Virtual Experiments on Mobile Robot Localization with External Smart RGB-D Camera Using ROS

Kirill Kononov
Laboratory of Intelligent Robotics Systems (LIRS),
Higher School of Information Technologies and Intelligent Systems,
Kazan Federal University
Kazan, Russia

Roman Lavrenov
Laboratory of Intelligent Robotics Systems (LIRS),
Higher School of Information Technologies and Intelligent Systems,
Kazan Federal University
Kazan, Russia
lavrenov@it.kfu.ru

Tatyana Tsoy
Laboratory of Intelligent Robotics Systems (LIRS),
Higher School of Information Technologies and Intelligent Systems,
Kazan Federal University
Kazan, Russia

Edgar A. Martínez-García
Department of Industrial Engineering and Manufacturing,
Institute of Engineering and Technology,
Autonomous University of Ciudad Juarez
Juarez, Mexico

Evgeni Magid
Laboratory of Intelligent Robotics Systems (LIRS),
Higher School of Information Technologies and Intelligent Systems,
Kazan Federal University
Kazan, Russia

2021 IEEE 3rd Eurasia Conference on IOT, Communication and Engineering (ECICE) | 978-1-6654-4516-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/ECICE52819.2021.9645644

***Abstract*—Precise robot localization is important for all mobile robots, which has to deal with accumulating odometry errors, onboard sensory noise, harsh environmental conditions, unstable or missing GPS signal, and absence or uncertainties of a global map. Yet, localization is considered in Smart Environments applying dynamic connectivity with external local sensors within the Internet of Things (IoT) paradigm. This paper presents experimental results of robot indoor localization using a single external smart RGB-D camera. The virtual experiments were performed in the ROS Gazebo simulator with Turtlebot3 Waffle Pi mobile robot model. Three types of robot motion within a virtual office environment were considered: static state, linear motion, and three different cases of curvilinear locomotion. In all cases, external RGB-D camera usage allowed to obtain a reasonably accurate location of the robot.**

Keywords—Robot Localization, Robot Recognition, Smart Environments.

I. Introduction

Mobile robots often experience difficulties in determining their position due to various problems. As a building's size and the number of rooms increase, it becomes difficult for a robot to localize itself within the building. Wrong localization causes errors in global planner algorithms and current location data publishing, which in turn makes the robot behavior irrational and incorrect. It also prevents proper execution of an intended task and, in a worst-case scenario, might lead to a complete loss of the robot [2]. It is necessary to support robot localization in a closed indoor environment with a help of an external RGB-D camera (considering IoT infrastructure available within the building, Fig. 1) that communicates with a robot and informs the robot about its position within a room. Such technology can be used in the robot location tracking over time.

II. Solution approach

There exist different methods for computing a distance to an object using a monocular camera. This approach is often used by vehicles for analyzing road safety. Nienaber *et.al*. used monocular depth estimation for pothole distance estimation [6]. Joglekar *et.al*. [7] estimated a distance to objects on a road to help a driver to avoid obstacles [8] and traffic accidents. Yet, these approaches require proper camera calibration, so we have used an RGB-D camera for a simplified estimation of a distance to an object.

IoT framework-based localization of a robot within a closed indoor environment is applied together with smart sensor networks in urban search and rescue scenarios [9]. To test the proposed solution, we constructed a typical office environment with a help of an automatic tool [10] for the Gazebo simulator [11]. A random room was selected within this environment and an RGB-D camera was mounted on a wall of the room. Next, the Turtlebot3 Waffle Pi [12] robot position within the room was tracked with the camera using the find_object_2d package [1]. A robot position tracking had certain inaccuracies that we carefully measured to estimate the packaging accuracy for a robot localization task with an RGB-D camera.

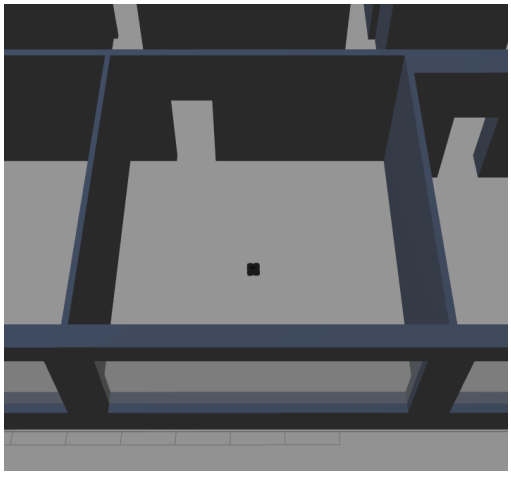

Fig. 1. Testing environment in the Gazebo simulator.

III. Find_object_2d Package Overview

Find-Object [3] is a simple application that allows finding a particular object within an image from a pre-created dictionary of objects employing different types of OpenCV [13] detectors and descriptors. We employed *find_object_2d* [1] package, which is the Robot Operating System (ROS, [4]) wrapper of the original application. The package uses an RGB camera for object detection and an RGB-D camera for object detection, localization, and SLAM tasks [5].

ISBN: 978-1-6654-4516-0

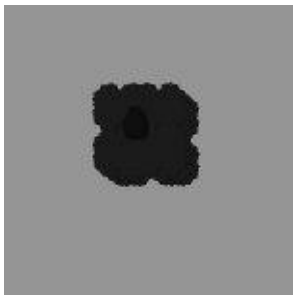

Fig. 2 The robot snapshot used by *find_object_2d.*

Find_object_2d subscribes to a ROS topic [14] with a camera image, finds known objects that it already knows (after familiarizing with them at the learning stage), and then publishes detected objects back into the ROS topic with information about their positions and orientations. The package uses RGB and RGB-D types of cameras with RGB cameras. The package only detects objects in an image, but with the RGB-D camera, it computes the 3D positions of detected objects relative to the camera.

To find an absolute position of a detected by a camera object the following steps are performed, Package *find_object_2d* publishes coordinate frames to ROS via *tf* [15] package, which is used for transformations between coordinate frames. When *tf* knows where the camera is, it publishes information into the ROS topic about the camera's child coordinate frames in absolute coordinates. These are camera frames, which in turn contain the detected object and thus the object position could be extracted.

Package *find_object_2d* supports different combinations of feature detectors and descriptors algorithms and provides an extensive number of configurable parameters. ORB (Oriented FAST and rotated BRIEF) [16] is selected as the most suitable combination of a detector and a descriptor [17–19] for the task of moving robot tracking and localization. ORB is a combination of FAST (Features from Accelerated Segment Test) detector [20] and BRIEF (Binary Robust Independent Elementary Features) descriptor [21].

ORB works fast and has appropriate performance, which is important for moving object tracking. In the beginning, the package requires a vocabulary with snapshots of objects. When the package finds feature points in an image, it compares them with feature points of objects from the vocabulary. If the similarity percentage is sufficient, the package informs on known object detection. If the RGB-D camera is used, the package [22] publishes a computed position of the detected object. We have tested different types of vocabulary and finally selected a case that provides enough accuracy.

To initialize the detection and recognition system, an image of an object is required, for example, a snapshot of a robot. The algorithm accuracy strongly depends on initial image quality, therefore selecting a proper location for a robot to obtain its snapshot is important. To provide the best coverage of a room with the camera, we placed the camera in the room center on a ceiling and pointing downwards (to a floor). Yet, a snapshot of the robot standing in the room's center directly under the camera was one of the worst cases since in this case at the initialization stage the camera saw it as a flat object. Then, as the robot was moving away from the center and obtaining a 3D shape (from the camera perspective), the detection rate decreased drastically due to feature points' detection. To solve this issue, the robot initialization snapshot (Fig. 2) was taken while it appeared between the room center and the boundary. In this position, the detector "saw" feature points of the robot's side. Thus, the detector was properly tracking the robot which was moving in a different direction around the room. For the detection process, all default configurations of *find_object_2d* were kept except disabling the *nndRatioUsed* parameter, which significantly improved package algorithms' performance.

IV. Testing Environment and Experiments

To test the proposed solution, we constructed a typical office environment (Fig. 1) with a help of an automatic tool [10] for the Gazebo simulator. A random square office-like empty room of 6x6 meters size was selected within this environment and an RGB-D camera was mounted on a wall of the room. Turtlebot3 Waffle Pi mobile robot was selected for experiments with the robot's base coordinate frame *base_link* located in the center of the robot's bottom.

Localization inaccuracy, which inevitably appears while using *find_object_2d*, is calculated as a distance between the real pose of the robot's *base_link* and a coordinate frame of a detected object (that should be the closest point of the robot body to the camera) received from this package. For computations of the localization inaccuracy, only projections of the robot and a detected object on the *XY* plane were utilized.

We constructed ROS plugin *tf_listener* (Listing 1) for computing inaccuracies of robot localization. It calculates a distance between the *base_link* frame and a coordinate frame of the detected object. Computation frequency is 10 iterations per second, which equals to *find_object_2d* package configuration of the camera's frame rate (10 fps). The plugin computes a distance between two frames (distance is the inaccuracy of the detector and descriptor algorithms) and writes computed values into a log file. The distance between the two coordinate frames is calculated as follows.

$$l = \sqrt{(X_b - X_f)^2 + (Y_b - Y_f)^2}\ , \qquad (1)$$

where X_b and Y_b are coordinates of *base_link* frame and X_f and Y_f are coordinates of the detected object within a current camera frame concerning the corresponding axis. To compute inaccuracies of localization, 3 types of experiments were considered: static state, linear motion, and curvilinear motion. For static state we measured the distance for 100 iterations, for linear motion we measured the distance for 250 iterations, and for curvilinear motion, we measured the distance for 1000 iterations. Each iteration means a computed value of the localization inaccuracy.

For static state experiments, the room was divided into 9 squares and the robot was placed in a center of each square to test the localization accuracy in different parts of the room (the left image in Fig. 3). For linear motion experiments, we selected eight straight segment paths throughout the room

from one corner to another (Fig. 4). For curvilinear motion, we constructed five different paths: one root with a large radius circle (2.0 m) [23], one root with a small radius circle (1.25 m) [24], and three chaotic locomotion scenarios [25–27]. Circular paths of different radii allowed to test localization accuracy while the robot turns (the right image in Fig. 3). Each path was applied 10 times to obtain average values.

```
import rospy
import tf
import math

if __name__ == '__main__':
  rospy.init_node('tf_listener')
  listener = tf.TransformListener()
  rate = rospy.Rate(10.0)
  file = open('dataset.txt', 'w')
  while not rospy.is_shutdown():
    try:
      (trans,rot) =
listener.lookupTransform('/base_link', '/object_1',
rospy.Time(0))
      trans = str(trans).replace(' ', '')[1:-1]
      arr = trans.split(",")
      x = float(arr[0])
      y = float(arr[1])
      d = math.sqrt(x*x + y*y)
      file.write(str(d) + '\n')
    except (tf.LookupException,
tf.ConnectivityException,
tf.ExtrapolationException):
      continue
    rate.sleep()
  file.close()
```

Listing 1. Source code of *tf_listener* plugin.

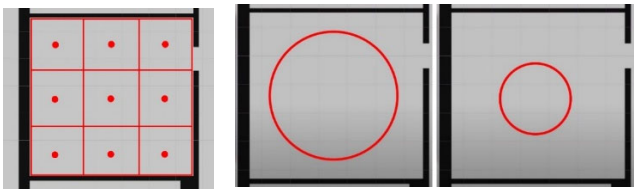

Fig. 3. The red dots denote the robot positions in static state experiments (left image). Routes for circular motion experiments, with a large and a small radii (central and right image respectfully).

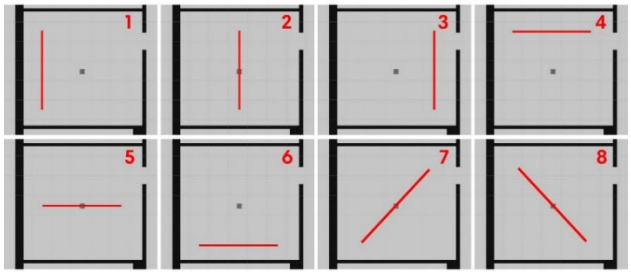

Fig. 4. The red lines depict routes for linear motion experiments.

V. Results of Virtual Experiments

To analyze obtained measurements, each type and case of locomotion data were clustered to form two distinguishable clusters: with localization error less than 0.25 meters (dense results with over 80% of data within the cluster) and over 0.25 meters (sparse results). Thus, we formed two types of labels for localization error: "Good" cases label with acceptable localization results, which were featured with a localization error that is less than 0.25 meters, and "Bad" cases with a localization error that are greater than or equal to 0.25 meters. The third label is "Unknown" data, which was selected when *find_object_2d* could not find the robot within the camera frame, and thus no new data could be sent to the corresponding ROS node. When the *tf_listener* plugin does not receive a new position from *find_object_2d* (i.e., *tf* package keeps publishing last remembered data until it receives a new transformation, which produces two or more repeated values in a row), the localization error is marked with "Unknown" label. Thus, "Unknown" datasets contain only duplicated values and this denotes that *find_object_2d* did not find the robot in the camera frame.

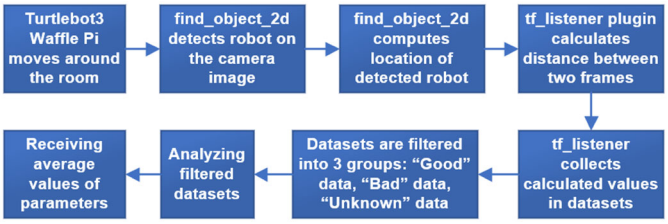

Fig. 5. Flow diagram of robot localization algorithm accuracy computing using an RGB-D camera.

The algorithm for determining the robot localization accuracy with an RGB-D camera using the *find_object_2d* package consists of the following steps (Fig. 5):

(1) A robot (e.g., Turtlebot3) moves along its pre-planned path
(2) *find_object_2d* package detects the robot and computes its position
(3) *tf_listener* plugin calculates localization inaccuracy value and collects these values into datasets
(4) Datasets are clustered and labeled.
(5) Average values of localization data are selected only within data that are labeled as "Good"

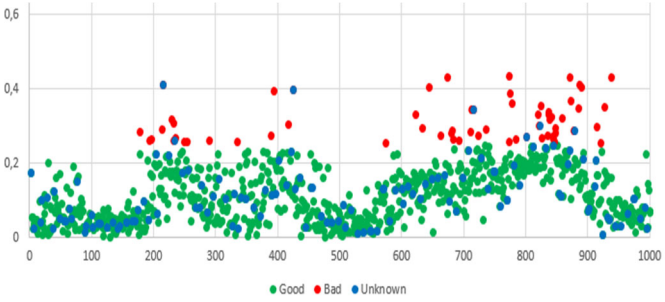

Fig. 6. Diagram example for one of the routes of the third type of curvilinear motion. The horizontal axis is the measurements' iteration. The vertical axis is the computed distance between two frames.

For analyzing localization accuracy, only data with "Good" labels were used. "Bad" and "Unknown" labeled data were used for counting cases with computational errors or false-negative errors (Fig. 6). Each locomotion type (9 static states, 8 linear paths, and 5 curvilinear paths) was tested 10 times to obtain average measurements. The analysis result (presented in Table 1) showed that the *find_object_2d* package with ORB detector and descriptor succeeded in correctly detecting and localizing the robot in about 82% of cases with an average inaccuracy distance of about 0.13 meters. The average minimum value of localization inaccuracy was not close to zero, so it could not be perfectly accurate while the

localization with the *find_object_2d* package provides acceptable results. As expected, the localization accuracy is higher for a static state of a robot and linear paths rather than for curvilinear paths.

TABLE 1. STATISTICS FOR ALL TYPES OF PATHS

	Static state	Linear motion	Curvilinear motion	Average value
Average percent of "Good" values (%)	83,30	83,82	79,04	82,05
Average percent of "Bad" values (%)	3,48	9,27	9,66	7,47
Average percent of "Unknown" values (%)	13,22	6,91	11,30	10,48
Average math expectation of inaccuracy (metres)	0,125	0,136	0,125	0,129
Average dispersion of inaccuracy (metres)	0,013	0,002	0,003	0,006
Average Minimum of inaccuracy (metres)	0,081	0,028	0,006	0,039

VI. CONCLUSIONS

This paper presented the experimental results of robot indoor localization using a single external smart RGB-D camera. The virtual experiments were performed in the ROS Gazebo simulator with Turtlebot3 Waffle Pi mobile robot model. Three types of robot motion within a virtual office environment were considered: static state, linear motion, and three different cases of curvilinear locomotion. In all cases, an external RGB-D camera usage with the find_object_2d ROS package allowed to obtain a reasonably accurate location of the robot in 82% of cases on average while keeping the average error of localization within 0.14 meters. Future works consider comparing the precision of odometry-based localization [28] and IoT-based approach to localization with an RGB-D camera using the find_object_2d package.

ACKNOWLEDGMENT

This paper has been supported by the Kazan Federal University Strategic Academic Leadership Program ("PRIORITY-2030").

REFERENCES

[1] Manual for find_object_2d package. [Online]. Available: http://wiki.ros.org/find_object_2d/

[2] N. Alishev, K.L. Su, R. Lavrenov, E. Magid, and K.-H Hsia, "Network failure detection and autonomous return algorithms for a crawler mobile robot navigation", in 11th International Conference on Developments in eSystems Engineering (DeSE), 2018, pp. 169-174.

[3] M. Labbé. Find-Object: Simple Qt interface to try OpenCV implementations of SIFT, SURF, FAST, BRIEF and other feature detectors and descriptors. [Online]. Available: http://introlab.github.io/find-object

[4] M. Quigley, et al., "ROS: An open-source robot operating system", in ICRA Open-Source Softw. Workshop, 2009.

[5] E. Mingachev, R. Lavrenov, E. Magid, and M. Svinin, "Comparative Analysis of Monocular SLAM Algorithms Using TUM & EuRoC Benchmarks", in 15th International Conference on Electromechanics and Robotics «Zavalishin's Readings» (ER(ZR)-2020), 2020, pp. 343-356.

[6] S. Nienaber, R. S. Kroon, and M. J. Booysen, "A Comparison of low-cost monocular vision techniques for pothole distance estimation", 2015 in IEEE Symposium Series on Computational Intelligence, 2015, pp. 419-426.

[7] A. Joglekar, D. Joshi, R. Khemani, and S. Nair, Depth estimation using monocular camera, Int. journal of computer science and information technologies, vol. 2, no. 4, 2011, pp. 1758-1763.

[8] R. Lavrenov, F. Matsuno, and E. Magid, Modified spline-based navigation: Guaranteed safety for obstacle avoidance, Lecture Notes in Computer Science, 10459 LNAI, 2017, pp. 123-133.

[9] E. Magid, et al., "Artificial intelligence based framework for robotic search and rescue operations conducted jointly by international teams", in International Conference on Electromechanics and Robotics "Zavalishin's Readings". Springer, 2019, pp. 15-26.

[10] B. Abbyasov, R. Lavrenov, A. Zakiev, K. Yakovlev, M. Svinin, and E. Magid, "Automatic Tool for Gazebo World Construction: From a Grayscale Image to a 3D Solid Model", in International Conference on Robotics and Automation (ICRA), 2020, pp. 7226-7232.

[11] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator", in IEEE/RSJ International Conference on Intelligent Robots and Systems, Sendai, Japan, 2004, pp. 2149–2154.

[12] R. Amsters and P. Slaets, "Turtlebot 3 as a robotics education platform", in International Conference on Robotics and Education RiE 2017. Springer, 2019, pp. 170–181.

[13] I. Culjak, D. Abram, T. Pribanic, H. Dzapo, and M. Cifrek, "A Brief Introduction to OpenCV", in 35th International MIPRO Convention, 2013, pp. 2142–2147.

[14] O'Kane and Jason M, "A Gentle introduction to ROS", 2014.

[15] T. Foote, "tf: the transform library", in Technologies for Practical Robot Applications (TePRA), 2013, pp. 1-6. IEEE.

[16] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: an efficient alternative to SIFT or SURF", in International Conference on Computer Vision (ICCV), Barcelona, Spain, 2011, pp. 2564–2571.

[17] P. Janku, K. Koplik, T. Dulík, and I. Szabo, Comparison of tracking algorithms implemented in OpenCV, MATEC Web of Conferences. vol. 76, no. 04031, 2016.

[18] F. K. Noble, "Comparison of OpenCV's feature detectors and feature matchers", in 23rd Int. Conf. on Mechatronics and Machine Vision in Practice (M2VIP), 2016, pp. 1–6.

[19] M. Patin, "Comparative analysis of image singular point descriptors with the implementation of algorithms under the Android operating system", 2016.

[20] D. G. Viswanathan, "Features from accelerated segment test (FAST)", 2009, pp. 1-5.

[21] M. Calonder, V. Lepetit, and P. Fua, "BRIEF: binary robust independent elementary features", in European Conference on Computer Vision, 2010, pp. 778-792.

[22] E. Magid, R. Lavrenov, T. Tsoy, M. Svinin, and R. Safin, "Real-time Video Server Implementation for a Mobile Robot", in 11th International Conference on Developments in eSystems Engineering (DeSE), 2018, pp. 180-185.

[23] 1st curvilinear r [Online]. Available: youtu.be/2qqdwDdo2oA

[24] 2nd curvilinear r. [Online]. Available: youtu.be/UUx2Wse9IU4

[25] 3rd curvilinear r. [Online]. Available: youtu.be/ALMcDjOn_UA

[26] 4th curvilinear r. [Online]. Available: youtu.be/x7KeEufGWe8

[27] 5th curvilinear r. [Online]. Available: youtu.be/DRWQg49Zi_U

[28] A. Gabdullin, G. Shvedov, M. Ivanou, and I. Afanasyev, "Analysis of onboard sensor-based odometry for a quadrotor UAV in outdoor environment", in International Conference on Artificial Life and Robotics (ICAROB), 2018.