A Virtual Experiment Platform for 2D Robot Autonomous Navigation Algorithm System Based on ROS

Yanshu Song, Tieshan Zhang, Bing Li*, Hailin Huang
Harbin Institute of Technology(Shenzhen), Shenzhen, 518055, China
Corresponding Author:libing.sgs@hit.edu.cn

***Abstract*— The robot autonomous navigation algorithm system enables the robots to autonomously plan path and move to target point in an unknown environment. Researchers have done a lot of work to optimize each sub module of this system, such as localization, global planning, local planning, etc. This paper proposes a virtual experiment platform, which can quickly verify the feasibility of an user own designed sub-algorithm while the user only needs to do some simple interface configuration work. It can play an important role in accelerating the research of robot autonomous navigation algorithm system. In this paper, the specific implementation process of each part is introduced and the final running results are obtained by selecting the most common algorithm of each sub module into the platform.**

***Index Terms*— Navigation algorithm. Virtual experiment platform. Robot.**

I. Introduction

A. Background

In recent years, SLAM(Simultaneous localization and mapping) algorithm is developing rapidly, which allow a robot in unknown environment to build the map of surrounding environment and calculate its location in real time. The autonomous navigation algorithm system[1] generated by the combination of path planning algorithm and SLAM algorithm can realize autonomous driving of mobile robot which has been applied to many fields, such as Unmanned vehicle, City search and rescue robot, Unmanned aerial vehicle and so on. As the Fig. 1 shows, this algorithm system is always composed of several sub algorithms: central processing part, global planning part, local planning part, localization part, bottom controller part, lidar driver part and map server part.

For the upper computer algorithm system, the most important three sub parts are the global planning partlocal planning part and the localization part. Each sub part has many corresponding algorithms that can be choosed ,for example, the EKF-Locolization algorithm[2] based on Extended Calman Filter and the AMCL[3] algorithm based on Monte Carlo localization are always used in localization part. Similarly, the A star[4] algorithm are always used in global planning and the Artificial Potential Field[5] method and Timed Elastic Band algorithm[6] are always used in local planning. In order to make the robot better adapt to the complex environment, scholars have constantly improved these sub algorithms and there are also many people research ways and tools to speed up the algorithm researching itself correspondingly.

*This work is supported by the Joint Funds of the National Natural Science Foundation of China [Grant No.U1613201], and in part by Science and Technology Project of Guangdong,China [Grant No.2016A010102004].

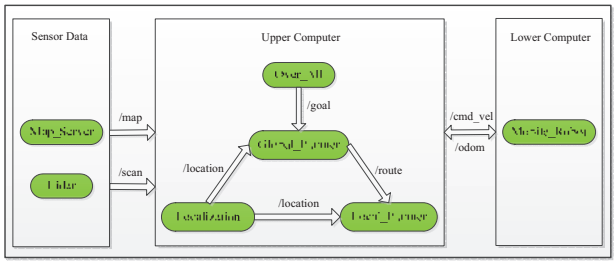


Fig. 1: Common framework for robot autonomous navigation algorithm system.

B. Related Work

Robot Operating System(ROS)[7] as a robotics middleware was started to support the Stanford AI Robot STAIR (Stanford AI Robot) project in 2007. It provides a lot of convenience for designing of heterogeneous computer cluster such as hardware abstraction, message-passing between processes, low-level device control and package management. With the way of node communication, the sub modules of navigation algorithm can be conveniently integrated into a system. Mobile Robot Programming Toolkit (MRPT) [8] is an open source tool set which provides many portable and well-tested applications and libraries covering data structures and algorithms employed in mobile robotics areas. MRPT makes it possible to add function in a mobile robot navigation system quickly. Arthur Huletski in St. Petersburg University put forward a SLAM research framework[9] based on ROS which provides a set of C++ classes which contain information and functions commonly called in robot autonomous navigation algorithm system.

But there is also a lot of work to do before getting a robot navigation platform for sub algorithm test, such as the configuration of the necessary data information, the definition of the data interface between sub algorithms and so on. To the best of our knowledge, a framework or platform that can quickly test sub algorithms of robot navigation system

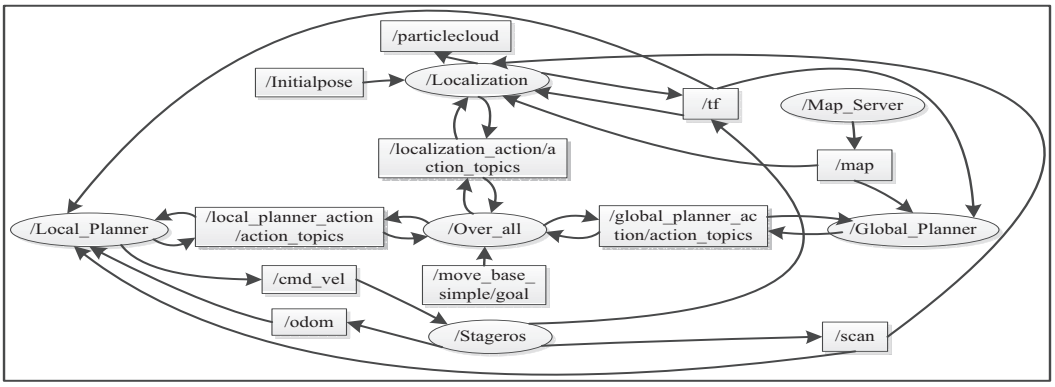


Fig. 2: System node diagram of the virtual experiment platform.

has't been put forward. So the paper introduces a ROS based virtual experiment platform which can play a role in accelerating the research of the robot autonomous navigation algorithm system, since the only thing users need to do to test whether their own designed sub algorithms are feasible is to put it in the corresponding location in our platform and simply configure the data interface.

C. The Arrangement of This Paper

The remainder of the paper is organized as follows: Section II introduces the basic architecture of our platform as well as the basic implementation methods of each sub module. Section III introduces the process of building robot motion environment with stage and details the environmental configuration parameters. In Section IV, the concrete realization method of each sub module is discussed. Afterwards, in Section V a commonly used algorithm in each sub module is selected and put into the experimental platform, and the final running result is obtained. Finally Section VI concludes the paper.

II. System Overview

A. System Node

The virtual experiment platform of this paper is based on ROS and it is composed of five nodes of Overall node, Global Planner node, Local Planner node, Localization node and Stageros node with each node responsible for a sub task. These nodes communicate between each other by publishing and subscribing messages. The specific system framework is shown in Fig. 2.

As the core node of this virtual experiment platform, the Overall node releases the command to Localization node to start or stop the localization function and subscribes to robot goal position topic published by the top and send it to the Global Planner node for global planning. It receives the feedback data(global route) from the Global Planner node in real time and then sends it to the Local Planner node for local path planning. In this way, the Overall node realizes the global control of the other nodes, and then realizes the coordinated and stable running of the system.

The Stageros node is produced by the 2D simulator stage which represents a robot moving in an unknown environment. It can publish the /odom information and the virtual lidar data based on the world environment provided by stage and subscribe the /cmd-vel topic to move the robot. The Global Planner node receives the goal position from Overall node and integrates information from /map and /tf topics to generate global route through certain global planning algorithm. The Local Planner node receives the route generated by Global Planner node and integrates information from /scan, /odom and /tf topics to calculate the direct control command /cmd-vel which contains the real-time speed information of the translation and rotation of the robot. Similarly, the Localization node subscribes to /map, /tf, and /scan topics and then calculates the location of the robot in real time through a certain localization algorithm, which is essential to all the other nodes of the autonomous navigation algorithm system.

B. Open Source Library

In the writing process of the virtual experiment platform, we used some open source librarys to ensure that the system code is portable and easy to maintain.

Actionlib of ROS[10] provides a standardized interface for interfacing with preemptable tasks such as moving the base to a target location, performing a laser scan and returning the resulting point cloud, detecting the handle of a door, etc. Compared to ROS services, it provides tools to create servers and clients interface conveniently and qucikly and the more important is that it allows users to cancel the request during execution and get periodic feedback about how the request is progressing. In the virtual experiment platform described in this paper, the three nodes of Global Planner node, Local Planner node, Localization node run as the server in each corresponding domain while the Overall node works as a public client. It sends the request and receives feedback result of the target execution from these servers

in real time, and then makes timely treatment according to different circumstances.

Protocol buffers[11] are Google's language-neutral, platform-neutral extensible mechanism for serializing structured data. It automatically generates a series of library functions which greatly facilitate users to read and write custom configuration files. Since there are a large number of configuration parameters need to be modified according to specific usage for different users, we need to read and write them many times during the implementation of the platform. As the official website introduces, the open source library plays a great role in this process.

In addition, to start this virtual experiment platform needs to configure many startup parameters, such as map selection, and start all the nodes introduced above which also has a amount of work. To deal it we write a .sh script which contains all the required boot commands together which makes the startup process of the virtual experiment platform simple and quick.

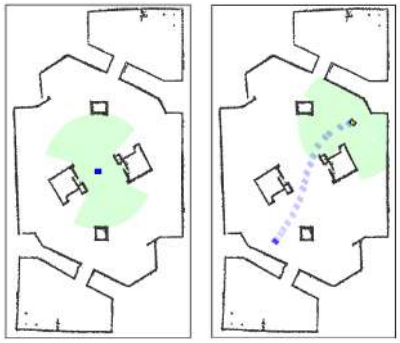
III. Simulation Robot Environment With ROS Stage

Stage is a 2D multi-robot simulator wraped by the stageros node via libstage. The environment simulated by it is defined in a .world file which contains all simulation information from obstacles, robots, to laser, camera. In this virtual experiment platform, stageros node works as a virtual robot that can publish and subscribe to the necessary topics for robot autonomous navigation algorithm, such as /scan, /odom and /cmd vel like a real robot. The specific implementation details will be introduced one by one below.

A. Configuration Process

As mentioned above, all information of the simulation environment comes from the .world file, so the core configuration process is to rewrite .world file's parameters according to the specific simulation needs. This file allows the user to define its own model parameters, including obstacles, robot, lidar, camera and floorplan model of the virtual environment. In this virtual experiment platform, we use the map of RoboMaster competition 2018 of DJI company as the floorplan model, and introduce a omnidirectional wheel moving robot which carries a lidar with a scanning angle of 360 degrees.

As the figture shows, the blue and gray areas represent the mobile robot and the obstacles while the green area represents the real-time scanning area of the vehicle radar and the outermost gray border represents the boundary of robot moving environment. Obviously, this is consistent with our expectations. In addition, the map of the vitual environment is a picture in pgm format which is a lowest common denominator grayscale file format and easily to learn and write programs. It can be produced by using the common SLAM algorithm to measure the actual environment, and can also be produced by the design of the picture making software which makes it easy to simulate different scientific research scenes.



(a) (b)

Fig. 3: Stage simulation.(a)Basic simulation environment;(b)The movement trajectory of virtual robot when subscribes to /cmd-vel topic.

(a) (b)

Fig. 4: Display results of stageros node publishing topics in rviz in different position.(a)Central position;(b)Upper right corner position.

B. Effect Verification

After configuring the .world file as described above, we get a virtual 2D robot simulation environment which is shown in Fig. 3(a). Before the other parts of the virtual experiment platform are written, we verify the validity of the simulation environment from two aspects: whether it can publish the correct data(include /lidar data and /odom data) and whether it can subscribe to the cmd-vel topic and make the robot move according to the corresponding speed information.

1) Publish Data: Open the stagoros node then move the virtual robot from the center to the upper right corner, we get the /scan and /odom topics data in rviz as shown in Fig. 4(a) and Fig. 4(b). The red dot line represents the obstacle contour obtained by lidar. The coordinates in the upper right corner and the diagram represent the origin and direction of /map and /odom respectively. It can be seen that the contour of

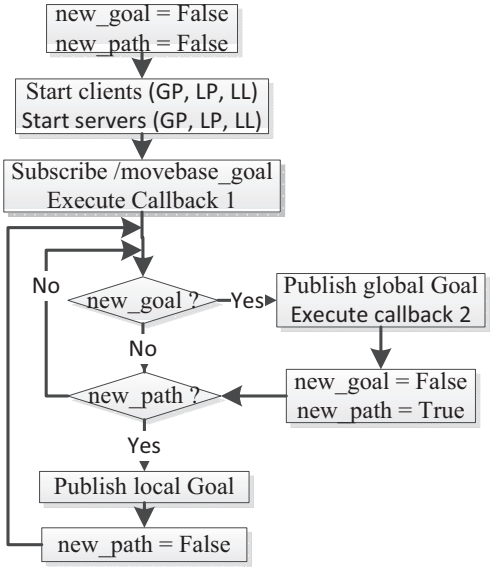


Fig. 5: Running process of the Overall node.

the obstacles scanned by the lidar basically coincides with the position of the virtual obstacle in the simulation environment, and the coordinate of the /odom can also accurately reflect the real time position of the virtual robot.

2) Subscribe Data: Turtlesim is a commonly used open source ROS package which can transform keystrokes into /cmd-vel topics and control robot movement. We use this package to control the virtual robot moving from the lower left corner to the upper right corner in stage and get the robot's footprint as shown in Fig. 3(b). It can be clearly seen that the robot can subscribe to /cmd-vel topic in real time and perform motion correctly.

To sum up, the robot simulation environment built through stage can meet the requirements of the robot autonomous navigation algorithm research from publishing to subscribing to the topics.

IV. Implementation of Autonomous Navigation Algorithm System

The virtual experiment platform is based on ROS and the upper algorithm of robot autonomous navigation system is mainly composed of four nodes: Overall node, Global Planner node, Local Planner node and Localization node. The writing of this platform is also around the implementation of these nodes and we will explain their implementation details one by one in this section, including the input and output data, the main structure, the code running process, etc.

A. Overall Node

Overall node is the core node of the virtual experiment platform. It works as a public client to coordinate the whole system while the other three nodes work as servers in their respective fields. Its general principle of work is that, it firstly start the Localization node to locate robot in a real time, then subscribes to the /move-base-simple goal topic and sends it to the Global Planner node as the global planning target, meanwhile, it accepts the feedback results(global route) of the global planning server and finally sends the route to the Local Planner node for local planning. The specific flow chart is shown in Fig. 5.

It is worth noting that:

Note1: the function of the callback 1 is to pass the data obtained by subscribing to the /move-base-simple goal topic to the Global Planner node as the goal of global path planning and then assign the entry parameter new_goal to true.

Note 2: the function of the callback 2 is to pass the route obtained from the feedback result of the Golbal Plannner node to the Local Planner node as the goal of local path planning and then assign the entry parameter new_path to true.

From the above flow chart, we can clearly know the specific running process of Overall node. It is worth mentioning that the Overall node is generated by compilation of the overall.cpp which defines a class called OverallNode containing all input and output data and necessary functions. The only thing its main function need to is to instantiate this class.

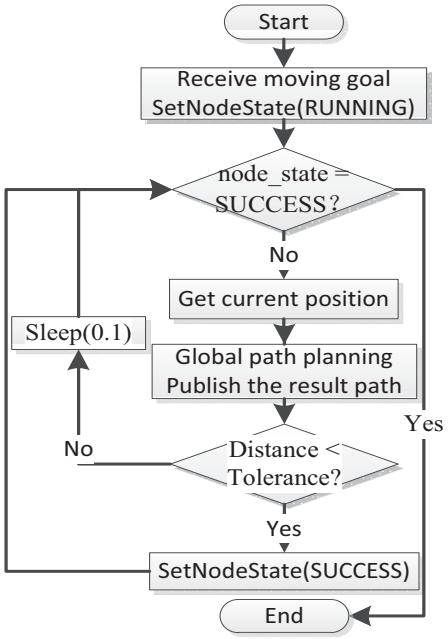


Fig. 6: Running process of the callback function of the Global Planner node.

B. Global Planner Node

The Global Planner node works as a global planning server. It is responsible for receiving the moving goal from Overall node then generates a global route which consists

of a set of points. Its general principle of work is that: first initializing some entry parameters including setting the node state to NodeState::IDLE state, then configuring global planning algorithm according to configuration file, finally, starting the global planning server and executing the callback function every time the moving goal is received. The specific running process of the callback function is shown in Fig. 6.

Similar to the Overall node, this node is generated by compilation of the global_planner_node.cpp and the only thing its main function needs to is to instantiate its GlobalPlannerNode class. It is important to note that this class not only needs to define the necessary interfaces and input and output data needed for the global path planning, but also the introduction of a specific global path planning subalgorithm. The Global Planner node provides information about the /tf, /map, initial point and goal point needed for global planning while the work of the specific calculating path is completed by these subalgorithms. We take the method of inheritancing class, that is, using the class GlobalPlannerNode to inherit the class of global planning subalgorithm. In this way, when applying different global planning subalgorithms to the virtual experiment platform, only the GlobalPlannerNode class is required to inherit the classes of different subalgorithms and call their functions when necessary.

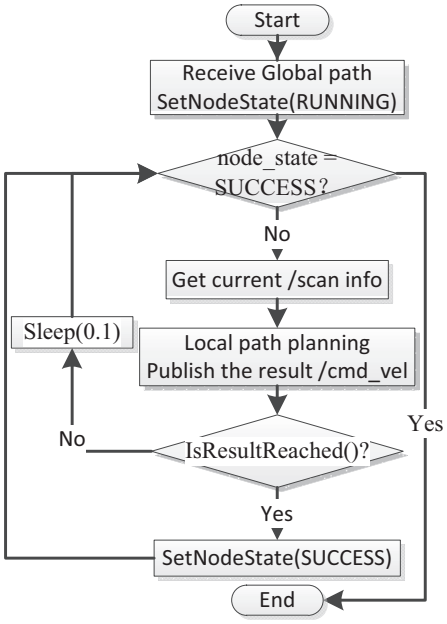


Fig. 7: Running process of the ExcuteCB function of the Local Planner node.

C. Local Planner Node and Localization Node

The Local Planner node works as a local planning server which receives the global route generated by the Global Planner node and combines the /scan, /odom, and /tf data for local planning. Then it publishes the control topic of /cmd-vel which contains the real time line velocity and angular velocity information of the robot. Its workflow is: first initializing some entry parameters including setting the node state to NodeState::IDLE state, then configuring local planning algorithm according to configuration file, finally, starting the local planning server and executing the ExcuteCB function every time the global route is received. The specific running process of the ExcuteCB function is shown in Fig. 7.

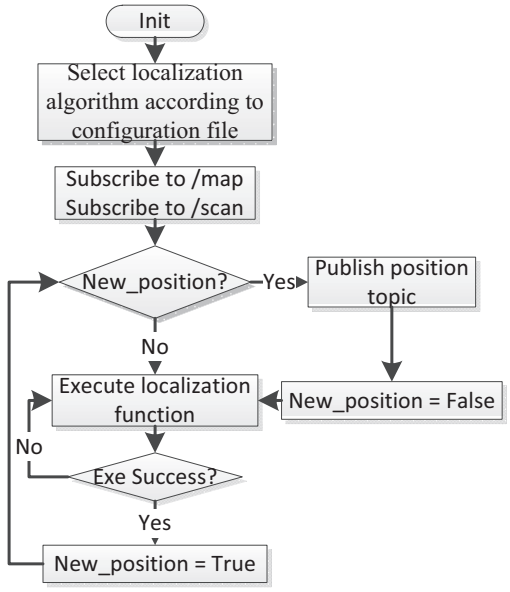


Fig. 8: Running process of the Localization node.

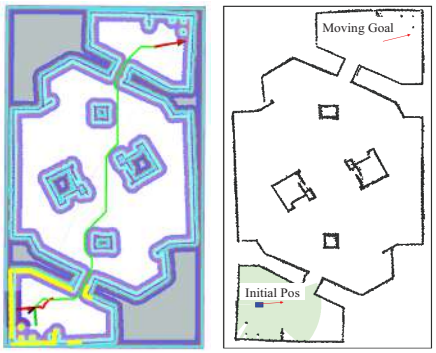
The Localization node works as a localization server which receives /scan data generated by lidar and combines /map, /tf data to calculate real time position information of robot. Its specific running process is shown in Fig. 8.

From the above two flow charts, we can clearly see the running process of Local Planner node and Localization node, and similar to the Global Planner node, these two nodes only need to call the corresponding classes when using different sub algorithms. It is worth mentioning that the actionlib goal of the Localization node only contains information of starting and ending, so once the starting goal is received, the Localization node will constantly check whether there is a new localization result in a loop and will publish it once it appears.

V. Experimental Result

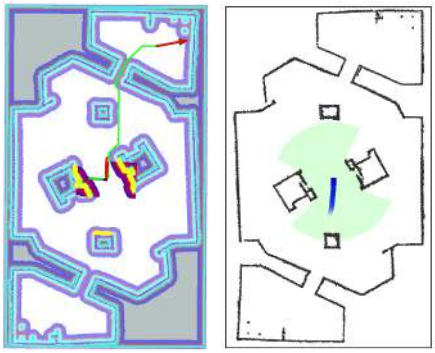
In the above, we introduce the architecture of the virtual experiment platform and the details of the implementation of each node. In order to verify its ability to quickly realize the fusion of each sub algorithm, we choose the corresponding common subalgorithm which is Monte Carlo algorithm, Dynamic Window Approach algorithm and Timed Elastic Band algorithm for localization, global planning and local planning

respectively. After each sub algorithm is integrated into the virtual experiment platform, we set the initial position and goal position in the lower left corner and the upper right corner of the simulation environment respectively, and the experimental results are shown in the following three figures.



(a) (b)

Fig. 9: The global planning and local planning results of the virtual experiment platform.(a)The effect shown in rviz;(b)The corresponding effect shown in stage.



(a) (b)

Fig. 10: The real time localization result of the virtual experiment platform.(a)The effect shown in rviz;(b)The corresponding effect shown in stage.

The long green line in Fig. 9(a) represents the global path planning result for the robot to move from the lower left corner of the simulation environment to the upper right corner while the Fig9(b) shows the location of moving goal in the stage environment. The red line in the lower left corner represents the local path planning result of the time. It can be seen that: the Global Planner node can quickly generate reasonable global path according to the given moving goal, and correspondingly, the Local Planner node can generate feasible local path according to the global path and the real time lidar information. The Fig. 10(a) represents the localization result calculated by the Localization node displayed in rviz during a robot movement, and Fig. 10(b) represents the position of the virtual robot in the stage simulation environment at that time. The comparison of these two figures shows that the Localization node can publish relatively correct localization results of the virtual robot in real time. It can be concluded from the above that the whole system can run steadily and reliably and the effect is basically in line with expectations.

VI. Conclusion

A virtual experimental platform for robot autonomous navigation algorithm system is presented in this paper which can easily integrate different sub algorithms for algorithm testing. This paper introduces its principle framework and necessary implementation details of each part. This experiment results proves that the virtual experimental platform can realize the expected function, and to some extent, it can play an important role in accelerating the research of the robot autonomous navigation algorithm system.

Acknowledgment

The idea of the robot autonomous navigation algorithm sysytem virtual experiment platform proposed in this paper comes from a competition called 2018 DJI RoboMaster AI Challenge held jointly by DJI company and ICRA. In the writing process of this platform, we refer to the open source algorithm framework provided by DJI company for the competition. Here we express our thanks.

References

[1] D. Dolgov, "Autonomous driving in semi-structured environments: Mapping and planning," Int. Conf. on Robotics and Automation (ICRA), 2009.
[2] Bailey T, Nieto J, Guivant J, et al, "Consistency of the EKF-SLAM Algorithm[C]," Ieee/rsj International Conference on Intelligent Robots and Systems. IEEE, 2006:3562-3568.
[3] Hanten R, Buck S, Otte S, et al, "Vector-AMCL: Vector Based Adaptive Monte Carlo Localization for Indoor Maps[M],"Intelligent Autonomous Systems 14. 2017.
[4] Arai H, Xu K, Maung C, et al, "Weighted A* algorithms for unsupervised feature selection with provable bounds on suboptimality[C]," Thirtieth AAAI Conference on Artificial Intelligence. AAAI Press, 2016:4194-4195.
[5] Chen Y B, Luo G C, Mei Y S, et al," UAV path planning using artificial potential field method updated by optimal control theory[J]," International Journal of Systems Science, 2016, 47(6):1407-1420.
[6] Rosmann C, Hoffmann F, Bertram T, "Online Trajectory Planning in ROS Under Kinodynamic Constraints with Timed-Elastic-Bands[M]," Robot Operating System (ROS). Springer International Publishing, 2017.
[7] Quigley M, Gerkey B P, Conley K, et al, "ROS: An open-source Robot Operating System[J]," 2009.
[8] Library W P, "Mobile Robot Programming Toolkit[J]," 2015.
[9] Kartashov D, Kartashov D, "A SLAM research framework for ROS[C]," Central and Eastern European Software Engineering Conference in Russia. ACM, 2016:12.
[10] Santos H B, Teixeira M A S, Oliveira A S D, et al, "Control of Mobile Robots Using ActionLib[J]," 2017.
[11] Chekanov S V, "Next generation input-output data format for HEP using Google's protocol buffers[J]," Computer Science, 2013.