SLAM-BASED 3D OUTDOOR RECONSTRUCTIONS FROM LIDAR DATA

Ivan Caminal, Josep R. Casas, Santiago Royo

Dept. d'Òptica i Optometria
Dept. de Teoria del Senyal i Comunicacions
Universitat Politècnica de Catalunya

ABSTRACT

The use of depth (RGBD) cameras to reconstruct large outdoor environments is not feasible due to lighting conditions and low depth range. LIDAR sensors can be used instead. Most state of the art SLAM methods are devoted to indoor environments and depth (RGBD) cameras. We have adapted two SLAM systems to work with LIDAR data. We have compared the systems for LIDAR and RGBD data by performing quantitative evaluations. Results show that the best method for LIDAR data is RTAB-Map with a clear difference. Additionally, RTAB-Map has been used to create 3D reconstructions with and without photometry from a visible color camera. This proves the potential of LIDAR sensors for the reconstruction of outdoor environments for immersion or audiovisual production applications.

***Index Terms*—** LIDAR cameras, mapping, time-of-flight, SLAM, 3D imaging, point-cloud processing

1. INTRODUCTION

Simultaneous localization and mapping (SLAM) is the computational problem of building a map of an unknown environment while simultaneously keeping track of an agent's location within it. Mapping allows to localize the sensor whereas a location estimate is needed to build the map. Some SLAM scenarios focus on location, such as in automotive where the map uses to be known beforehand, while in audiovisual and special effects the focus is rather on mapping, i.e. reconstruction of the scene environment. LIDAR imaging [1] is a powerful measurement technique where a laser pulse is shone onto an object and the beam reflected back is recovered at some solid-state detector. The time elapsed is measured, allowing for an automated measurement of the distance to the target, without any further calculation. The concept is also referred to as ladar or time-of-flight imaging. Popular applications involve landing aids, object recognition or self-guided vehicles.

This paper focuses on adapting two state of the art SLAM strategies to work with LIDAR sensors. The two strategies are evaluated quantitatively with one real LIDAR dataset and two RGBD datasets (one real and the other synthetic). This evaluation allows to objectively compare the two systems. The best system is used to obtain 3D reconstructions even without photometric images, just with a LIDAR sensor developed at Beamagine (a spin-off of UPC developing LIDARs based on proprietary technology).

The paper is organized as follows. Section 2 reviews the state of the art in 3D SLAM systems. Section 3 explains the adaptations of SLAM systems to LIDAR data. Sections 4 and 5 provide evaluation results and conclusions.

2. STATE OF THE ART

The basics of SLAM systems capable of creating three dimensional maps were investigated in the form of 3D grids [2] and 3d geometric features [3]. The first 3D SLAM systems used mono cameras [4], stereo cameras [5] or 3D LIDARs [6].
More recently, the availability of real-time dense depth sensors (RGBD) has eased the live reconstruction of real scenes. Microsoft develops Kinect Fusion [7] in 2011, an algorithm allowing 3D reconstructions at 30fps taking advantage of the recently launched Kinect matricial depth sensor. One year later, PCL [8] incorporates a similar open-source tool known as KinFu [9]. Both systems use a voxelized representation of the scene named TSDF model (Truncated Signed Distance Function model [10]), where each voxel stores the distance to the closest surface and a confidence weight. The main limitation of these systems is the inability to map areas larger than the model. This limitation was eliminated at the same time by Kintinuous [11] and KinFu large-scale [12].
Kintinuous implements an unbounded mapping of the environment on top of KinFu. Precisely, it incorporates the ability to virtually translate the TSDF model when new estimated camera poses exceed a dimension independent threshold. Kintinuous was improved to be more robust against challenging scenes [13], such as large camera displacements or lack of 3D depth features, while also aiming to eliminate the accumulated drift of previously registered frames [14]. The drift elimination is known as loop closure. It happens when the sensor revisits a previous location by optimizing all the affected transformations with a pose optimizer (iSAM) and a non-rigid

method that corrects the reconstruction. KinFu large-scale is now a simpler tool similar to the original Kintinuous without the real-time map extraction.

RGB-D SLAM [15] is another real-time system with Robot Operating System (ROS) support [16]. The transformations between poses are obtained by detecting key-points of incoming frames, computing features and finding correspondences with older ones. The system also does loop closure with a pose optimizer (g^2o). The OctoMap framework is used to create reconstructions using the optimized trajectory. The system was improved [17] and now includes: a beam-based environment measurement model (EMM) that validates the estimated transformations according to occlusion probabilities, a selection strategy of candidate frames for comparisons based on exploring the geodesic graph neighborhood of the previous frame, and the use of key-frames to simplify the search.

ElasticFusion [18] is another real-time system developed by some of the authors of Kintinuous. It is based on a surface model instead of TSDF, the loop closure is done without a pose optimizer by non-rigidly deforming the affected surfaces.

RTAB-Map is another real-time system with ROS support that can work with 2D LIDARs and stereo setups (apart from RGBD cameras). It is based on a graph of links and nodes. The nodes contain information about the poses of the robot and the links store rigid transformations between nodes. The transformations are obtained using 3D visual words correspondences and maintained with TORO (Tree-based netwORk Optimizer) allowing to propagate the error through links after loop closures. Additionally, RTAB-Map incorporates a proximity module to find loop closures with 2D LIDARs that helps in situations when the RGBD cameras do not have enough information. The strong point of RTAB-Map is a memory-efficient loop closure detection approach.

3. LIDAR ADAPTATION

From the SLAM strategies explored in the previous section, we have selected both Kintinuous and RTAB-Map (available on Github) to work with LIDAR data. The reasons are that Kintinuous is supposed to perform better than ElasticFusion with noisy LIDAR data and that RTAB-Map is expected to improve RGB-D SLAM with LIDAR, since the EMM of RGB-D SLAM assumes dense depth measurements, and the loop closure approach of RTAB-Map seems to be more efficient.

We have adapted the SLAM algorithms to LIDAR data, and we describe the adaptations according to the specific sensor setup of the LIDAR dataset and Beamagine data.

3.1. Adaptation of LIDAR dataset

The KITTI dataset [19] is the one chosen for the adaptation of SLAM algorithms to LIDAR data as it allows quantitative evaluation. It consists of 22 sequences about diverse traffic environments (highway, rural and city). Regarding the sensor setup, it is composed of: 2x gray-scale and color cameras, 1x rotating 3D LIDAR and 1x inertial and GPS unit. In this adaptation, we only use the images of the left color camera and the scans of the LIDAR from the already rectified, undistorted and synchronized version of the dataset. Both the projection transformation to the rectified cameras and the extrinsic transformation from 3D LIDAR coordinates to camera coordinates are provided in [19].

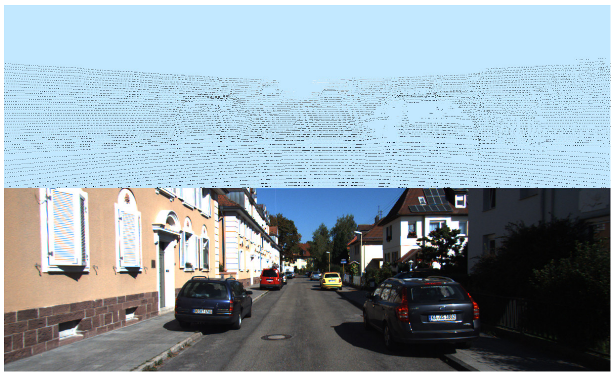


Fig. 1: Depth image obtained after the projection of a KITTI LIDAR scan, with its corresponding color image below. Pixels without depth values in the LIDAR scan are colored in blue in the depth image to ease visualization.

In the first part of the adaptation, we converted the eleven KITTI sequences with available ground-truth to the PNG format of the RGB-D SLAM dataset [20]. This was done with a tool that basically projects the 3D LIDAR scans to the selected camera (in our case the left one with color) and calculates its depth values. Then, every value is quantized to a 16-bit unsigned representation considering the maximum LIDAR range (120 meters). The resulting quantization step is much lower than the one provided by the manufacturer ($1.8 \ll 20$ millimeters). Given the LIDAR properties and the camera FOVs, only about 32% of the points of a complete scene are projected to the camera plane within the sequence dependent image size, where half of these points are front-projected. A result for the depth projection from a LIDAR scan is shown along with the corresponding color frame in figure 1.

The remaining parts of the adaptation specific for each system are described below.

3.1.1. Kintinuous applied to KITTI data

The implementation of Kintinuous uses log files in KLG format as input. This format consists of storing in a single file all the information of a sequence: the timestamps and a compressed version of the depth and color images. The main author of Kintinuous provides some tools to create KLG log files directly from data-streams of sensors like Kinect and Xtion Pro Live. That said, a conversion from PNG RGB-D SLAM format to KLG format was needed. Fortunately, the implementation of this conversion was already done in a GitHub repository [21]. This repository contains a tool called *png_to_klg* that essentially creates a KLG log file from the frame pairs provided by an associations text file, converts the timestamps from seconds to micro-seconds and the scaled depth measurements to millimeter units.

The core of Kintinous is the cubic TSDF model that, in its default configuration, has a side length in voxels of 512 and a real world equivalence of 6 meters. The converted LIDAR data has a theoretical maximum range of 120 meters. These two statements make the system and the data incompatible. The only two ways to solve this is by adapting the data to Kintinuous or Kintinuous to the data. Regarding the system adaptation, increasing the number of voxels of the cubic TSDF model may be an option, but it requires a complex code modification and is expected to fail due to low density of points within the model. This lack of points would be produced by the low number of LIDAR points (100K per scan) and the low scan-rate (10 Hz) related to the average LIDAR movement (car motion). On the other hand, the data adaptation could be achieved either by increasing the real world equivalence of a single voxel or scaling the real world dimensions, both of them at the cost of losing precision. The second option was chosen, and implemented by scaling the dataset with a world scale factor, which was implicitly introduced along with the depth quantization factor in the *png_to_klg* tool. This allows the generation of KLG files with different world scale factors. After some testing, the definitive world scale factor was set to 20 (1 scaled meter for the algorithm corresponds to 20 world meters). This comes from the fact that the actual maximum depth of LIDAR scans was about 80 m. and the depth limit that Kintinuous implementation allows to project is 4 m. (as it considers that larger Kinect depths are too noisy).

When executing Kintinuous, we set the shifting threshold to 16 voxels (maximum according to the author) since the distance traveled by the camera at different frames is larger, due to high velocity (car in KITTI vs hand-held cameras in RGB-D SLAM data) and low frame rate (30 vs 10 fps). Also, the parameter subsample pose graph was deactivated to export all optimized poses of the graph when loop closure is enabled.

3.1.2. RTAB-Map applied to KITTI data

The RTAB-Map implementation uses images stored in regular files as input, thus not requiring the *png_to_klg* tool. The images need to be already associated in disc since it does not accept an associations text file as synchronization information. Luckily, the administrator of RTAB-Map already provides a modified version of the RGB-D SLAM associations tool that, instead of exporting the pairing information in a text file, creates directories and moves the synchronized images resulting from the association process.

In this case, the world scale was unnecessary since RTAB-Map system is not restricted to the ranges of structured light sensors. Nevertheless, we decided to execute both versions, thus allowing to verify the correct implementation of the world scaling factor and its effect. We had to locally modify the RTAB-Map implementation to allow for a depth scale factor lower than 1 step/millimeter which was the case when using the converted version of the dataset with *kitty_to_png*. We used the RGB-D dataset command-line tool for executing with RTAB-Map instead of the GUI interface. This tool saves a SQLite database with all the information related to the SLAM and exports the poses in the selected format. The maps can be created without RTAB-Map GUI using the export example (available in the examples folder of the Github repository) implemented by one author (thanks to Mathieu Labbé) after asking a question in the official RTAB-Map forum.

3.2. Adaptation of Beamagine data

Unlike the KITTI dataset, Beamagine data comes from a single sensor: a 3D LIDAR with its infrared light based range measurements, without a registered color camera. This LIDAR, different from the one in KITTI, is static and front-facing, and its main specifications are 5 Hz, 0.5Mpoints/s, FOV: 54.5° h, 20° v, range 165 m., 200x600 sampling points. The lack of a photometric camera in the Beamagine set-up opens the challenge to test SLAM systems without exploiting photometric RGB data in a dense, regularly sampled raster image. Visual SLAM detects singular image points to find correspondences between frames to be registered. At this point, we propose to replace the dense photometric information by the infrared intensity of the LIDAR points. This idea was implemented in a tool called *beamagine_to_klg* similar to the one implemented for KITTI. Basically, it reads the LIDAR scans (stored in separated pcd files), converts the metric units from millimeters to meters, projects the points to a simulated camera plane and calculates their associated depth and intensity values. As camera parameters, we only used a focal value for each dimension (to account for perspective projection) and the image size, since no intrinsic LIDAR calibration was available. The image size was selected simulating the highest sampling frequency in each dimension. It was directly set to 200 pixels vertical, and 1364 pixels horizontal, since the off-center points have higher resolution than the center ones (in angular measures: 0.04° vs. 0.15°) due to design constraints of the sensor. Then, the two focal parameters were obtained considering both the image dimensions and the two LIDAR FOVs. Given the simulated camera parameters, about 99% of points are correctly projected. After projection, each value is quantized to unsigned 16-bits considering the dynamic range of the measure. The depth quantization step is 2,5 mm (a bit larger than KITTI's), but again it is considered to be sufficient. A result for the depth calculation is shown in figure 2.

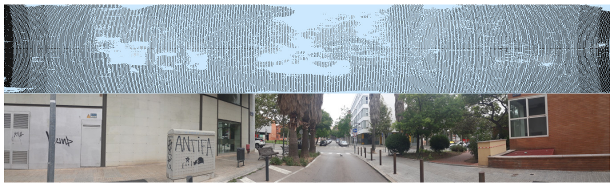


Fig. 2: Depth image obtained from the projection of the point cloud of a Beamagine LIDAR scan. Pixels without depth values are set to blue in the depth image to ease visualization. The photo below was taken some days after the captures from a similar point of view.

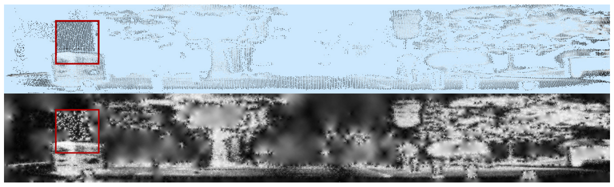


Fig. 3: Infrared values from a Beamagine scan projected in an image plane before and after interpolation. Pixels with no value are set to blue in the upper image to ease visualization. The content within the red boxes corresponds to the same images without eliminating the saturated and null values of the distribution that produced high spatial frequencies.

Furthermore, in order to obtain infrared data similar to a dense raster image (without holes), the generated intensity images were post-processed with an inpainting stage. This was done to fulfill the SLAM systems requirements and to ease the detection of singular points for correspondences. For the inpainting, we used the 8-connectivity version of a morphological interpolation technique [22], that preserves the original infrared values of the projected points and its transitions, thanks to the use of geodesic distance. This technique is efficiently implemented as an iterative process: first, the set of initial pixels are propagated using geodesic dilation and, then, the transitions generated are recovered by applying the morphological Laplacian, where the average values are used along with the originally projected ones for propagation in subsequent iterations. After applying the inpainting stage, some sensor noise with a high spatial frequency between the saturated and null values was discovered in the infrared values of the Beamagine sensor. The noise was eliminated by discarding the two histogram peaks which implied to end up using about 30% of the dynamic range.
In order to evaluate the effect of using the intensity of the points instead of the color images, the same adaptation was done in the *png_to_klg* tool for the KITTI dataset. Figures 3 and 4 show a frame and its interpolated version for Beamagine and KITTI data, respectively.

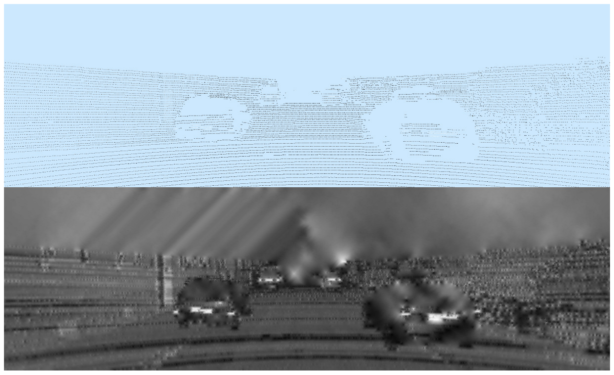


Fig. 4: Infrared image before and after interpolation from a KITTI LIDAR scan.

The specific adaptation for each SLAM system is similar to the one explained in section 3.1 since the generated data format is the same.

4. RESULTS

In this section, we present and discuss quantitative results of evaluating the systems with the LIDAR adaptation mentioned in section 3.1, and the evaluation done with RGBD datasets both natural and synthetic. Qualitative results obtained for the LIDAR adaptation of section 3.2 are also shown.

4.1. Trajectory evaluation

The quantitative trajectory evaluation was done with the Absolute Trajectory Error metric (ATE) [23] for both real Kinect and LIDAR data modalities using ground-truth available in RGB-D SLAM and KITTI datasets.

4.1.1. Estimated Trajectories on the RGB-D SLAM dataset

Depending on the system, we performed different tests switching the loop closure component and varying parameters of SLAM algorithms. For Kintinuous, we tried all possible cost-combinations except FOVIS alone (that could not be set). The combinations are: *ICP*, *RGB-D*, *FOVIS/ICP*, *FOVIS/RGB-D*, *FOVIS/ICP+RGB-D* and *ICP+RGB-D*. For RTAB-Map, we tried to modify its key-point and feature descriptor extractors, always using frame-to-map odometry and 3D to 3D motion estimation. This includes: *surf*, *sift*, *orb*, *kaze*, *brisk*, *gftt/brief*, *gftt/orb*, *gftt/freak*, *fast/freak*, *fast/brief*, *surf/brief* and *surf/orb*.

The default behavior of RTAB-Map when it cannot compute a transformation (minimum of 20 inliers by default) for an incoming frame is to discard it. Conversely, the default behavior of Kintinuous is to repeat the last pose. This fact, renders the trajectory evaluation of both systems somewhat

Sequence	Kintinuous	RTAB-Map
desk	**0,052**	0,082
room	0,224	**0,128**
desk2	0,073	**0,045**
large no loop	0,465	**0,332**
pioneer slam2	2,186	-
long office household	0,048	**0,037**
AVERAGE	*0,172*	*0,125*

Table 1: Best RMSE results of ATE [23] on the Kinect RGB-D SLAM dataset (best results per sequence in bold).

biased. The way we approached a fair comparison was by executing RTAB-Map in a fixed and an adaptive form, and using only the fixed form for the comparison. The fixed form consists of setting a maximum inlier distance of the feature correspondences to a fixed value for all the sequences and discarding the executions where the system is not able to compute the transformation for any of the sequence frames. The same was applied for the executions where Kintinuous outputs repeated transformations. The adaptive form consists of starting with a low inlier distance and, if at any frame of the sequence the transformation cannot be computed, the inlier distance value is increased by a factor and starts again, until success or until reaching a maximum value. While this later form tends to give more accurate results, it is sequence dependent and would not be applicable to real-time situations.

From all the evaluations we run, we picked the best performing combination of parameters for each system, based on the average RMSE of the sequences. For Kintinuous, the best combination was *ICP+RGB-D*, while for RTAB-Map, it was a tunned version of the *gftt/brief* combination. Loop closure was enabled in both cases. Specifically, the parameters modified where the quality level of the *gftt* (good features to track) key-point extractor, that was set to 0.005, and the minimum Euclidean distance between detected corners, set to 5 pixels. The results are shown in table 1, where RTAB-Map performs about 40% better in average than Kintinuous in trajectory estimation. And this happens consistently in all sequences but for the pioneer slam 2 sequence, where it is not able to compute all the transformations when the inlier distance is fixed at 0.1 meters. This fact happens with all tested combinations and only yields results when RTAB Map is executed with the adaptive modality that allows for a greater inlier distance. Also note that, in this sequence, Kintinuous is able to compute the trajectory but with an average RMSE of about 2m, with values in a range of [0,196m, 3,614m]. Note that RTAB-Map is more accurate than Kintinuous for about one fourth of the trajectory length, whilst, in the remainder, Kintinuous maintains its performance while RTAB-Map drifts.

4.1.2. Estimated Trajectories on the KITTI dataset

Unlike for the previous dataset, here we evaluated the trajectory for all ten sequences with available ground-truth. For RTAB-Map, we switched again the loop closure component but only considering the combination that gave best results (*gftt/brief*) in the Kinect trajectory baseline. The quality level of the gftt key-point extractor was changed from 0,005 to 0,0005 and the minimum Euclidean distance between detected corners was increased by one pixel. Again, we executed all cost-combinations for Kintinuous.

In the evaluation of this LIDAR dataset, the best cost-combination for Kintinuous was the *RGB-D* independent one, whether executed with or without loop closure, while the one for RTAB-Map is with loop closure enabled. Table 2 compares these results. Note that RTAB-Map is about 5 times better than Kintinuous in average in trajectory estimation. Apart from this, in figure 5 we show a plot from sequence *07* comparing the translational part of the estimated trajectory with its corresponding ground-truth. The plot visually proves that the trajectory is better estimated by RTAB-Map than by Kintinuous. The projection does not allow to visualize the vertical component of the differences.

4.2. Evaluation of the 3D reconstructed map

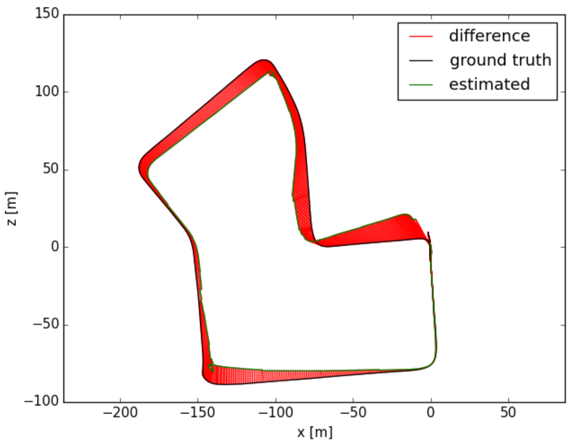
For the quantitative evaluation of the 3D mapping generation functionality of SLAM algorithms we have chosen to use the tool provided by the main author of Kintinuous. This tool computes as metric the point-to-point distance between the ground-truth and the estimated maps on a synthetic dataset of *living-room* sequences known as ICL-NUIM [24].

4.2.1. Evaluation of the mapping for the ICL-NUIM dataset

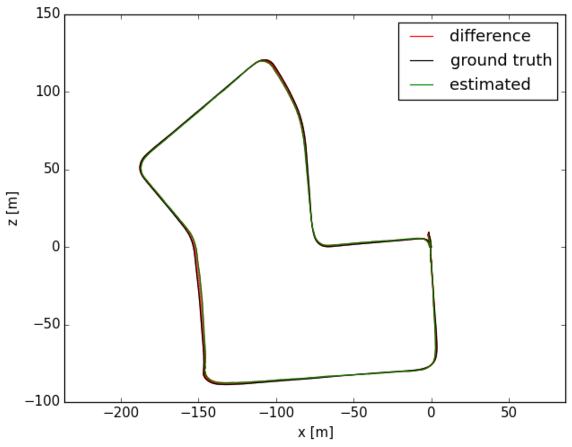
All the four *living-room* sequences *lr kt0..3* were used with and without simulated Kinect noise. For RTAB-Map, we used the same best combination found in the trajectory baseline of section 4.1.1, with and without the loop closure component. Regarding the RTAB-Map reconstruction extraction, and in order to perform a comparison, a voxel grid filter with the same leaf size as the one used by Kintinuous (6/512) was used. For the creation of the point clouds, the same maximum length as Kintinuous is used (4 meters) and a decimation in the color image by a factor of 9 was applied to obtain a similar number of points for the maps of both systems. Additionally, for the version of the dataset with noise, a local smoothing filter was tried in RTAB-Map but, as the computational time for extraction increased and some fine walls of the reconstructions were filtered before the removal of some noisy parts, it was not included for the comparison. Again, for Kintinuous, all cost-combinations were tried, filtering the noisy extracted points from the zero crossing surface of the slices with a minimum voxel weight threshold of 8 (default).

Sequence	Kintinuous	Kintinuous, LC	RTAB-Map	RTAB-Map, LC
00	149,7	149,7	**30,9**	**11,5**
01	488,6	489,1	-	-
02	289,8	289,8	**34,5**	**29,0**
03	**2,3**	**2,3**	6,9	7,3
04	11,8	11,9	**11,7**	**11,7**
05	93,3	93,4	**21,7**	**18,5**
06	203,8	203,7	-	-
07	21,9	21,9	**3,2**	**2,2**
08	65,1	65,1	**30,0**	**26,6**
09	77,1	77,2	**17,9**	**15,7**
10	38,0	38,1	**8,8**	**8,5**
AVERAGE	*83,2*	*83,3*	*18,4*	*14,6*

Table 2: Best RMSE results of ATE [23], with and without loop closure (LC), on the KITTI LIDAR dataset (best results per sequence in bold).



(a)



(b)

Fig. 5: Differences between the estimated trajectories of sequence 07 (in green) and the ground truth (in black) projected to the xz plane: (a) Kintinuous, (b) RTAB-Map.

Table 3 summarizes the results for both data modalities (with and without noise). The best results for Kintinuous are obtained with *ICP* cost and with loop closure. However, the sequence *lr kt0* is considered without loop closure, since the deformation graph failed without saving any result for Kintinuous in the original version of the sequence, and for RTAB-Map it improved more than double without loop closure. Similarly, for RTAB-Map all results are picked with loop closure except for the first sequence.

Sequence	Kintinuous	RTAB-Map	Kintinuous	RTAB-Map
Modality	Original	Original	Noise	Noise
AVG points	*471K*	*555K*	*441K*	*863K*
lr kt0	**4,4**	12,7	**6,4**	50,2
lr kt1	5,6	**4,7**	**8,9**	69,9
lr kt2	**4,3**	7,8	**9,0**	51,0
lr kt3	74,2	**6,3**	77,2	**58,5**

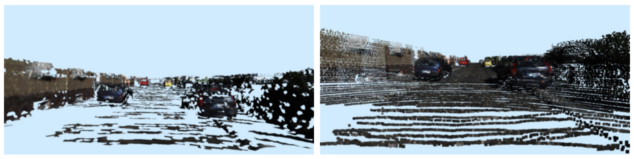
Table 3: RMS point-to-point distance for evaluation of the reconstructed 3D map in the ICL-NUIM dataset (in bold, best technique results for each modality).

4.3. Reconstructions

As final qualitative results for this section, we present the obtained RTAB-Map reconstructions with 3D LIDAR data for both the KITTI and Beamagine scaled datasets.

4.3.1. 3D reconstruction for the KITTI dataset

For this dataset 3D reconstructions are generated for the eleven sequences evaluated in section 4.1.2. The export tool mentioned in section 3.1.2 is used in place of the RTAB-Map with



(a) (b)

Fig. 6: RTAB-Map reconstructions for sequence 07 unscaled, from a similar point of view to the one of the intensity, depth and color frames showed in section 3.1, figure 1. The reconstructions modes are: (a) Mesh (b) Point Cloud.

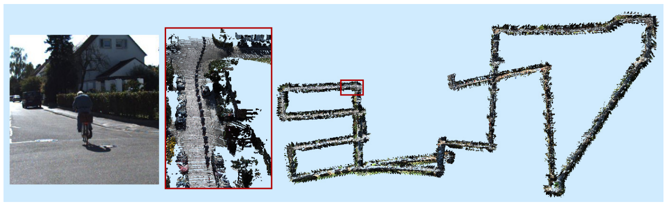


Fig. 7: RTAB-Map reconstruction for sequence 08 unscaled. The snapshot on the left shows a biker in front of the car, with the 3D overall reconstruction of the trajectory on the right. The central image shows a zoom in on the red rectangle, with the darker trace of the moving bike clearly visible.

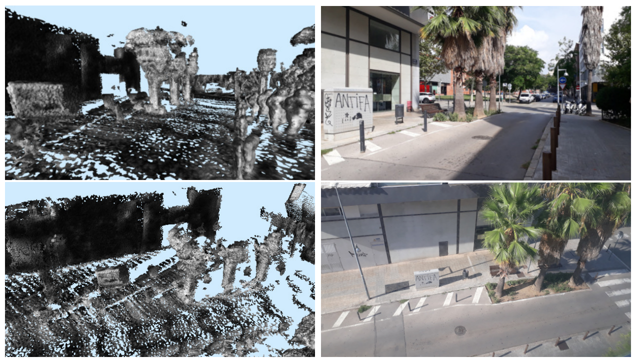
GUI installation, having as input the databases generated with the same configuration that produced the compared trajectory results. As a reminder, for those comparisons, we used the left color camera and the 3D LIDAR of the car sensor setup. Due to the large number of reconstructions and the difficulty of showing the 3D reconstructions in a paper report, we only show a few of them.

For example, figure 6 shows a detail of the reconstruction of sequence 07. Sequence *08* is one of the most complex and large. A top view of its 3D reconstruction is shown in figure 7.

As mentioned in section 3.2, we also tried to discard the photometric information and use only the LIDAR data provided in the KITTI dataset. Unfortunately, we could not obtain any good reconstruction at the time of writing this report.

4.3.2. 3D reconstruction for the Beamagine data

In this case, we used the adaptation described in section 3.2 with a dataset of 8 sequences, where each one contains a hundred frames. As a reminder, in these sequences, we only had data coming from the 3D LIDAR. In spite of this situation, we were able to obtain some reconstruction results. For instance, figure 8a shows part of a reconstruction that corresponds to the photo on the side (8b). The photos were taken some days after the dataset capture from a similar point of view. Also, the point of view is similar to the one of the depth and intensity frames from figures 2 and 3.



(a) (b)

Fig. 8: RTAB-Map unscaled reconstructions, (a) Mesh reconstructed from LIDAR infrared and depth data, and (b) Photo taken some days after the capture (for comparison purposes).

5. CONCLUSIONS

We have successfully adapted two SLAM systems (Kintinuous and RTAB-Map) to work with LIDAR data. We have obtained a quantitative baseline with indoor RGBD data by evaluating the mapping (reconstruction) and location (trajectory) performance of both systems. Besides this, we have carried out a trajectory evaluation with outdoor LIDAR data. All these objective evaluations have been performed on publicly available datasets with annotated ground-truth.

Additionally, we have tested the best system in a LIDAR dataset lacking a visible color camera, thus only exploiting the metric information of the LIDAR and the infrared values of the projected scan points. We propose an interpolation method for the empty areas to allow for feature detectors needed by SLAM algorithms for correspondence matching. With this challenging data we have obtained some reconstructions from the streets of Terrassa, a city near Barcelona where UPC has one of its campuses. We would like to highlight the following points resulting from our exploration:

- In indoor real scenarios RTAB-Map is slightly better than Kintinuous for trajectory estimation. However, based on point-to-point map differences, Kintinuous is better in 3D reconstruction for synthetic indoors with simulated Kinect noise, probably thanks to its TSDF model. For outdoor real data RTAB-Map undoubtedly performs better based on ATE.
- With the scan projections done in the KITTI dataset about 84% of the available 3D points are lost, hence, the obtained reconstructions have low density of points in the parts that are not captured by the camera FOV.
- For data captured with less than 6DoF (like KITTI), the cubic shape of the TSDF volume (used in Kintinuous) is a waste of resources, since a large part of it is never used.

- The use of a sparsely sampled infrared image in place of a high resolution visible image makes the SLAM problem more difficult, but simplifies the sensor setup.
- Dynamic movements of objects break the assumption of a static world producing duplications in the map.

As future work, we would like to continue with: a precise intrinsic calibration of the Beamagine LIDAR, a registration of a hi-res color camera with the Beamagine LIDAR data, and exploiting the advantages of the real-time ROS wrapper for RTAB-Map.

6. REFERENCES

[1] P. F. McManamon, "Review of ladar: a historic, yet emerging, sensor technology with rich phenomenology," *Optical Engineering*, vol. 51, no. 6, 2012.

[2] H. Moravec, "Robot spatial perception by stereoscopic vision and 3d evidence grids," *Perception*, 1996.

[3] N. Ayache and O. D. Faugeras, "Building, registrating, and fusing noisy visual maps," *The International Journal of Robotics Research*, vol. 7, no. 6, pp. 45–65, 1988.

[4] E. Eade and T. Drummond, "Scalable monocular slam," in *Computer Vision and Pattern Recognition*, vol. 1. IEEE Computer Society, 2006, pp. 469–476.

[5] M. A. Garcia and A. Solanas, "3d simultaneous localization and modeling from stereo vision," in *Robotics and Automation ICRA'04*, vol. 1. IEEE, 2004, pp. 847–853.

[6] H. Surmann, A. Nüchter, and J. Hertzberg, "An autonomous mobile robot with a 3d laser range finder for 3d exploration and digitalization of indoor environments," *Robotics and Autonomous Systems*, vol. 45, no. 3-4, pp. 181–198, 2003.

[7] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *IEEE Intl. Symposium on Mixed and Augmented Reality*, Oct 2011, pp. 127–136.

[8] "Point cloud library." [Online]. Available: http://pointclouds.org/

[9] M. Pirovano, "Kinfu - an open source implementation of Kinect Fusion+ case study: implementing a 3d scanner with PCL," UniMi, Tech. Rep., 2012.

[10] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *23rd Annual Conference on Computer Graphics and Interactive Techniques*. New York: ACM, 1996, pp. 303–312.

[11] T. Whelan, M. Kaess, and M. Fallon, "Kintinuous: Spatially extended kinectfusion," *RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, p. 7, 2012.

[12] F. Heredia and R. Favier, "Kinect Fusion extensions to large scale environments," 2012. [Online]. Available: http://www.pointclouds.org/blog/srcs/fheredia/

[13] T. Whelan, H. Johannsson, M. Kaess, J. J. Leonard, and J. McDonald, "Robust real-time visual odometry for dense RGB-D mapping," *IEEE Intl. Conference on Robotics and Automation*, pp. 5724–5731, 2013.

[14] T. Whelan, M. Kaess, J. J. Leonard, and J. McDonald, "Deformation-based loop closure for large scale dense RGB-D SLAM," *IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, pp. 548–555, 2013.

[15] F. Endres, J. Hess, N. Engelhard, and J. Sturm, "An evaluation of the RGB-D SLAM system," *ICRA*, vol. 3, no. c, pp. 1691–1696, 2012.

[16] "ROS.org | Powering the world's robots." [Online]. Available: http://www.ros.org/

[17] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard, "3-D Mapping With an RGB-D Camera," *IEEE Trans. on Robotics*, vol. 30, no. 1, pp. 177–187, 2014.

[18] T. Whelan, S. Leutenegger, R. Salas Moreno, B. Glocker, and A. Davison, "ElasticFusion: Dense SLAM Without A Pose Graph," *Robotics: Science and Systems XI*, 2015.

[19] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[20] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," *IEEE International Conference on Intelligent Robots and Systems*, pp. 573–580, 2012.

[21] L. Jacky, "png_to_klg." [Online]. Available: https://github.com/HTLife/png_to_klg

[22] J. R. Casas, P. Salembier, and L. Torres, "Morphological interpolation for texture coding," *IEEE Intl. Conference on Image Processing*, vol. 1, pp. 526–529, 1996.

[23] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," *IEEE International Conference on Intelligent Robots and Systems*, pp. 573–580, 2012.

[24] A. Handa, T. Whelan, J. McDonald, and A. J. Davison, "A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM," *IEEE International Conference on Robotics and Automation*, pp. 1524–1531, 2014.