An Open Source Tele-Operation Application for Arbitrary N-DOF Manipulators

Steven Macenski[1]

***Abstract*— Human operation and tele-operation of manipulators is required in a number of operation, validation, and experimental scenarios. An application, termed the Command Center, for commanding and receiving feedback from an arbitrary manipulator is presented to improve human-robot interaction. The Command Center dynamically updates available operations, feedback, command handling, and disabling automatically for *any* degree of freedom (DOF) manipulator utilizing MoveIt! for it's path planning in the robotic operating system (ROS). The application is capable of joint and Cartesian space moves with variable speed control and MoveIt!'s built-in perception handling and obstacle avoidance. Features for gripper actuation are supported through the Command Center. No additional setup is required to operate under the existing ROS-Moveit! standard architecture. The application facilitates lowering the barrier to entry for robot interaction and provides ROS users with tightly integrated support for its current state of the art for a robot arm. Considerations for uses and comparisons of applications are explored.**

I. INTRODUCTION

Many robotics applications continue to be based on the human control of robotic manipulators in joint space and Cartesian space with supplemental force or torque limitations. Applications such as International Space Station robot arm operations by the National Aeronautics and Space Administration (NASA), handling of materials in nuclear power plants, jogging robotics out of convoluted positions in a lab environment, or remote tele-operation require human operators to handle positioning of a manipulator. Many applications demand human operators due to the risks associated with automation and intelligent robotics in regions like space and nuclear power plants. Human spaceflight and public safety robotics applications, such as presented in the DARPA robotics challenge, will require a human operator for the foreseeable future. [1]

ROS is a powerful communications tool in robotics that has resulted in a breadth of easy-to-use standardized tools for robotics developers. MoveIt! has made autonomous path planning in ROS simple containing the same powerful path planning algorithms from the Open Motion Planning Library (OMPL) with visualization and simulation tools. [2] MoveIt! does not currently support a tele-operation application to specific goal poses for users. The Command Center application presented is a unique extension of the ROS-MoveIt! architecture developed for robotic manipulators with tight native integration. It fills the gap with a standardized application ready for any robot arm in minutes with a pre-generated MoveIt! configuration package, making robot tele-operation easier than ever. The Command Center is a Qt stand-alone application that utilizes MoveIt! services and namespaces to determine the number of axes in the manipulator and configure the application dynamically for any non-zero DOF robot. It is capable of generating plans utilizing all parts of MoveIt!'s architecture including obstacle avoidance when moving the robot using this application. [3]

[1]Department of Aerospace Engineering, University of Illinois at Urbana-Champaign, email: smacens2@illinois.edu.

II. COMPARISON

The Command Center was built with ROS-MoveIt! integration in mind. There are other options that have ROS bindings for their software architecture, however they are generally highly specialized for their specific use cases and arms. One major strength in this tele-operation software is the ability to control many types and configurations of manipulators utilizing the same hardware portability that MoveIt! and ROS have been popularized for.

The simplified interface and portability make it the ideal tele-operation software for new users or those who want to move their ROS enabled robot within hours of installation with commercially supported manipulators. Section III and IV covers the straight-forward process for users with custom hardware without ROS support. The barrier to entry of robotic manipulation and operation is dramatically decreased; a user is capable of installing ROS and MoveIt! then simply run an application to move their robot. Surgical and NASA servicing tele-operation software, shown in Table 1, are highly specialized and cannot be easily modified to work with an arbitrary manipulator due to their separate custom control and planning software. While they offer real-time control, many tasks outside of medicine and contact operations do not require it. [4]-[6]

Robonaut is not highly integrated with ROS. It has roots in ROS for its system architecture but does not entirely utilize the current manipulator software stack the ROS community has and continues to develop. The presented software is tightly integrated with that open source software stack and will continue to develop new features with the continuous integration of the ROS manipulator software stack. [7]

The MoveIt! client that exists in an RVIZ plugin does not allow the user to move to a specific numeric goal pose or receive direct feedback from the robot.

III. ROBOT SETUP

Manipulators must be configured with ROS before they are capable of using the Command Center. Three components are required to configure the robot for use with MoveIt!, the

TABLE I

APPLICATION FEATURE COMPARISON

	Realtime	ROS	Portable	Feedback
Proposed		✓	✓	✓
MoveIt! Plugin		✓	✓	
Tele-Op. Surgery [4]	✓			✓
Robonaut [7]	✓*	✓*		✓
Sat. Servicing Software [5]	✓			✓

* with OROCOS

robot controllers, drivers, action client. These components allow for the robot to receive paths from MoveIt!'s action server and translate them to joint commands to send to the robot controllers.

A. Drivers

Drivers for interacting with the robot controller in Ubuntu are needed and many robots have drivers written in Linux for ROS. The drivers must write to the robot controller the desired positions from the ROS message types to the specific manipulator messages using the manufacturers communications protocols and methods.

The drivers should handle either a streaming or downloading interface to the robots. A streaming driver will send a single waypoint over the connection to the robot at a time. A downloading driver will send a subset or full trajectory to the robot for execution. This is effective for applications which have a re-programmable robot controller to receive multi-waypoint trajectories. The controller is then responsible for timing, execution, and monitoring of each waypoint in the trajectory. [8]

B. Action Server

MoveIt! interfaces to the drivers through an action server. This server receives the joint trajectory messages from MoveIt's action client, breaks down the messages for dispersion to the drivers, and reports robot telemetry from the drivers. It is expected that the action client has an action server started to receive the goals from MoveIt! in a unique namespace.

The action client and server interface must have these five topics implemented;

- Feedback
- Results
- Status
- Goal
- Cancel.

The feedback, results, and status topics are available for triggering other ROS nodes, error handling, trajectory status updates, and trajectory completion messages.The action server should receive a trajectory message from the goal topic. Cancel is used to cancel the entire trajectory or a subset of future points in the trajectory as determined by the execution time or ID of the trajectory waypoint in the sequence. Fig. 1 shows the action server and client interfaces. [9] [10]

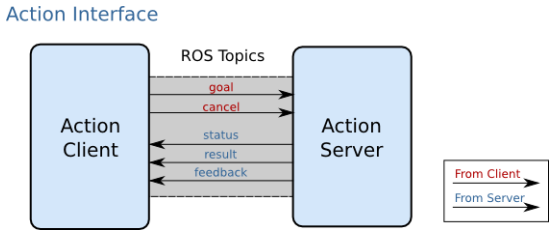

Fig. 1. Depiction of an action interface [9]

IV. MOVEIT! SETUP

MoveIt! requires setup for a robot. The setup assistant was created as a tool to help users generate a MoveIt! package for an arbitrary manipulator. The setup assistant is shown in Fig. 2. Its input is a Unified Robot Description File (URDF) generated from meshes or simple geometry with similar parameters as the robot. The assistant will guide through kinematic chains to build a path planning package. Multiple planning groups may be created for multi-arm or macro-micro systems and work with multiple instances of the Command Center. A separate planning group should be generated for any end-effectors that are to be actuated through MoveIt! and the Command Center.

With a generated package, a handful of additional steps are required to configure the package with the controllers and action server.

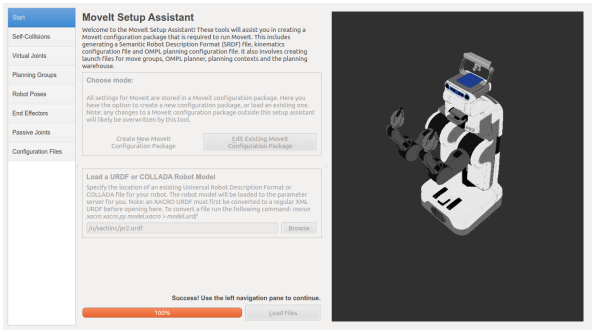

Fig. 2. MoveIt! Setup Assistant GUI [12]

The controller manager must be populated to launch the appropriate launch and configuration files. When perception devices are used, a sensor manager file must also be populated. [12]

Additionally, the joint names and controllers configuration files must be either added or filled out corresponding to the robot specifics in the action server or controller configuration. A MoveIt! planning and execution file should be generated. This file connects to the robot and launch the required drivers, controllers, and MoveIt! programs.

The connection between MoveIt!'s action client and an action server for the robot is through the controllers.yaml configuration file. The `action_ns` field should correspond to the ROS topic namespace that the controllers will be started in.

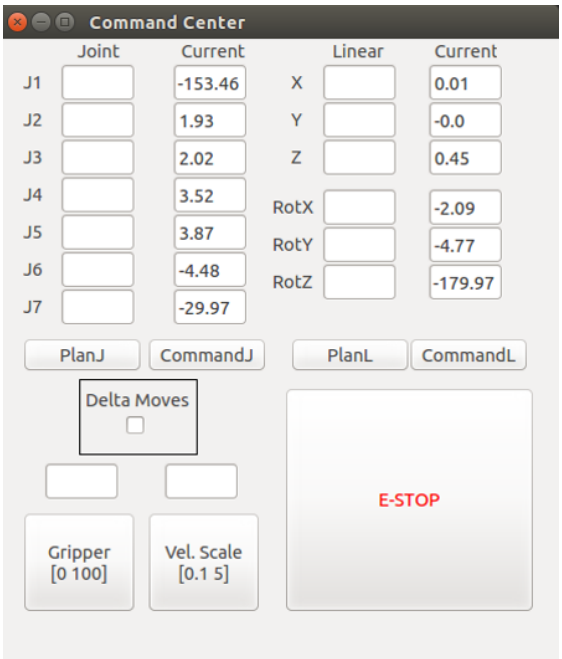

Fig. 3. Command Center GUI

V. Application

After a proper ROS and MoveIt! configuration, the Command Center is very simple to setup and operate. The power of this application is in its portability for any robot, tight native integration in ROS capable of updates with new MoveIt! functionality such as force control, and ease for novice robot operators.

A. Capabilities

The application will allow the user to complete speed variable joint and Cartesian space moves on an arbitrary robot while respecting any planning scene or collision objects added or identified by the perception system. Safety features have been implemented to stop robot motion and operate gripper-style or symmetrically actuated end-effectors.

The GUI will automatically update the number of joints available for planning, commanding, and receiving feedback from based on the manipulator group joints. The gripper execution button and field will only appear when a gripper-style group is available.

1) Joint and Cartesian Commands:

The Command Center, seen in Fig. 3, is able to complete basic joint and Cartesian space moves using MoveIt!'s functions `plan` and `execute`. All values displayed to the operator are in degrees. [1]

The forward and inverse kinematics are determined from the URDF of the manipulator loaded into MoveIt! on start-up via the parameter server. The planning groups in MoveIt! are used to determine the number of active joints in the robot and whether an end-effector is present for use.

A current pose in Cartesian coordinates is determined relative to the manipulator's base frame to the manipulator's distal joint. A virtual end-effector is the preferred method of planning for the end-effector pose. The rotation angles in the linear commanding module are base frame fixed axis angles.

The "Current" fields for joints and Cartesian pose, labeled by "Linear", display the feedback from the robot controller on the current robot position to aid in planning and goal pose determination.

A call to the planning buttons for joint space "PlanJ" and Cartesian space "PlanL" will plan the entered goal position through MoveIt!'s `plan` function. After planning, the proposed plan can be visualized in the `move_group`'s RVIZ window opened upon initialization of MoveIt!.

Fig. 4 displays an instantaneous view of operations that a user would see while visualizing a path in Cartesian space. Fig. 5 shows a time lapse of a simulated path in joint space.

The Commanding buttons, "CommandJ" and "CommandL", will send the plans to the the action server, in joint or Cartesian space, after speed scaling. The robot can be seen moving in either Gazebo or with hardware depending on the experimental setup.

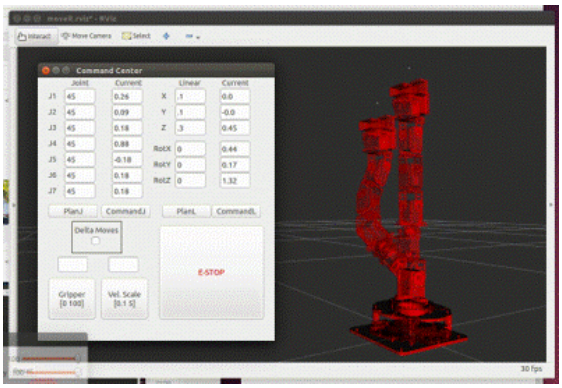

Fig. 4. Demonstration of Application and RVIZ

2) Delta-Position Moves:

A common move for many robot operators is to change the pose of the robot in joint or Cartesian space by a small amount. The delta moves check box gives the operator the freedom to input "change in" values from the current position for either types of moves. When not checked, the moves are absolute relative to base frame.

The same planning and execution buttons are used and the output plans can be again visualized or executed in RVIZ, Gazebo, or lab environments. The velocity scaling control is also available for delta-position moves.

3) Speed Control:

The ability to change the speed of a manipulator is critical for safety of people in proximity or when operating under uncertainty.

The velocity scaling field allows an operator to scale the speed of execution by an specified amount. After a submission to execute a trajectory, the plan is conditioned with the velocity scale factor for the final execution in Gazebo or physical world.

MoveIt! does not directly implement any form of dynamic velocity scaling. This feature was created in the application

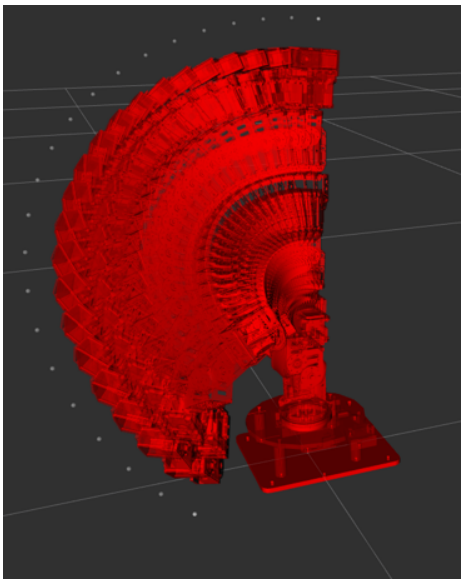

Fig. 5. RVIZ plan propagation over time

by taking the planned path and scaling the velocities, accelerations, and `time_from_start` duration fields for the plan outputted directly from MoveIt!'s OMPL plan function call. The default value is 1, corresponding to the speed automatically used in MoveIt!'s planning process.

RVIZ will not reflect the speed scaling during it's trajectory simulation.

4) Emergency Disable:

The emergency disable feature allows the user to stop any trajectory being executed on the robot. This safety feature facilities necessary precautions for robot operations especially when human operators are involved.

The E-Stop button will send a message to the action server to stop all waypoints from being executed. The action server `/cancel` topic takes a message to cancel the execution of way points after the indicated time stamp or way point identification number. In the absence of a time stamp or trajectory point identifier, the action server must cancel all pending waypoints in a stored trajectory on-board the controller or being fed to the controller over a streaming interface. [9]

The E-Stop sends an immediate actionlib_msgs/GoalID message with empty time stamp and point identifier to cancel all pending points. The E-Stop relies on a properly implemented action server. [9]

5) End-Effector Operations:

The end-effector can be operated from the fields and buttons labeled by "gripper". This icon and field only appear when a end-effector planning group is available. The gripper actuation calls the planning group's actuators to move the specified amount by the GUI symmetrically. When a single actuator gripper is present, the application will simply actuate the single joint. If multiple exist, they will all actuate the specified amount.

For more complex or unconventional end-effectors such as hands, planning groups in MoveIt! should be separately generated and controlled from the application rather than use of the gripper command structure.

6) With Obstacles and Perception:

MoveIt! is able to handle obstacle avoidance from obstacles declared in the planning scene and the output of perception sensors. [1]

The application utilizes the planning features of MoveIt! in combination with the action client and respects all constraints imposed by the planning scene and occupancy grids generated from the sensor data. [1]

This powerful result gives some autonomous capabilities for manual operation of robot manipulators, yielding safer use and correcting for user error.

B. Usage

Using the application is simple, run the ROS node from a terminal or launch file. RVIZ and Gazebo will function the same as with MoveIt! alone, planning in the application will graphically display the path in RVIZ and commanding it will move the simulated robot in Gazebo. Physical robots can be replace Gazebo when drivers, action servers, and robot controllers have been configured.

1) Launch File:

An example launch file is shown below.

```
<launch>
  <node
    name="robot_ops_application"
    pkg="moveit_operations_application"
    type="robot_ops_application.py"
    args="[controller name]
     [manipulator planning group]
     [opt. end-effector planning group]"
  />
</launch>
```

The arguments passed to the script contain the necessary static information to connect the application to MoveIt!. The controller name is the namespace of the robot controllers. The manipulation and optional end-effector planning groups are the names of the groups in MoveIt! configuration that correspond to the planning groups desired to operate.

Multiple instances of the application can be open at the same time for multi-arm or complex end-effector applications.

2) RVIZ Compatibility:

RVIZ will continue to display the paths planned and current state of the robot as ordinarily expected from MoveIt!. The application contains a MoveIt! wrapper for the planning process to use the same pipeline for autonomous use cases.

RVIZ may be used in the human operations context as a path verification tool to ensure operator comfort of the move before execution. Additional characteristics or sensor feeds may be visualized in RVIZ within the same window simultaneously.

3) Gazebo or hardware robot:

Gazebo will also execute the paths commanded and return state information about the robot as expected. The application contains a MoveIt! wrapper for the execution process to use the same pipeline used in autonomous simulation use cases.

Gazebo may be used to replace hardware for simulation with sensor feedback and environmental testing. Fig. 6 displays the application's planning process in RVIZ with before and after execution in Gazebo.

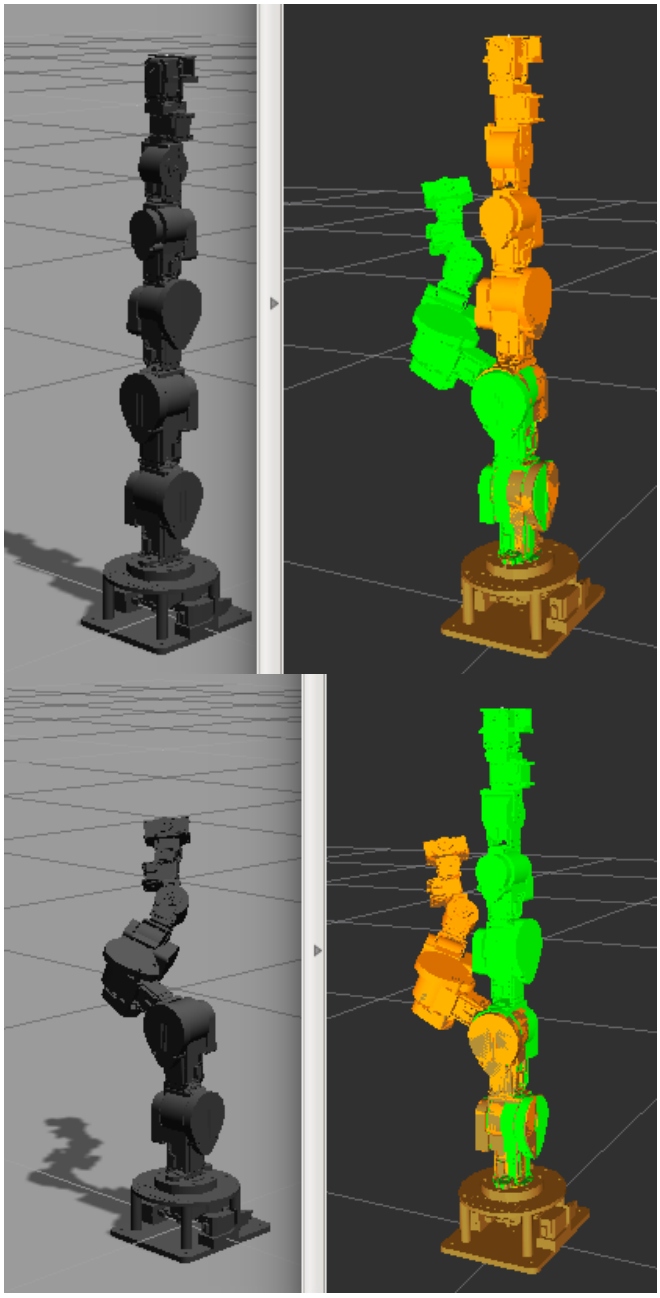

Fig. 6. RVIZ (right) displays plan with Gazebo (left) execution

VI. Use Cases

The application was created to service a number of groups in need of an GUI for visualizing and simulating user commands through a front-end application in MoveIt!.

A. Novice Users

Many new robotics personnel are attracted to ROS because of its simplicity (as compared to the alternatives), implemented packages, and strong developer support. These users often require manual tele-operation capabilities for their applications and MoveIt! abstracts the motion planning problem from them.

Many research and industrial robots on the market have configuration packages for the robots pre-generated for use. The application would allow a new operator to tele-operate their hardware manipulator or simulation in under an hour. The only configuration required is editing a single launch file for the application. This very effectively lowers the barrier to entry of new robotics users with elementary tele-operation needs with any manipulator.

B. Space Operations

International Space Station, satellite servicing, and larger space manipulation projects conducted by NASA and international partners have relied on human operators to interface with the robots remotely. Particularly in human spaceflight and high value assets in orbit, the automation of robotic manipulation on-orbit is highly unlikely to be adopted in the near term. [3], [6], [14]

The International Space Station (ISS) robotics workstation aboard the ISS relies on a custom application for operating their robot in orbit. These applications have more capabilities than currently supported by MoveIt!, open source control, and planning communities. As the capabilities of MoveIt! and ROS grow, in conjunction with the creation of more capable flight processors, the barrier to space applications shrinks. [6] [14]

While it is not expected ROS becomes the primary architecture in the five year outlook, ROS integration with other software architectures are in preliminary considerations. When MoveIt! grows to include force control and other state of the art algorithms, the interface, with minor updates, will become a serious option in trade studies. The ability to train operators on a single software system that extends to a larger family of space robots aboard multiple spacecraft without significant development costs is appealing. This application is tightly integrated into ROS and through the development of MoveIt!, the application will grow with it. [13] [14]

Currently, many robot operators in several locations exist within NASA alone. Each flight program utilizes similar manipulators and has separate applications for interacting with their robots with similar base requirements. With a unified platform, there is potential for a smaller number of human operators capable of operating a myriad of manipulators on-orbit. Standardization within the space robot community

homologous with the open source community will strengthen the capabilities in both.

C. Jogging Robots in a Lab Environment

A nominal application for the application is for jogging manipulators in research labs. A frequent need is to jog a manipulator to position to change tools or move the robot to the home position for resetting an experiment.

Research labs tend to have more relaxed requirements on the storing objects in the robot workspace. For robots already being operated with MoveIt!, it is a simple transition and does not require manual scripting of desired positions to move an industrial robot to a tool change pose. Safety is critical for robot motion, thusly the application's user position feedback from the robot and motion visualization could potentially save assets from damage.

VII. CONCLUSIONS

The Command Center presented is simple to use and powerful. It allows the user to command robot motion with ROS accessing MoveIt!'s native collision avoidance properties ensuring asset safety. The application lowers the barrier to entry for new users and experts, is capable of easily interfacing with any manipulator, and has tight integration with ROS and MoveIt!. It is simple to update with MoveIt! development to include advanced capabilities for continuous integration with future capabilities. For future work, the Command Center will be added as a RVIZ plugin for a single window simulation and command application launching directly from MoveIt!.

APPENDIX

Project can be found at the following link, github.com/stevemacenski/CommandCenter. The project is open source under the MIT License.

REFERENCES

[1] "MoveIt! Concepts," in MoveIt ROS, 2016.
[2] S. Chitta, I. Sucan, and S. Cousins, "MoveIt!," in IEEE Robotics & Automation Magazine, 2012.
[3] N. Curry, "International Space Station Robotic Systems Operations A Human Factors Perspective,", 2004.
[4] Tobergte, Konietschke, Hirzinger, "Planning and Control of a Teleoperation System for Research in Minimally Invasive Robotic Surgery," in IEEE International Conference on Robotics and Automation, 2009.
[5] Washington, Alessandro, "NASA tests new technologies for robotic refueling," Feb 2014.
[6] "On-Orbit Satellite Servicing Study Project Report," Satellite Servicing Projects Division, NASA Goddard Spaceflight Center, Oct. 2010.
[7] Badger, Goodling, Ensley, Hambuchen, Thackston, "ROS in Space: A Case Study on Robonaut 2".
[8] "Working with ROS-Industrial Robot Support Packages," in ROS Wiki, 2016.
[9] "Actionlib Detailed Description," in ROS Wiki, 2013.
[10] "Create a MoveIt Package for an Industrial Robot," in ROS Wiki, 2015.
[11] "ROS Concept," in ROS Wiki, 2014.
[12] "MoveIt! Setup Assistant Tutorial," in MoveIt!, 2013.
[13] D. Colman, "MoveIt! Strengths, Weaknesses, and Developer Insights," in ROSCon, 2015.
[14] S. Loff, "Robotics workstation in the international space stations Cupola," [Online]. Available: `nasa.gov`.