# Transport Problems

Volume 20 Issue 4

# Problemy Transportu

WYDAWNICTWO POLITECHNIKI ŚLĄSKIEJ

GLIWICE 2025

SCIENTIFIC
JOURNAL

# TRANSPORT PROBLEMS

*Volume 20 Issue 4*

# PROBLEMY TRANSPORTU

*Tom 20 Zeszyt 4*

QUARTERLY

WYDAWNICTWO POLITECHNIKI ŚLĄSKIEJ
GLIWICE 2025

# CONTENTS

Page

1. Asenov A., Pencheva V., Geogriev A., Mineva K.: Multifactor analysis of mass transit digitalization in the five largest cities in the republic of Bulgaria ..... 5
2. Charłampowicz J.: The role of customer knowledge in improving maritime container terminal service quality: insights from forwarders..... 15
3. Kubík J.: Statistical assessment of indicators measuring public transport use in cities ..... 25
4. Sicińska K., Zielińska A., Olszewski P., Osińska B.: Safety of e-scooters on polish roads between 2022 and 2024 ..... 37
5. Alturki S.F.: Towards green transportation: an evaluation of strategies for carbon emissions mitigation in cement-based transportation infrastructure..... 51
6. Vaičiūnas G., Steišūnas S.: Study of the impact of passenger train control algorithms on fuel consumption..... 65
7. Palczewska A.: Identification of risks and opportunities related to the road transport of dangerous goods, flammable liquids, and gases ..... 75
8. Czerniak R., Borowski P.F.: Logistical challenges of rail transport concerning suicidal incidents on tracks: operational, psychological, and ecological perspectives ..... 85
9. Bosyi D., Sablin O., Potapchuk I., Usenko A.: Embedded AI for audio-based drone detection in critical railway infrastructure ..... 99
10. Wang X., Wang C., Sładkowski A., Gao K.: Using the lightweight YOLOv8 system for visual detection, logistics package grasping, and autonomous navigation for mobile manipulators ..... 113
11. Mańka A., Pawlikowski P., Wachnik R., Staszek J.: A tool for analyzing the passage of a train with specific technical parameters on a railway line considering various models of movement resistance..... 127
12. Murawski J., Panek A., Franke P., Bolzhelarskyi Y.: Decision-making support for locating logistics facilities in transport networks..... 141
13. Koting S., Shamsul Harumain Y.A., Muhamad M.R., Sinoh S.S., Matsunaga Ch., Morimoto A.: Intergenerational transport habits and children's mobility choices: a study of car-dependent households ..... 155
14. Żelazny R., Kocot D., Kocot M.: Facing the challenges of Industry 5.0 – the case of the automotive sector in Poland ..... 167
15. Chelidze M., Zviadauri V., Natriashvili T., Didebulidze A., Javakhishvili G., Kuntchulia T.: Vibrational transport of materials considering the physical and mechanical characteristics of the “material-working element” system ..... 181
16. Awad H.A., Mhana K.H., Alrawi F., Sierpiński G., Staniek M.: AHP-based construction of a smart sustainable transportation index for the Ramadi districts..... 195
17. Passoli A., Aholou C.: Urban sprawl and accessibility in downtown Lomé: mobility issues and socio-spatial inequalities..... 209
18. Matijošius J., Rimkus A., Gruodis A.: Data-driven prediction of fuel consumption and thermal efficiency in CI engines using HVO/diesel mixtures and neural networks ..... 223

List of reviewers..... 235

TRANSPORT PROBLEMS 2025 Volume 20 Issue 4
PROBLEMY TRANSPORTU DOI: 10.20858/tp.2025.20.4.10

**Keywords:** mobile manipulator; visual grasping; YOLOv8; autonomous navigation; parcel grasping

**Xuelin WANG[1]*, Changlin WANG[2], Aleksander SŁADKOWSKI[3], Kailan GAO[4]**

# USING THE LIGHTWEIGHT YOLOV8 SYSTEM FOR VISUAL DETECTION, LOGISTICS PACKAGE GRASPING, AND AUTONOMOUS NAVIGATION FOR MOBILE MANIPULATORS

**Summary.** In recent years, the global industrial production and manufacturing paradigm from mass production has continued to shift to customized production. Enterprise order processing is also increasingly showing strong timeliness, variety, small batch, and batch characteristics. The market demand for highly flexible robotic automated production lines is also continuing to grow. Mobile manipulator is a new type of robot that integrates two functions of mobile robots and robotic arms, which can plan routes, navigate accurately, avoid obstacles, and identify, sort, grasp, and transport items. It is widely used in logistics and warehousing, factories, indoor exhibition halls, etc., which can save the cost of manpower, improve efficiency, and create differentiated competitiveness. Despite its promising application, it also faces some problems, especially in the cooperative operation of the mobile platform and robotic arm. Understanding how to realize vision-based target intelligent recognition and grasping is still a current research challenge. In this paper, we build a mobile manipulator platform, deploy and verify a YOLOv8n-SCS-CE lightweight detection network proposed in the previous stage (which can detect common logistics parcels), and test the robot's autonomous mobile grasping of parcels and autonomously navigating to the target place. The test demonstrates that it can utilize the improved YOLOv8 to achieve intelligent grasping and autonomous navigation, thereby solving the challenges of intelligent grasping and autonomous transportation for indoor mobile manipulators. This study provides key technologies and methodologies for the intelligent grasping and manipulation capabilities of mobile manipulators.

## 1. INTRODUCTION

A mobile manipulator, a highly flexible robot with multi-scene adaptability, typically consists of a vision sensing module, a multi-axis robotic arm, a mobile platform, and an end-effector. This system enables object recognition, inspection, grasping, and other task-specific functions. Its core technological architecture integrates artificial intelligence, mobile manipulation, sensor fusion, and localization/navigation technologies [1-3].

The research and development of mobile manipulators began abroad earlier, resulting in relatively mature technologies. Notable international examples include the PR2 robot developed by Willow

[1] Institute of Automation, Qilu University of Technology (Shandong Academy of Sciences); Keyuan Road 19, Lixia, Jinan 250014, China; e-mail: wangxuel@sdas.org; orcid.org/0000-0002-4934-5494
[2] Institute of Automation, Qilu University of Technology (Shandong Academy of Sciences); Keyuan Road 19, Lixia, Jinan 250014, China; e-mail: 3089263363@qq.com; orcid.org/0009-0005-0719-1676
[3] Silesian University of Technology; Krasiński 8, 40-019 Katowice, Poland; e-mail: Aleksander.Sladkowski@polsl.pl; orcid.org/0000-0002-1041-4309
[4] Institute of Automation, Qilu University of Technology (Shandong Academy of Sciences); Keyuan Road 19, Lixia, Jinan 250014, China; e-mail: 1837948379@qq.com; orcid.org/0009-0009-0072-757X
* Corresponding author. E-mail: wangxuel@sdas.org

Garage (USA); HERB, developed by Intel Pittsburgh Lab and Carnegie Mellon University; the Fetch robot by Fetch Robotics (USA); Stretch, Boston Dynamics' mobile warehouse automation solution; and KUKA's (Germany) KMR iiwa robot for mission-critical industrial applications. Although China started later in this field than other countries, progress has been rapid. Key domestic brands include HSCR5, China's first intelligent hybrid collaborative robot, launched by Shenyang Siasun Co., Ltd; Daystar robot, independently developed by Lenovo Corporation; STAR robot, developed by Shenzhen Han's Robot Co., Ltd; and Scara hybrid robot by RobotPhoenix LLC. These mobile manipulators can be applied across various fields, including logistics and warehousing, electronics manufacturing, food processing, and pharmaceutical production [4-6].

Composite mobile robots have a broad application prospect, but they also face some problems. Substantial research has been carried out internationally on fixed robotic arm visual grasping technology, YOLO detection network algorithms, and autonomous navigation of mobile robots [7-11]. However, these studies have primarily focused on aspects such as the lightweight design of YOLO algorithms, the visual grasping for robotic arms, autonomous navigation, and collaborative control. There remains insufficient research on integrated YOLO-based visual grasping and navigation control for mobile manipulators as a complete system. In particular, several technical challenges and bottlenecks persist in areas like lightweight YOLO recognition algorithms, eye-hand coordinated grasping, and autonomous navigation control [12, 13]. Achieving vision-based efficient target recognition, autonomous mobile grasping, and intelligent navigation for mobile manipulators remain a current research hotspot.

To address these challenges, this study develops a mobile manipulator platform that combines advanced perception, manipulation, and navigation capabilities. Using the detection and grasping of common logistics box-shaped parcels as a case study, the platform integrates three key technological components: our previously developed lightweight YOLOv8n-SCS-CE detection algorithm [14] for robust object recognition; a vision-guided robotic arm grasping system for precise manipulation; and LiDAR-based autonomous navigation technology for environment-aware mobility. The synergistic combination of these technologies provides a practical solution for logistics automation applications requiring both precise manipulation and mobile transportation capabilities.

## 2. CONSTRUCTION OF A MOBILE MANIPULATOR EXPERIMENTAL PLATFORM

### 2.1. Components of the Test Platform

The mobile manipulator platform depicted in Fig. 1 has been developed to meet the needs of indoor, warehousing and logistics fields, overcome the limitations of traditional fixed-base robots and mobile AGV platforms, and improve the flexibility, safety and reliability of a variety of processes. The experimental platform comprises the following core components: a circular indoor intelligent mobile base (SHANSU Intelligence Co., Ltd.), an FR5 collaborative robotic manipulator (FAIR Innovation Robot Systems, Suzhou), a two-finger parallel gripper (DaHuan Co., Ltd.), an Intel RealSense D435i RGB-D camera, and a 2D LiDAR sensor. The platform measures 700×545×500 mm (L×W×H), and it integrates a six-axis robotic arm with a 922-mm workspace, complemented by an autonomously navigating chassis. Its ROS-based control system offers extensive sensor interfaces and API options to streamline secondary development and system integration.

（1）Collaborative Robot Arm

The FAIR FR5 collaborative robot from the FR series was selected for this application. This system features an integrated robot button box and compact control box, offering an extensive working envelope within a minimal footprint. Its compatibility with diverse end-effectors and sensor packages ensures optimal adaptability to meet all project specifications. The FR5 robotic arm utilizes the frcobot_ros software package to create a TCP client-server communication channel with the robot's state feedback system. This implementation facilitates the continuous real-time data acquisition of the robot's operational status parameters. The FR5 features a 5-kg payload capacity, a total weight of 20.6 kg, and an exceptional ±0.03-mm repeatability.

（2）Autonomous Mobile Platform

The mobile robot chassis from SHANSU Intelligence was selected; this system integrates multiple sensors to facilitate environment perception and autonomous navigation. The system incorporates the WLR-716-Mini, a high-precision 2D LiDAR, primarily employed for environmental mapping, obstacle detection, and simultaneous localization and mapping (SLAM) applications. The IMU Brick 2.0 is a high-performance inertial measurement unit (IMU) that integrates a three-axis accelerometer, magnetometer (e-compass), and gyroscope, capable of measuring motion in nine degrees of freedom. The mobile chassis adopts two-wheel differential speed, built-in odometer, LIDAR and IMU, with high-precision environment sensing, motion state monitoring, and autonomous navigation capability for the mobile manipulator.

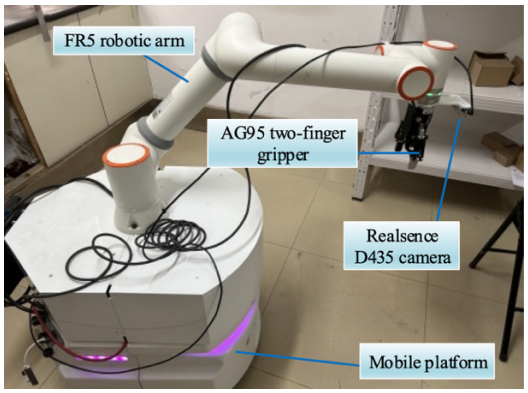


Fig. 1. Composition of mobile manipulator

（3）Intel RealSense D435 Camera

The Intel RealSense D435 binocular stereo camera was chosen to process color images, recognize, and (stereo vision) compute depth information, and the associated driver ROS package was provided. As shown in Fig. 2, its forward sensing array is distributed in a horizontal line configuration as follows: the first and third from the left are infrared sensors (IR stereo camera), the second from the left is an infrared projector, and the fourth from the left is a color camera (i.e., a color sensor).


Fig. 2. Intel RealSense D435 camera

（4）End Effector

The DaHuan AG95 articulated adaptive electric two-finger gripper was selected for its robotic arm compatibility. Its compact structural design enables adaptive grasping of workpieces with varying geometries, ensuring stable manipulation of diverse objects. AG95 has the features of plug-and-play, drive, and control as a high-precision adjustable gripping force, double gripper control, interchangeable fingertips, fast installation, etc., and was combined with the robotic arm FR5 to enhance the flexibility of the gripping system.

## 2.2. Overall Design of Control System

The experimental platform operates on an Ubuntu 20.04 LTS environment with ROS Noetic framework integration. System control is centralized through an industrial-grade mini-PC, which orchestrates all hardware modules via unified coordination and real-time scheduling protocols. The primary development languages are C++ and Python. The platform system architecture encapsulates

various functional units, including the robotic arm, depth camera, LiDAR, Gmapping module, and end effector, as independent ROS nodes seamlessly integrated into the robot's Ubuntu operating environment. These nodes communicate via ROS topics to receive information, which is then converted into hardware commands to ultimately drive the robotic arm in executing target tasks.

The system employs multiple communication protocols to enable efficient inter-module coordination. The chassis differential motor communicates with the main control system through CAN bus to realize the motion control and state feedback of the chassis. the robotic arm adopts Ethernet (TCP/IP) communication to support high-frequency control and precise positioning. The gripper interacts with the main control system through the serial port for grasping and releasing. Environmental perception is achieved through an Intel RealSense D435 RGB-D camera, interfaced via USB 3.0 Gen 1, providing registered color-depth image pairs and dense 3D point clouds. These hardware modules are encapsulated as independent nodes in the ROS noetic system, and each node exchanges data and transmits control instructions through topics, etc. The master control system uniformly manages the collaborative scheduling of all the hardware to realize the intelligent operation of the robot. The overall control system is shown in Fig. 3.

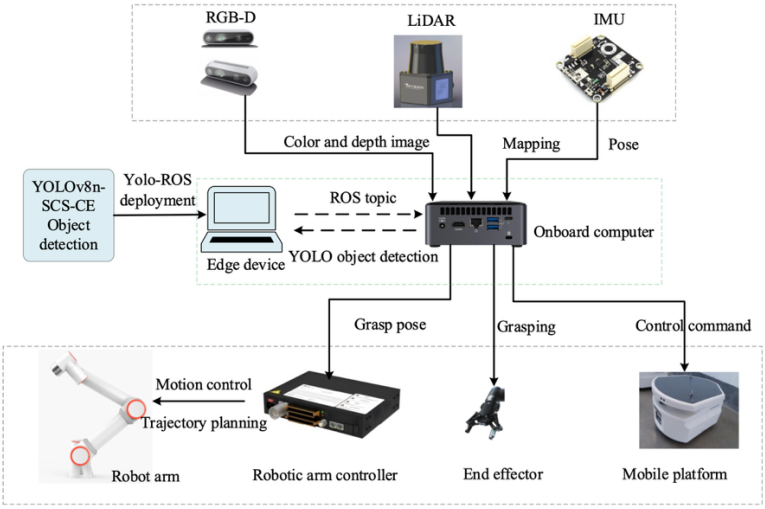


Fig. 3. Composition of the overall control system

## 3. KEY TECHNOLOGIES FOR THE MOBILE GRASPING OF MOBILE MANIPULATORS

The mobile grasping capability of a mobile manipulator combines the precision grasping of a robotic arm with the mobility provided by a mobile chassis. This integration primarily encompasses several key aspects: a YOLO lightweight target detection network, visual calibration, and autonomous navigation (SLAM).

### 3.1. YOLOv8n-SCS-CE Lightweight Detection Algorithm

YOLOv8, when used in warehouse logistics field detection, requires a large target detection model and computational complexity, thus leading to a large amount of computation, resulting in a slow operation speed, which makes it difficult to meet the demand for deployment in embedded or mobile devices. The authors of this paper have previously proposed a YOLOv8n-SCS-CE target detection algorithm, which improves the model feature extraction capability while maintaining a low computational complexity. Its network structure is shown in Fig. 4, with the changes mainly concentrated in the backbone part and neck parts, which are mainly used for detecting common logistics packages [15].

In the backbone network part, conventional convolution is first used to capture the initial spatial features and downsample the feature maps, and then the proposed SCS network structure is introduced to optimize the data flow method and improve the computational efficiency through the strategies of channel equalization, optimization of channel shuffle, reducing the amount of 1×1 convolutional

computation, and reducing the cost of memory access. Through these optimizations, the SCS structure effectively reduces the computational burden of the YOLOv8 backbone network. In the neck part, the C2f module of the original YOLOv8 is replaced by the C2f_ECA module, which further improves the feature extraction capability while controlling computational complexity. The main optimization points of C2f_ECA include the incorporation of efficient channel attention, which avoids additional computational burdens while improving the feature selection capability through local cross-channel information interactions compared to the traditional CBAM or SE attention mechanisms.

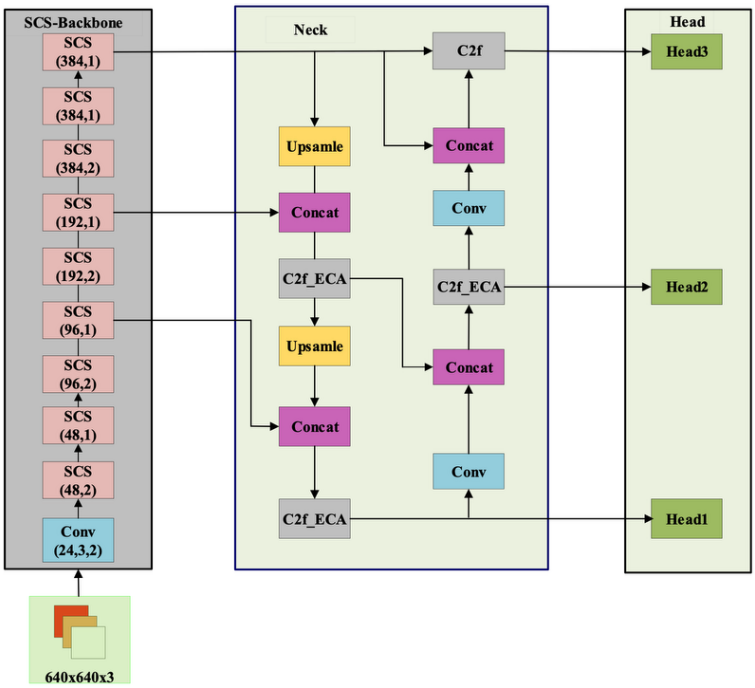


Fig. 4. YOLOv8n-SCS-CE lightweight detection network

### 3.2. Robotic Eye-on-Hand Calibration

#### 3.2.1. Camera Intrinsic Calibration

First, the depth camera is fixed on the end rotary joint of the robotic arm to ensure that the coordinate system of the camera to acquire data can be consistent with the default coordinate system direction of the actuator at the end of the robotic arm to facilitate the coordinate transformation between the camera and the robotic arm. Then, the camera is activated to acquire the image-aligned RGB scene and depth map.

This experiment uses MATLAB's own toolbox (Camera Calibrator) for calibration, the advantages of which are the accuracy of the calibration and the simplicity of the operation process in practice. We employed a 12×8 checkerboard grid (comprising 11×7 internal corner points) as the calibration target, with each square measuring 25 mm × 25 mm. The board was rotated while keeping the camera stationary to capture a sufficient number of images. A total of 80 groups of images were collected for the same calibration plate in different attitudes, and the camera was calibrated by importing these images into MATLAB and extracting the corner points of the images using Camera Calibrator. During the calibration process, the system calculates the average calibration error for the left and right images, as well as the simulated attitude between the camera and the calibration target. Images with large errors are then removed, leaving 40 valid image pairs. This information can be visualized graphically, as shown in Figs. 5 and 6.

The end of calibration yields the camera intrinsic matrix [607.9263 0 0; 0 607.4232 0; 315.0026 251.6894 1.0000], the translation vector of principal point coordinates [-88.0890 -184.4831 608.3623], and the rotation matrix [0.9988 -0.0406 -0.0290; 0.0405 0.9992 -0.0022; 0.0291 0.0011 0.9996].

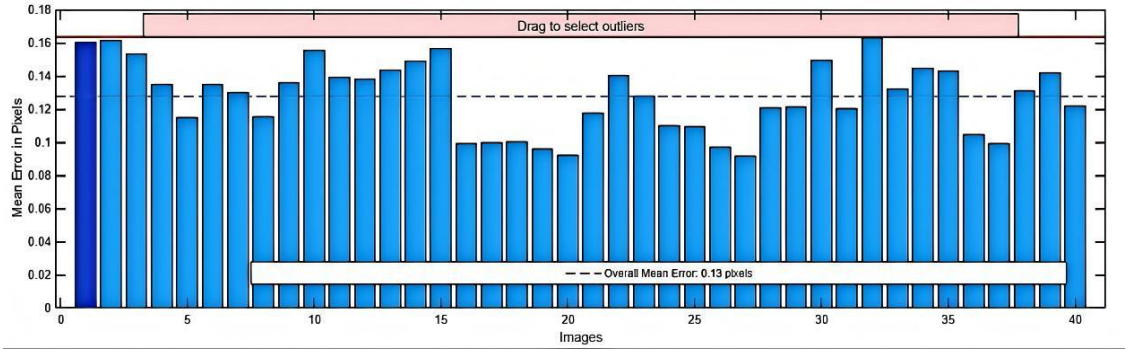


Fig. 5. Reprojection errors

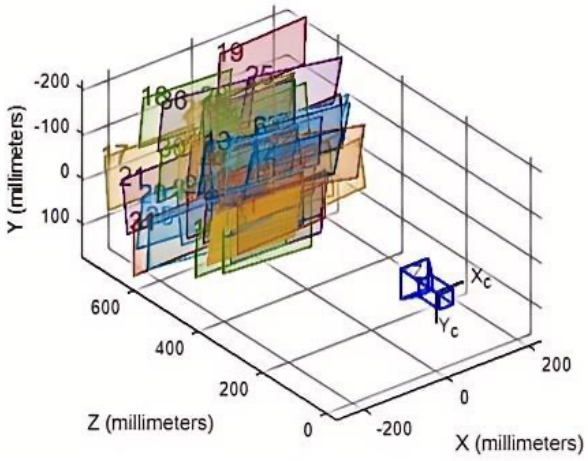


Fig. 6. Pose and spatial position of calibration board

### 3.2.2. Eye-on-Hand Calibration

Hand-eye calibration experiments are used in robotics and computer vision to determine the transformation between a robot's end-effector and a camera mounted on it. This calibration is crucial for tasks like visual servoing and object manipulation.

Let the transformation from the camera coordinate system to the joint coordinate system at the end of the robotic arm be $T_{cam}^{arm}$, the transformation from the joint end coordinate system to the base coordinate system be $T_{arm}^{base}$, and the position of the object in the camera coordinate system be $Pose_{cam}$. Then, the position of the object in the base coordinate system can be obtained as

$$Pose_{base} = T_{arm}^{base} * T_{cam}^{arm} * Pose_{cam} \quad . \tag{1}$$

During calibration, to keep the object's pose unchanged, the robot arm's base is not moved, that $Pose_{base}$ remains unchanged. In order to ensure the calibration effect, *n* groups of poses are generally selected for calibration, from which *n* position and orientation transformation relations can be derived

$$Pose_{base_n} = T_{arm_n}^{base} * T_{cam}^{arm} * Pose_{cam_n} \qquad (n = 1,2, \dots, n) \tag{2}$$

where $T_{arm}^{base}$ can be acquired using the robot's teaching pendant, $Pose_{cam}$ can be obtained by the camera calibration, and $T_{cam}^{arm}$ is an unknown transformation to be solved, but it remains constant across all *n* positions during hand-eye calibration. Based on this condition, it is possible to obtain Eq. (3):

$$T_{arm_1}^{base} * T_{cam}^{arm} * Pose_{cam_1} = \dots = T_{arm_n}^{base} * T_{cam}^{arm} * Pose_{cam_n} \tag{3}$$

Any set of equations can be transformed to obtain the equation that

$$T_{arm_2}^{base\ -1} * T_{arm_1}^{base} * T_{cam}^{arm} = \dots = T_{cam}^{arm} * Pose_{cam_2} * Pose_{cam_1}{}^{-1} \tag{4}$$

Let $A = T_{arm_2}^{base\ -1} * T_{arm_1}^{base}$, $B = Pose_{cam_2} * Pose_{cam_1}{}^{-1}$, $X = T_{cam}^{arm}$. Then, we get *AX=XB*, where both *A* and *B* are known, solving for the unknown transformation *X*. Hand-eye calibration is performed using MATLAB's built-in Camera Calibrator toolbox. To ensure calibration accuracy, the calibration plate remained stationary within the camera's field of view throughout the process. The

robotic arm is systematically repositioned to capture 15 distinct image sets, optimizing pose coverage for robust parameter estimation. This multi-image acquisition strategy enhances calibration precision by mitigating measurement noise and improving spatial constraint resolution. The experimental setup is illustrated in Fig. 7.

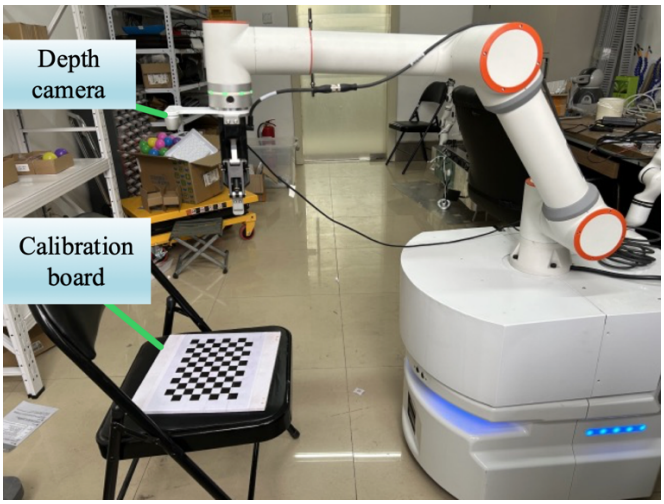


Fig. 7. Eye-on-hand calibration

Here are several professionally refined versions of the hand-eye calibration results presentation: rotation matrix $R$=[0.9774 0.2073 0.0422; -0.1993 0.9691 -0.1451; -0.0710 0.1334 0.9885]; translation vector $t$=[-72.8437 -5.9962 586.1100] (units: *mm*).

After the arm hand-eye calibration, according to the obtained rotation matrix and translation vector, which can be deduced from the coordinate transformation matrix, the main program publishes coordinate system conversion, converting the object to the camera coordinates, and then to the object to the robot arm coordinates, enabling the robot arm to grasp the object.

### 3.3. Autonomous Navigation via Gmapping SLAM

Gmapping SLAM map construction for a mobile chassis requires LiDAR, odometer, IMU, etc. Gmapping is a SLAM algorithm based on 2D LiDAR using the Rao-Blackwellized particle filters (RBPF) algorithm to complete 2D raster map construction. Its core problem is that it struggles with simultaneous robot position estimation and environment map construction. Accurate position estimation relies on high-quality maps, while high-quality map construction relies on accurate position estimation, making SLAM a complex computational problem. Gmapping shows the advantages of high computational efficiency, stable accuracy, and a lower requirement for LIDAR frequency in small-scale environments.

Fig. 8 shows the autonomous navigation experimental environment, which uses a virtual machine to remotely log in to the host computer of the robot; open the LiDAR node, mobile chassis drive node, and odometer node; run the Gmapping map building node; and run the Rviz visual interface, which can be viewed in real time during the map building process. The user can use the handle to move the robot slowly to complete the construction of the map of the entire environment and then save the map. When autonomous navigation is required, the navigation node needs to be turned on, the built map file needs to be loaded, and Rviz needs to be opened on the local machine. The 2D Pose Estimate is used to adjust the initial pose to the initial position of the mobile manipulator in the field at this time, with the arrow pointing in the direction of the robot's x direction. Ensure that the laser scanning obstacle data and the map obstacles basically coincide, use 2D Nav Goal to set the target point and target posture for robot navigation, and the robot will automatically plan a feasible path to move to the target point autonomously.

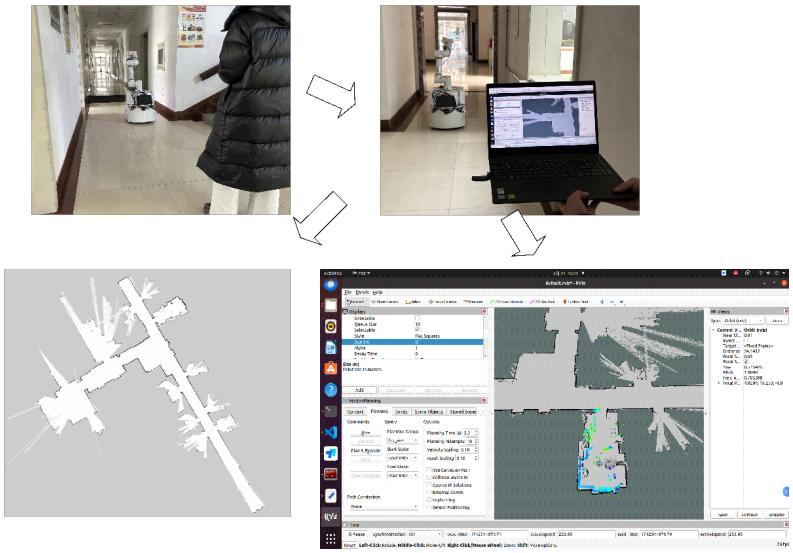

Fig. 8. Mapping and map loading tests

## 4. MOBILE MANIPULATOR PROGRAMMING FRAMEWORK DESIGN

The mobile manipulator was developed on the UBUNTU 20.04 operating system with ROS Noetic, a widely used open-source framework that provides integrated communication mechanisms, tool packages, and other essential functionalities for robotics development. The system implementation employs C++ and Python as primary programming languages, with the mobile chassis and FR5 robotic arm control functions developed using the ROS MoveIt framework. As shown in Fig. 9, all SRC source files within the catkin package are fully accessible and modifiable through VS Code.

The architecture incorporates multiple customized ROS packages (Fig. 10) organized into 10 core functional modules: (1) battery management system' (2) gripper modeling and control subsystem' (3) robotic arm driver for hardware communication and operations' (4) mobile chassis 3D modeling and navigation control package' (5) pointcloud_grasp application layer (containing move_group.cpp and detect.py)' (6) RGB-D camera driver with integrated calibration parameters' (7) ROS-control hardware interface simulator and templates' (8) LiDAR driver package and SDK' (9) custom communication packages (topics/messages/actions)' and (10) IMU driver package and SDK. This modular design ensured robust system integration while maintaining flexibility for future enhancements.

### 4.1. Autonomous Mobile Grasping Program Design for the Mobile Manipulator

The mobile grasping test of a mobile manipulator involves the cooperative operation of multiple functional nodes to realize the detection, path planning, and grasping operation of target objects. This subsection mainly introduces the program development of the move_group node. The move_group node relies on the MoveIt motion planning framework, which can realize the functions of path planning, path execution, inverse kinematics and Cartesian path. Its main program, move_group.cpp, needs to introduce header files and initialize robotic arm control parameters. To implement mobile grasping capabilities, the user defines a C++ grasp_demo class for encapsulating the complete grasping pipeline. This object-oriented design provides a clean interface for programmatic control of the grasping process. In C++, the grasp_demo class contains some member functions and variables, of which the class constructor is as follows:

grasp_demo::grasp_demo (ros: NodeHandle &nh):move_group_(PLANNING_GROUP).

In the grasp_demo class, where the armGrasp() function is used to execute the complete grasp process (as shown in Fig. 10), the grasp_demo class contains the following core variables and functions:

(1) ros: init(argc, argv, arm_grasp_demo): initialization function, where argc indicates the number of parameters, argv indicates the parameter list, and arm_grasp_demo indicates the ROS node name;

(2) ros: NodeHandle nh: create ROS handle, nh is responsible for managing ROS network communication, such as topic subscription and publication;

(3) ros: AsyncSpinner(4): create an asynchronous callback manager and allocate 4 threads for processing ROS callback functions to improve system responsiveness;

(4) spinner.start(): start the callback processing to ensure uninterrupted message delivery;

(5) gototarget(nav_goal): set the robot's autonomous navigation goal to ensure that it moves to the grasping position;

(6) armGrasp(): responsible for executing target detection, robot arm motion planning, grasping and placing operations.

The main logic of the main program mainly includes the steps of ROS node initialization, asynchronous callback processing, autonomous navigation, and grasping execution, as shown in Fig. 11.

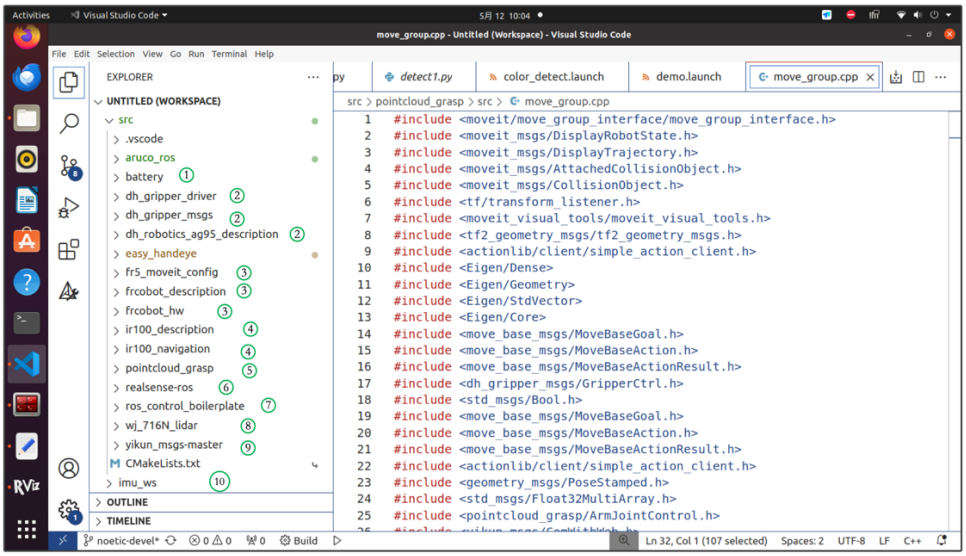


Fig. 9. ROS source (SRC) files in VS Code

```
class grasp_demo
{ private:
  public:
  grasp_demo(ros::NodeHandle &nh);
  ~grasp_demo();
  void graspObject();
  MovebaseGoalClient *move_client_;
  void GripperContrl(ros::Publisher open_gripper, float position_);
  void ArmDoControl(ros::Publisher arm_do_pub, int do_num, bool value);
  void NavResultCallback(const move_base_msgs::
        MoveBaseActionResult::ConstPtr& nav_result);
  void TagetCallback(const geometry_msgs::PoseStamped::ConstPtr& target_msg);
  bool goToTarget(move_base_msgs::MoveBaseGoal goal);
  void timerCallback(const ros::TimerEvent &event);
  tf::TransformListener listener_;
  std::string nav_text_, target_link, target_link1;
  moveit::planning_interface::MoveGroupInterface move_group_;
  const robot_state::JointModelGroup* joint_model_group;
  ros::NodeHandle private_nh_;
  ros::Subscriber nav_res_sub_, target_sub_;
  ros::Publisher gripper_pub_, arm_pose_pub_, arm_joint_pub_, arm_do_pub_;
  ros::Publisher detect_pub_;
  ros::Timer timer_;
  ros::ServiceServer arm_joint_control_srv_, arm_grasp_srv_;
  geometry_msgs::PoseStamped current_pose_;
  std_msgs::Float32MultiArray current_joint_;
  std::vector<double> init_position_; ready_position_; put_position1_; put_position2_;
  void armGrasp( ros::Publisher detect_pub,
      moveit::planning_interface::MoveGroupInterface *move_group,
      std::vector<double> init_position, ready_position, put_position1, put_position2,
      std::string arm_name, int max_object_num, int timeout);
      bool ArmJointcontrolSrv(pointcloud_grasp::ArmJointControl::Request &req,
          pointcloud_grasp::ArmJointControl::Response &res);
      bool ArmGraspSrv(yikun_msgs::ComWithWeb::Request &req,
          yikun_msgs::ComWithWeb::Response &res);
      moveit::planning_interface::MoveGroupInterface *get_move_group(); };
```

Fig. 10. Design of a grasping class in C++

```
int main(int argc, char** argv)
{
  ros::init(argc, argv, "arm_grasp_demo");
  ros::NodeHandle nh;
  ros::AsyncSpinner spinner(4);
  spinner.start();
  grasp_demo grasp(nh);
  move_base_msgs::MoveBaseGoal nav_goal;
  nav_goal.target_pose.header.stamp = ros::Time::now();
  nav_goal.target_pose.header.frame_id = "map";
  nav_goal.target_pose.pose.position.x = 2.0;
  nav_goal.target_pose.pose.position.y = 1.20;
  nav_goal.target_pose.pose.position.z = 0.0;
  nav_goal.target_pose.pose.orientation.x = 0.0;
  nav_goal.target_pose.pose.orientation.y = 0.0;
  nav_goal.target_pose.pose.orientation.z = -0.069;
  nav_goal.target_pose.pose.orientation.w = 0.997;
  grasp.goToTarget(nav_goal);
  nav_goal.target_pose.pose.position.x = 2.00;
  nav_goal.target_pose.pose.position.y = 1.20;
  nav_goal.target_pose.pose.position.z = 0.0;
  nav_goal.target_pose.pose.orientation.x = 0.0;
  nav_goal.target_pose.pose.orientation.y = 0.0;
  nav_goal.target_pose.pose.orientation.z = -0.069;
  nav_goal.target_pose.pose.orientation.w = 0.997;
  grasp.goToTarget(nav_goal);
  grasp.armGrasp(grasp.detect_pub_,
    grasp.get_move_group(),
    "object_link", "object_link1", grasp.init_position_,
    grasp.ready_position_, grasp.put_position1_,
     grasp.put_position2_, "RM", 1, 50);
  ros::waitForShutdown();
  return 0;
  }
```

Fig. 11. Main control architecture for move_group.cpp

### 4.2. Design of Lightweight YOLOv8 Network Target Detection System

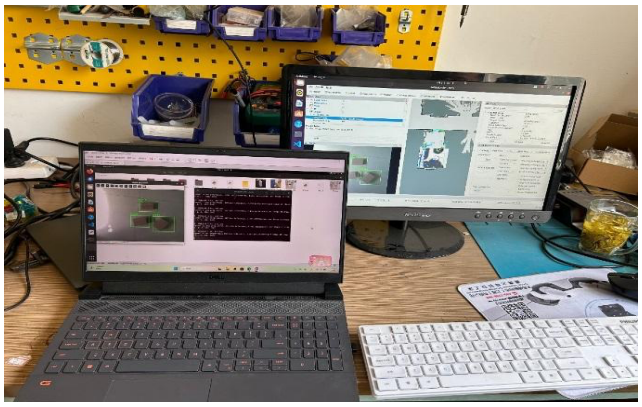

Fig. 12. Improved lightweight YOLOv8 model for parcel detection

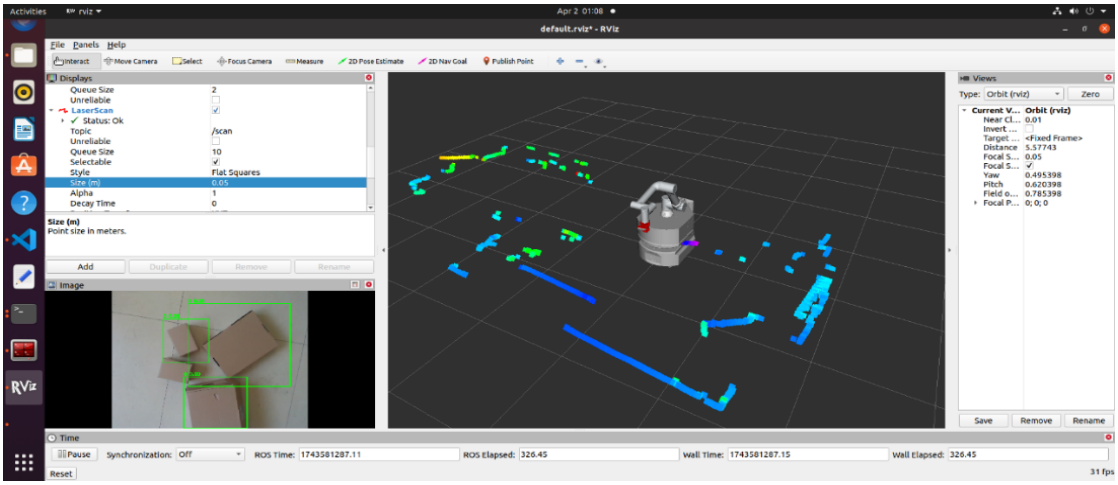


Fig. 13. Robot system subscribes to YOLO topic data

(1) Install a VMware virtual machine on a laptop, allocate 8G of memory to the virtual machine, and set 40G of disk capacity. To install Ubuntu20.04, ROS noetic version, the deployment of YOLO into the need to install Anaconda, create virtual environments, install UItralytics and PyTorch libraries. Next, install ROS-related dependencies, such as the dependency libraries installed by rospkg, etc., to

ensure that the trained weights can be deployed and used in the ROS environment. For the robot's main controller, a laptop serves as an edge device to process real-time YOLO-based image detection algorithms.

(2) Store the pre-trained weights file into the ROS workspace and name it best.pt. Write a ROS node to call the YOLOv8 lightweight model for inference. Publish image messages from the depth camera to ROS for topic communication. First, subscribe to the /camera/image_raw topic to get real-time images. After processing, the /yolo_detections topic is published to send the detected target coordinates to the robot control system, which subscribes to the /yolo_detections topic to parse the detected target position and plan the grasping.

(3) Open the demo_hardware.launch file in the function package fr5_moveit_config so that the mobile manipulator can be launched, and under the workspace launch the rs_rgbd. launch camera node of the realsense2_camera function package; the dh_gripper_driver function package's dh_gripper. launch node; the pointcloud_grasp package's move_group_demo node and color_detect.launch node. As shown in Figs. 12 and 13, after starting the corresponding nodes and loading the detection model, the remote VM successfully runs the YOLOv8n-SCS-CE lightweight detection algorithm after two-way topic communication, and the master and slave detect the target parcel in real time.

## 5. COMPREHENSIVE EXPERIMENT OF MOBILE GRASPING AND AUTONOMOUS NAVIGATION

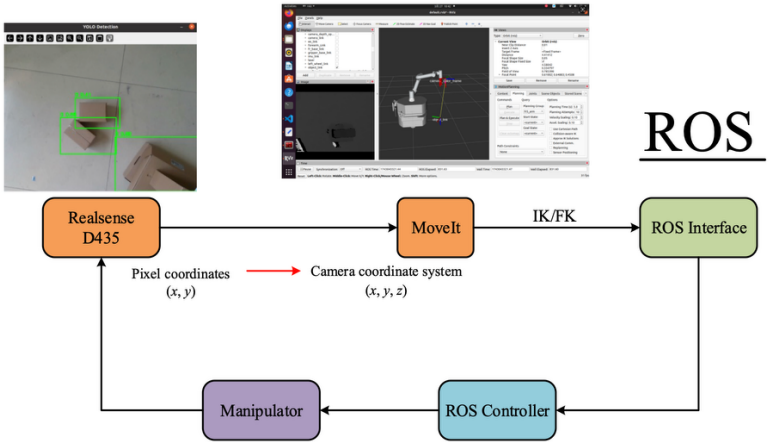


Fig. 14. Visual grasping pipeline for target objects

The target parcel recognition and grasping test procedure (Fig. 14) operates as follows. First, the system acquires aligned RGB and depth images via the depth camera, then publishes the visual calibration parameters through a TF-generated .yaml file. It subsequently calculates the target object's centroid coordinates derived from RGB, depth value, and rotational orientation. These parameters are processed by the programmable move_group interface, which leverages MoveIt for kinematic motion planning. Finally, the planned trajectory is executed via ROS control, guiding the robotic arm's end-effector to the designated grasp pose.

### 5.1. Mobile Manipulator Grasp Tests

The purpose of the tests was to evaluate the mobile grasping performance of the mobile manipulator. The remote virtual machine subscribed to the camera image topic published by the host, while simultaneously activating the lightweight YOLOv8 detection node. After image processing, the robot host received the target detection results output by the lightweight YOLOv8 node on the virtual machine via the topic. For each detected target package, the center coordinates ($x$, $y$) of the bounding box were extracted, and the rotation angle of the gripper was determined based on the principal axis direction of the box-shaped package in the image. The pixel coordinates of the target's center point were

read to obtain the corresponding depth value z in the depth map; in this way, the complete spatial position ($x$, $y$, $z$) of the target needed to construct the grasping information was acquired. In the move_group program, the target points for mobile grasping—including the initial pose, recognition pose, grasping pose, placement pose, and final pose—were predefined. During the mobile grasping process, the robot automatically planned its path and moved according to the preset navigation points.

Small box-shaped packages were selected to validate the mobile manipulator's grasping performance. The image resolution during the grasping process was set to an image resolution of 640×480. The gripper's control function was configured with a position value of 950, a grip force of 100, and a speed set to the maximum range of 100 in the registers.

The robot's camera node was activated to capture real-time images, and a pre-trained weight model was used for object detection. The detection results, combined with the 3D coordinates obtained from depth information, were transmitted to the robot control system via the tf_package. After completing the grasping task, the robot moved to the designated placement location and then returned to its initial position. In this study, the target grasping process was simplified to a 2D planar grasping problem. Throughout the experiment, the camera remained within the optimal viewing range of the box-shaped package, and the gripper performed the grasping operation vertically from above. The experimental process is illustrated in Fig. 15.

Fig. 15. Vision-based mobile manipulation for parcel grasping

Due to the lack of GPU acceleration support on the virtual machine, the system exhibited suboptimal real-time performance, achieving a grasping success rate of 66%. Subsequent studies will employ higher-computing-capacity devices for validation and analysis.

### 5.2. Autonomous Navigation Testing Using Gmapping SLAM

The mobile robot utilized a LiDAR, odometry, and IMU sensors, employing the particle filter-based SLAM algorithm to construct a 2D grid map from collected laser scans and pose relationship data. In ROS, move_base served as the core node for autonomous navigation, integrating both global and local path planning to enable self-guided robot movement. During autonomous navigation, the global costmap facilitated global path planning by computing an optimal route from the starting point to the target within the known map. It avoided static obstacles while minimizing travel distance. The local costmap handled local path planning by dynamically detecting real-time environmental changes (e.g., moving obstacles). It continuously adjusted the trajectory during execution to ensure obstacle avoidance and smooth navigation. Upon receiving a navigation target, move_base first invoked the global path planner (Dijkstra's algorithm) to generate an optimal route on the global costmap. Subsequently, the local path planner employed the dynamic window approach (DWA) algorithm to dynamically adjust the robot's trajectory in response to real-time environmental changes.

As illustrated in Fig. 16, the autonomous navigation experiment proceeded as follows: The robotic arm visually grasped the target package. The mobile platform followed predefined navigation waypoints, automatically generating global and local paths. move_base computed feasible linear and angular velocities, transmitting them via the ROS topic /cmd_vel to the low-level controller, driving the robot along the planned trajectory. The red arrow in Fig. 16 indicates the robot's initial heading direction. The trajectory of the red arrow's variation reflects real-time adjustments in movement and orientation.

The experimental results demonstrate that when navigation waypoints comply with the costmap constraints, the system achieves autonomous transport by placing the grasped target into the designated collection bin. This experiment demonstrates the mobile robot's autonomous transport capabilities using the Gmapping SLAM algorithm. The system integrates data from LiDAR, odometry, and IMU sensors to construct and update a 2D occupancy grid map in real-time while simultaneously localizing the robot within this map. Mobility tests demonstrate a 96% success rate in autonomous transportation, with navigation and localization accuracy maintained at ±3 cm.

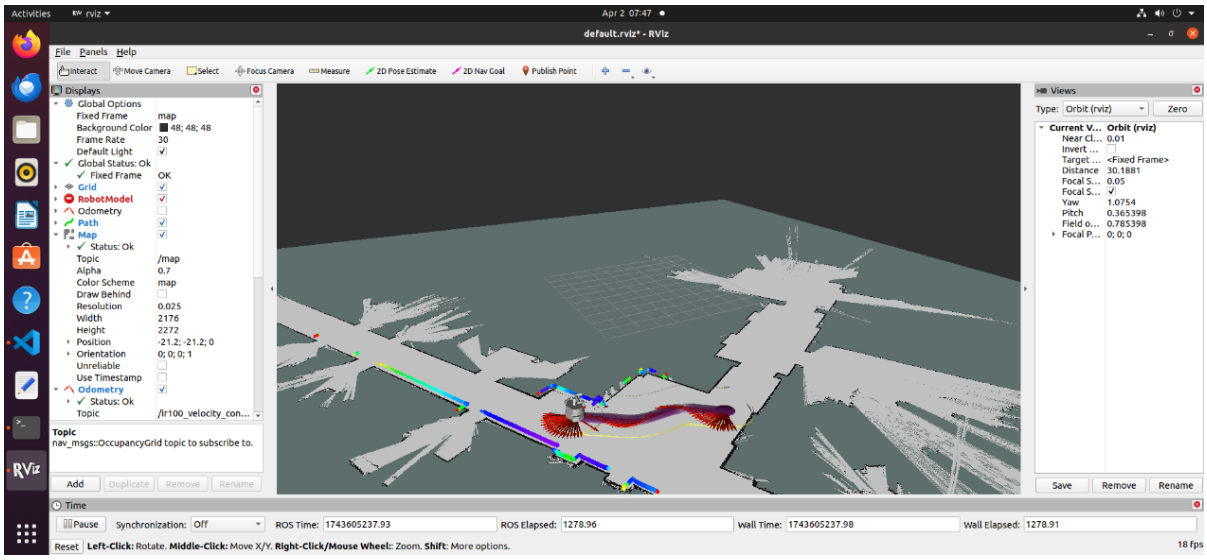


Fig. 16. Autonomous navigation experiment with Gmapping SLAM

## 6. CONCLUSIONS

This study established an integrated robotic test platform to investigate core technologies in mobile grasping and autonomous navigation. The principal findings are as follows:

(1) The integrated robotic system was developed using Ubuntu 20.04 and the open-source ROS Noetic framework. The platform comprises an intelligent differential wheel mobile platform, an FR5 series 6-DOF collaborative robotic arm, a two-finger gripper, an RGB-D depth camera, etc. The robotic arm is located on the mobile robot, and it can be flexibly moved along with the robot to realize the visual grasping, navigation and transportation of specific objects.

(2) We addressed the computational limitations of conventional YOLO algorithms for mobile deployment by implementing our previously developed lightweight YOLOv8n-SCS-CE network on an edge device. The system achieved efficient detection of common logistics box-shaped parcels through bidirectional ROS topic communication between the host machine and virtual machine. This distributed architecture implements YOLOv8-based target detection on the remote virtual machine while reserving visual grasping and motion control functions for the host machine, thereby significantly reducing the host's computational resource consumption.

(3) We developed an autonomous navigation system based on Gmapping SLAM that enables the mobile manipulator to perform lightweight YOLOv8 visual grasping and autonomous transportation tasks. The system achieved a success rate of approximately 66% in object grasping tasks, a 96% success rate in indoor autonomous navigation, and a positioning accuracy of ±3 cm. This integrated solution

effectively addresses three key challenges in indoor parcel handling: intelligent detection, YOLOv8 grasping, and autonomous transportation. Future work will further improve system performance by focusing on optimizing critical grasping parameters, including end-effector pose accuracy, grasping target recognition, and object approach trajectories.

**Acknowledgments**

This project was supported by the Shandong Provincial Natural Science Foundation of China (ZR2023MF077) and the Interdisciplinary Innovation Guidance Program of Qilu University of Technology (Shandong Academy of Sciences) (2025XKJC0108)

**References**

1. Inoue, S. & Urata, A. & Kodama, T. et al. High-precision mobile robotic manipulator for reconfigurable manufacturing systems. *International Journal of Automation Technology*. 2021. Vol. 15(5). P. 651-660.
2. Pan, Y.J. & Buchanan, S. & Chen, Q. et al. Survey on recent advances in planning and control for collaborative robotics. *IEEE Journal of Industry Applications*. 2025. Vol.14(2). P. 139-151.
3. Jia, Y. & Liu, Y. & Xi, N. et al. Design of robotic human assistance systems using a mobile manipulator. *International Journal of Advanced Robotic Systems*. 2012. Vol. 9(5). DOI: 10.5772/50828.
4. Øvsthus, Ø. & Robsrud, D.N. & Muggerud, L. et al. Mobile robotic manipulator based autonomous warehouse operations. In: *11th International Conference on Control, Mechatronics and Automation (ICCMA)*. IEEE. 2023. P. 278-283.
5. Ghodsian, N. & Benfriha, K. & Olabi, A. et al. MSOA: A modular service-oriented architecture to integrate mobile manipulators as cyber-physical systems. *Journal of Intelligent Manufacturing*. 2024. Vol. 36. P. 3207-3226.
6. Ma, Y. & Zhu, W. & Zhou. Y Automatic grasping control of mobile robot based on monocular vision. *The International Journal of Advanced Manufacturing Technology*. 2022. Vol. 121(3). P. 1785-1798.
7. Gao, W. YOLO-based gripping method for industrial robots. *International Journal of Computer Applications in Technology*. 2024. Vol. 75(1). P. 48-57.
8. Zhong, X. & Chen, Y. & Luo, J. et al. A novel grasp detection algorithm with multi-target semantic segmentation for a robot to manipulate cluttered objects. *Machines.* 2024. Vol. 12(8). No. 506.
9. Tzafestas, S.G. Mobile robot control and navigation: A global overview. *Journal of Intelligent & Robotic Systems*. 2018. Vol. 91. P. 35-58.
10. Lin, Y. & Sun, Y. Robot grasp planning based on demonstrated grasp strategies. *The International Journal of Robotics Research.* 2015. Vol. 34(1) P. 26-42.
11. Huang, Y.Y. & Song, K.T. Human-to-robot handover control of an autonomous mobile robot based on hand-masked object pose estimation. *IEEE Robotics and Automation Letters*. 2024. Vol. 9(9). P. 7851-7858.
12. Pearson, E. & Mirisola, B. & Murphy, C. et al. Robust autonomous mobile manipulation for substation inspection. *Journal of Mechanisms and Robotics.* 2024. Vol. 16(11). No. 115001.
13. Zhang, G. & Wang, S. & Xie, Y. et al. A task-oriented grasping framework guided by visual semantics for mobile manipulators. *IEEE Transactions on Instrumentation and Measurement*. 2024. Vol. 73. P. 1-13. DOI: 10.1109/TIM.2024.3381662.
14. Gao, K.L. & Wang, X.L. & Xu, Y.L. et al. Intelligent logistics express parcel real-time detection system based on improved YOLOv8. *Transport Problems*. 2025. Vol. 20(1). P. 179-192.

Received 10.07.2024; accepted in revised form 06.11.2025