Comparison of ROS-based Visual SLAM methods in homogeneous indoor environment

Conference Paper · October 2017

DOI: 10.1109/WPNC.2017.8250081

2 authors:

Ilya Afanasyev
Innopolis University
95 PUBLICATIONS **303** CITATIONS
SEE PROFILE

Ilmir Ibragimov
1 PUBLICATION **18** CITATIONS
SEE PROFILE

Comparison of ROS-based Visual SLAM methods in homogeneous indoor environment

Ilmir Z. Ibragimov and Ilya M. Afanasyev

***Abstract* — This paper presents investigation of various ROS-based visual SLAM methods and analyzes their feasibility for a mobile robot application in homogeneous indoor environment. We compare trajectories obtained by processing different sensor data (conventional camera, LIDAR, stereo ZED camera and Kinect depth sensor) during the experiment with UGV prototype motion. These trajectories were computed by monocular ORB-SLAM, monocular DPPTAM, stereo ZedFu (based on ZED camera data) and RTAB-Map (based on MS Kinect 2.0 depth sensor data), and verified by LIDAR-based Hector SLAM and tape measure.**

Keywords — ROS, visual SLAM, visual odometry, ORB-SLAM, DPPTAM, RTAB-Map, Kinect, LIDAR, ZED camera

I. Introduction

One of the most important challenge for mobile robotics applications is the ability to navigate mobile robots in an unknown complex environment. Having a map and exact location on the map allows to navigate in environment and to predict a path to a target. Both robot location determination and mapping do not make any difficulties if onboard sensors are precise enough. The difficulty is to solve both of these problems at the same time. Often the challenge arises when a robot does not know neither map nor location. Simultaneous Localization and Mapping. SLAM problem is related to constructing a map of unknown environment while simultaneously keeping track of an agent's location within it. Depending on applied algorithm the robot can use various sensors: laser rangefinder, IMU-sensor, sonar, altimeter, a depth camera or conventional RGB camera and so on. Over the last 10 years actively began to develop SLAM methods based on computer vision algorithms [2 – 8, 11, 13 – 19]. Problems of autonomous navigation and mapping require fast, robust and precise real-time SLAM algorithms. Robot navigation with laser rangefinder is quite expensive, but one of the most effective way. However, nowadays computer vision algorithms allow to use RGB camera [5], stereo camera (ZED camera) [7, 13] or RGB-D Kinect depth camera [5, 13]. The main challenge is to use computer vision-based algorithms for indoor navigation within office-style environment, where walls can be painted in homogeneous colors and can contain glass and mirrors.

One of the first real-time monocular SLAM methods was created by Andrew Davison [4] about 10 years ago and this method is called MonoSLAM. Over the last decade the number of monocular SLAM-related methods has grown, including Parallel Tracking and Mapping (PTAM, [11]), Dense Tracking and Mapping (DTAM, [15]), Large-Scale Direct Monocular SLAM(LSD-SLAM, [5]), Regularized MOnocular Depth Estimation (REMODE, [16]), Oriented FAST and Rotated BRIEF [2] (ORB-SLAM, [14], [17]) etc. One of the most modern methods is Dense Piecewise Planar Tracking and Tracking (DPTTAM, [3]), which contains all the best features of other monocular SLAM methods. So recently the developers of LSD-SLAM [5] have created a new monocular SLAM method Direct Sparse Odometry (DSO, [6]). Although a qualitative comparative analysis of ROS-based monocular SLAM methods was performed in the article [1], in this paper we present also qualitative comparison of different SLAM methods based on visual monocular, stereo, and RGB-D sensors, verified by ground truth (that was established by processing of LIDAR data with ROS-based Hector SLAM package, which has demonstrated high precision and robustness to different deviations of robot motion [20]).

I. Z. Ibragimov and I. M. Afanasyev are with the Robotics Institute, Innopolis University, Universitetskaya str. 1, 420500 Innopolis, Russia (+7-843-203-9253 ext. 177; e-mail: {i.ibragimov, i.afanasyev}@innopolis.ru).

II. System Setup

For our experiments we created a human-operated prototype of Unmanned Ground Vehicle (UGV), which was equipped with a computing system and a set of sensors (Fig. 1). To analyze ROS-based visual SLAM-related methods we recorded raw videodata streams from industrial camera Basler acA2000-50gc GigE1 (Fig. 2b) mounted on UGV prototype. The UGV computational platform is based on Intel Core i3 processor and GeForce GT740M graphics card, which supports CUDA technology, producing on-board parallel computation. Table I describes UGV hardware and software parameters and sensors set.

TABLE I. The UGV prototype system configuration

Parameters	Configuration
UGV hardware	
Processor	Intel Core i3-416 CPU @ 3.60GHz x 4
GPU	GeForce GT 740M
RAM	8 GB
Sensors	
LIDAR	HOKUYO UTM-30LX[a]
Camera	Basler acA2000-50gc GigE[b]
Stereo camera	Stereolabs ZED camera[c]
RGB-D sensor	Microsoft Kinect 2.0[d]
Software	
OS	Ubuntu 14.04
ROS	Indigo Igloo

[a.] Hokuyo Automatic Co. 2D Lidar (Fig. 2a), *www.hokuyo-aut.jp*

[b.] Area Scan Camera from Basler AG Company (Fig. 2b), *www.baslerweb.com/en/products/cameras/*

[c.] ZED camera from Stereolabs company (Fig. 2c), https://www.stereolabs.com

[d.] MS Kinect 2 depth sensor (Fig. 2d), www.xbox.com/xbox-one/accessories/kinect

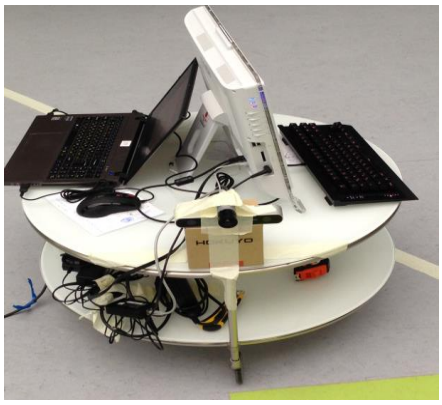


Figure 1. Human-operated prototype of Unmanned Ground Vehicle (UGV).

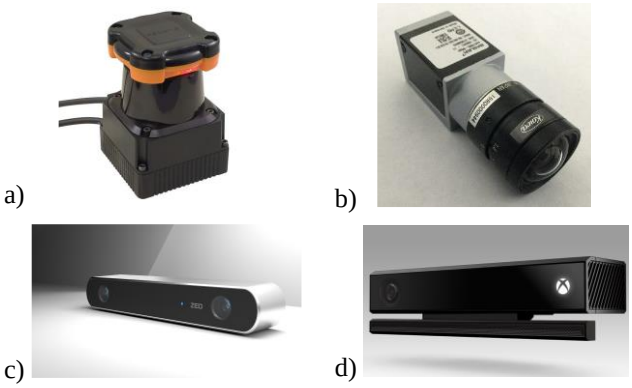


Figure 2. Sensors of the UGV prototype: a) Hokuyo UTM-30LX laser rangefinder. b) Basler acA2000-50gc camera. c) Stereolabs ZED camera. d) Microsoft Kinect 2.0 depth sensor. Courtesy of sensors manufacturers.

III. Description of the Experimental Environment

We marked trajectory inside typical office-style indoor environment. This means that we know all of values (scale, angles) of the real UGV trajectory. Test environment contains monochrome, homogeneously painted walls and glass protection fences (Fig. 3). The UGV prototype has strictly moved along a marked white line. Thus, initially we turned on all the sensors of the UGV prototype, which was started moving under a human operator control along the marked white line. Simultaneously, we recorded raw videodata stream *ros_image_raw* in *rosbag*[1] file. At the same time, ROS-based LIDAR package *hector_slam*[2] is running. After a successful check-in, the video stream was processed by monocular SLAM algorithms. We made verification LIDAR trajectory with real UGV trajectory. After that, we verified trajectories of monocular SLAM algorithms with LIDAR trajectory and real UGV trajectory along white line. The same experiments we provided with the ZED stereo camera and RGB-D Kinect sensor. To keep track of statistics with experimental goals, we conducted these tests with UGV prototype motion several times. The results are discussed in the next sections.

[1] *Rosbag* is a set of tools for recording and playback ROS topics: *http://wiki.ros.org/rosbag*

[2] ROS package *hector_slam* is available at https://github.com/tu-darmstadt-ros-pkg/hector_slam

Figure 3. Workspace for UGV prototype motion along white line.

IV. Tests of SLAM Methods

A. Lidar SLAM method

Hector SLAM [12] is based on data obtained from the 2D LIDAR. It was created by PhD students of the Darmstadt University in 2011 with designing your own UGV prototype. Hector SLAM is one of the most popular SLAM methods and is used in many projects related with mobile robots. Hector SLAM requires only 2D LIDAR data in LaserScan format. Algorithm is able to build a 2D map and localization in the same frequency at which LIDAR scans. In this case IMU-sensor data or odometry data is not necessary. LIDAR can be mounted with some offset and at an angle to the surface that robot moves. To correctly build a map we need to convert from coordinate system of the LIDAR to the coordinate system of the surface movement.

Hector SLAM build 2D occupancy grid map (Fig. 4), which publish in ROS topic map in (*nav_msgs/OccupancyGrid*) format. Each cell of the map is painted to its color, this is probability of occupation (black – cell is busy, light grey – cell is free, dark grey–cell is not scanned area). Also, we can see bright green trajectory line on the map (Fig. 4).

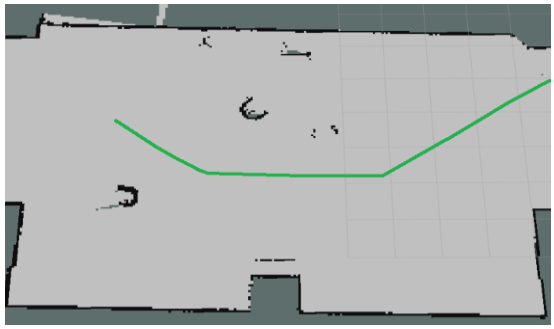


Figure 4. 2D occupancy grid map by Hector SLAM.

As mentioned before, we take 2D LIDAR data as Ground Truth. Lidar trajectory is presented in Fig. 5 as red line, black line is marked trajectory with the length of 11.5 m. It is important to note that deviation from the marked line is small enough, therefore, LIDAR data can be trusted.

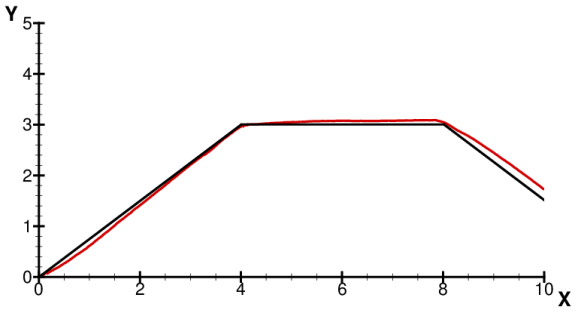
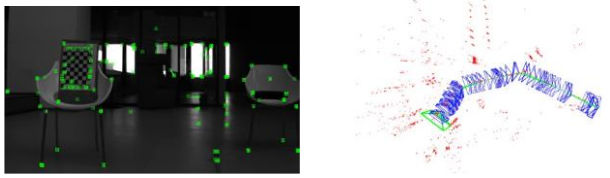


Figure 5. Trajectory (red – Hector SLAM, black – marked line). Black curve is the ground truth. The units of measure for the XY axes are meters.

B. *Monocular ORS-SLAM method*

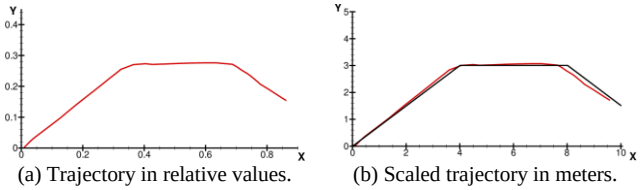
Oriented FAST and Rotated BRIEF [2] (ORB-SLAM[3]), [14], [17]) is a feature-based real-time SLAM library for monocular, stereo and RGB-D cameras. Using the Bundle Adjustment algorithm, the features from different images are placed in 3D space. ORB-SLAM calculates a camera trajectory and recovers a sparse 3D scene. Algorithm use ORB (Oriented FAST and Rotated BRIEF) fast feature detector. This detector allows to work in real-time. ORB-SLAM uses advanced approaches for loop closing, keyframe selection and localization for each frame. To process live monocular streams, the library is used with a ROS node. This library provides required calculations and a graphical user interface (GUI). Visualization of the ORB-SLAM feature detector is presented in Fig. 6a. Map Viewer visualizes 3D map (Fig. 6b) as a red point cloud, shows trajectory as a green line and (blue rectangle) plane in which the camera shoots at particular point in time.



(a) Extracted features. (b) Point cloud and trajectory

Figure 6. ORB-SLAM features and point clouds visualization.

As the result of the experiment, we obtained ORB-SLAM trajectory in relative values (Fig. 7a). The scaled trajectory is shown in Fig. 7b.

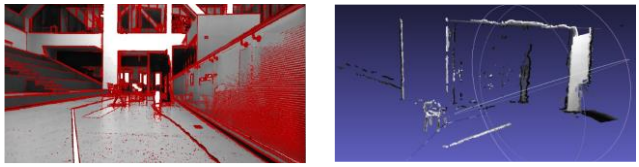


(a) Trajectory in relative values. (b) Scaled trajectory in meters.

Figure 7. ORB-SLAM trajectories (in red), marked line (in black).

C. *Monocular DPPTAM method*

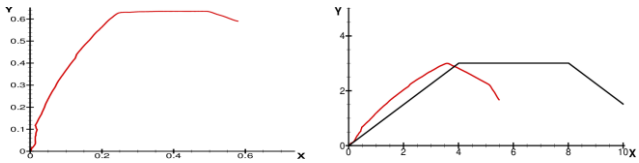
Dense Piecewise Planar Tracking and Mapping from a Monocular Sequence (DPPTAM) is one of the newest direct real-time visual SLAM methods, which adapts positive ideas of previous SLAM algorithms .DPPTAM estimates dense 3D reconstruction of scene and saves trajectory as a sequence of points in point cloud. It works with an assumption that homogeneous and monochrome regions belong to approximately planar areas. DPPTAM implementation in ROS is real-time ROS node, which does not have graphical user interface (GUI). Therefore, we visualize extracted features with standard ROS package RViz[4] (Fig. 8a) and point clouds in MeshLab[5] (Fig. 8b).



(a) Extracted features. (b) Point cloud and trajectory

Figure 8. DPPTAM features and point clouds visualization.

Finally, we estimated DPPTAM trajectory in relative values (Fig. 9a), and scaled trajectory according to the marked white line (Fig. 9b). As far as there is no coincidence of these trajectories, is means the failure of visual odometry with DPPTAM.



(a) Trajectory in relative values. (b) Scaled trajectory in meters.

Figure 9. DPPTAM trajectories (in red), marked line (in black).

D. *Stereo SLAM method with ZED camera*

Stereolabs developers provide ZED camera with various software tools and interfaces, from which we used ZED SDK[6], ZED Calibration Tool and ZEDfu software for real-time 3D mapping and 3D scene generation in the form of point cloud. ZED SDK has the integration with ROS, which is performed with *zed-ros-wrapper*[7] package. After the experiment with UGV prototype, a point cloud was recorded in SVO format and processed by ZEDfu tool (Fig. 10). We can see from Fig. 10 that the map is dense and all the important obstacles such as chairs and tables are displayed.

To build the UGV trajectory based on the ZED camera visual odometry data we used the ROS package and existing Position Tracking visualization tool in OpenGL window. The calculated trajectory is displayed in Fig. 11a. The comparison of trajectories from ZED camera visual odometry and marked line is shown in Fig. 11b, where we can see that maximal error of trajectories mismatch is about 10 cm.

[3] ORB-SLAM2 package, github.com/raulmur/ORB SLAM2

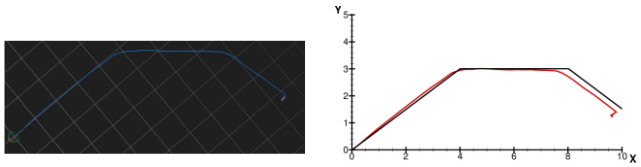
[4] RViz is 3D visualization tool for ROS, wiki.ros.org/rviz

[5] 3D mesh processing, viewing and editing software: www.meshlab.net

[6] ZED SDK: www.stereolabs.com/developers/documentation/API/

[7] ROS package for ZED: github.com/stereolabs/zed-ros-wrapper

Figure 10. ZEDfu 3D map.

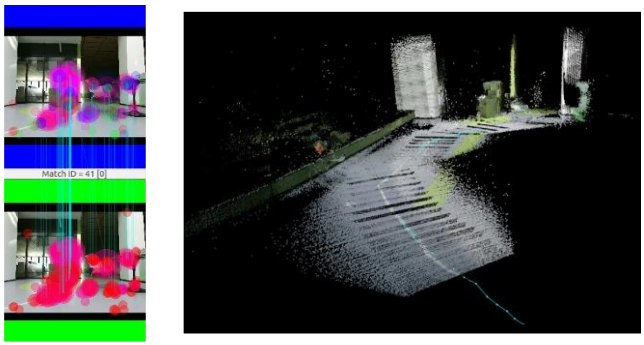


(a) Path in ROS Position Tracking. (b) Scaled trajectory in meters.

Figure 11. ZED Camera trajectories (in red), marked line (in black).

E. RTAB-Map SLAM with Kinect RBG-D camera

Real-Time Appearance-Based Mapping (RTABMap, [13]) is an algorithm of visual graph SLAM, based on closure detection. Detector uses an algorithm *bag-of-words* to determine the level of equality of new camera images to images of already visited locations. Each of such closures adds a new edge to a camera position graph, and then graph is optimized. RTAB-Map[8] provide a graphical interface, which visualizes the output of loop closure detector (Fig. 12a), visual odometry and point cloud (Fig. 12b). Also it allows to filter and save point cloud in PLY or PCD formats. RTAB-Map supports operation with Kinect RGB-D camera, ZED stereo camera, conventional stereo pair, etc.



(a) Loop closure detection. (b) Point cloud in RTAB-Map.

Figure 12. DPPTAM features and point clouds visualization.

Integration with ROS is implemented via *RTAB-Map ros-pkg*[9] package. The results of the algorithm are a 3D map in form of a point cloud, visual odometry and 2D map in the form of occupancy grid map, where each cell represents the probability of space occupancy. These data were visualized by RViz standard package.

Thus, as the results of experiments we built a map (Fig. 12b) in the form of dense point cloud and obtained the data of visual odometry, which were used to compute the UGV prototype trajectory in XY coordinate plane (Fig. 13). The Fig. 13 shows that at the beginning of UGV prototype motion there are deviations from marked trajectory, but then Kinect RBG-D camera-based RTAB-Map odometry gives the results, which are close to the Ground Truth trajectory.

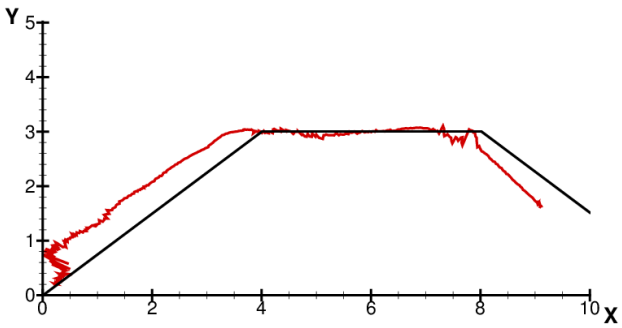


Figure 13. Kinect RBG-D camera-based RTAB-Map trajectories (in red), marked line (in black).

V. COMPARATIVE ANALYSIS OF SLAM METHODS

A. Maps analysis

ORB-SLAM is the feature-based method, which builds sparse map from point cloud (Figure 14a), which has too little details to determine some objects. However, this method creates less amount of outliers.

DPPTAM is the direct method, which is unlike ORB-SLAM and builds dense map from point cloud (Figure 14b), trying to detect monochrome flat objects (walls). The number of outliers is acceptable. Some obstacles were not found during the tests, forming empty spaces instead of walls.

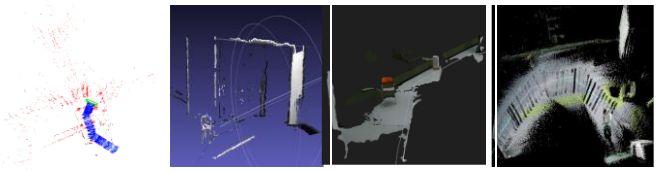
ZEDfu builds dense map (Figure 14c) from point cloud, triangulating area and overlaying colored surfaces. The number of outliers on the map is minimal because of good visual odometry and point cloud filtering. This method visualize all main obstacles and walls, showing them on the 3D map.

RTAB-Map builds dense map from point cloud (Figure 14d). The number of outliers is small, because of filtering. Some surfaces are repeated, showing disperse character because of odometry noises, but it is insignificant. When using for navigation of a real robot in a closed space, OctoMap[10] can be a valuable tool to translate qualitative dense point cloud into obstacles map [10].

[8] RTAB-map package: http://introlab.github.io/rtabmap/

[9] ROS package for RTAB-Map: http://wiki.ros.org/rtabmap_ros

[10] OctoMap is 3D occupancy mapping tool: http://octomap.github.io/



a) ORB-SLAM; b) DPPTAM; c) ZEDfu; d) RTAB-Map.

Figure 14. 3D Point Clouds.

B. *Odometry comparative analysis*

In this section, we provide the comparative analysis of all trajectories obtained by processing different sensor data (conventional camera, LIDAR, stereo ZED camera and Kinect depth sensor) during the experiment with UGV prototype motion in indoor environment (Fig. 15). For monocular SLAM methods (ORB and DPPTAM) the computed trajectories were properly scaled.

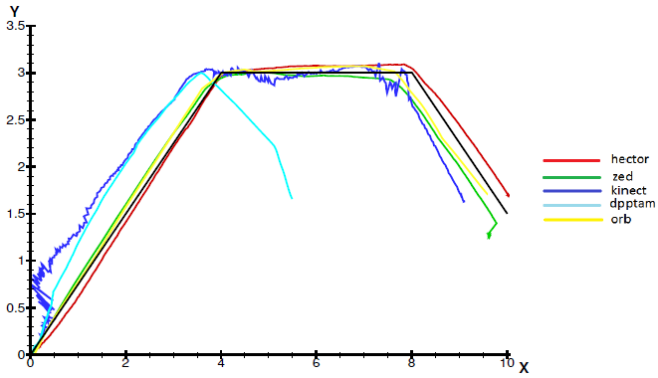


Figure 15. Comparison of UGV prototype trajectories (in meters) computed by monocular ORB-SLAM (yellow curve), monocular DPPTAM (cyan curve), LIDAR data (Hector SLAM – red curve), stereo ZED camera (green curve), and Kinect depth sensor (RTAB-Map – blue curve). Black curve is the ground truth. The units of measure for the XY axes are meters.

Error estimation was calculated as follows: marking line and movement trajectory line were split into uniform grid with a uniform step. To compute the maximum and average deviations we used the corresponding formulae:

$$max.\ deviation = \max\left(\sqrt{(x_i^* - x_i)^2 + (y_i^* - y_i)^2}\right), \quad (1)$$

$$aver.\ deviation = \frac{1}{N}\sum_{i=1}^{N}\sqrt{(x_i^* - x_i)^2 + (y_i^* - y_i)^2}, \quad (2)$$

where x_i^*, y_i^* – marked line coordinates, x_i, y_i – computed trajectory coordinates, N – number of grid nodes.

Table II contains deviations of tested visual SLAM methods.

TABLE II. TRAJECTORY DEVIATIONS

SLAM method	Sensors	Average deviation, m	Maximal deviation, m
Hector SLAM	2D LIDAR	0.11	0.18
ORB-SLAM	Monocular Camera	0.19	0.43
DPPTAM	Monocular Camera	2.05	4.26
ZEDfu	Stereo ZED camera	0.14	0.32
RTAB-Map	Kinect 2.0 depth sensor	0.42	0.67

As it was mentioned earlier, LIDAR is one of the most precise and fast sensors. Odometry Hector SLAM have shown the least error. ORB-SLAM has shown also good results. Despite the fact that the number of features was small, the odometry data were robust. Disadvantage of feature points affected DPPTAM method. The angles of camera rotation were not precisely determined.

ZED camera odometry has also shown good results and errors were close to LIDAR data. However, it loses tracking on sharp turns. RTAB-map and Kinect camera are exposed to slight noises (one of the worst tests is on the graph). The algorithms is good at determination of camera rotation angles and absolute scales of movement trajectory. It should be noticed, that in such kind of test (movement by the strait line forward) all the SLAM methods have shown good results of visual odometry. The problems begin on turns, when the scan area changes fast. Table III represents comparison of visual SLAM methods by the major parameters.

TABLE III. COMPARISON OF VISUAL SLAM METHODS

Parameter	ORB-SLAM	DPPTAM	ZEDfu	RTAM-Map
Visualization	Built-in	ROS/RViz	Built-in	Built-in
CUDA	No	No	Yes	No
Odometry	No	No	Yes	Yes
Trajectory	Yes	Yes	No	No
Noise level	Low	Low	Low	Middle
3D map quality	Low	Average	Good	Good
3D map type	Sparse	Dense	Dense	Dense
Odometry quality (Max. deviation)	Good (0.43m)	Poor (4.26m)	Good (0.32m)	Good (0.67m)

VI. CONCLUSION AND FUTURE PLANS

In this paper we investigated various SLAM methods, providing the comparative analysis of all trajectories obtained by processing different sensor data (conventional camera, LIDAR, stereo ZED camera and Kinect depth sensor) during the experiment with UGV prototype motion in indoor environment. We compared such ROS-based visual

SLAM methods as monocular ORB-SLAM, monocular DPPTAM, stereo ZedFu (based on ZED camera data) and RTAB-Map (based on MS Kinect 2.0 depth sensor data). To monitor the UGV prototype motion we used LIDAR data and ROS-based Hector SLAM trajectory, which was also used as the ground truth (along with marked white tape on the floor).

The main conclusions from our measurements are: (1) There is no a monocular SLAM algorithms which can estimate the absolute scale of the received map and, therefore, localization. Therefore, it is necessary to use a ground truth to verify the size of map objects or to estimate absolute value of camera displacement. (2) Visual odometry, which is based on ORB-SLAM is more robust to homogeneous indoor environment than DPPTAM. Although, DPPTAM creates denser 3D map.

To answer on the question about applicability of visual SLAM methods we provide the following recommendations: (1) ORB SLAM method should be used if there is a high performance requirement and the environment does not have any flat monochrome objects. (2) DPPTAM method should be used if a dense area map is required to build an obstacles map or the environment does not contain enough number of features and the hardware has enough performance power. (3) Stereo cameras or RGB-D sensors usage gives good results and builds a map and a localization in absolute values. There is no significant difference in terms of resource consumption. (4) If it is necessary to build a map with high depth, then it is preferable to use a stereo camera. (5) If there are large monochrome walls, stained glass or mirrors, then RGB-D sensor is a better option to use.

Our future plans deal with realization of other visual SLAM and odometry methods (like Google Cartographer, DSO, LSD-SLAM, ElasticFusion), paying attention to building quality dense point cloud, improving experimental technique and providing tests in more complex environment.

REFERENCES

[1] A. Buyval, I. Afanasyev, E. Magid. "Comparative analysis of ROS-based monocular slam methods for indoor navigation", in *Proc. SPIE 10341 of Int. Conf. on Machine Vision (ICMV)*, Nice, France, 2016.

[2] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. "Brief: Binary robust independent elementary features", in *Proc. of European Conf. on Computer Vision (ECCV)*, Springer, 2010, pp. 778–792.

[3] A. Concha and J. Civera. "DPPTAM: Dense Piecewise Planar Tracking and Mapping from a Monocular Sequence", in *IEEE/RSJ Int. Conf. on Intel. Robots and Sys. (IROS)*, 2015, pp. 5686-5693.

[4] A.J. Davison, I.D. Reid, N.D. Molton, and O. Stasse. "MonoSLAM: Real-time single camera SLAM". *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 29(6), pp. 1052–1067, 2007.

[5] J. Engel, T. Schöps, D. Cremers. "LSD-SLAM: Large-Scale Direct Monocular SLAM", in *Proc. of European Conf. on Computer Vision (ECCV)*, Zurich, Switzerland, 2014, pp. 834-849.

[6] J. Engel, V. Koltun, and D. Cremers. "Direct sparse odometry", in *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2017.

[7] J. Engel, J. Stuckler, and D. Cremers, "Large-scale direct slam with stereo cameras", in *Proc. IEEE/RSJ Intelligent Robots and Systems (IROS)*, 2015, pp. 1935–1942.

[8] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry", in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2014.

[9] Wolfgang Hess, Damon Kohler, Holger Rapp, and Daniel Andor, "Real-time loop closure in 2d LIDAR slam", in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2016, pp. 1271–1278.

[10] A. Hornung, K.M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees", in *Autonomous Robots*, 2013.

[11] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces", in *Proc. IEEE/ACM Int. Symp. on Mixed and Augmented Reality (ISMAR)*, Nara, Japan, Nov. 2007.

[12] S. Kohlbrecher, J. Meyer, O. von Stryk, and U. Klingauf, "A flexible and scalable SLAM system with full 3D motion estimation", in *Proc. IEEE Int. Symp. on Safety, Security and Rescue Robotics (SSRR)*, Nov. 2011.

[13] M. Labbe and F. Michaud. "Online Global Loop Closure Detection for Large-Scale Multi-Session Graph-Based SLAM", in *Proc. IEEE/RSJ Intelligent Robots and Systems (IROS)*, 2014, pp. 2661–2666.

[14] R. Mur-Artal, J.M.M. Montiel and J.D. Tardos, "ORB-SLAM: A Versatile and Accurate Monocular SLAM System", in: *IEEE Trans. on Robotics*, 31(5), pp. 1147-1163, 2015.

[15] R.A. Newcombe, S.J. Lovegrove and A.J. Davison, "DTAM: Dense Tracking and Mapping in Real-time", in *IEEE Int. Conf. on Computer Vision* (ICCV), 2011, 2320-2327.

[16] M. Pizzoli, C. Forster and D. Scaramuzza, "REMODE: Probabilistic, monocular dense reconstruction in real time", in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2014, pp. 2609-2616.

[17] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF", in *Proc. Int. Conf. on Computer Vision (ICCV)*, 2011, pp. 2564–2571.

[18] T. Whelan, S. Leutenegger, R Salas-Moreno, B. Glocker, and A. Davison, "Elasticfusion: Dense slam without a pose graph", in *Robotics: Science and Systems*, 2015.

[19] B. Williams, M. Cummins, J. Neira, P. Newman, I. Reid, and J. Tardos, "A comparison of loop closing techniques in monocular SLAM", in *Robotics and Autonomous Systems*, 57(12), pp. 1188–1197, 2009.

[20] M. Sokolov, O. Bulichev and I. Afanasyev, "Analysis of ROS-based Visual and Lidar Odometry for a Teleoperated Crawler-type Robot in indoor environment", in *Proc. Int. Conf. on Informatics in Control, Automation and Robotics (ICINCO)*, Madrid, Spain, 2017.