Industrial robot modeling and Simulation in Gazebo

MaXu
maxu2023@shanghaitech.edu.cn

Yongqi Zhang
zhangyq12023@shanghaitech.edu.cn

January 28, 2024

1 Abstract

We have developed a system designed to autonomously guide robots to their designated positions with the correct pose for tasks related to the deployment of solar panels. Furthermore, we have created a simulation environment to virtually assess and test the performance of these robots.

This proposal outlines a project aimed at creating URDF files for robots, establishing a Gazebo simulation environment, and conducting experiments to evaluate a car moving algorithm within the simulation. The project aims to provide a robust framework for algorithm development and testing in a simulated environment, reducing the need for extensive physical testing.

2 Introduction

Solar energy is widely acknowledged as a sustainable energy source. In western China, the favorable spatial and resource conditions for solar energy generation present a significant opportunity to establish extensive arrays for solar power generation. Given the considerable size and weight of solar panels, the automation of their deployment necessitates the involvement of robots to ensure efficiency and safety. To enhance the precision and effectiveness of the robotic deployment, an intelligent system is imperative for accurate target localization and motion planning.

In response to this need, we have developed a system that enables robots to autonomously plan routes and navigate to designated targets based on sensor input. The complexity of the task is compounded by challenging environmental conditions such as air quality and sand impact. Conducting direct real-world tests under such conditions can result in prohibitively high costs. Therefore, we have implemented a simulation environment to evaluate and enhance the performance of robots in a virtual setting, ensuring both safety and efficiency improvements.

3 Methodology

3.1 URDF Design

- URDF File Design: Obtain STL files of different parts from CAD drawings.

(a) STL of Track

(b) STL of Truck

- URDF File Implementation: Utilize STL files to create the URDF files.
  - Define and set up the robot's links and joints.
  - Calculate the inertia matrix and other parameters.
- Add Transmission to joints

```
<!-- ===============  wheel Transmission =============== -->
<xacro:macro name="wheel_trans" params="prefix">
  <transmission name="${prefix}_wheel_trans">
    <type>transmission_interface/SimpleTransmission</type>
    <joint name="${prefix}_wheel_joint">
      <hardwareInterface>hardware_interface/VelocityJointInterface</hardwareInterface>
    </joint>
    <actuator name="${prefix}_wheel_motor">
      <hardwareInterface>hardware_interface/VelocityJointInterface</hardwareInterface>
      <mechanicalReduction>1</mechanicalReduction>
    </actuator>
  </transmission>
</xacro:macro>

<xacro:wheel_trans prefix="front_right"/>
<xacro:wheel_trans prefix="front_left"/>
<xacro:wheel_trans prefix="rear_right"/>
<xacro:wheel_trans prefix="rear_left"/>
```

Figure 1: Code of Tranmission

- The whole urdf in rviz

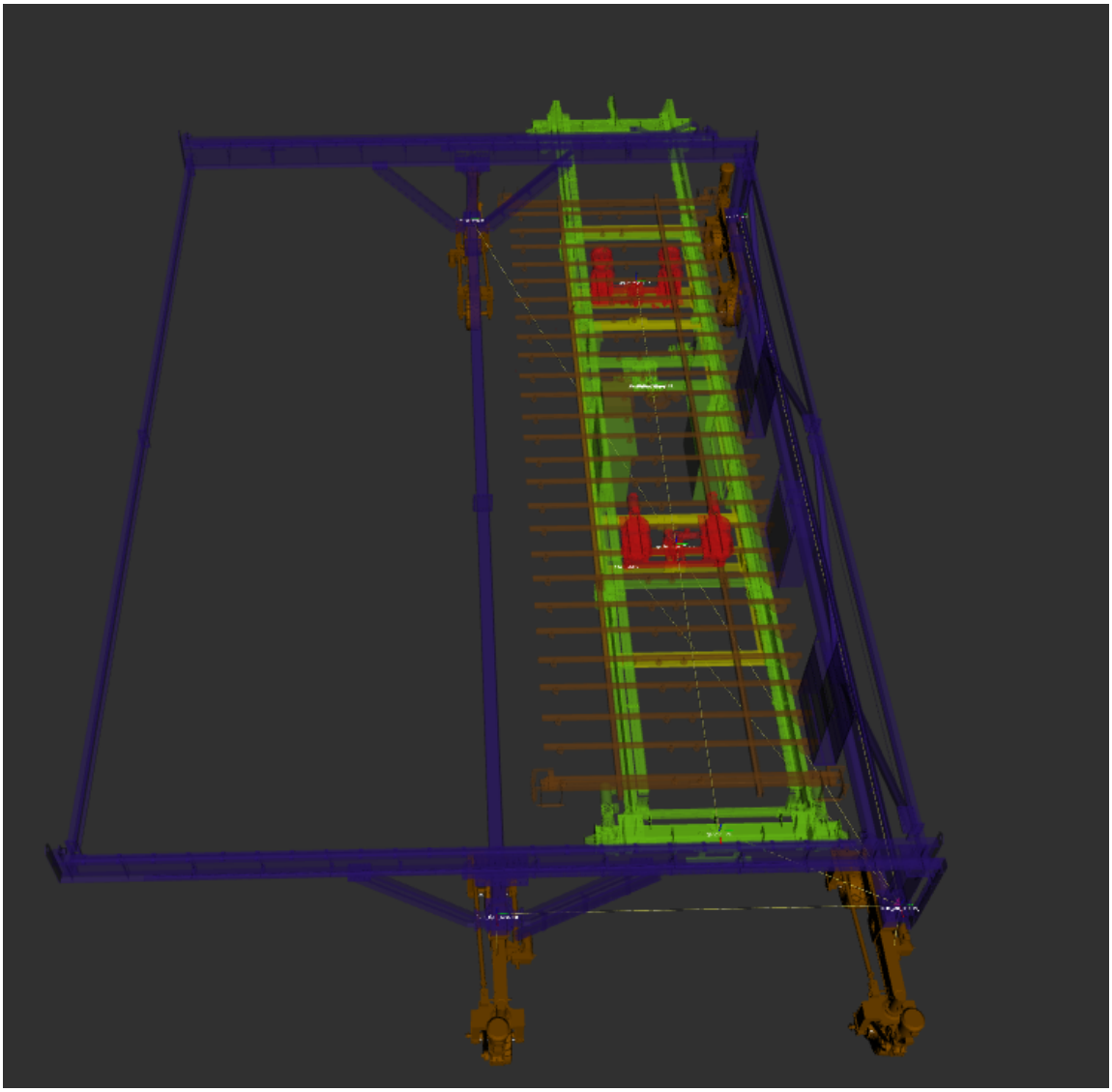

Figure 2: URDF of robot

3.2 Gazebo Design

- Create a flat terrain environment with a desert texture.
- Create rack and solar panel models to facilitate reuse of gazebo simulation world in the future.
- We set the appropriate parameters (density, mass impact volume, etc.) for the rack, solar panel, and our own robots to make the simulation in gazebo conform to the formal situation.
- Based on the parameter information in the map.yaml file, the model is automatically imported into a specific location in the gazebo simulation world.

- Adding sensor plugins such as DGPS and lidar to the robot can provide real-time feedback on the robot's own latitude and longitude and yaw, as well as detect surrounding objects.

```
<plugin name="gazebo_ros_gps" filename="libgazebo_ros_gps.so">
  <gpsTopic>gps</gpsTopic>
  <fixRate>10</fixRate>
  <horizontalPositionNoise>0.2</horizontalPositionNoise>
  <verticalPositionNoise>0.2</verticalPositionNoise>
  <horizontalVelocityNoise>0.01</horizontalVelocityNoise>
  <verticalVelocityNoise>0.01</verticalVelocityNoise>
</plugin>
```

Figure 3: Plugins of GPS

- Control the movement of our differential model robot in gazebo simulation environment. We use ros control packages.

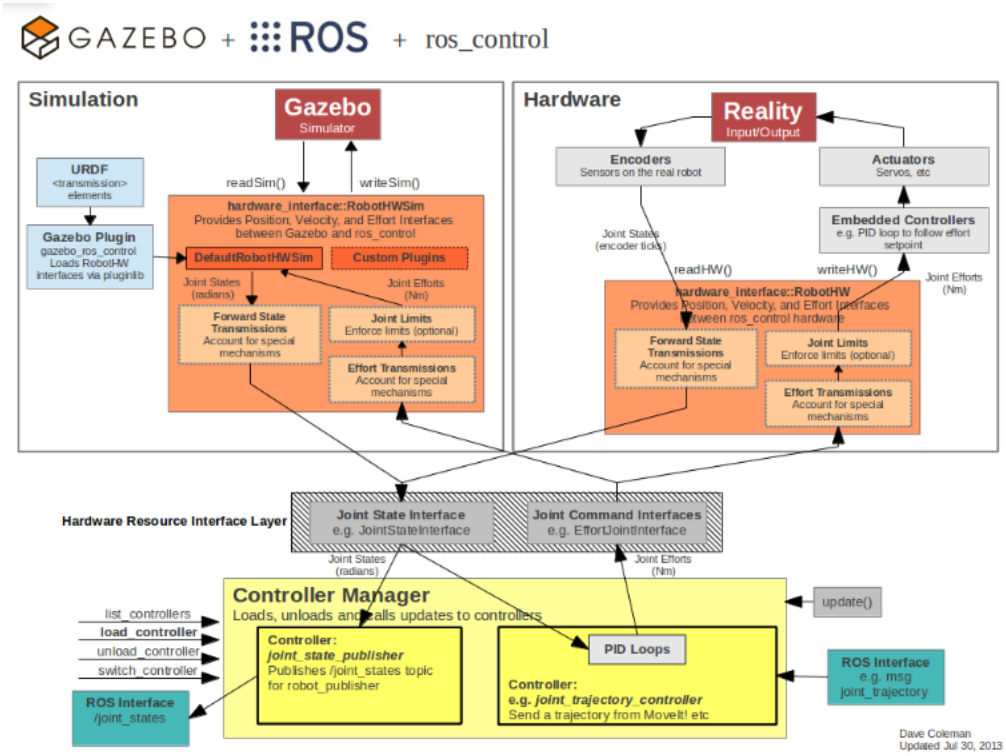

Figure 4: Overview of ros control

```
class Diffbot : public hardware_interface::RobotHW
{
public:
  Diffbot()
  : running_(true)
  , start_srv_(nh_.advertiseService("start", &Diffbot::start_callback, this))
  , stop_srv_(nh_.advertiseService("stop", &Diffbot::stop_callback, this))
  {
    // Intialize raw data
    std::fill_n(pos_, NUM_JOINTS, 0.0);
    std::fill_n(vel_, NUM_JOINTS, 0.0);
    std::fill_n(eff_, NUM_JOINTS, 0.0);
    std::fill_n(cmd_, NUM_JOINTS, 0.0);

    // Connect and register the joint state and velocity interface
    for (unsigned int i = 0; i < NUM_JOINTS; ++i)
    {
      std::ostringstream os;
      os << "track_" << i+1 << "_speed";

      hardware_interface::JointStateHandle state_handle(os.str(), &pos_[i], &vel_[i], &eff_[i]);
      jnt_state_interface_.registerHandle(state_handle);

      hardware_interface::JointHandle vel_handle(jnt_state_interface_.getHandle(os.str()), &cmd_[i]);
      jnt_vel_interface_.registerHandle(vel_handle);
    }

    registerInterface(&jnt_state_interface_);
    registerInterface(&jnt_vel_interface_);
  }

  ros::Time getTime() const {return ros::Time::now();}
  ros::Duration getPeriod() const {return ros::Duration(0.01);}
```

Figure 5: Abstraction of Robot

- The whole robot in gazebo

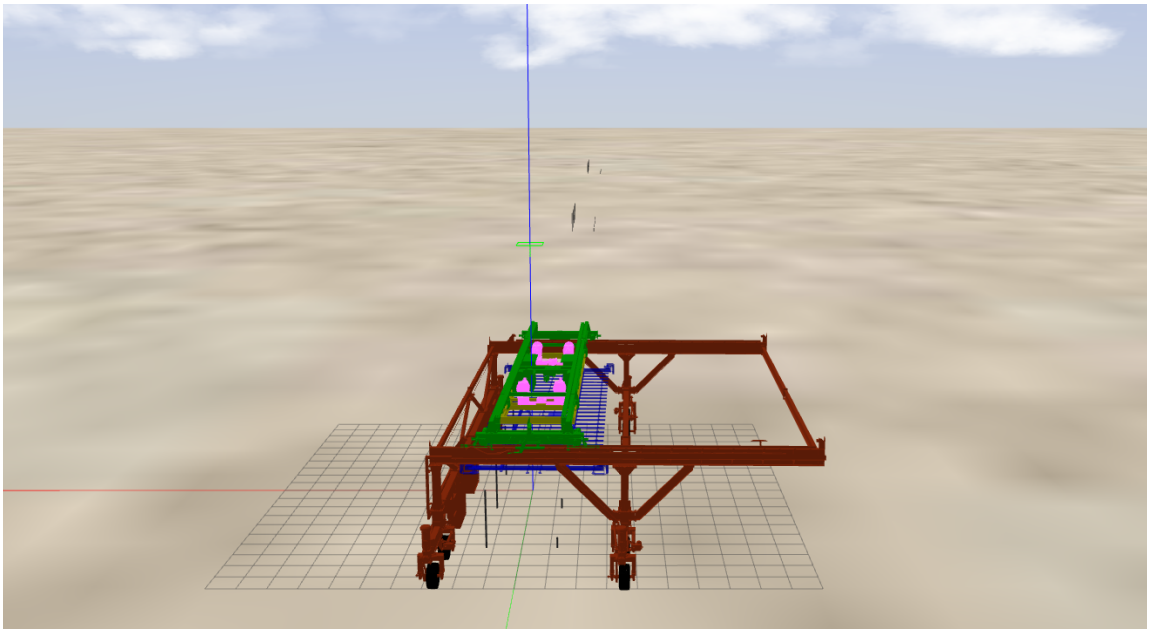

Figure 6: Robot in gazebo

4 Conclusion

In conclusion, this project successfully achieved its objectives of building a comprehensive robotic simulation environment through URDF modeling, Gazebo simulation, and the integration of radar and DGPS technologies. The combination of these elements provided a realistic and dynamic platform for testing and validating the performance of robotic systems in a controlled virtual environment.

The URDF modeling process allowed for the accurate representation of the robot's structure, joints, and sensors. Gazebo, as the simulation environment, proved to be a powerful tool, offering a high-fidelity representation of the robot's interactions with its surroundings. The successful integration of radar and DGPS enhanced the robot's perception and localization capabilities within the simulated environment.

The results demonstrated the effectiveness of the simulation setup, showcasing the robot's movement and interactions, as well as the accurate outputs from radar and DGPS sensors. The data analysis provided valuable insights into the system's performance, allowing for a deeper understanding of the robot's behavior and the reliability of sensor data.

5 Recommendations and Future Work

While this project has achieved notable success, there are several avenues for future work and improvement:

1. Enhanced Realism: Explore ways to further enhance the realism of the simulation environment. This could involve refining the physics engine in Gazebo, incorporating more detailed sensor models, and improving the accuracy of URDF models.

2. Advanced Sensor Integration: Investigate the integration of additional sensors, such as cameras or lidars, to broaden the robot's sensing capabilities. This expansion could contribute to a more comprehensive simulation of real-world scenarios.

3. Algorithm Optimization: Focus on optimizing algorithms related to navigation, perception, and control within the simulated environment. This could involve implementing more advanced algorithms for path planning, obstacle avoidance, and sensor fusion.

4. Scenario Complexity: Increase the complexity of simulated scenarios to challenge the robot's capabilities. This may involve designing intricate environments, introducing dynamic elements, or simulating adverse weather conditions to test the system's robustness.

5. Hardware-in-the-Loop (HIL) Testing: Explore the possibility of incorporating Hardware-in-the-Loop testing, allowing for the integration of physical hardware components into the simulation. This step could bridge the gap between simulation and real-world deployment.

6. User Interface and Interaction: Develop a user-friendly interface for controlling and monitoring the robot in the simulation. This could facilitate easier

experimentation and testing for researchers and engineers.

In summary, the success of this project lays the foundation for future advancements in robotic simulation and testing. By addressing the outlined areas for improvement, we can ensure a more sophisticated and reliable simulation environment, ultimately contributing to the development of robust and capable robotic systems in real-world applications.