Towards development of reliable mobile robot navigation system.

Andrey Ustyuzhanin, Denis Shepelev
Moscow Institute of Physics and Technology (State University)
andrey.ustyuzhanin@mipt.ru, shepelev@phystech.edu

***Abstract*—This paper presents a combined control system for mobile robot. The developed system allows to teleoperate a mobile robot and switches to autonomous movement when the connection with the mobile robot is lost. To create this control system the comparative analysis of existing sensors was performed, based on this analysis the sensors for localization robot were chosen, the virtual model of the 4-wheeled mobile robot and a controller were developed. Then based on ROS(Robot Operating System) the combined control system was created.**

***Index Terms*—Sensors; Autonomous navigation; Robotic Operating System.**

I. Introduction

For a long time mobile robots have been operated remotely by human. With the algorithms improvements and equipment cost reducing, the development of partially or fully autonomous robotic systems becomes feasible.

Let us imagine the following situation. A special teleoperated mobile robot is used at a nuclear power plant for work in areas with a high radiation background. An accident occurred at the nuclear power plant and we decided to use the robot to eliminate emergency situation. During its movement to the accident location, the robot lost a connection with the operator. We can't send someone to pull the robot because of the life risk. Also we can't use a robot-rescuer because we may lose it too. To solve this problem we can create a control system that will allow the mobile robot to return autonomously if the connection is lost. This method could be useful for other mobile robots. In this paper we develop this system.

To achieve this goal we:

- analyze the existing solutions of the problem of autonomous navigation;
- choose sensors to localize a mobile robot in space;
- design the robot's virtual model, controller and environment to test algorithms reliability;
- develop the autonomous robot return system (*ARRS*) that should be used when connection is lost.

II. Sensors

Choice of sensors determines the reliability of localization accuracy hence the control system reliability. We analyze and compare some sensors in this work. In this paper the summary of this analysis is showed in the Tables I and II. Analyzing features of the sensors we decide to choose 2D Lidar as a main sensor because it provides the best accuracy and resolution, long range measurements and its drawbacks are not critical. Also we use IMU sensor for the odometry estimation.

III. Robot's model

We design the robot's virtual model in order to test algorithmic methods and to save costs on real equipment. The model of 4-wheel mobile robot (the Fig. 1) and the environment is created in the Gazebo software.The package Gazebo allows to emulate physical properties of robots and their interaction with the virtual environment. The robot is equipped with a 2D laser rangefinder HOKUYO and an IMU sensor [8]. We consider 2D navigation case. Therefore the Lidar is mounted fixedly and the laser beam is parallel to the plane of environment [7].

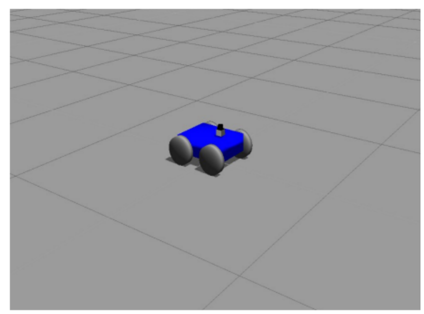

Fig. 1. The robot's model.

IV. Robot's controller

The aim of controller is to provide desired (e.g. given by user) linear and angular velocities of robot by setting rotational speeds of wheels Initially we tried to use the following kinematic model from [4]

$$v_{lin} = u_{lin}^{des}/r \qquad v_{rot} = u_{ang}^{des} \cdot c/2r$$

$$\omega_{left} = v_{lin} - v_{rot} \qquad \omega_{right} = v_{lin} + v_{rot}$$

where u^{des} - desired speed, r - wheel radius, c - distance between the centers of the right front and the left front wheels, ω_i - speed of left or right wheels. According this equations the Gazebo-plugin we develop the controller for the mobile robot.

But tests showed that this control model couldn't provide the required angular velocity of the robot. We change the controller model as follows

$$v_{lin} = u_{lin}^{des}/r \qquad v_{rot} = u_{ang}^{des} \cdot k \cdot c/2r$$

$$w_{left} = v_{lin} - v_{rot} \qquad w_{right} = v_{lin} + v_{rot}$$

TABLE I

SENSORS ADVANTAGES AND DISADVANTAGES.

Sensor	Advantages	Disadvantages
Rotary encoders	Simple and cheap sensor.	Is used only for measuring the angle of rotation and speed. Do not consider the effect of slippage.
Gyroscopic systems	Can measure the angle and the speed of rotation.	Requires calibration.
Accelerometers	Simple and cheap sensor.	Is used only to measure acceleration.
GPS	Estimates coordinates of any point of the planet.	Low accuracy. Doesn't work indoors.
Stereo cameras	Can provide full 3-D range images.	Area correlation introduces expansion in foreground objects (Smearing). These are areas where no good matches can be found because of low texture energy (Dropouts). Stereo range accuracy is a quadratic function of distance.
Sonars	Inexpensive. Acceptable accuracy and measurement speed. Good angular resolution. Not vulnerable to conditions lighting and light-reflectivity of the environment.	Provides only 2D map. Crosstalk errors. Phantom echos.
Lidars	Can provide 2D and 3D estimations. Excellent accuracy. Good angular resolution. Rapid data collection. Long range measurements.	Specular reflections from metallic and polished objects which may cause incorrect calculation of where the laser spot is illuminating. The most expensive sensor.

TABLE II

SUMMARY COMPARATIVE TABLE OF QUANTITATIVE SENSORS PARAMETERS.

	GPS	Stereo cameras	Sonars	Lidars 2D	Lidars 3D
Distance measurement accuracy	3-5m to 50m	10cm	20cm	10mm	5mm
Range of distance measurements (m)	$-$	1-50	1-5	0-80	0-120
Measurements frequency (Hz)	$-$	10-30	100	75	15
3D mapping	$+$	$+$	$-$	$-$	$+$

where k - the unknown enhancer factor control. It depends on physical parameters of the robot's model and the environment. In this paper k is calculated using linear regression method. Using the first controller we carry out a series of experiments where a desired speed is an input and the actual speed is an output. We obtain that $k = 1.26402 \pm 0.01564$.

Then using ROS-package *joy* [9] we create the node of the teleoperation.

V. ODOMETRY

To obtain reliable odometry data, firstly they are estimated as

$$v_{ang} = (w_{right} - w_{left}) \cdot r/c$$

$$v_{lin} = (w_{right} + w_{left}) \cdot r/2$$

$$v_x = v_{lin} \cdot \cos\theta$$

$$v_y = v_{lin} \cdot \sin\theta$$

$$v_\theta = v_{ang}$$

Then using ROS-package *robot_pose_ekf*(extended Kalman filter) [10] and IMU data the we get final odometry.

VI. EMERGENCY CONTROL MODULE.

Based on ROS navigation stack we develop the combined emergency control system. The ROS package *move_base* [11] provides an implementation of an action that, given a goal in the world, will attempt to reach it. The *move_base* builds a path and provides appropriate speeds to achieve the goal. It uses sensors data, a robot position in space that can be provided by the *amcl* [12] or the *fake_localization* [13] ROS packages. Using ROS *actionlib* [14] we can send to the *move_base* goals thus mobile robot will move autonomously wherever we want. The ideas of the module operation are:

- While operator is controlling mobile robot, robot current position is stored every l meters in a special *queue of the intermediate goals*(*QIG*).
- If the connection is lost, the control system will go into autonomous mode. Using the *actionlib* the control system will send the *intermediate goals* to the *move_base* if the *QIG* isn't empty. Thus the robot autonomously returns. If the current *intermediate goals* aren't reachable, the next goal will be taken from the *QIG*.

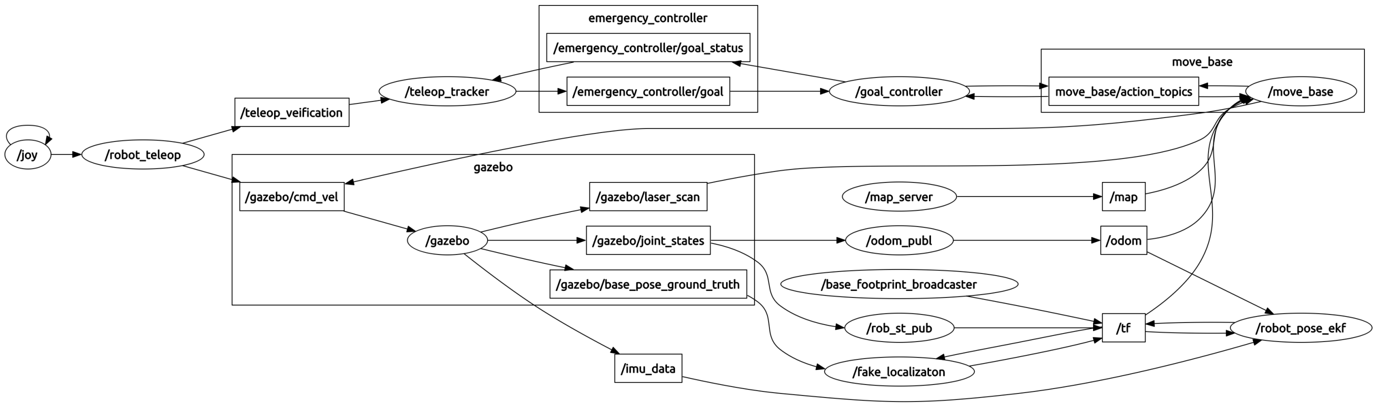

Fig. 2. Operation graph of the emergency control module.

- If the connection recovers, control will return to the operator.

To implement this ideas we create the following programs [15]:

- The *teleop_tracker* fills the *QIG* relying on *amcl* data and verifies the connection.
  1) In case of the connection failure the *teleop_tracker* sends goals to the *goal_controller*.
  2) If the current goal processing is over (achieved or aborted), the *teleop_tracker* will send the next goal.
  3) If the connection recovers, the *teleop_tracker* will send a request to stop processing goals to the *goal_controller*.
  4) If the *QIG* is empty, the movement stops.
- The *goal_controller* waits for the goals from the *teleop_tracker* and then sends them to the *move_base*.
  1) If the goal achieved or aborted by the *move_base*, the *goal_controller* notifies the *teleop_tracker*.
  2) If it gets stop-processing request, it sends stop-processing request to the *move_base*.

VII. Evaluation

The method solves the problem using the ROS opportunities. Advantage and at the same time disadvantage of this method is that, the main work relies on the robot's controller and the *move_base*. If there is a malfunction in the controller or the *move_base* work, operation of the system will be disrupted. So if we want to test our *ARRS* on a real robot, first of all we need to make sure the *move_base* reliability and create new reliable robot's controller, because our controller can be used only for simulation.

The system is tested on a common laptop with a necessary software [16]. The map of the environment (the Fig. 3, 4, 5, 6) is static with unmapped obstacles, which can appear or disappear during the robot movement. The *ARRS* can rebuild a path if the environment changes. There are some problems with a rate of the movement calculations, sometimes the robot can freeze for a minutes and then continue its movement. We think there are some issues with *move_base*, but this problems aren't fully understood.

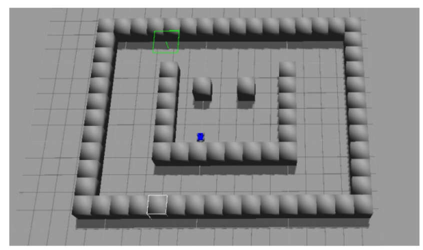

Fig. 3. First environment.

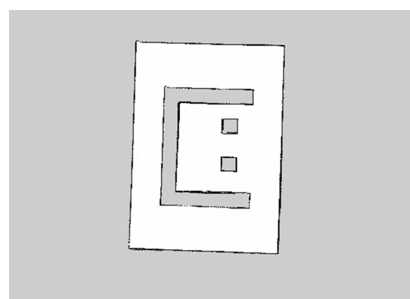

Fig. 4. First environment map.

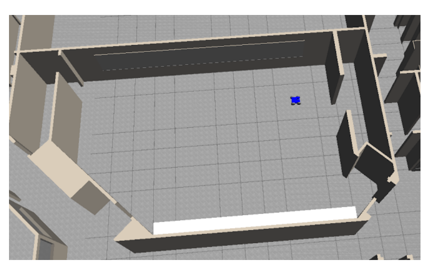

Fig. 5. Second environment.

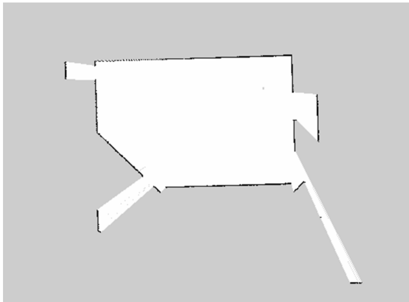

Fig. 6. Second environment map.

VIII. Conclusions

In this paper we analyzed different sensors for localization mobile robot in the environment, developed the *ARRS* in case of connection loss based on ROS packages and tested in the Gazebo simulation. The next step is adding sonar ring to ensure reliable operation in cramped conditions, testing this system on a real prototype and improving the system.

References

[1] *Springer Handbook of Robotics, Sensing and Perception.* SpringerVerlag Berlin Heidelberg. 2008. 477-540.

[2] Kiyoshi Okuda, Masamichi Miyake, Hiroyuki Takai, Keihachiro Tachibana. *Obstacle arrangement detection using multichannel ultrasonic sonar for indoor mobile robots.* Artificial Life and Robotics. September 2010, Volume 15, Issue 2, pp 229-233.

[3] Kok Seng Chong, Lindsay Kleeman. *Mobile Robot Map Building from an Advanced Sonar Array and Accurate Odometry.* MECSE-1996-10, available at http://sbp.ri.cmu.edu/papers/sbp_papers/integrated2/chong_kleeman_sonar.pdf.

[4] Krzysztof Kozlowski, Dariusz Pazerski. *Modeling and control of a 4-wheel skid-steering mobile robot* International Journal of Applied Mathematics and Computer Science, 2004, Vol. 14, No. 4, 477–496.

[5] A. Minin. *Navigation and control of the mobile robot equipped with a laser rangefinder.* the dissertation of the candidate of technical sciences: 50.02.05, Moscow, 2008. (in Russian)

[6] *Robotic Operating System. Navigation.* Available at http://wiki.ros.org/navigation.

[7] *Robotic Operating system. Gazebo 1.9 documentation.* Available at http://gazebosim.org/wiki/Tutorials#Gazebo_Version_1.9.

[8] *Robotic Operating system. Gazebo 1.9. ROS Motor and Sensor Plugins* Available at http://gazebosim.org/wiki/Tutorials/1.9/ROS_Motor_and_Sensor_Plugins

[9] *Robotic Operating system. joy documentation.* Available at http://wiki.ros.org/joy.

[10] *Robotic Operating system. robot_pose_ekf documentation.* Available at http://wiki.ros.org/robot_pose_ekf.

[11] *Robotic Operating system. move_base documentation.* Available at http://wiki.ros.org/move_base.

[12] *Robotic Operating system. amcl documentation.* Available at http://wiki.ros.org/amcl.

[13] *Robotic Operating system. fake_localization documentation.* Available at http://wiki.ros.org/fake_localization.

[14] *Robotic Operating system. actionlib documentation.* Available at http://wiki.ros.org/actionlib.

[15] *Project repository.* Available at https://github.com/denzist/robot-navigation/tree/master/src/robot_2dnav/src.

[16] *ROS Hydro.* Available at http://wiki.ros.org/hydro.