SLAM method: reconstruction and modeling of environment with moving objects using an RGBD camera

Korney Tertychny
Volgograd State University
Volgograd, Russia
tertychny@volsu.ru

Dmitry Krivolapov
Volgograd State University
Volgograd, Russia
krivolapov@volsu.ru

Sergey Karpov
Volgograd State University
Volgograd, Russia
karpov.sergey@volsu.ru

Alexander Khoperskov
Volgograd State University
Volgograd, Russia
khoperskov@volsu.ru

Abstract

We have analyzed high and low dynamic environment problems of Simultaneous Localization and Mapping (SLAM) methods, which we use for automatic 3D models construction. Constructed models are convenient for developing lifelike VR (virtual reality) systems of real places. Our software provides preliminary processing of data before transferring it to the main SLAM method. Developed software allows utilizing various computer vision algorithms or artificial neural networks for recognition of objects which may change their location. Microsoft Kinect sensor allows easily recognize movable objects like human while testing.

1 Introduction

Navigation systems development is one of the most important problem in robotic engineering. To make decisions on further action a robot needs information about environment, surrounding objects, and its localization. There are a lot of various navigation methods based on odometry [13], inertial navigation, magnetometer, active labels (RFIDS, GPS) [7], label and map matching. Simultaneous Localization and Mapping (SLAM) approach is one of the most promising way of navigation [6, 7]. Modern SLAM solutions provide mapping and localization in an unknown environment [2]. Some of them can be used for updating the map which has been made before. SLAM approach can be used in mobile autonomous systems like robots, cars, etc. SLAM approach is the only method which provides its processing without any external information sources or the preparation of the environment (for example, placing labels) [10]. SLAM is the general methodology for solving two problems [4, 9, 18]: 1) environment mapping and 3D model construction; 2) localization using generated map and trajectory processing.

In: Marco Schaerf, Massimo Mecella, Drozdova Viktoria Igorevna, Kalmykov Igor Anatolievich (eds.): Proceedings of REMS 2018 – Russian Federation & Europe Multidisciplinary Symposium on Computer Science and ICT, Stavropol – Dombay, Russia, 15–20 October 2018, published at http://ceur-ws.org

Efficacy of SLAM solution determines navigation and environment reconstruction accuracy [13, 13]. SLAM is one of the general directions in computer vision and robotic engineering for the last than 25 years. It appears that, all animals and human beings use this principle of navigation.

However, there are some problems in modern SLAM solutions [6, 7]. One of them is dynamic environment processing, where objects can move both in sight or out of sight. The problem becomes even more serious, since a change of the position of objects may lead to a complete disorientation of the robotic system. Our research is focused on the algorithm for removing moving objects to increase accuracy of 3D reconstruction of a real environment using SLAM approach development.

There are four types of robots [13]: indoor, outdoor, underwater and airborne robots. Our solution is applicable with all four types, but the main relevance are indoor and outdoor robots. We have applied and tested it with indoor robots due to technical features of our equipment.

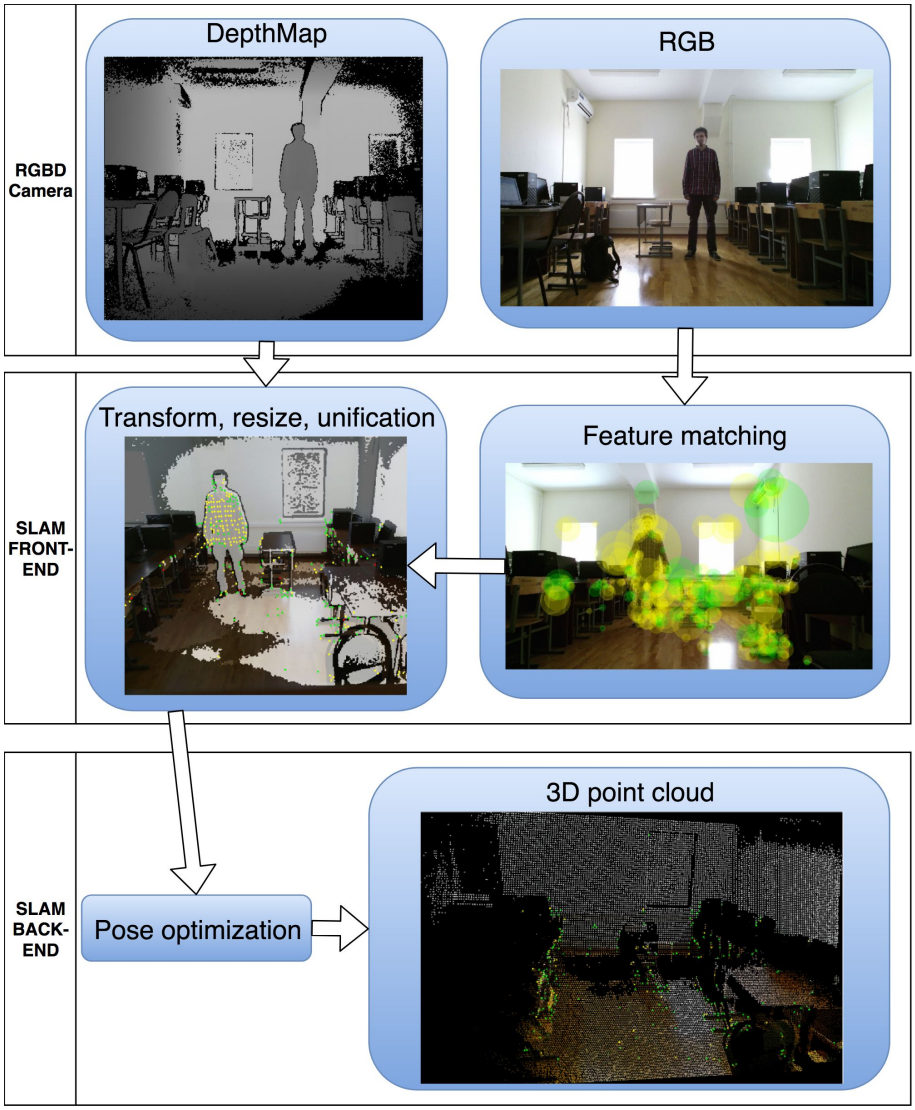

Figure 1: diagram of common SLAM solution, which uses RGBD sensor

Figure 1 presents a common operation SLAM algorithm, which uses RGBD sensor. The output data of a RGBD camera is an RGB image and a depthmap. In the depthmap image pixels contain data about the distance between the sensor and the object. RGB image and the depthmap are transfered to the SLAM front-end from a RGBD sensor. A feature matching algorithm processes the RGB image. Selected features are used

for a localization (See yellow and green points on "Feature matching" panel Figure 1). ORB, SURF, SIFT are examples of such algorithms [6, 15]. Then the image with the selected features and the depthmap are resized, cut and calibrated. Calibration, cutting and resizing are required because the RGB and the depthmap may have different resolution, view angle and these cameras are located in some distance from each other [3]. Some SLAM solutions combine RGB, depthmap and features information into a single structure. Then the data is transfered to SLAM back-end, where the localization of the image, the camera and the features are provided. Afterwards SLAM solution can make and locate point cloud on the 3D scene. Also it can make 2D maps, a camera trajectory, and a feature map (features and their location).

Usually SLAM solutions based on graph-based representation may be subdivided into two parts: the front-end and the back-end. Using sensor data the front-end provides preliminary solution as a feature map. The back-end uses the data from the front-end and a probabilistic approach to make solution with maximum probability. As a rule, such approach is faster and more reliable than filter–based approaches due to better ability to cope with SLAM nonlinearity [4, 13].

In the classical SLAM scheme, when a SLAM algorithm starts working there is no knowledge on surrounding environment, no preliminary map or labels [4]. Thus, the solution is only the result of sensor measurements.

The other peculiarity is that the environment, where the robot works, must be static. It means that objects seen and memorized by the robot must not change their positions and properties [4]. The lighting conditions should be taken into the account depending on used equipment. All of them is hardly possible in a real environment. We can highlight several important SLAM problems, which appear in the real world: dynamic or changeable environment [9, 18] and accumulative mistakes eliminating (See discussion in works [3, 13, 9, 18, 8, 12, 1, 17]). In the current work we focus on the first problem.

2 The dynamic environment problem

Emergence of moving and movable objects in an environment which is surveyed by a robot is a significant problem. We call high-dynamic environment, if objects move in sight, and low-dynamic environment, if objects move out of sight [9, 18]. The presence of such objects may cause errors in the resulting 3D model, or localizing failure. Any objects in a dynamic environment can be divided into objects moving in sight or out of sight. Some SLAM solutions, such as DP-SLAM and RTAB-Map, may cope with small out of sight changes [14, 5]. DP-SLAM can include an overlapping frames, and after some time passes it can replace wrong frames with overlapping ones. RTAB-Map overwrites voxels, but the object don't disappear while all voxels are overwritten. Actually some parts of the object will never eliminate. It looks like transparent object (See Figure 2). It is important that such objects can contain some features, which cause localization failure or errors.

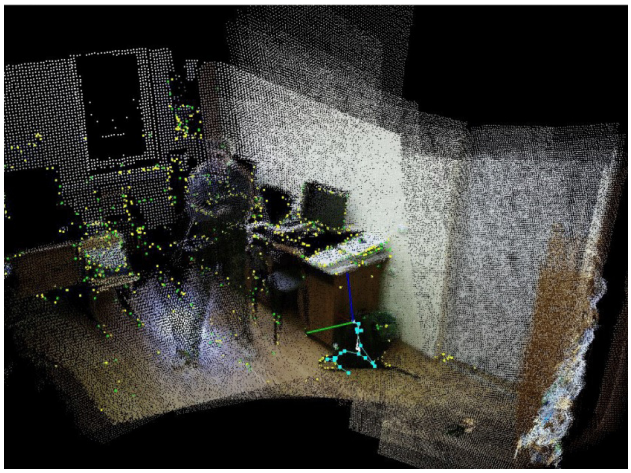

Figure 2: Example of a "transparent" object (human being), which is gone out of sigh.

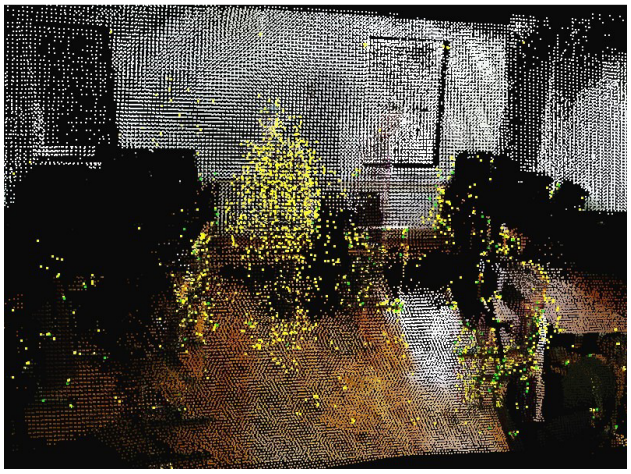

Figure 3: Moving object traces in the point cloud

We can see the point cloud made from the frames set. Half of them contain a human being who left the frame. In that case some features are located at the defunct object. So there are several errors, but the method still works because greater amount of features are on static objects. Also the model is extended with the parts which

have been invisible before the human being has gone. The place which was invisible looks like a shadow on the floor.

If there are too many features on a defunct or replaced object, the robot can't understand that it is the same place. It may cause unexpected behavior [13, 9]. Any dramatic global changes may cause such problems too.

In a high dynamic environment sensors provide erroneous data on environment with moving objects. If moving objects don't carry features, modern SLAM solutions don't have any problems. Nevertheless such objects leave traces in point cloud (See Figure 3). All point cloud images in this article are rotated relative to their RGB and depth analogs for better visibility.

There are a lot of traces and features (yellow dots) left by a moving object. We can see significant amount of features, which are not placed on static objects. Figures 4 and 5 shows how features are located on the image with a human being. A lighter space on the RGBD image shows where the distance was correctly measured. Only points with correct distance are placed to the point cloud.

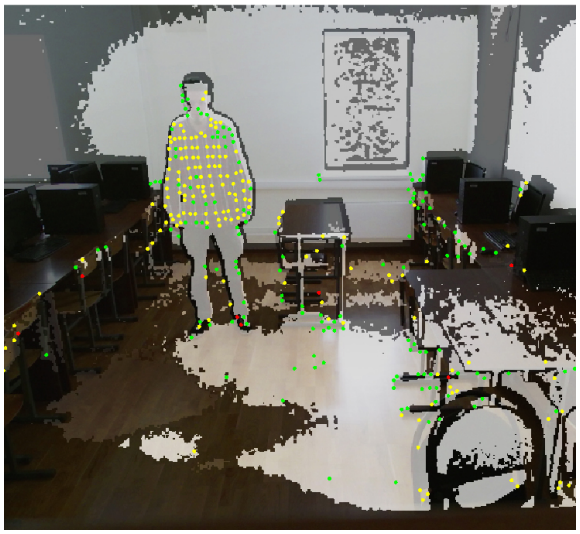

Figure 4: Example of features matching on the RGBD image (RGB+DepthMap) with the human being

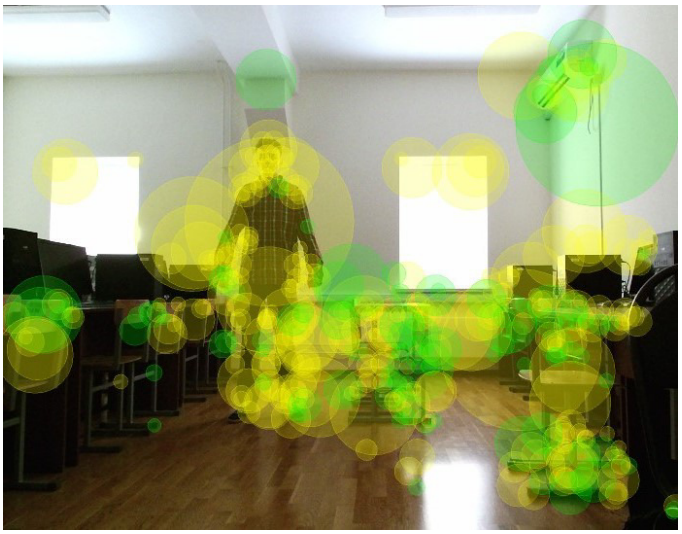

Figure 5: Example of features matching on the RGB image with the human being

It is easy to observe that on the RGB image there are approximately 30% of features located on the human being. The transition to the RGBD image shows that the human being has already occupied more than 50% of features. This is caused by the fact that all the features without correct distance (not located on the RGBD image) are discarded.

A human being is usually separated form any environment and the distance to him can be correctly measured. Also a human being is a good source of features. When several human beings appear in the frame incorrect work of SLAM solutions is practically guaranteed. On the RGB image different circle colors and sizes mean reliability level and location accuracy of features. The smaller the radius, the more accurate they are determined.

It should be mentioned that in the current work we consider human as an object which can change its position, since it is easy to locate a human being using Kinect. Thus all the argumentation above is valid for different movable and moving objects, such as animals, cars, papers, chairs etc. All of those objects are not a reliable features source.

The solutions for the dynamic environment problem have been already offered. Authors [9] propose to make static and dynamic maps for high-dynamic environment separately. The localization is provided by using a static map. Not real objects but only moving elements of the frame are determined. If a pixel has changed its color, it is considered as a moving object and removed from the static map. This method does not work in a low-dynamic environment with objects which can change their position out of sight.

In Ref.[18] a framework solving the problem with a low-dynamic environment has been proposed. In their work an environment changes between sessions. Their algorithm finds outdated frames from last session and replace them with fresher ones. The algorithm does not treat moving objects in frame leading to the instability when moving object has too many features. Both of these solutions solve the problem in the back-end part.

3 Modificated method

We propose a solution which can work both with a high and low-dynamic environment in the front-end. It processes data before transfering it to the main SLAM method. So a different back-end can be used in order to transform input data. Our solution removes movable objects from data stream. Our algorithm can allocate objects which don't move but can move unlike the solution presented in Ref.[9].

We establish human beings as removing objects for our algorithm in this article. This type of an object has been chosen because it is easy to locate people using Kinect sensor and Kinect SDK, which provide a body frame stream. We use a body frame to remove human being image from data streams. We overwrite a distance as incorrect (less than minimum) in pixels where a human location is specified. It is displayed with a red colored human silhouette on Figure 6. On Figure 6 the red color was chosen to make images more visible.

Figure 6: The depthmap frame modification

After processing our algorithm transfers RGB and depthmap frames to feature matching algorithms and then to SLAM back-end. An examples of solutions which uses an RGB, depthmap and feature map are RTAB-Map (Real-Time Appearance-Based Mapping), and DP-SLAM (Distributed Particle).

After several tests we noticed that sometimes some pixels, which belong to removed objects, stay away of overwritten part. We determined that there are 0 to 3 pixels around an overwritten space. Figure 7 shows the sequence of updated SLAM method operations. Our algorithm processes data from any RGBD sensor overwriting unwelcome objects on the depth frames, and transfers data to the main SLAM solution. It should be mentioned that in the "Object rejection" stage we can remove any unwelcome objects, which we can allocate by computer vision algorithms or artificial neural networks.

4 Results and discussion

In the current work the algorithm which provides preliminary data processing to remove objects which may cause errors in simultaneous localization and mapping is described. The examples of such objects are people, cars, animals and other movable or moving objects. We test developed software using a Kinect 2 sensor, and RTAB-Map as a SLAM solution. We take out a human as an sample of moving objects from data provided by sensors, since Microsoft Kinect SDK provides simple and convenient work with human contours. We use an output point cloud for our VR (virtual reality) system. It provides virtual tours to reconstructed real places.

Figure 9 shows the output point cloud of our hardware and software system. Our algorithm creates the point cloud with no erroneous data caused by moving objects. If there is some place hidden by removing objects, there will be empty space in the point cloud. After it becomes visible, system will add information.

If removing objects occupy a significant part of the field of vision so the SLAM solution can't localize, it may show an error and continue localizing after the object moves out. There is no problem if there are enough features outside a moving object. As it was before, yellow and green dots on Figure 9 indicate features. SLAM solutions such as RTAB-Map use features for localization and place memorization.

While testing we have found out that there may be some pixels which are not overwritten by our algorithm around the removing object. So we have to modify our solution to overwrite at least 3 more pixels around allocated objects.

There is an empty space that looks like a shadow which is made by human. Standard SLAM solution doesn't add points there (See Figure 8). Without moving of the camera, the algorithm continues to assume that the

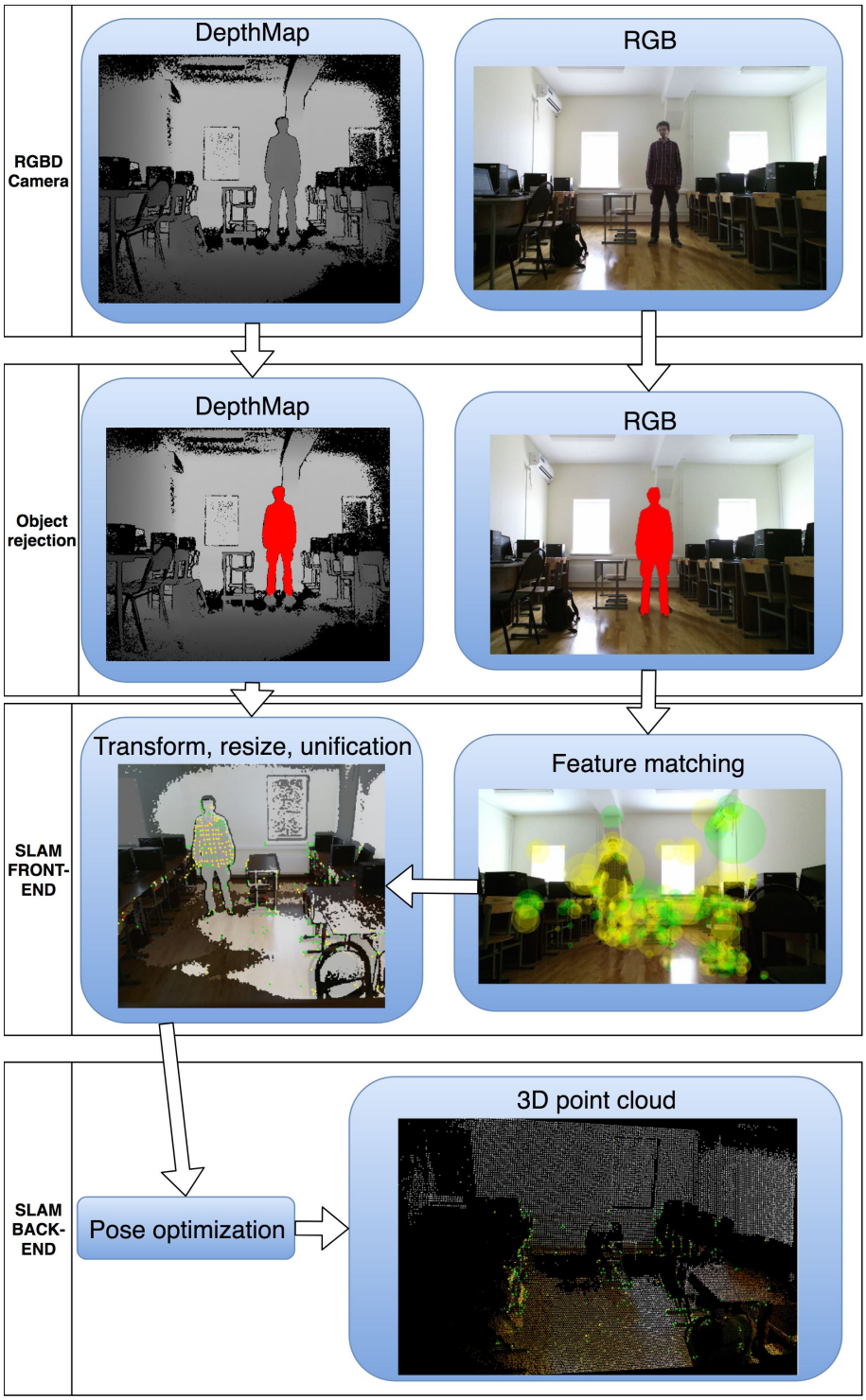

Figure 7: The scheme of operation of modified SLAM method

cloud of points is complete and true and the empty place is out of view. The key feature of our algorithm is an ability to create a list of unwelcome objects, that are not desirable to transmit to SLAM input. We can use different computer vision algorithms or artificial neural networks to recognize objects that can change their position. Then such objects can be removed, as it has been shown in this article using a human being as an example. Our algorithm has an ability to remove any recognizable object. In addition to the modular principle,

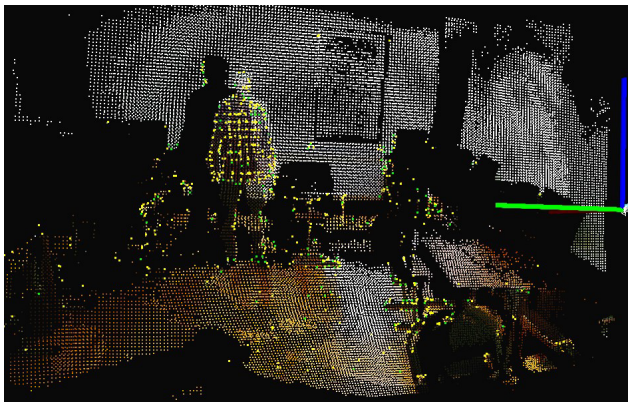

Figure 8: Standard SLAM solution output point cloud. The silhouette of a human is clearly seen. Green and yellow point are features used by methods for localization.

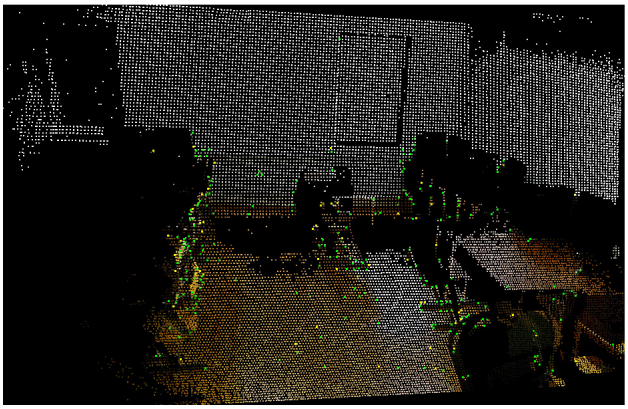

Figure 9: Modified SLAM solution output point cloud. There are no features tied to non-stationary objects.

it is possible to work with the method in combination with various solutions that implement the SLAM approach.

The tests of the developed algorithm give satisfactory results completely excluding undesirable objects from the input data of the main SLAM algorithm. SLAM solution without using the proposed algorithm results shown on Figure 8. It is seen that in the cloud of points there is a large number of features, which are not tied to real objects. Such features are not a reliable source of information about an environment. Figure 9 shows the cloud of points obtained with the proposed algorithm. It does not contain any erroneous points which always appear due to the presence of a moving object. Tests show that it is enough to remove an object image from the depthmap, because SLAM solutions use only the features with correctly measured distance for localization and mapping. Furthermore, removing one object like a human doesn't cause performance decline using medium-power personal computer.

Acknowledgements

Work has been supported by the Ministry of Education and Science of the Russian Federation (government task No. 2.852.2017/4.6).

References

[1] Bakkay, M.C., Arafan, M., Zagrouba, E.: Dense 3D SLAM in dynamic scenes using kinect. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 9117, pp. 121-129 (2015)

[2] Cao, F., Zhuang, Y., Zhang, H., Wang, W.: Robust Place Recognition and Loop Closing in Laser-Based SLAM for UGVs in Urban Environments. IEEE Sensors Journal 18(10), 4242-4252 (2018) https://doi.org/10.1109/JSEN.2018.2815956

[3] Civera, J., Bueno, D., Davison, A., Montiel, J.: Camera self-calibration for sequential Bayesian structure from motion. IEEE International Conference on Robotics and Automation 2009, pp. 403-408. IEEE (2009)

[4] Cummins, M., Newman, P.: Appearance-only SLAM at Large Scale with FAB-MAP 2.0. The International Journal of Robotics Research, Vol. 30, pp 1100-1123 (2010). https://doi.org/10.1177/0278364910385483

[5] DP Developers, https://github.com/jordant0/DP-SLAM. Last accessed 27 March 2018

[6] Durrant-Whyte, H., Bailey, T.: Simultaneous localization and mapping: Part I. IEEE Robotics and Automation Magazine 13(2), pp. 99-110 (2006) https://doi.org/10.1109/MRA.2006.1638022

[7] Durrant-Whyte, H., Bailey, T.: Simultaneous localization and mapping: Part II IEEE Robotics and Automation Magazine 13(3), pp. 108-117 (2006) https://doi.org/10.1109/MRA.2006.1678144

[8] Endres, F., Hess, J., Sturm, J., Cremers, D., Burgard, W.: 3-D Mapping with an RGB-D camera. In IEEE Transactions on Robotics 2014, Vol 30(1), pp. 177-187. IEEE (2014) https://doi.org/10.1109/TRO.2013.2279412

[9] Hahnel, D., Triebel, R., Burgard, W., Thrun, S.: Map building with mobile robots in dynamic environments. Robotics Automation, 2003, pp. 1557-1563. IEEE (2003) https://doi.org/10.1109/ROBOT.2003.1241816

[10] Kim, P., Chen, J., Cho, Y.K.: SLAM-driven robotic mapping and registration of 3D point clouds. Automation in Construction 89, 38-48 (2018) https://doi.org/10.1016/j.autcon.2018.01.009

[11] Lachat, E., Macher, H., Landes, T., Grussenmeyer, P.: Assessment and calibration of a RGB-D camera (Kinect v2 Sensor) towards a potential use for close-range 3D modeling. Remote Sensing 7(10), 13070-13097 (2015) https://doi.org/10.3390/rs71013070

[12] Majdi, A., Bakkay, M., Zagrouba, E.: 3D modeling of indoor environments using Kinect sensor. In IEEE 2nd International Conference on Image Information Processing pp, 2013,pp. 67-72. IEEE (2013) https://doi.org/10.1109/ICIIP.2013.6707557

[13] Nister, D., Naroditsky, O., Bergen, J.: Visual Odometry. IEEE Computer Society Conference on Computer Vision and Pattern Recognition 2004, vol. 1, pp. 652-659. IEEE (2004) https://doi.org/10.1109/CVPR.2004.1315094

[14] RTAB_map Developers, http://introlab.github.io/rtabmap/. Last accessed 27 March 2018

[15] Rublee, E., Rabaud, V, Konolige, K., Bradski, G.: ORB: an efficient alternative to SIFT or SURF. IEEE International Conference on Computer Vision 2011, pp. 2564-2571, IEEE (2011) https://doi.org/10.1109/ICCV.2011.6126544

[16] Shao, B., Yan, Z.: 3D Indoor Environment Modeling and Detection of Moving Object and Scene Understanding, Transactions on Edutainment XIV,40-55, IEEE International Conference on. IEEE, pp. 2564-2571 (2011)

[17] Walcott-Bryant, A., Kaess, M., Johannsson, H., Leonard, J.J.: Dynamic pose graph SLAM: Long-term mapping in low dynamic environments. IEEE International Conference on Intelligent Robots and Systems vol. 6385561, pp. 1871-1878. IEEE (2012) https://doi.org/10.1109/IROS.2012.6385561

[18] Wang, Y., Huang, S., Xiong, R., Wu, J.: A framework for multisession RGBD SLAM in low dynamic workspace environment. CAAI Transactions on Intelligence Technology 1, 90-103. CAAI (2016) https://doi.org/10.1016/j.trit.2016.03.009

[19] Zhang, Z.: Microsoft kinect sensor and its effect. IEEE Multimedia 19(2), pp. 4-10 (2012) https://doi.org/10.1109/MMUL.2012.24