2016 IEEE International Conference on Multisensor Fusion
and Integration for Intelligent Systems (MFI)
Kongresshaus Baden-Baden, Germany, Sep. 19-21, 2016

TinySLAM Improvements for Indoor Navigation

Arthur Huletski[1], Dmitriy Kartashov[2] and Kirill Krinkin[3]

Abstract— **TinySLAM [1] is one of the most simple SLAM methods but the original implementation [2] is based on the specific robot model and provided as the ad-hoc application. Its key feature is simplicity of implementation and configuration at cost of accuracy (as our tests shown). Some changes were made in the original algorithm in order to minimize an error of estimated trajectories. The introduced model of cell leads to an error decrease on almost all tested indoor sequences. The proposed dynamic probability estimator improves usage of coarse-grained maps when memory efficiency is more desirable than accuracy. Obtained quantitative measurements justify the changes made in tinySLAM in case the method is used in a relatively small environment.**

I. Introduction

Simultaneous localization and mapping (SLAM) is the problem of simultaneous computation of a mobile robot position within an unknown environment and reconstructing the environment structure without prior knowledge about it. There are plenty of different SLAM methods, but in practice some "complicated" methods require tuning of a very large number of parameters to fit a specific environment or involve manual processing of output data [4]. Taking into account the above, a quantitative evaluation of localization quality of the simple SLAM method (in terms of implementation, configuration and description) with respect to more sophisticated ones seems to be useful.

SLAM methods based on probabilistic estimation of a robot position and environment characteristics are widely used as an approach to the SLAM problem [5]. The key idea of such probabilistic methods is a computation of an initial estimation of the robot position using odometry data and adjustment of this estimation by matching current sensor data with the previously constructed environment map. The naïve method that calculates probability of the robot state with an exhaustive search is not useful due to exponential domain size of a probability distribution function. So a variety of methods which approximate the required distribution (e.g. extended Kalman filter or particle filter) are used in practice.

One of the factors that determines the structure of a SLAM method is a set of available sensors. An odometry measurement unit and a laser scanner are often used as an input source for a method. Examples of the methods that use such sensor set are GMapping [3], Hector SLAM [6], tinySLAM [1], DP-SLAM [7] and some others.

[1]Arthur Huletski is with the Academic University, Saint-Petersburg, Russia; hatless.fox@gmail.com.

[2]Dmitriy Kartashov is with The Academic University, Saint-Petersburg, Russia; dmakart@gmail.com.

[3]Kirill Krinkin is with the St. Petersburg State Electrotechnical University "LETI", Saint-Petersburg, Russia; kirill.krinkin@fruct.org.

According to the tinySLAM authors [1], it was created with the aim to be simple to implement and easy to understand. Unfortunately, the available implementations are provided either as the ad-hoc application [2] that makes testing on public datasets a bit difficult or incompatible with modern ROS versions [8]. The CoreSLAM implementation of tinySLAM was evaluated in [9] and compared with some other 2D SLAM methods. According to the testing results, CoreSLAM has poor accuracy and builds low-quality maps, especially in comparison with GMapping which is based on Rao-Blackwellized particle filter [10] and tracks multiple hypotheses of a world state. Results also show that GMapping is one of the most accurate 2D SLAM methods available in ROS [11].

The authors of the tinySLAM method have proposed several improvements for the original implementation focused on a localization quality increase [12]. It was mainly suggested to use filtering to determine whether an estimated pose is good enough to add to a map. Another improvement is a new scan matcher based on a genetic algorithm. A parallel localization with two scan matchers configured with different parameter sets was also proposed. The negative effect of suggested filtering is an extra threshold value that should be determined empirically, i.e. an extra assumption about an environment is introduced. The optimized parallel localization approach and usage of genetic algorithms improve accuracy of the method in cost of code simplicity and size which makes the method less clear and less simple.

Considering all of the above, it was decided to implement the tinySLAM algorithm as a ROS node, improve its accuracy keeping the code compact, and compare it with GMapping in relatively small environments. In such environments most of laser scanner measurements are valid, i.e. they are maximally covered by a laser scanner. Changes proposed in this paper are mainly focused on improving of the map cell model rather than addition of heuristics to the scan matcher or the method itself.

The rest of the paper is organized as follows. In the next section a brief overview of the tinySLAM algorithm is given. In Section III. implementation details and suggested optimizations and improvements are described. Section IV. discusses performance and accuracy of the implemented method. Finally, in Section V. a summary of this paper is given and some subjects for further development are listed.

II. TinySLAM

TinySLAM uses odometry and laser scans as input data and represents knowledge about an environment as a certainty grid and a robot position as a 2D oriented pose. The

978-1-4673-9708-7/16/$31.00 ©2016 IEEE

same is true for GMapping. The main difference between these two methods lies in the fact that tinySLAM tracks a single hypothesis about the world state.

The robot pose is refined by a Monte-Carlo-based method. The method changes it by addition of independent normally distributed random values to each of robot state's component (x, y, θ). Then it makes a hypothesis about the best robot position. The hypothesized position is checked by matching the current laser scan data in it against the map state. The matching function computes the sum of range measurements that hit occupied map cells. If the hypothesis has a better matching score then it becomes the current robot pose.

The certainty map used by tinySLAM is updated with scan values by the following formulae:

$$map_{n+1}(x, y) = \bar{q} \cdot map_n(x, y) + q \cdot value \qquad (1)$$

where:

- $map_n(x, y)$ is the current cell value;
- q, $\bar{q}$ are the constants that define the "variance" of the value being fused into the cell;
- $value$ is the binary value that needs to be incorporated into the map: 1 if the cell is occupied and 0 otherwise.

TinySLAM also implements the following ad-hoc features: obstacle blurring and scan interpolation. Blurring of occupied map parts increases an occupancy probability value of the cells near the occupied one depending on the distance.

Also the original implementation preprocesses laser scan data and increases a number of range readings using interpolation.

III. TinySLAM implementation

ROS (Robot Operating System) provides various system services such as hardware abstraction, low-level device control, implementation of commonly used functionality and message-passing between processes [11]. Although ROS itself is not a real-time OS, these features simplify development and testing of robot software. It was decided to implement tinySLAM as a ROS package.

The following changes were made in the original algorithm:

- an explicit limit on a total number of iterations of the Monte-Carlo method was added to bound from above the time spent on scan matching;
- robot movement data come from ROS topics and are not computed by a specific robot motion model to abstract the algorithm from robot's internals;
- laser scans are added to the map with a higher "disparity" value (i.e. lower quality is assumed) when the robot pose was refined by the scan matcher. It makes the algorithm more robust since the pose correction returned by the matcher is not guaranteed to be optimal;
- the interpolation of laser scan data was removed.

A. Modified cell model

Another modification of the algorithm is a new model of a grid map cell that allows to build more accurate maps and increases robustness of the method. The new model combines benefits of the "average" nature of GMapping's model with the notion of quality used by tinySLAM. The average of all stored effective values is treated as cell's occupancy probability. A new effective value used for a cell update is computed by the following formulae:

$$value_{\text{eff}} = 0.5 + (value - 0.5) \cdot quality \qquad (2)$$

where:

- $value$ is the assumed cell value ($0 \leq value \leq 1$);
- $quality$ is the quality (the absence of noise) of the given value.

As the formulae states, the quality argument controls the deviation of cell's default probability of being occupied. This semantics differs from the one used by the original tinySLAM cell model where the argument is used to weight a given cell value. This is done to mix with the previously computed probability by an affine combination.

B. Dynamic occupancy estimation

The original tinySLAM implementation uses constant values for a cell update but actual cell's occupancy probability depends on how a laser beam splits the cell. Some SLAM algorithms use this information to increase accuracy of an estimated map. For example, DP-SLAM [7] uses a path traversed by a beam inside a cell to estimate effective probability: the longer the path the greater cell's probability of being empty.

An estimator that computes occupancy probabilities for cells that contain obstacles is proposed. Probability computation is based on areas' ratio: the area of the cell chunk supposed to be occupied to cell's total area. To estimate the occupied chunk area the cell is split into two parts by the normal to a laser beam. The part that lies on the side opposite to the robot's one with respect to the normal is treated as occupied (Fig. 1).

The estimator was added to the implementation as an option disabled by default. The details on the estimator efficiency are provided in Section V.

IV. TinySLAM evaluation

The TinySLAM evaluation was performed using the MIT Stata Center dataset [14] that provides laser scan data. The dataset is one of a few datasets of that kind designed to be used with ROS, i.e. provides ROS bag-files. It is obtained using the PR2 robot platform that is equipped with the

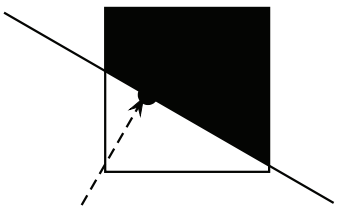


Fig. 1. An occupancy estimator example. The dashed line is a beam, the solid line is the normal to the beam, the cell chunk filled with black is supposed to be occupied.

TABLE I
ACCURACY TESTING RESULTS FOR TINYSLAM AND GMAPPING

Sequence	Length, m	Trajectory RMSE, m		
		tinySLAM (base)	tinySLAM (mod)	GMapping
2011-01-18	86	3.773 ± 0.353	2.458 ± 0.099	3.217 ± 0.022
2011-01-19	68	0.756 ± 0.372	0.945 ± 0.286	0.216 ± 0.012
2011-01-20	76	0.276 ± 0.063	0.164 ± 0.036	0.206 ± 0.040
2011-01-21	87	0.773 ± 0.303	0.201 ± 0.046	0.198 ± 0.017
2011-01-24	87	0.784 ± 0.130	0.107 ± 0.014	0.332 ± 0.104
2011-01-25	109	0.528 ± 0.058	0.141 ± 0.021	0.222 ± 0.008
2011-01-27	94	3.477 ± 1.493	0.198 ± 0.065	0.242 ± 0.029
2011-01-28	145	3.945 ± 1.251	1.481 ± 0.612	2.388 ± 1.949
2011-03-11	245	2.352 ± 0.548	0.605 ± 0.134	0.490 ± 0.028
2011-03-18	80	0.379 ± 0.029	0.103 ± 0.008	0.145 ± 0.001
2011-04-06	95	0.396 ± 0.148	0.105 ± 0.017	0.185 ± 0.002
2011-10-20	264	3.498 ± 0.659	2.214 ± 0.314	0.317 ± 0.007
Edmonton 2002	327	1.407	1.283	—

Hokuyo UTM-30LX laser scanner (40 Hz scan frequency, 0.1 to 30 m detection range). The dataset contains 84 sequences in total but some of them are multi-floor (both tinySLAM and GMapping are not supposed to work in 3D) and some ones do not have corresponding ground truth data. It was selected 12 out of 20 suitable sequences that traverse a single floor only and have confirmed ground truth.

Additionally, the ROS-compatible version of the single-sequence dataset "Edmonton 2002" [15] was used to test the methods on longer distances. Unfortunately, the authors of this dataset do not provide details about the equipment was used.

A. Performance

Performance of the available GMapping package and our tinySLAM implementation can not be directly compared due to non-constant overhead introduced by ROS which is not a real-time system. The information below is provided for a relative comparison of methods' performance.

The total time of pose refinement and map update operations was measured during accuracy testing of tinySLAM. The testing was performed on Core i7-6700HQ CPU (2.6 GHz) with 8 GB DDR4 RAM using only a single core. On this hardware robot localization and map update operations took 6 ms on the 200 m × 200 m map (4000 × 4000 cells, 256 MB total data size). However, performance of a map update operation depends on a scan size and measured distances to obstacles rather than the map size, since the scan matcher analyzes only a local area near a robot. Processing time increases on bigger maps due to map publishing to ROS topics. Thereby the implemented algorithm can process sensor data and build a map in real time, especially on indoor environments which typically have a lot of obstacles.

Runtime analysis of GMapping was performed in [13] on a 2.8 GHz CPU. According to the authors, GMapping requires 2-3 s to perform a map update operation on the 40 m × 40 m map (800 × 800 cells) with 30 particles. Note that this is a total update time, i.e. it summarizes map update time measurements of each particle. GMapping consumes 150 MB of memory to store all the particles in this setup. TinySLAM which use a single hypothesis only requires approximately 10 MB under the similar conditions.

B. Accuracy

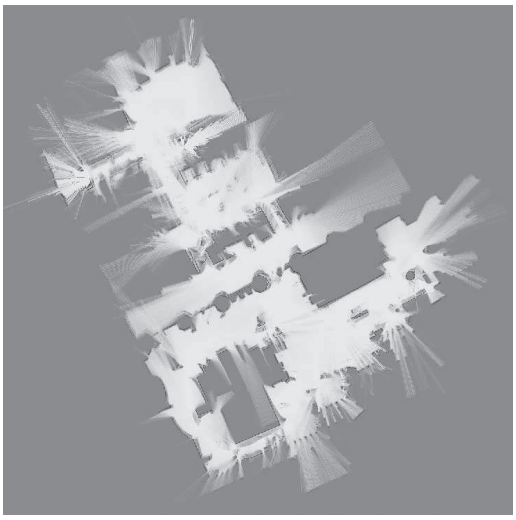
The trajectories obtained with tinySLAM and the available GMapping implementation [16] were compared with the ground truth data to evaluate their accuracy. GMapping was used "as is", i.e. default values were used for all parameters except the map resolution which was set to 0.2 m for all tested sequences. The trajectories were compared using the TUM's SLAM evaluation tool [17]. The mean of RMSE over several runs were used as an accuracy measure. Unfortunately, the "Edmonton 2002" dataset provides only the graphical image of its map as ground truth (Fig. 2a) and doesn't contain any numeric data for robot's trajectory. So we compared tinySLAM and GMapping results assuming the GMapping trajectory as the ground truth for this dataset.

The testing results for tinySLAM with modified and original cell models are provided in Table I. Resulting maps for the "Edmonton 2002" dataset are shown in Fig. 2. A comparison of resulting trajectories for some sequences are shown in Fig. 3. The figures show that the generated maps and trajectories are rather close to each other.

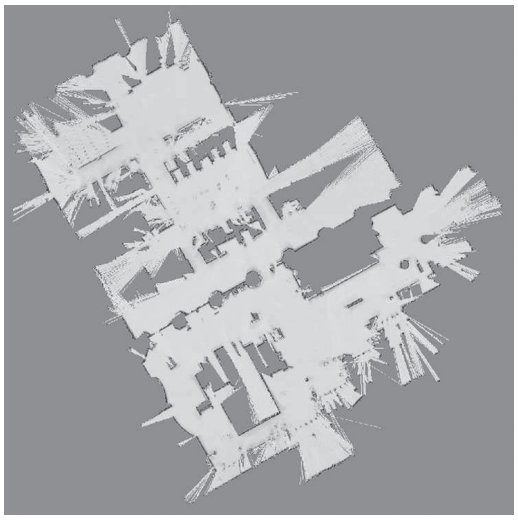
The results show that the accuracy of tinySLAM with the modified cell model is comparable with the accuracy of GMapping, especially in irregular environments that can be maximally covered by a laser scanner. Unfortunately, tinySLAM poorly handles sequences with long featureless passages (e.g. "2011-10-20"). In this situation the scan matcher tends to stick robot's coordinates to a fixed position. Tracking of multiple hypotheses implemented in GMapping produces far more accurate results than tinySLAM on such data sequences.

(a) Ground truth

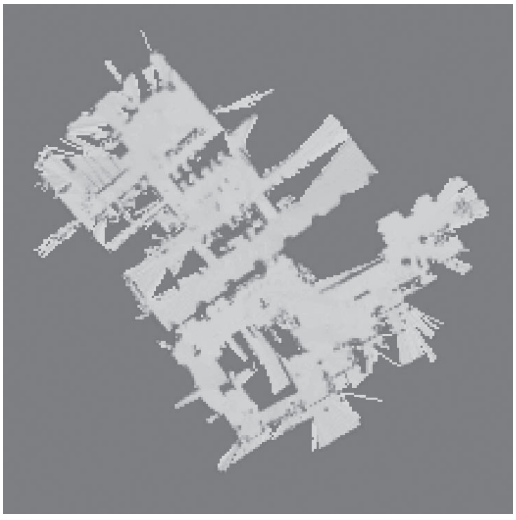
(b) GMapping



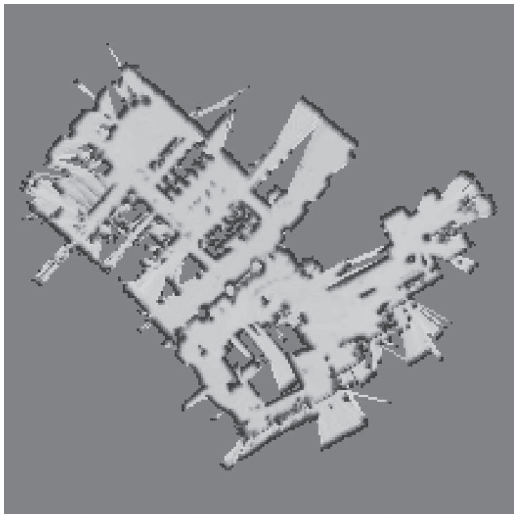
(c) tinySLAM (the base cell model)



(d) tinySLAM (the modified cell model)

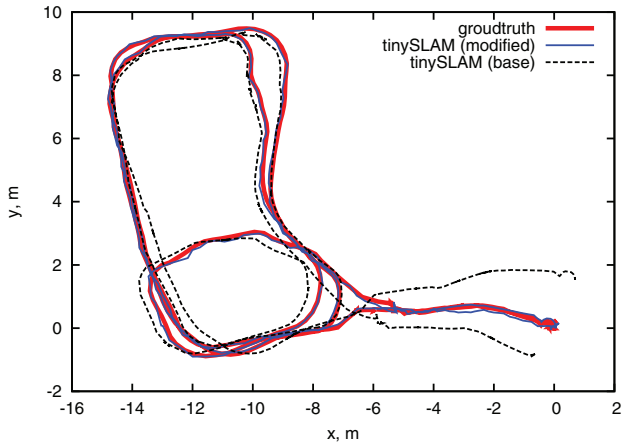


(e) tinySLAM with the dynamic occupancy estimator (the map has 0.6 m cell size)

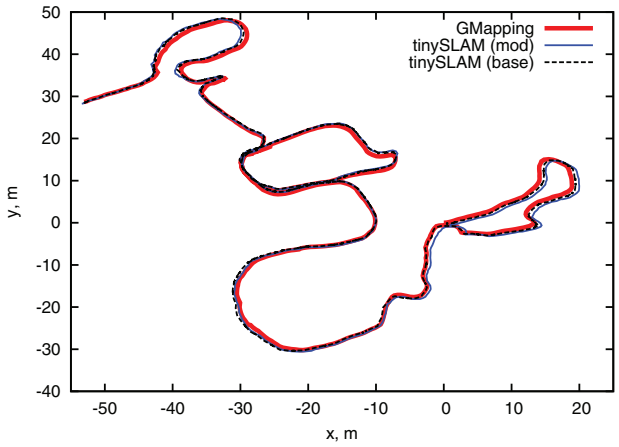


(f) tinySLAM with the modified cell only (the map has 0.6 m cell size)

Fig. 2. SLAM maps for "Edmonton 2002"



(a) tinySLAM with the base cell model vs the modified cell model



(b) Edmonton 2002

Fig. 3. SLAM trajectories examples

TABLE II
COMPARISON OF CELL MODELS

Actual occupancy	Fwd Base Cell	Bwd Base Cell	Fwd Mod. Cell	Bwd Mod. Cell
0.25	0.105	0.502	0.245	0.245
0.5	0.225	0.737	0.48	0.48
0.75	0.460	0.858	0.715	0.715

C. Comparison of base and modified cell models

As it was mentioned above, tinySLAM's cell model switching has the greatest impact on accuracy of the algorithm. Better performance can be explained by the following reasons:

- The modified cell model tracks the average of ever estimated occupancies of a cell. This feature makes the model more robust to a sensor noise comparing with the base one.
- The occupancy probability estimated by the base cell model depends on the order of cell state updates: the more cell updates are performed after a given one the less the update influences on cell's occupancy probability. In contrast, the probability estimated by the modified cell model doesn't depend on the order of updates.

Fig. 4 shows the scheme of an experiment that was performed to justify the last reason and to obtain quantitative data. A robot model with a single laser beam is used in this experiment. The robot tries to estimate the occupancy probability of a single cell which is partially occupied. The cell is split into two parts by a line parallel to the laser beam: the left part of the cell is treated as occupied, the right one – as empty. The robot moves in two directions: forward (the occupied part is observed at first) and backward (the empty part is observed at first). For the sake of the experiment no random noise is assumed, so the quality parameter is set to 1.0 for the modified cell model and to 0.2 for the base cell model (as in the original tinySLAM implementation). The observed obstacle leads to a cell update with occupancy probability equals to 0.95, the absence of the obstacle – 0.01. The robot moves parallel to a side of the cell with the speed of 1/12 of cell's side (so 12 measurements are taken into account).

Table II provides probabilities estimated by the models depending on the actual cell occupancy. The obtained results show a significant probability estimation error (approx. 50%) on a small number of trials (12) and a dependency on the update order (forward or backward). Such errors can significantly decrease accuracy of a scan matcher when an unknown part of an environment (e.g. a room) is explored.

D. Occupancy estimator accuracy

The dynamic occupancy estimator accuracy was evaluated on the "2011-01-24" sequence of the MIT Stata dataset using different sizes of map cells. RMSE of the trajectories obtained using the modified cell model only and the same model with the estimator are shown in Table III. The table

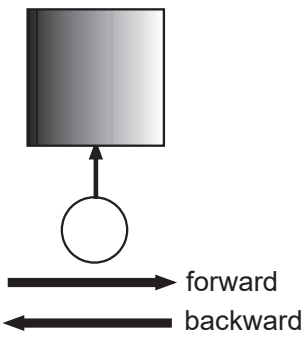


Fig. 4. The scheme of the cell evaluation experiment

TABLE III
DYNAMIC OCCUPANCY ESTIMATOR ACCURACY

Method	Trajectory RMSE for different cell sizes, m					
	0.1 m	0.2 m	0.3 m	0.4 m	0.5 m	0.6 m
Modified cell only	0.107(±0.004)	0.093(±0.009)	0.116(±0.028)	0.183(±0.032)	0.248(±0.038)	0.409(±0.147)
Mod. cell + dyn. occ. estimator	0.107(±0.025)	0.111(±0.013)	0.153(±0.014)	0.188(±0.019)	0.197(±0.037)	0.308(±0.067)

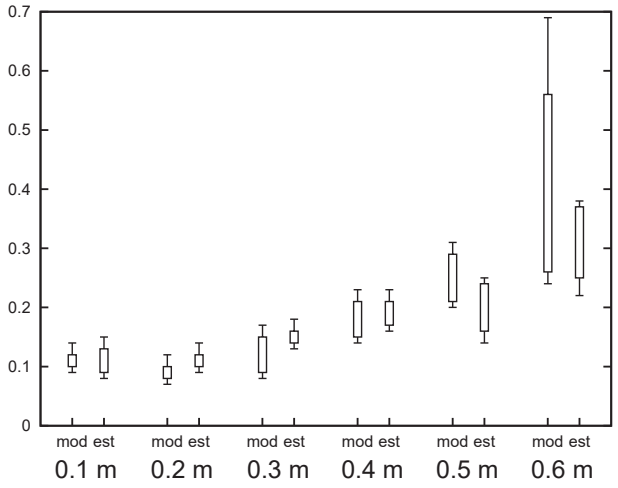


Fig. 5. RMSE of the modified cell model (mod) and the model with the dynamic occupancy estimator (est) on different cell sizes

contains means and standard deviations of RMSE over 20 runs. The testing results are also visualized in Fig. 5. Fig. 2e and Fig. 2f show the maps with 0.6 m cell size for the "Edmonton 2002" dataset that are obtained using the estimator and the modified cell model only respectively.

According to the results, the estimator significantly improves localization quality on big enough cells but is inefficient on maps with small cells. Thus the estimator may be used as a trade-off between localization quality and map granularity when available memory is limited.

V. CONCLUSION

The developed tinySLAM modification shows acceptable results for indoor robot navigation. The proposed modified cell model increases accuracy of an estimated trajectory for small indoor environments without patterns repeated with high frequency. The introduced dynamic probability estimator decreases accuracy penalty when a coarse-grained map is used.

Taking into account high performance and compactness of its code, the tinySLAM method fits well low-cost mobile robots which operate in indoor environments (e.g. autonomous vacuum cleaners).

Further development may involve accuracy testing on outdoor datasets. Also, the scan matcher may be improved by analyzing free space near obstacles. The dynamic probability estimator may also estimate quality of an "empty cell pass" evidence that may be used by a cell model based on Dempster-Shafer theory [18].

All programs and algorithm implementations are published as an open source ROS-ready package and can be accessed by the following link: `https://github.com/OSLL/tiny-slam-ros-cpp`.

ACKNOWLEDGMENT

Authors would like to thank JetBrains Research for provided support and materials for working on this research. Some parts of the paper has been prepared within the scope of project part of the state plan of the Board of Education of Russia (task # 2.136.2014/K).

REFERENCES

[1] B. Steux, O. E. Hamzaoui, "tinySLAM: A SLAM algorithm in less than 200 lines C-language program", *in Proc. of the 11th International Conference on Control Automation Robotics Vision*, pp. 1975-1979, 2010.
[2] OpenSLAM.org, Web: https://openslam.org/tinyslam.html.
[3] G. Grisetti, C. Stachniss, W. Burgard, "Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters", *IEEE Transactions on Robotics*, pp. 34-46, 2007.
[4] A. Huletski, D. Kartashov, K. Krinkin, "Evaluation of the Modern Visual SLAM Methods", *in Proc. of the AINL-ISMW FRUCT Conference*, pp. 19-25, 2015.
[5] S. Thrun, W. Burgard, D. Fox, "Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)". Cambridge: The MIT Press, 2005.
[6] S. Kohlbrecher, J. Meyer, O. von Stryk, U. Klingauf, "A Flexible and Scalable SLAM System with Full 3D Motion Estimation", *in Proc. of IEEE International Symposium on Safety, Security and Rescue Robotics*, pp. 150-160, 2011.
[7] A. I. Eliazar, R. Parr, "DP-SLAM 2.0", *in Proc. of IEEE International Conference on Robotics and Automation*, vol.2, pp. 1314-1320, 2004.
[8] coreslam — ROS Wiki, Web: http://wiki.ros.org/coreslam.
[9] J. M. Santos, D. Portugal, R. P. Rocha, "An evaluation of 2D SLAM techniques available in Robot Operating System", *in Proc. of IEEE International Symposium on Safety, Security, and Rescue Robotics*, pp. 1-6, 2013.
[10] A. Doucet, N. de Freitas, K. P. Murphy, S. J. Russell, "Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks", *in Proc. of the 16th Conference on Uncertainty in Artificial Intelligence*, pp. 176-183, 2000.
[11] ROS.org — About ROS, Web: http://www.ros.org/about-ros/.
[12] O. E. Hamzaoui, B. Steux, "SLAM algorithm with parallel localization loops: TinySLAM 1.1", *in Proc. of IEEE International Conference on Automation and Logistics*, pp. 137-142, 2011.
[13] G. Grisetti, C. Stachniss, W. Burgard, "Improving Grid-based SLAM with Rao-Blackwellized Particle Filters by Adaptive Proposals and Selective Resampling", *in Proc. of the IEEE International Conference on Robotics and Automation*, pp. 2432-2437, 2005.
[14] M. Fallon, H. Johannsson, M. Kaess, J. Leonard "The MIT Stata Center dataset", *The International Journal of Robotics Research*, vol. 32, no. 14, pp. 1695-1699, 2013.
[15] Repository of robotics and computer vision datasets — MRPT, Web: http://www.mrpt.org/robotics_datasets.
[16] gmapping — ROS Wiki, Web: http://wiki.ros.org/gmapping.
[17] Computer Vision Group — Useful tools for the RGB-D benchmark, Web: http://vision.in.tum.de/data/datasets/rgbd-dataset/tools.
[18] G. Shafer, *A mathematical theory of evidence*, Princeton university press, vol. 1, 1976.