. Fortuna and G. Muscato, "A roll stabilization system for a monohull ship: modeling, identification, and adaptive control," *IEEE Transactions on Control Systems Technology*, vol. 4, no. 1, pp. 18–28, 1996.

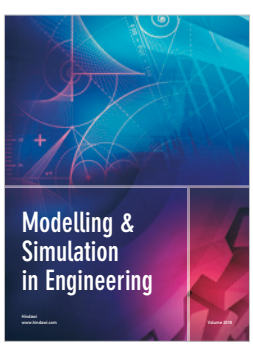


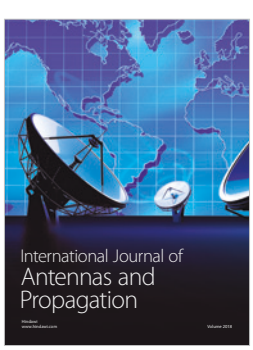


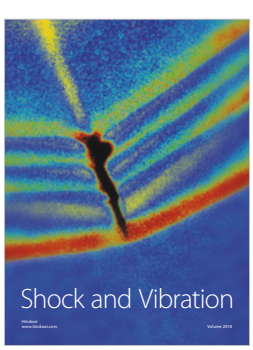


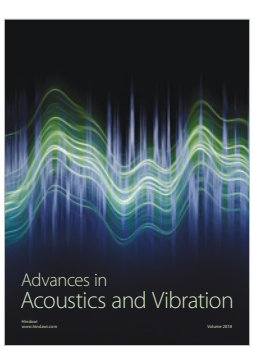












Hindawi

Submit your manuscripts at
www.hindawi.com