ISSN No:-2456-2165

Bridge Gateway for Solid Works to URDF and Integration with Robot Operating System

Raghul T.; Nagalaxman G.; Anbarasi M. P.
Robotics and Automation PSG College of Technology Coimbatore, India

Abstract:- Unified Robot Description Format (URDF) integration of mechanical designs from SolidWorks, a well-known computer-aided design (CAD) software, into the Robot Operating System (ROS) ecosystem is essential for seamless robotic simulations and control. In order to bridge the gap between SolidWorks CAD models and ROS-compatible URDF files, this paper presents a thorough methodology for the conversion process. Exporting SolidWorks models, comprehending the structure and syntax of URDF, defining links, joints, visual, and collision properties, and integrating transformational and inertial data are all necessary steps in the procedure. Within the URDF framework, particular focus is placed on precisely positioning joints and defining their axes of motion. The resulting URDF files make it possible to precisely represent robotic structures, making it easier to simulate, analyze, and control robotic environments that are powered by ROS. Case studies and useful insights demonstrate. This conversion process' accuracy and efficiency demonstrate how it can be used in a variety of robotic applications.

Keywords:- *SolidWorks, URDF, ROS, CAD, Robot, Modelling, Simulation, Interoperability, Integration, Conversion.*

I. INTRODUCTION

The process of converting a SolidWorks model to the Unified Robotic Description Format (URDF) involves converting engineering designs from SolidWorks, a popular computer-aided design (CAD) software, to a format that Can be used in simulation and robotics applications. URDF is a widely used XML format to describe the structure and properties of robots, commonly used in ROS (Robot Operating System) to control and simulate robots. Here we are going to build a Solidworks model and integrate it in ROS using URDF and thereby making the 6 DOF robot to perform the operation.

Solidworks is one of the model designing software which is used widely. The parts required for our design are made separately using Solidworks [1] and we mate all the parts we had designed before and made the final design of the product as 6 Degrees Of Freedom Robot [2], likewise we are able to design any model we want to design using Solidworks. This is in contrast to existing modeling approaches, which the robot manipulators consider extensions of the standard and modified Denavit-Hartenberg (DH) conventions [3] or the Product of Exponential (PoE) formulation [4].

We aim to achieve simplicity in implementation for obtaining an assembled modular robot description as a Unified Robot Description Format (URDF) file, which can be easily utilized for applications with software tools from Robot Operating System (ROS) [5] libraries. Robot modeling knowledge and a lot of time are needed when configuring the modular robot control for every new assembly in order to take into consideration the dynamics and kinematics of the new system. To avoid this we create the Unified Robot Description Format (URDF) file for robots automatically by utilizing the kinematic and dynamic descriptions that are stated in accordance with the URDF of the individual modules that make up the manipulators[6,7].

The building and simulating a six-degree-of-freedom (DOF) robotic arm specifically for harvesting coconuts, as well as collaborative robots for use in research and assistive robots like wheelchairs and home robots, as well as search and rescue robots for dangerous situations. For the robotic arm [8], a kinematics-based solution has been created that facilitates use and operation. The Robot Operating System (ROS) was used in the development of the robotic control interfaces that were presented in the study. RVIZ was utilized for the design and visualization of the 6-DOF articulated robotic arm. The robotic arm's ease of manipulation with the end effector was made possible by the kinematics.

ROS, an open source Robot Operating System An overview of ROS, an open-source robot operating system, it offers a structured communications layer on top of the host operating systems in a heterogeneous compute cluster. and we see how ROS relates to existing robot software frameworks, and briefly overview some of the available application software which uses ROS[9]. This paper explains how to create a custom URDF file and covers the mechanics and electronics used in the OpenDog. To port it to ROS and run path planning algorithms on the same, a custom URDF built on the OpenDog model that is currently available is required. To do this, more independent packages and launch files are made for the model's visualization and simulation. Rviz is used to visualize the movement simulations that are carried out in Gazebo[10]. The structure of this paper was subsequently followed as Assembly on Guide for Robot model is described in Section 2, and Positioning of Joints and Movement of Axis are described in Section 3, The URDF module is presented in Section 4, Robotic Arm Package and working in Ros are described in Section 5 and 6, Followed By Conclusion in section 7.

ISSN No:-2456-2165

II. ASSEMBLY ON GUIDE FOR ROBOT MODEL

The process of creating an assembly guide for a robot model in SolidWorks entails delineating the steps and procedures required to assemble the various components using the SolidWorks CAD software. Presented below is a generalized framework for such a guide:

- Initiate SolidWorks and generate a new assembly document.
- Import the necessary CAD files for the robot components.
- Position the base component in the assembly by mating itto the origin.
- Place subsequent components in their respective positions using appropriate mates, such as coincident and concentric.
- Utilize mates to constrain the relative motion between components, including parallel, perpendicular, and distance mates.
- Ensure proper alignment and connection of components using mates for rotational and translational degrees of freedom.
- Incorporate screws, bolts, nuts, and other fasteners using the Toolbox feature in SolidWorks.
- Position and mate the fasteners appropriately within the assembly.
- Verify that all components move and interact as intended within the assembly.
- Make necessary adjustments to ensure smooth functioning of the robot model.
- Generate exploded views to illustrate the assembly process.
- Annotate the exploded views to provide a clear understanding of the assembly sequence.
- Create detailed drawings of the assembled robot model.
- Annotate the drawings with part numbers, dimensions, and other relevant information as shown in Fig 1.

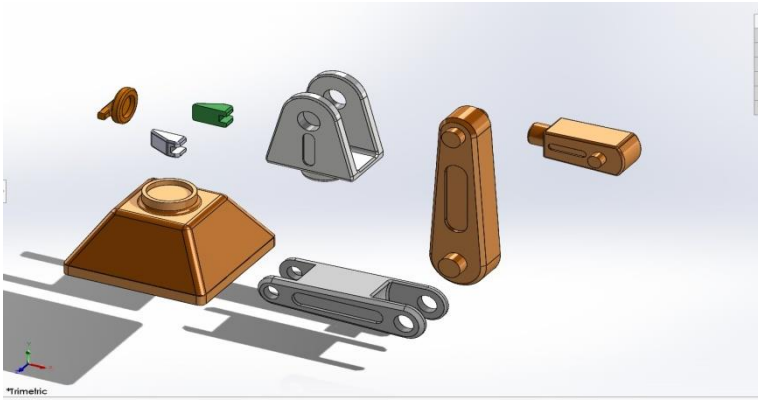

Fig. 1: Assembling parts of Manipulator

III. POSITIONING JOINTS AND AXIS OF MOVEMENT

In SolidWorks, positioning the joint at position 0 and determining its axis of motion is fundamental to accurately representing the robot model. Here are the steps to achieve this in SolidWorks:

A. Positioning mismatch:

In SolidWorks, joints are often created during assembly to define the relationships between components. To zero a joint, follow these steps:Open your assembly in SolidWorks. Select the "Mate" command:Go to the "Assembly" tab in Order Manager. Click "Mate" or use the keyboard shortcut "M."Choose the right type of companion for your joints:For rotating joints (e.g., swivel joints), select "Concentric" or "Coincident" depending on the specific joint configuration.Select the desired faces or axes for the joint parts and assemble them to align the joint in the 0 position.These procedures will help you make a mate that accurately aligns the components of your joint, zeroing the joint and enabling precise movement or contact between the pieces. This procedure is essential to guaranteeing that your assembly will perform as intended, particularly in dynamic systems or mechanisms.

B. Create motion axis:

To create a joint's axis of motion, you can use reference geometry to define the rotational or translational axis. Follow these steps:Open the component part file associated with the assembly. Create reference axis:Go to the “Reference Geometry” drop-down menu under the “Features” tab.Select "Axis".Choose the appropriate references for the axis: Select two cylindrical or planar faces or any suitable reference aligned to the desired axis of motion.Click "OK" to create the axis. Save the section file. Return to the congress. When defining a joint, select the axis you created as the rotation or translation axis of the joint. Make sure the axis is aligned with the desired axis of movement of the joint. Inspect the joint to verify its joint movement along the specified axis. By following these steps, you can position the joint at position 0 and precisely define its axis of motion in SolidWorks, providing a clear representation of the kinematics of the robot model as shown in Fig 2.

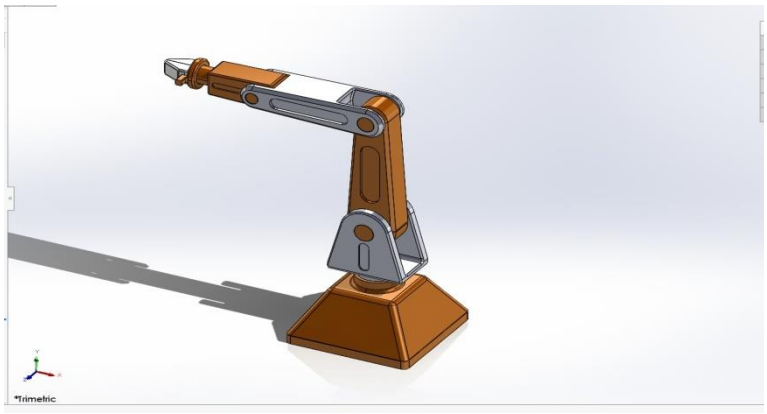

Fig. 2: Assembled Manipulator

IV. THIRD PARTY TOOL FOR EXPORTING

"SW2URDF" is not a standard or built-in tool in SolidWorks. Instead, you can refer to custom scripts or third-party tools developed by the community to help convert SolidWorks models into URDF files. Here's a general approach to exporting a SolidWorks model to a URDF file using a custom script or tool (assuming "sw2urdf" is a custom script or tool):

- Install and configure the "sw2urdf" tool: Make sure the "sw2urdf" tool is installed and configured in your SolidWorks environment. Typically this involves downloading a script/tool from a trusted source and following the installation instructions as shown in Fig 3.
- Prepare your SolidWorks model: Open your SolidWorks model and make sure it is properly organized, including the necessary joints, links, and components that define the robot's structure. Use "sw2urdf" to export: Run the "sw2urdf" script/tool from SolidWorks. This tool will guide you through the process of exporting a SolidWorks model toa URDF file.

  Note: Specific export steps and options may vary depending on the script/tool you are using.
- Follow the export instructions: Follow the instructions provided by the "sw2urdf" tool to customize export settings, including match type, name, link properties, and other necessary settings.

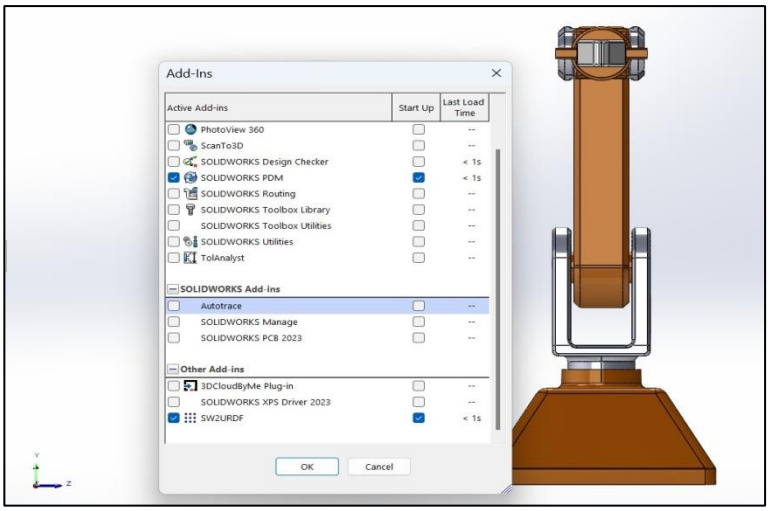

Fig. 3: SW2URDF tool on Solidworks

- Create URDF file: Once you have configured the export settings to your liking, proceed with creating the URDF file using the "sw2urdf" tool.
- Validate URDF files: Open the created URDF file to ensure that it accurately represents the robot model and its kinematics. Make any necessary adjustments or corrections.
- Import into ROS:After creating and validating the URDF file, import it into ROS and test it in a simulated environment to ensure it works as expected.

Third-party tools to export Solid Works models to URDF:

- Sw2URDF Solid Works to URDF exporter: The "sw2urdf" (Solid Works to URDF) project is a third-party tool that allows you to export Solid Works models in URDF format. This tool typically consists of a combination of SolidWorks macros and Python scripts.
- Exporter ROS Industrial Universal Robot Description Format (URDF): ROS Industrial provides a set of tools, including scripts and Python packages, that can help convert SolidWorks models to the URDF format. It is part of the ROS ecosystem and is widely used for

ISSN No:-2456-2165

robotics applications.

- ROS URDF export plugin for SolidWorks: Some community members have developed custom plugins or scripts for SolidWorks that allow model export to the URDF format. These plugins may have different features and capabilities.

V. ROBOTIC ARM PACKAGE IN ROS

After creating the URDF file for the robot arm using SOLIDWORKS, a ROS package folder will be created. Copy this package folder as is to your ROS workspace (e.g. catkin_ws/src) without renaming it. If you don't have ROS installed, follow the instructions for your specific ROS distribution. ROS Melodic on Ubuntu 18.04. Open a terminal. Update system package list to upgrade and install new packages: sudo apt-get update

- Access your personal folder disk ~Create a new
- Catkin workspace (you can name it anything youwant, e.g. moveit_workspace):
- mkdir -p moveit_workspace/src
- Access the workspace: cd moveit_workspace
- Initialize the Catkin workspace: initialize kitten
- Create a Catkin workspace: build kittens

Source the setup.bash file to configure the workspace to use: development source/setup.bash Copy the ROS package and configure MoveIt.In the next section, copy the ROS packageinto the Catkin workspace and configure it for use with MoveIt to simulate a robotic arm in ROS.

- Intersection:Nodes are individual software processes that perform specific tasks. Multiple nodes can operate simultaneously and communicate with each other through ROS.
- Objects:Nodes communicate with each other by publishing and subscribing to topics. Topics are called buseson which nodes can send and receive messages.
- Posts:Messages are data structures used to communicate information about ROS topics. They are defined in messagefiles and have a specific format.
- Service:The service allows nodes to send requests and receive responses, allowing more complex interactions between nodes.
- Act:Actions are similar to services but are designed for tasks that take longer to complete. They help set goals, provide feedback, and report results.
- Kitty:Catkin is the official build system for ROS. It helps create packages and manage their dependencies.
- Package:Packages in ROS are organizational units that contain nodes, configuration files, and other resources needed for a specific task.
- Launch file:Launch files are XML files used to launch multiple ROS nodes with specific settings and configurations.
- RViz:RViz is a powerful 3D visualization tool in ROS used to visualize sensor data, robot models, and other information.
- Move it:MoveIt is a popular ROS package used for motion planning, manipulation, and control of robotic arms.

VI. WORKING IN ROS

Navigate to the src folder of your ROS workspace. Create a new package (replace your_package_name with an appropriate name). In your new package, create an urdf directory to store your URDF files. Place your URDF files (e.g. robots.urdf) inside the urdf folder. category. Navigate to the root of your package (replace your_package_name with your package's actual name). Open the CMakeLists.txt file to edit it. Add the necessary lines to the CMakeLists.txt file to create the URDF file. Open the package.xml file for editing. Add the appropriate dependencies to your package, including urdf. Build your Catkin workspace from the workspace root directory. Find the install script to display your package in the current terminal session. You can now use URDF files and any buttons or programs associated with your package.Configure a URDF (Unified Robot Description Format) file generated from SOLIDWORKS for use in ROS and Gazebo.

Export your robot arm model from SOLIDWORKS to a URDF file. Place the URDF file and associated meshes (if any) into a folder in your ROS. wrap. Let's say you have a package named your_package_name. Open the URDF file “your_robot.urdf”; in a text editor. Make sure all file paths to meshes are relative and point correctly to the meshes in the meshes/ directory. Update URDF to include required ROS and Gazebo components, such as global properties and sensor plugins. Make sure all joints and connections are placed correctly. with precise properties, such as inertia, joint boundaries, joint type, etc., to accurately represent the robot arm. For compatibility with Gazebo, add Gazebo specific tags in the URDF to define physical properties and other details related to the simulation. Create a Gazebo launch file (for example: your_robot_gazebo.launch) in your package to launch URDF in Gazebo. Launch URDF in Gazebo using the launch file you created, Launch URDF in Gazebo using the launch file you created.Create a YAML file to configure the ROS generic trajectory controller to control the joints of the robot arm including specifying the necessary parameters for the controller. Below is an example of a YAML configuration for a typical robot arm with general trajectory control. Creating the ROS initialization file to initialize the URDF, buttons, and controllers of the robot arm includes defining the necessary buttons, parameters, and controllers. Load the robot's URDF description from a file using the param tag. Start the General Status Editor to publish general statuses.

Create a robot model in Gazebo using URDF descriptions. Start Controller Manager. Load the controller configuration using the rosparam tag, in specifying the path to your common path controller configuration YAML file. Start the pipeline controller using the controller manager. Remember to replace "your_package_name" and "your_robot.urdf" with the name of the appropriate package and URDF file for your specific robot arm. Also, adjust the path according to your file structure. You will also need to create a YAML file for the joint trajectory controller configuration (e.g. Joint_trajectory_controller.yaml) and adjust the launch file accordingly. Launch this file using roslaunch your_package_name your_launch_file.launch to

ISSN No:-2456-2165

launch the URDF, buttons, and controllers of your robot arm. To create a Catkin workspace and launch the robotic arm in Gazebo using a launch file, follow these steps: Go to your Catkin workspace, root folder, create workspace, create launch file "e.g. launch_robot_arm_gazebo.launch". in your ROS package (your_package_name) to launch the robot arm in Gazebo. Launch the robot arm in Gazebo using the created launch file. Make sure to replace "your_package_name", "your_robot.urdf", and "your_world.world" with the appropriate values for your robot arm and environment. This launch file loads the URDF, starts Gazebo with an empty world, creates the robot model in Gazebo, starts the global state editor, and loads and starts the global path controller.

Adjust settings and paths based on your specific robot and setup.Use the MoveIt Setup Assistant to create a URDF model control package imported from SOLIDWORKS that includes configuring the robot's motion planning and control capabilities. Make sure MoveIt is installed. If not, install it using the following command. (sudo apt-get install ros-melodic-moveit) Replace melodic with your ROS version if you are using a different version. Launch the MoveIt setup wizard. Click “Create new MoveIt configuration”. to start a new configuration. Click "Import File" and select the URDF file you exported from SOLIDWORKS. Determine the robot's semantic information such as robot name, end effector link, and group name. Set calendar groups for MoveIt. This often includes joints used for movement planning. Set agent information last, if any. Proceed to the next steps and create the self-collision matrix and the allowed collision matrix if needed. Calibrate your robot drive system, configure the kinematic solver and charge controller if applicable. If your robot has virtual joints, configure it accordingly. Configure all of your robot's passive joints, if any.

Configure scene planning settings based on your robot's requirements and environment. Determine the final impact pose for your robot, if applicable. Save configuration. Click “Create Package” to create a MoveIt configuration package. The MoveIt Setup Wizard will create a MoveIt setup package. You will find the package in the specified directory. After creating the package, you can launch MoveIt with the command(roslaunch_your_generated_moveit_config_package_moveit_planning_execution.launch).Replace the given your_generated_moveit_config_package with the actual name of the package generated by the MoveIt Setup Assistant. You now have a MoveIt configuration package for your robotic arm, imported from SOLIDWORKS URDF, ready for motion planning and control. You can use the MoveIt RViz plugin or the MoveIt Python or C++ API to plan and execute movements for your robotic arm.

You can use the MoveIt Setup Assistant to create a Unified Robot Description Format (URDF) package imported from SOLIDWORKS by following these general instructions. MoveIt is a powerful motion design framework used with ROS (Robot Operating System) to control robotic arms and other robotic systems.Install and configure ROS and MoveIt: First, make sure you have ROS and MoveIt installed on your system. You can follow the official ROS and MoveIt installation instructions for your specific ROS distribution.Import URDF from SOLIDWORKS: Make sure the URDF file is exported from SOLIDWORKS before continuing.

Create a new ROS package: Open a terminal and navigate to the ROS workspace (usually catkin_ws/src) where you want to create a new package.Use the catkin_create_pkg command to create a new ROS package containing the MoveIt definition: hit catkin_create_pkg my_robot_moveit_config Run the MoveIt Setup Assistant: Run the MoveIt Setup Assistant with the following command: hit roslaunch moveit_setup_assistant setup_assistant.launch Download the URDF: In the MoveIt Setup Assistant, use the Download button to download the SOLIDWORKS URDF file. Set the robot location and design group: Set the starting position of the robot using the setup wizard. This helps MoveIt understand the constraints and kinematics of the robot and joints.

Create a design group for your bot. This determines which parts of the robot you want to control. Collision Matrix: Configure collision detection between different parts of the robot. Self collision control: Set self-collision checking parameters. Kinematics and controls:Define kinematic solvers for your robot. Determine which drivers you intend to use and configure them.

Create a configuration package:After completing all the necessary settings in the MoveIt Setup Assistant, click "Generate Package". a button Build and get a workspace: Build the package in your ROS workspace directory (eg catkin_ws) and get the setup.bash file: hit cat_do source devel/setup.bash Run MoveIt in Rviz: You can view and control your bot in Rviz using the following command: hit roslaunch my_robot_moveit_config demo.launch You now have a ROS package (called "my_robot_moveit_config" in this example) that contains all the necessary configuration files and startup files of MoveIt to control the movement of your robot and #039; You can customize and refine the configuration according to your robot system as shown in Fig 4.

ISSN No:-2456-2165

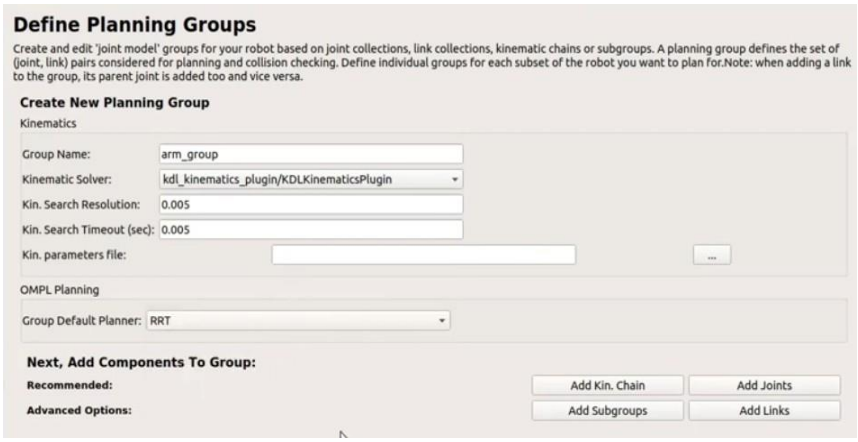

Fig. 4: Moveit Setup-Assistant on ROS

To verify the MoveIt processing package created for the URDF of the custom robotic arm, follow these steps to ensure that the manipulation attributes are set and working correctly.Run the manipulation demo: The MoveIt package you create should contain a startup file to run the MoveIt manipulation demo. With this demo, you can interactively design and execute the movements of a robotic arm. Run the demo: hit roslaunch my_robot_moveit_config demo. launch Replace my_robot_moveit_config with the name of your specific MoveIt configuration package. RViz visualization: RViz should open to provide a GUI for your robot arm. You should see your robot model, environment and various interactive characters.

Choose and plan a destination: In RViz, select the "Movement Planning" tab. This tab allows you to interact with the robot and power booster and plan moves. You can: Choose an exit booster or a specific link in your bot. Set the target position and final effect of your bot. You can click on "2D Pose Estimate" and "2D Nav Goalandquot; To set a destination in RViz. Plan your move: Once you have set your goal, you can click the "Plan" button. the RViz button to calculate the business plan. MoveIt uses motion planning algorithms to find the appropriate path for your robot to reach the target position. View the plan: You can see the planned movement by clicking andquot;Execute and quot; button RViz.

The robot and your model simulate the intended movement to reach the target position. Follow the collision check: During design and implementation, MoveIt must perform collision checks to ensure that the robot does not collide with obstacles in its environment. Verify that collision detection works as expected. Handling extensions: Check if you have defined handling plugins or handlers. You can interact with these plugins in RViz to define capture strategies or other manipulation tasks. Custom Power Amplifier and Gripper Control: If you have a custom end effector or gripper for your robot arm, make sure you can control and visualize its movements and interactions in RViz. Performance and Durability: Make sure the handling package is working efficiently and reliably. Experiment with different target positions and scenarios to ensure manipulation design and execution.Additional testing and validation:Depending on your specific robot and application, perform additional tests to verify special capabilities such as pick and place functions, object manipulation, or other manipulative tasks that your robot is designed for as shown in Fig 5.

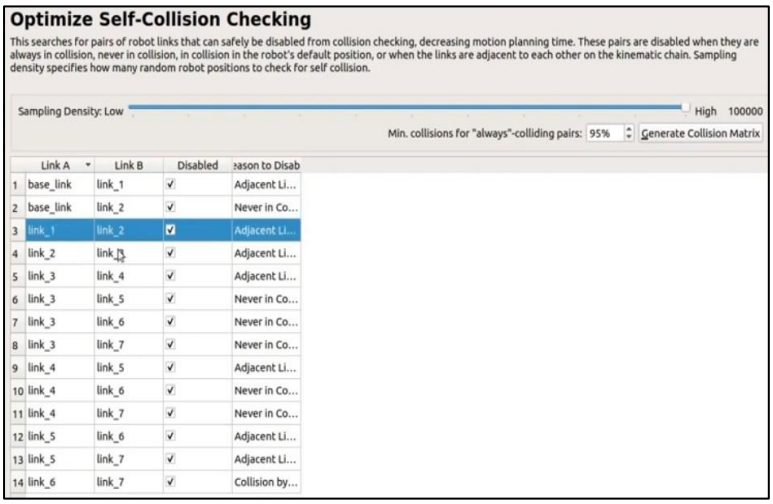

Fig 5: MoveIt processing package on ROS

ISSN No:-2456-2165

Creating a new ROS controller for the MoveIt rendering package for a custom robotic arm requires defining a controller for your specific hardware and connecting it to the MoveIt motion design framework. Below are the general steps to create a new ROS controller. Hardware User Interface:First, you must develop a hardware interface that communicates with the controllers of the robot armind#039;. This interface usually involves writing ROS nodes that send your bot and #039; commands to motors or servos.ROS controller configuration: Create a configuration file for your new controller. This file defines the device interface and controller parameters. You usually use the ROS Management Framework for this.

Create a .yaml file with the controller and parameters.Joints This example sets up a pose controller for the robot and joints.Implementation of ROS controller:Enable the ROS controller code that turns MoveIt commands into control actions for your robot arm. Normally, you would write a ROS node to manage this translation. Here's a simplified Python Run the file: Create a startup file to start the ROS controller node. Integration with MoveIt: To integrate your new controller with MoveIt, you need to update the MoveIt configuration package (created using the MoveIt Configuration Assistant) to configure your new controller for robot connections.Testing: Test your controller by running it with the MoveIt app. Make sure it can correctly receive and execute the move plans created by MoveIt. Calibration and tuning: Refine the controller and perform the necessary calibrations to ensure it moves the robot accurately. Security: Check your controller to avoid unexpected behavior and ensure safe operation of the robot arm. Documentation: Document our controller and its integration into the custom robot arm for future reference. Keep in mind that the details of how to implement a controller will depend on your robot's hardware, control system, and the type of controller you want to create (such as position, velocity, or force control) as shown in Fig 6.

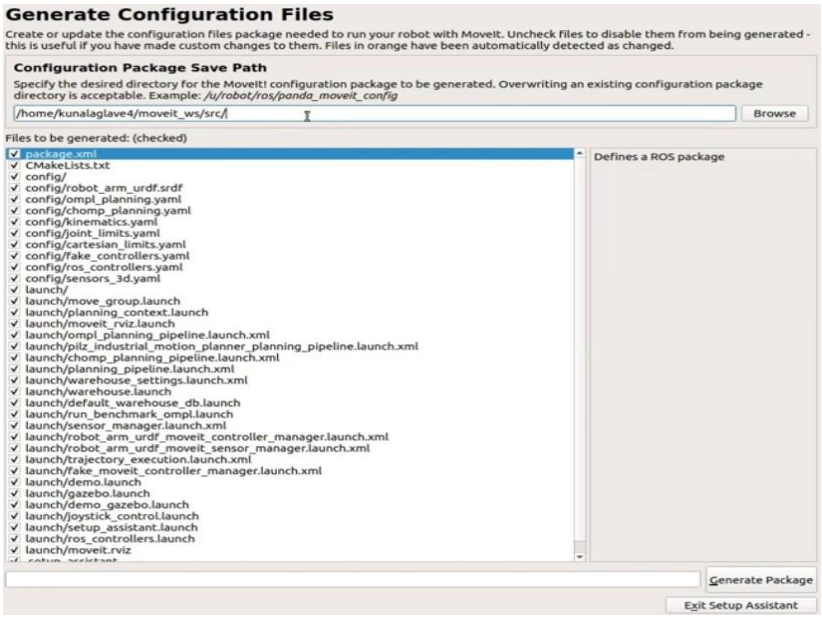

Fig. 6: File configurations on ROS

To load a full Move It manipulation simulation of a custom robot arm to create a new startup file, you must specify the components and configuration required for the simulation.

Below is a simplified example startup file to get you started: Create startup file: Create a new launch file in your ROS package and launch directory, e.g. manipulator_simulation.launch Specify the launch file: Run the boot file specifying the boot description. You can use the launch tag to specify the launch file #039 name and package as follows: xml andlt; launchgt; andlt;!-- Information about packages and startup files --andgt; andlt; arg name= and quot; robe_description and quot; default=andquot;$(find_your_package_name)/urdf/my_robo t.urdfandquot; /andgt; Replace package_name with the actual name of your ROS package and your_robot.urdf with the URDF file of your custom robot arm. Run Robot State Publisher:You want to run robot_state_publisher to publish the robot and state in /robot_description. This allows MoveIt to take advantage of the robot's kinematic and dynamic capabilities. Run the MoveIt Setup Assistant: If you have previously used the MoveIt Setup Assistant to create configuration files for your robot, you can download those settings.we define a MoveIt configuration package (your_robot_moveit_config) and an SRDF file that defines the robot's self-collision matrix and design groups.

Run the MoveIt RViz plugin: You can visualize the robot in RViz and interact with MoveIt using the MoveIt RViz plugin. Specify the configuration package to download. xml andlt; node name=and quot; moveit_rviz and quot; pkg=andquot; rvizand quot; type= and quot; rviz and quot; args=andquot;/launch/moveit.rvizandquot;andgt; Start the simulation node: Start the simulation node of your robot armand#039;. The details depend on the simulator you are using (eg Watchtower, V-REP, etc.). xml andlt;node name= and quot; my_robot_simulations and quot;

ISSN No:-2456-2165

pkg=andquot;your_robot_simulation package andquot; type = and quot; your_robot_simulation_nodesquot; output=andquot;screenandquot;andgt; andlt;/nodeandgt; Replace your_robot_simulationpackage and your_robot_simulationnode with the actual simulation package and node names.Run the MoveIt design and implementation nodes:Run the MoveIt nodes that handle the planning and execution of the move.

Include safety measures and additional components: Depending on your specific configuration, you may need to add additional components such as security controls, detection modules, and more. Test the launch file: Run the startup file with roslaunch to run the MoveIt simulation for the custom robot arm. hit roslaunch your_pact_name manipulator_simulation.launch.This startup file is the starting point for running a custom MoveIt robot arm simulation. You may need to modify and extend this to suit the configuration and simulation of your particular bot #039. To verify that the MoveIt action package was created for your URDF custom robotic arm, follow these steps to test and understand the generated package.

Navigate to the folder where the MoveIt configuration package was created. Inside the created package you will find various folders and directories. Below is a breakdown of the main folders and their contents: configuration: Contains YAML files for configuring MoveIt and setting up MoveIt configuration, including kinematics.yaml, joint_limits.yaml, ompl_planning.yaml, etc.launch: Contains launch files for launching MoveIt buttons and configuring MoveIt paths. Important files may include move_group. launch, demo. launch, etc.stitches:Contains your robot meshes, if any.see: Contains RViz configuration files for visualizing your robot and planning movements.script: Contains scripts that can be used to control robots using MoveIt. src:Contains source code files that may have been created or added for custom control logic.urdf:Contains the URDF file for your robotic arm.configuration:Contains configuration files for controllers and planners.

Open and examine the YAML configuration files in the configuration folder to understand the configuration of the robot control and scheduling settings. Check the joint_limits.yaml file to see the joint limits specified for your robot. Open and examine the launch files in the launch folder. For example, the move_group.launch file configures the MoveIt's MoveGroup button. Find the launch files that start the necessary components of MoveIt, such as scene planning, path execution, and RViz visualization. Open and examine the RViz configuration files in the rviz folder to understand how your robot is displayed in RViz. Check out Planning_execution.launch or similar launch files to understand how RViz is integrated with MoveIt. Explore any scripts or source code files contained in the scripts and src folders to understand any custom logic or controllers that may have been created or added. By exploring the contents of the generated MoveIt manipulation package, you will better understand how to configure MoveIt for your URDF custom robotic arm.

Creating a new ROS controller for the MoveIt manipulation package involves defining a ROS controller that interfaces with MoveIt to control the robotic arm. Create a new ROS package to contain your controller. Navigate to your packages folder. Create a new C++ source file for your controller (e.g. Joint_position_controller.cpp) in your package's src directory. Open the file and add the necessary include and control logic. Open the CMakeLists.txt file in your package. Add the following lines to bind your controller execution. Create your ROS package. Run your controller. It represents a simple generic position controller.

Adjust and extend the code based on your specific robot arm and control requirements. You may also need to connect to MoveIt for trajectory planning and execution. To create a launch file that loads the full MoveIt driver simulation for the custom robotic arm, we will configure the necessary components including MoveIt, Gazebo, RViz, and the controller. Create a new launch file, for example moveit_simulation.launch. #41; in your package. Open the launcher file to edit it. Determine your launcher file structure and import the necessary components. Customize the launch file based on your specific robot arm and requirements:Update your_moveit_config_package with the name of the MoveIt configuration package you created.Update your_world.world with the appropriate Gazebo world file.Replace your_robot with your robot model name.Adjust the path to your URDF file accordingly. Launch the MoveIt simulation using the generated launch file. Replace your_package with the actual name of your package. This launch file will launch MoveIt for motion planning, Gazebo for simulation, and RViz for visualization.

The robot model will be created in Gazebo and you can use MoveIt to plan and execute movements for your robotic arm in a simulated environment. Adjust the launch file as needed to fit your robot arm's specific configuration and environment.To launch the final MoveIt driver simulation of your custom robotic arm in RViz and Gazebo. Create a new launch file “e.g. custom_robot_simulation.launch”; in your ROS package.

- your_package with the actual name of your ROS package.
- your_world.world with the appropriate Gazebo world file.
- your_robot with your robot model name.
- your_moveit_config_package with the name of your MoveIt configuration package.

ISSN No:-2456-2165

Launch the MoveIt simulation with Gazebo and RViz using the generated launch file. This launch file will start Gazebo for simulation, create the robot model in Gazebo, launch RViz for visualization, and launch MoveIt for motion planning. You will have a simulated environment in which you can plan and execute movements for your personalized robotic arm using MoveIt combined with Gazebo and RViz. Adjust launch file paths and settings to suit your specific robot arm configuration and environment as shown in Fig 7 & Fig 8.

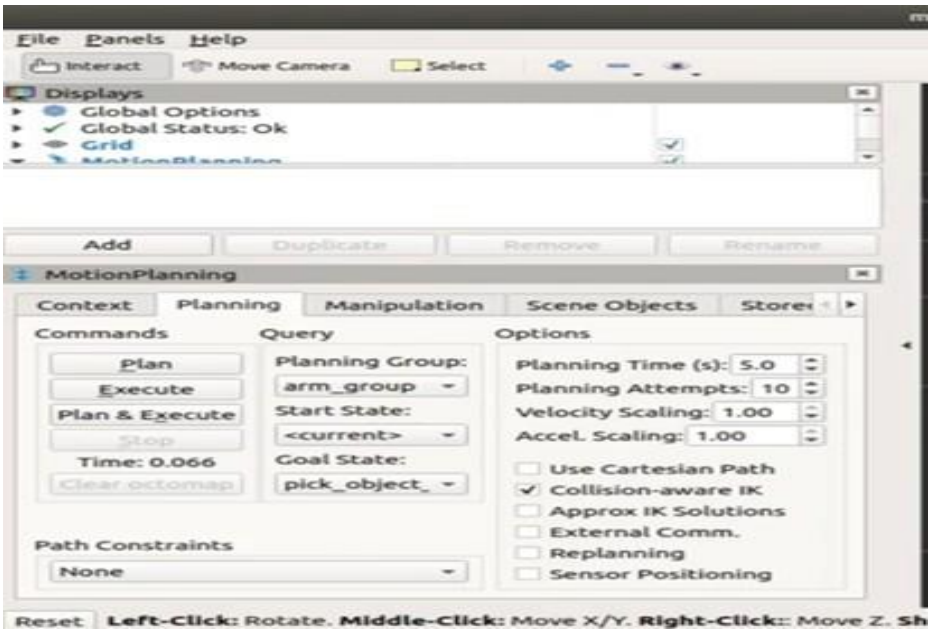

Fig 7: Robotic arm Motion planning in Rviz

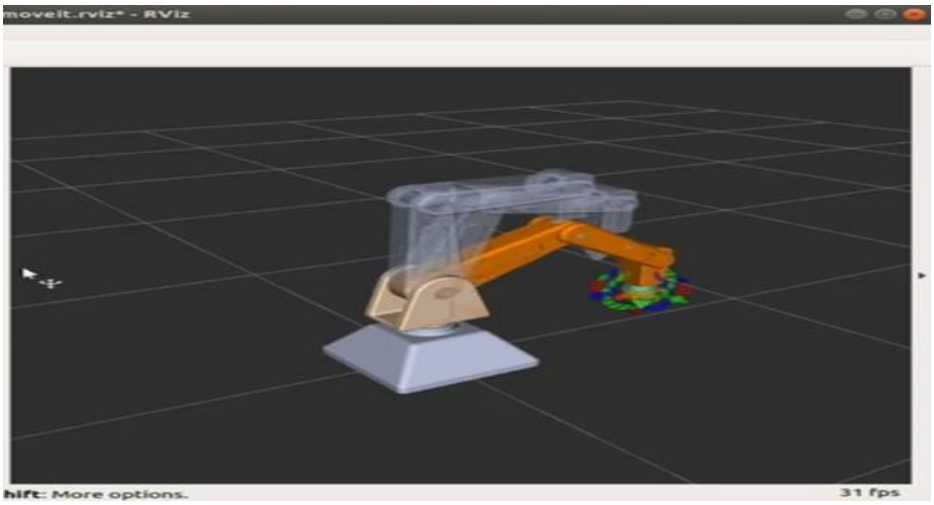

Fig. 8: Robotic arm in Rviz and Gazebo

VII. CONCLUSION

Exporting SOLIDWORKS models to URDF for ROS is an important step in robotics research and development. This integration improves collaboration and accelerates the development lifecycle by providing a standardized platform for simulating, planning, and controlling robotic systems. The methods presented in this article provide researchers and practitioners with an effective method to integrate SOLIDWORKS-designed robot models into ROS, opening the door to further advances in the robotics field. This structured conclusion provides an overview of the article's objectives, methods, findings, and results. implications, summarizing the importance of exporting SOLIDWORKS models to URDF for ROS in robotics research.

REFERENCES

[1.] Guiyun Huang , Yong Li , Jian Cui, Research on Modeling of Cutting Parts Based on Solidworks

[2.] Congguo shi, Weizhen wu, Xun Qiao Jianshe Huang, ” Secondary development of Solidworks Based Parts”

[3.] Nainer, M. Feder and A. Giusti, ”Automatic Generation of Kinematics and Dynamics Model Descriptions for Modular Reconfigurable Robot Manipulators”, in IEEE 17th International Conference on Automation Science and Engineering, 2021, pp. 45-52.

[4.] I.-M. Chen and G. Yang, “Automatic model generation for modular reconfigurable robot

dynamics," J. Dyn. Syst., Meas., Control, vol. 120, pp. 346–352, 1998.

[5.] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Ng, "ROS: an open-source Robot Operating System", in Proc. ICRA Open-Source Softw. Workshop, 2009.

[6.] P. Vyavahare, S. Jayaprakash and K. Bharatia, "Construction of URDF model based on open source robot dog using Gazebo and ROS," 2019 Advances in Science and Engineering Technology International Conferences (ASET), 2019, pp. 1-5, doi: 10.1109/ICASET.2019.8714265.

[7.] Maddalena Fedder a b,Andrea Giusti a,Renato Vidoni b, An approach for automatic generation of the URDF file of modular robots from modules designed using Solid Works

[8.] Rajesh Kannan Megalingam, Raviteja Geesala, Ruthvik Rangaiah Chanda, Nigam Katta ,Multimode Control and Simulation of 6-DOF Robotic Arm in ROS.

[9.] Morgan Quigley, Brian Gerkey , Ken Conley , Josh Faust , Tully Foote , Jeremy Leibs , Eric Berger , Rob Wheeler , Andrew Ng, ROS: an open-source Robot Operating System

[10.] Pratik Vyavahare, Sivaranjani Jayaprakash, Krishna Bharatia,Construction of URDF model based on open source robot dog using Gazebo and ROS.