Improving model of crawler robot Servosila ”Engineer” for simulation in ROS/Gazebo

Alexandra Dobrokvashina, Roman Lavrenov
Laboratory of Intelligent Robotic Systems
Intelligent Robotics Department, ITIS
Kazan Federal University
Kazan, Russia
AlSDobrokvashina@stud.kpfu.ru, lavrenov@it.kfu.ru

Edgar A. Martinez-Garcia
Laboratorio de Robotica
Institute Engineering and Technology
Universidad Autonoma de Ciudad Juarez
Juarez, Mexico
edmartin@uacj.mx

Yang Bai
Information Science and Engineering Department
Ritsumeikan University
Kyoto, Japan
fc.ritsumei.ac.jp

2020 13th International Conference on Developments in eSystems Engineering (DeSE) | 978-1-6654-2238-3/20/$31.00 ©2020 IEEE | DOI: 10.1109/DeSE51703.2020.9450233

***Abstract*—Crawler robots are popular in tasks of urban search and rescue. Because of using tracks most of these robots have good traversability on rough terrains. For creating and testing these abilities and crawler robot itself often used simulators, which could help to reconstruct most of the conditions and environment without any difficulties. And this is a reason why most of the robots must have its simulation model.**

In this paper, we present a fully equipped and moving model of tracked robot Servosila ”Engineer”. It contains all of the sensors that used a real robot with the ability to get information from them via ROS topics through all the time of the simulation. Also, it has similar to real robot control elements, physic characteristics and useful crawler simulation.

***Index Terms*—crawler robot, modeling, track simulation, Servosila Engineer, Gazebo, ROS, urban search and rescue, USAR**

I. Introduction

The main goals of robotics was and always will be serving and helping people. Nowadays robots could be found nearly in all areas of human lives: manufactures, medicine, entertainment and etc. They play with humans, work for humans and even help to heal and save humans.

Urban search and rescue (USAR) is one of the application which robots are successfully used for. There are many ways to use this kind of technologies in such extreme conditions as natural and technological disasters and many other places, where it is too dangerous for human to work. For example, first time rescue robots were tested was during events of disaster at the World Trade Center in 2001. Center the robotic assisted search and rescue (CRASAR) used different sized radio controlled robots for eight times there [1]. Importance of USAR robotics was also emphasized by situation that took place in Fukushima [2]. Working in condition of radioactive pollution which is deadly dangerous for human could be not a problem for specified robot [3]. More than that robots because of various of their configurations could be not only human replacement, but also do any work better and faster [4].

One of the most useful type of the robots in conditions of uneven terrain which is very common things for USAR operations is crawler UGV [5]. Because of tracks traversability of this kind of robots could be extremely high. There are several works about moving algorithms for caterpillar vehicle in random step environment such as [6], [7]. And in this article we talk about one of those robots - Mobile Robot ”Engineer” [8] created by Servosila company (Fig. 1).

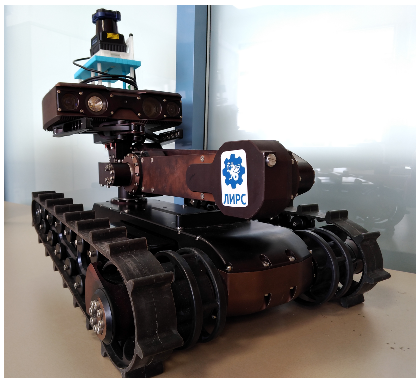
Fig. 1. Servosila “Engineer” crawler-type mobile robot. Courtesy of Servosila company.

Talking about search and rescue robotics we should remember that environment for testing created algorithms could be very hard to produce - radioactive pollution, ruined town don’t look like everyday routine. So it is obviously that it could be hard to simulate working environment for those robots in real life, and here comes simulators. Robot simulators [9] provide abilities to create various virtual environment for the robot to work in [10], that is why most of the robots should have simulation model [11], especially robots for the purposes of USAR [12].

For a couple years there were several attempts to work on ”Engineer” model such as [13], [14]. Most of them work directly on track simulation, manipulator or controllers. There were presented different ways of simulating tracks and ways of control this robot. But this article is going to provide information about creating completely new model for the Servosila ”Engineer” robot, which maximally repeats the behavior and the physic characteristics of real robot.

II. System setup

A. Crawler robot Servosila ”Engineer”

Crawler-type mobile robot ”Engineer”, which designed and produced by Russian company Servosila [15] specially for applications in human-unfriendly environment such as firefighting, search and rescue, industrial engineering and maintenance of tunnels and pipelines. It has protection from water and dust, so it could be used in different weather conditions in open air. Couple of crawlers and moving flippers make ”Engineer” easily overcome most of the obstacles on its way. Created in metal body with radiation-hardened electronics and several sensors on the board it weight 16,3 kg (Table I). Robot equipped with robotic arm with the gripper and so-called ”head” on the end. ”Head” of the robot contains 4 cameras (three in front and one in back), lantern, IMU sensor laser scan and most of computing powers. Last one presented with powerful computer which could control all the joints and work with data from sensors. Also because of modular construction and different abilities this robot became a popular education platform for laboratories and universities [16], [17].

TABLE I
”Engineer” robot’s weight.

Equipment	**Weight parameter**
Robot chassis with two main reversible tracks, two traction motors and motor control electronics	**8.8 kg**
On-board control and power system	**2.1 kg**
Sealed connector for external payloads or external computer	**0.1 kg**
LiFePo battery	**3.7 kg**
Power supply for standard robot battery (with cable)	**1.6 kg**

B. Simulation environment of ROS/Gazebo

As already been mentioned main goal of robot simulator is allow to test robot in different simulated environments, that could easily safe time of development and decrease its cost. In this article we perform ”Engineer” robot’s model and simulation in simulator Gazebo 9, which integrated with ROS Kinetic and have possible integration with ROS Melodic.

We consider ROS (Robot Operating System) as useful open-source framework for a mobile robot development. It provides hardware abstraction, low-level motor control, package management and communication between different services and processes via topics and messages. From the box ROS has very good integration with Gazebo, so this couple is nearly the best decision for mentioned task.

III. Model improvement

In this article we base on results of previous works and present new model of the robot with such improvements as collision meshes and models itself, adding moving gripper, sensors, cameras, lantern and simple track simulation.

A. Model editing and texturing

First step of our work was editing collision meshes, so that simulation could be counted faster [18]. For a note most heavy calculations of simulation absolutely counting of physics which uses shapes of collision meshes, that is why our golden rule is ”the easier collision meshes are, the faster simulation is” [19].

Meshes that used earlier for calculating collisions were simply automatic generated from original meshes of parts. And as the result they were absolutely not optimized for their purpose, mostly because of geometry. Example of different meshes illustrated in ‘Fig. 2’.

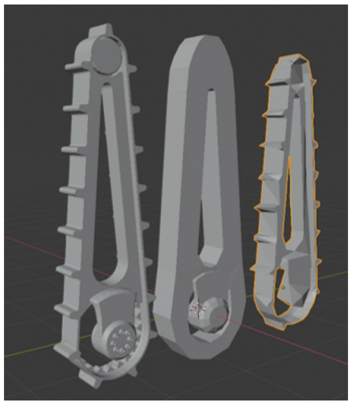

Fig. 2. Mesh examples (left – original mesh of robot, middle – new, right – old)

In the end of working on model count of polygons was decreased by one hundred from old auto-generated collision meshes of robot. At the same time count of parts in the model grow from nine to fifteen due to breaking previously single model of ”head” combined with not moving gripper into model of ”head” and divided in different links gripper (also called end effector or EE). In the Fig. 3 illustrated difference in collision meshes of initial 9 parts.

Fig. 4 shows final difference between start model and final one. Here could be found such differs as corrected normal of one of the flippers (it is black on the picture), putting both of them on right places and already mentioned division of mesh of the ”head”.

As a final point of working on model we added textures to give model of robot maximally realistic look. Result illustrated in Fig. 5

In the end we compare several different collision meshes, such as visual meshes, old auto-generated meshes and new ones. The result shown in the Table II.

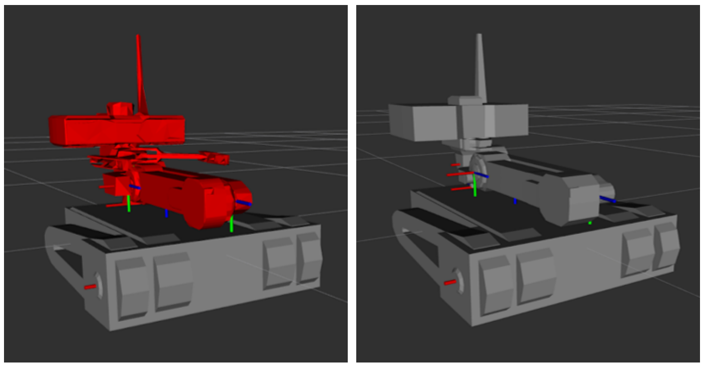

Fig. 3. Collision meshes of robot without end effector (red – old, gray – new)

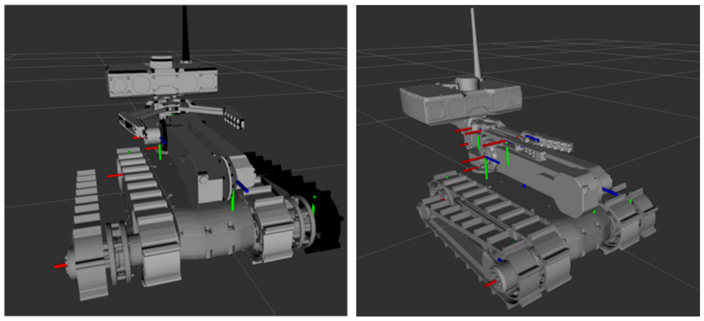

Fig. 4. Full robot model (left – old, right – new)

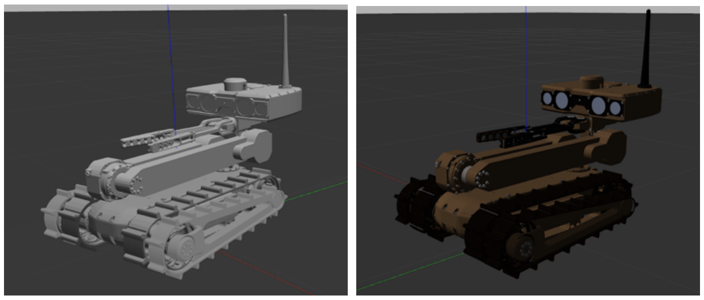

Fig. 5. Robot before and after texturing (Gazebo)

TABLE II
COMPARE OF DIFFERENT COLLISION MESHES

Mesh	**Visual meshes used as collision**	**Old auto-generated collision meshes**	**New manually created collision meshes**
Elements count	**15**	**9**	**15**
Polygons	**967 219**	**5 712**	**5 644**
RTF (Real Time Factor)	**$\approx$ 0.2**	**$\approx$ 1**	**$\approx$ 1**

B. Adding sensors and lantern

One of the most important things for the robot is the way to interact with environment and all the ways to sense it. ”Engineer” as it has been mentioned has several sensors (laser scan and IMU sensor), 4 cameras and as addition help for cameras there is powerful lantern, which could be turn on if there is need.

Laser scan basically used to count distance to the objects around robot. This sensor definitely is one of the most popular sensors in the robotics, because it took place nearly in every mobile robot.

Next goes IMU sensor or inertial measurement unit. It used to take information about position of the robot in space: angular rate, specific forces and orientation.

The last but not the least - cameras. Through being very simple from the first view, camera could be very important source of data for robot. Its easiest mode of application is teleoperation [20]. On the other hand if data from camera will be processed good enough we could now nearly anything: position, orientation, get information about environment e.t.c [21].

Gazebo simulator has huge amount of plugins to simulate many different things: lasers, cameras, bumpers, controllers and others. Because of being open-source project any new plugin could be manually wrote and successfully added [22]. In this work we mostly use basic plugins for laser scan, cameras and IMU sensor. They have all necessary for us settings such as topics for data, ranges of sensing and even noise level. They took their places in the ”head’ according to the real robot, and it could be seen in Fig. 6. Below listed example of XML code from URDF (Unified Robot Description Format) [23] file with description of the robot which used for setting up IMU sensor in our simulation model:

```
<gazebo reference="imu_link">
<gravity>true</gravity>
<sensor name="imu_sensor" type="imu">
<always_on>true</always_on>
<update_rate>100</update_rate>
<visualize>false</visualize>
<topic>__default_topic__</topic>
<plugin filename="libgazebo_ros_imu_sensor.so"
name="imu_plugin">
<topicName>/imu</topicName>
<bodyName>imu_link</bodyName>
<updateRateHZ>10.0</updateRateHZ>
<gaussianNoise>0.0</gaussianNoise>
<xyzOffset>0 0 0</xyzOffset>
<rpyOffset>0 0 0</rpyOffset>
<frameName>engineer/imu_link</frameName>
</plugin>
<pose>0 0 0 0 0 0</pose>
</sensor>
</gazebo>
```

Of course through working with sensors in simulation and robot itself we used visualisation tool rViz. It helps not only check if created sensors are working, but also visualize data from most of them them. For example in Fig. 7 illustrated four pictures from all of the cameras(three in front and one in

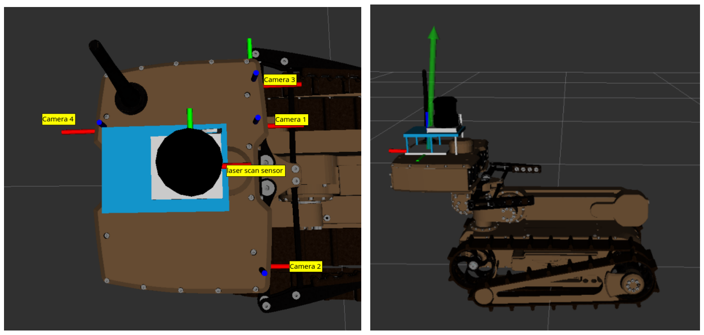

Fig. 6. Sensors and cameras(left - laser scan and 4 cameras, right - IMU sensor and its data

the back of the ”head”). The bottom picture in Fig. 7 shows actual place of robot in the scene.

Another moment in this part of work was adding the lantern to the robot.

One of the solution for this task was Flash Light plugin for Gazebo that used for attaching light source to objects. Using it light source need to be created apart from the robot. And in the end it appears to be not very comfortable if there is need to move robot to new simulation world.

So after some changes in hierarchy it turned out that attaching light source to robot was not so hard from the beginning: light source (type of spot light which mostly common to real life torch) just added in the description of the robot [19]. Below listed part of XML code URDF file with description of the robot which responsible for that:

```
<gazebo reference="head_link">
<light type="spot" name="spot">
<pose>0.08 0.11 -0.04 0 0 0</pose>
<diffuse>1 1 1 1</diffuse>
<specular>.2 .2 .2 1</specular>
<attenuation>
<range>10</range>
<linear>0.01</linear>
<constant>0.2</constant>
<quadratic>0.0</quadratic>
</attenuation>
<direction>0 0 -1</direction>
<spot>
<inner_angle>0.1</inner_angle>
<outer_angle>0.5</outer_angle>
<falloff>1.2</falloff>
</spot>
<cast_shadows>true</cast_shadows>
</light>
</gazebo>
```

Realisation of its turning off and on was done by calling service called ’set light properties’ produced by Gazebo via ROS and changing state of the light source. Example of working lantern in environment without any further light sources shown in Fig. 8.

C. Adding controllers

Next step in our work was adding controllers to the joints. Some of them had already took their places such as shoulder, waist and head joints. We added controllers to flippers and to

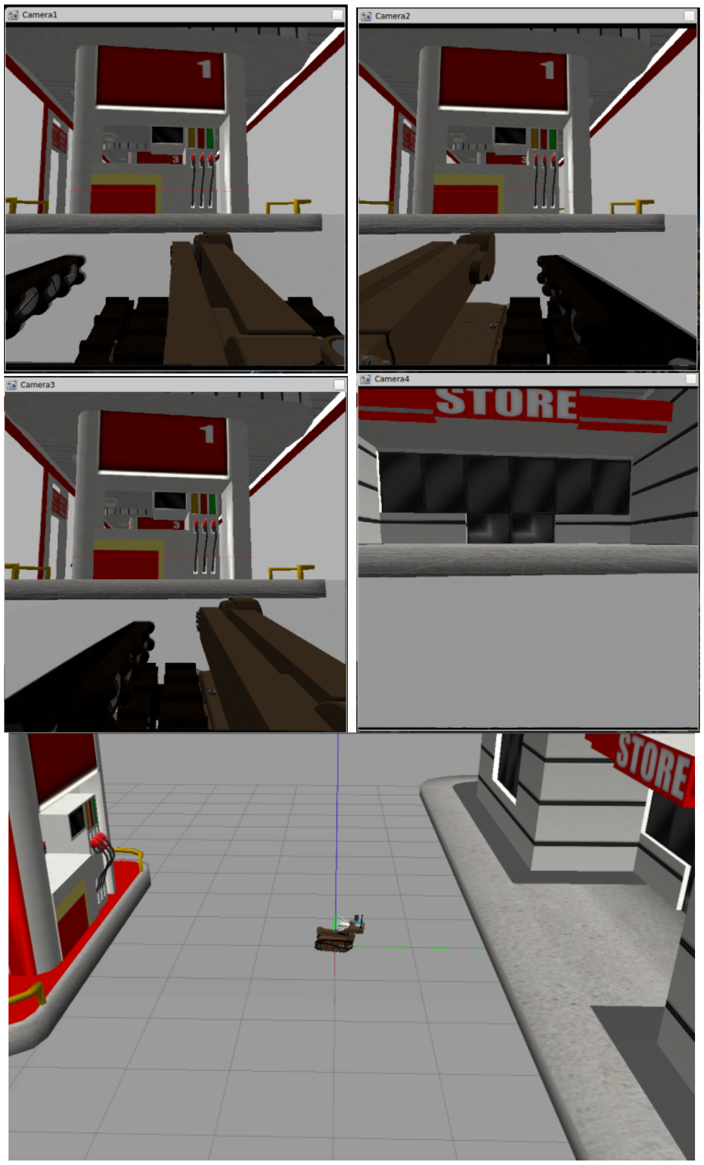

Fig. 7. Data from cameras and place of Engineer in the scene.

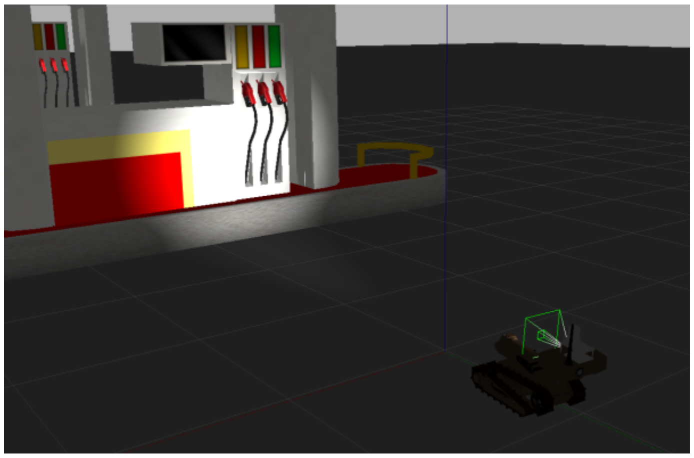

Fig. 8. Enabled robot lantern

make it similar to real ”Engineer” which flippers move in sync we added to one the flippers joint mimic option. For simulating such behavior in Gazebo was used Mimic Joint plugin which could be found in [24].

Also several position controllers where attached to end effector. Need to be mentioned that gripper on ”Engineer” has hierarchy of closed loop, so there again were used several mimic joints. Below listed example of code from URDF description of robot which responsible for one of the mimic joints in end effector:

```
<joint name="p3_left_ee" type="continuous">
<parent link="part_one_left"/>
<child link="part_three_left"/>
<dynamics friction="1.0" damping="1.0"/>
<axis xyz="0 1 0"/>
<mimic joint="p1_left_ee" multiplier="-1"/>
<origin xyz="${reflect*0.008} 0 0.145"
rpy="0 ${-reflect*pi/2-reflect*pi/10} 0"/>
</joint>
<gazebo>
<plugin name="p3_left_ee_mimic_plugin"
filename="libroboticsgroup_
gazebo_mimic_joint_plugin.so">
<joint>p3_left_ee</joint>
<mimicJoint>p1_left_ee</mimicJoint>
<multiplier>-1.0</multiplier>
<maxEffort>1.0</maxEffort>
<hasPID />
<robotNamespace>engineer</robotNamespace>
</plugin>
</gazebo>
```

In the end five out of six joints in end effector became mimic. So that to make entire gripper move you need to set position only for one joint. In Fig. 9 there are illustrated several intermediate positions of end effector.

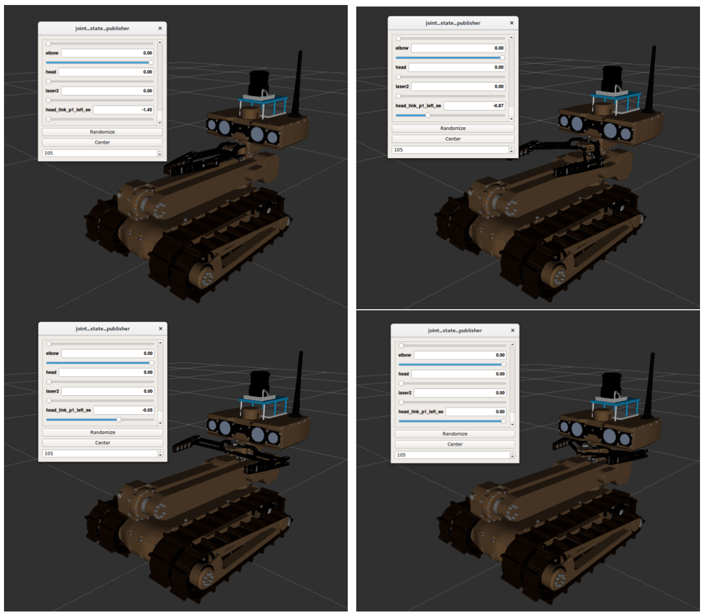

Fig. 9. Example of configurations of end effector (first - closed, second - open, third - middle, fourth - hidden).

IV. Crawler simulation

One of the most hard thing about creating model of crawler robot is making simulation of the tracks [25]. For now there are a lots of different solutions for this problem, especially for non-deformable tracks like ones that used by ”Engineer” [26], [27].

As already mentioned there were several proposed variant for simulation exactly ”Engineer” crawlers and this time for its simulating we create fifty six imaginary wheels for each of the tracks and connect them with differential drive controller as it was already proposed in [13]. It helps to drive robot in all directions maximally accurate. Below listed XML code from yaml-file with example of configuration of differential drive controller:

```
mobile_base_controller:
type: diff_drive_controller/DiffDriveController
publish_rate: 50
left_wheel: [...]
right_wheel: [...]
pose_covariance_diagonal: [0.001, 0.001,
1000000.0, 1000000.0, 1000000.0, 1000.0]
twist_covariance_diagonal: [0.001, 0.001,
1000000.0, 1000000.0, 1000000.0, 1000.0]
base_frame_id: engineer/base_footprint
enable_odom_tf: true
linear:
x:
has_velocity_limits: true
max_velocity: 20.0  # m/s
min_velocity: -20.0 # m/s
has_acceleration_limits: true
max_acceleration: 0.15  # m/s^2
min_acceleration: -0.15 # m/s^2
has_jerk_limits: true
max_jerk: 0.30  # m/s^3
angular:
z:
has_velocity_limits: true
max_velocity: 6.7  # rad/s
has_acceleration_limits: true
max_acceleration: 0.08  # rad/s^2
has_jerk_limits: true
max_jerk: 0.08  # rad/s^3
```

For the simulation tracks on flippers similar type of wheels were used. In this case to fully cover contact surface of each of them we used fifty wheel. Wheels on the both of the flippers are controlled by differential drive controller.

Together four tracks use two hundred and twelve wheel and visualise all of them is a quite hard task. So for making view of robot more clear and realistic we make visual part of the links of these imaginary wheels much smaller so they become literally invisible. The difference in the view could be seen in Fig. 8.

V. Conclusion

In this paper we presented fully equipped and moving model of Russian crawler-type robot ”Engineer” produced by Servosila company. Model has all of the sensors on board, crawler simulation and controllers to manipulate every joint. All of the sensors (IMU and laser scan) and cameras are

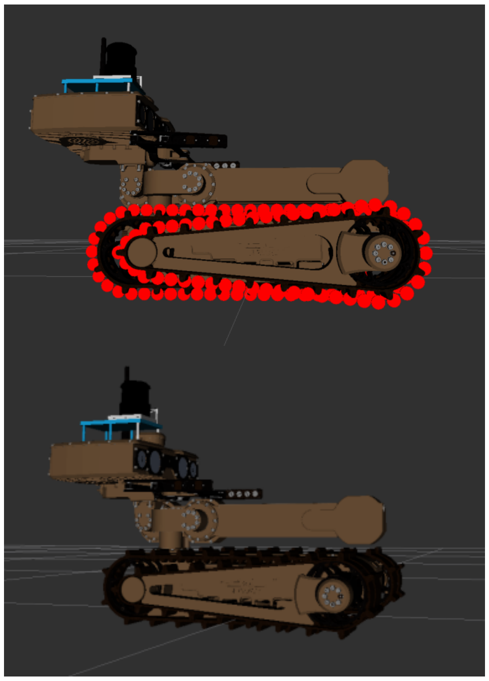

Fig. 10. Crawler simulation(top-visible wheels, bottom - invisible wheels)

working in Gazebo environment and sharing their data via topics in ROS. Crawlers simulated by eighteen imaginary wheel for each of main tracks and thirty one wheel for each of flipper tracks, all of the wheels finally controlling by the two differential drive controllers proposed by ROS.

VI. Future work

Creating realistic model of the robot for simulations is a very complicated task and it gets even more difficult if we talk about crawler robot such as "Engineer".

In the nearest future we are going to make more simulation experiments with this model to compare behavior of the model and real robot in the same conditions. This will help us detect any problems and erase them by improving our model.

Acknowledgment

The reported study was funded by the Russian Foundation for Basic Research (RFBR), according to the research project No. 19-58-70002. This work was partially supported by the research grant of Kazan Federal University.

References

[1] M. Blackburn, H. Everett, and R. Laird, After action report to the joint program office: Center the robotic assisted search and rescue (CRASAR) related efforts at the world trade center (No. SSC/SD-TD-3141). Space And Naval Warfare Systems Center San Diego CA, 2002.

[2] Nagatani, Keiji, et al. Emergency response to the nuclear accident at the Fukushima Daiichi Nuclear Power Plants using mobile rescue robots. Journal of Field Robotics 30.1 (2013), pp. 44-63.

[3] Iborra, Andres, et al. Robots in radioactive environments. IEEE Robotics & Automation Magazine 10.4 (2003), pp 12-22.

[4] R. R. Murphy, A decade of rescue robots. International Conference on Intelligent Robots and Systems, 2012, pp. 5448-5449.

[5] K. Ohno, S. Morimura, S. Tadokoro, E. Koyanagi, T. Yoshida. Semi-autonomous control system of rescue crawler robot having flippers for getting over unknown-steps. IEEE/RSJ International Conference on Intelligent Robots and Systems, 2007, pp. 3012-3018.

[6] E. Magid and T. Tsubouchi, Static Balance for Rescue Robot Navigation: Translation Motion Discretization Issue within Random Step Environment, International Conference on Informatics in Control, Automation and Robotics (ICINCO), Portugal, 2010, pp 415-422.

[7] K. Nagatani, A. Yamasaki, K. Yoshida, T. Yoshida and E. Koyanagi, Semi-autonomous traversal on uneven terrain for a tracked vehicle using autonomous control of active flippers, International Conference on Intelligent Robots and Systems (IROS), 2008, pp. 2667-2672.

[8] I. Mavrin, R. Lavrenov, M. Svinin, S. Sorokin, and E. Magid, Remote control library and GUI development for Russian crawler robot Servosila Engineer, MATEC Web of Conferences, vol. 161, 2018, p. 03016.

[9] N. Koenig, A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. International Conference on Intelligent Robots and Systems (IROS), 2004, Vol. 3, pp. 2149-2154.

[10] B. Abbyasov, R. Lavrenov, A. Zakiev, K. Yakovlev, M. Svinin, E. Magid, Automatic tool for Gazebo world construction: from a grayscale image to a 3D solid model. In 2020 IEEE International Conference on Robotics and Automation (ICRA), 2020, pp. 7226-7232.

[11] Shimchik, I., Sagitov, A., Afanasyev, I., Matsuno, F., Magid, E. Golf cart prototype development and navigation simulation using ROS and Gazebo. In MATEC Web of Conferences, 2016, Vol. 75, p. 09005.

[12] A. Wolf, H. H. Choset, B. H. Brown, R. W. Casciola. Design and control of a mobile hyper-redundant urban search and rescue robot. Advanced Robotics, 2005, vol. 19(3), pp. 221-248.

[13] I. Moskvin, R. Lavrenov, E. Magid and M. Svinin. Modelling a Crawler Robot Using Wheels as Pseudo-Tracks: Model Complexity vs Performance. In 7th International Conference on Industrial Engineering and Applications (ICIEA), 2020, pp. 1-5.

[14] I. Moskvin and R. Lavrenov, Modeling Tracks and Controller for Servosila Engineer Robot, 14th International Conference on Electromechanics and Robotics "Zavalishin's Readings", 2020, pp. 411-422.

[15] Servosila official site https://www.servosila.com/en/index.shtml

[16] M. Yim, D. G. Duff, K. D. Roufas. PolyBot: a modular reconfigurable robot. In Proceedings International Conference on Robotics and Automation, 2000, Vol. 1, pp. 514-520. IEEE.

[17] S. Carpin, M. Lewis, J. Wang, S. Balakirsky, C. Scrapper. USARSim: a robot simulator for research and education. In Proceedings International Conference on Robotics and Automation, 2007, pp. 1400-1405.

[18] S. Chitta, I. Sucan, S. Cousins. Moveit![ros topics]. Robotics and Automation Magazine, 2012, vol. 19(1), pp. 18-19.

[19] L. Joseph, J. Cacace. Mastering ROS for Robotics Programming: Design, build, and simulate complex robots using the Robot Operating System, 2018, Packt Publishing Ltd.

[20] J. Vertut, P.C. Coeffet. Robot Technology; vol. 3A Teleoperation and Robotics Evolution and Development, 1986.

[21] Ramil, S., Lavrenov, R., Tsoy, T., Svinin, M., & Magid, E. Real-Time Video Server Implementation for a Mobile Robot. In 11th International Conference on Developments in eSystems Engineering (DeSE), 2018, pp. 180-185. IEEE.

[22] Joseph, Lentin, and Jonathan Cacace. Mastering ROS for Robotics Programming: Design, build, and simulate complex robots using the Robot Operating System. Packt Publishing Ltd, 2018.

[23] Universal robot description format (urdf). http://www.ros.org/urdf/.

[24] Roboticsgroup Gazebo Plugins. https://github.com/roboticsgroup/roboticsgroup_gazebo_plugins.

[25] M. Pecka, K. Zimmermann and T. Svoboda, Fast simulation of vehicles with non-deformable tracks, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2017, pp. 6414-6419.

[26] S. Xie, B. Shilong, Z. Bin, P. Huayan, L. Jun, and G. Jason, The research on obstacle-surmounting capability of six-track robot with four swing arms, IEEE International Conference on Robotics and Biomimetics (ROBIO), 2013, pp. 2441-2445.

[27] D. Calisi, D. Nardi, K. Ohno and S. Tadokoro, A semi-autonomous tracked robot system for rescue missions, IEEE SICE Annual Conference, 2008, pp. 2066-2069.