Construction of URDF model based on open source robot dog using Gazebo and ROS

Pratik Vyavahare
B.Tech Mechatronics
Manipal University
Dubai, UAE
pratik_vyavahare@mail.com

Sivaranjani Jayaprakash
Asst. Professor
Manipal University
Dubai, UAE
sivaranjani.jayaprakash
@manipaldubai.com

Krishna Bharatia
B.Tech Mechatronics
Manipal University
Dubai, UAE
krishbharatia@yahoo.com

***Abstract*— In this paper, the mechanical design and electronics used in the OpenDog are explained along with the process of creating a custom URDF file. The custom URDF based on the available OpenDog model is needed to port it to ROS and perform path planning algorithms on the same. This is carried out by further creating separate packages and launch files for visualization and simulation of the model. The simulations of the movements are performed in Gazebo and visualized in Rviz.**

1. Introduction

There are several different types of robotic vehicles under development for varying applications. They have different characteristics, some are at very tiny micro level while some are very large human sized. Regardless of size, they can perform tasks which aren't possible by humans. They can lift objects several times their weight or move at incredible speeds and in some cases also have provisions for camouflage.

Based on previous studies, vehicles consisting of legs for movement have proven to be potentially better than vehicles consisting of wheels for movement. Due to the unique design of legs, it enables the robot to perform complex motion and enable them to be apt for environment unsuitable for humans and wheeled robots. This flexibility in terms of movement is achieved due to the high amount of degrees of freedom when performing motion.

Motion planning is the process of planning the path of system so that the configuration space is obtained in the end. This configuration space is calculated on the basis of several factors such as dynamic constraints on the model, environmental conditions in the simulation etc. In order to perform this motion planning, we first need to construct a dynamic model based on a particular CAD 3-d model. This can be done by combining several methods ranging in terms of complexity. I decided to use Solidworks in conjunction with URDF exporting plugin to generate a robot description and later using the same for analysis.

In this paper, we have designed a joint control system for the open source robot 'OpenDog' based on its available CAD model. The model was converted into relevant format for followed by constructing joints and defining the parent and child links. Then respective inertia and forces were applied to the model. Two models were constructed for the same robot, the display model for visual demonstration purposes in Rviz and a dynamic model to perform simulations in gazebo. Hence this allowed to check the real time ragdoll physics of the model based on the dynamic collision model.

2. OpenDog

OpenDog is an open source robot designed by James Bruton, its basis are similar to the Boston Dynamics dog. The robot is designed to resemble the visuals of a dog while containing functionality of a mobile robot which can hence be used ahead in order to perform several tasks from home surveillance to disaster rescues based on the attachments provided on board the robot.

Due to the presence of 3 DOF on each leg, it is capable of performing complex motion enabling it to be potentially be capable of search and rescue, interplanetary exploration and exploration of volcanoes and dangerous areas where humans are incapable of going.

Therefore the robot is primarily made of a combination of plastic and aluminium, runs using ODrives with Arduinos as the brains of the whole operation. The CAD and code for the project is licensed under GPL3, making it truly open source.

3. *Model Explanation*

The OpenDog has a design very similar to the Boston Dynamics dog robot with a few optimizations for easier calculations. The first prototype was constructed completely out of 3-d printed parts but it led to quick wear and tear in sections with high pressure. Therefore in the next version, majority of the parts were casted in aluminium and only the least stressed parts were kept as plastic.

The legs of the robot are constructed in a 90° triangular fashion in order to facilitate easier calculation of reverse kinematics of the model. The top section of the robot is a separate attachment formed from 3d printed plastic in order to store all the electronics and additional sensors in a systematic manner.

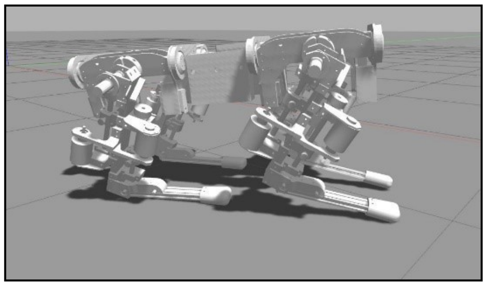


Fig. 1. Side view of OpenDog model

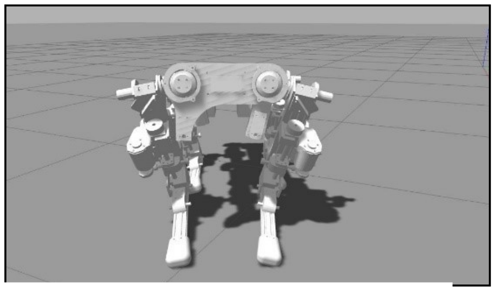


Fig. 2. Front view of OpenDog

The robot assembly consists of a combination of parts, the main base of the robot is the section to which all the legs are connected, but before connecting to the upper section of legs, there are hip joints present which offer that extra degree of freedom motion to each leg. There is an upper leg section attached to these hip joints and lower leg sections are added to these upper leg sections. All of the joints present in the motion are revolute joints.

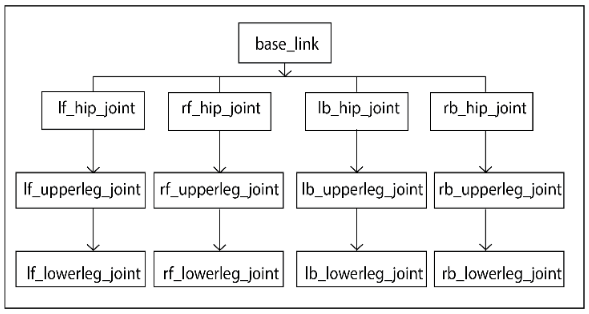


Fig. 3. Flowchart of parts of OpenDog

The same setup is repeated across all 4 legs of the robot. The high stress areas are made from aluminium and the ends of the legs are made using NingaFlex flexible 3d printer filament in order to avoid rigid steps.

4. *Electronics*

The robot design consists of a custom made caged design which is used to hold all of the electronics in a systematic fashion. This cage holds 3 Arduino Mega 2650s and 6 ODrives. The motors used for motion are all 149KV 6374 motors. One of the Arduinos is the master while the other 2 are slaves which follow all commands relayed by the master.

Each of the motors have an encoder attached to them in order to provide accurate feedback and data logging for programming based on steps. There are currently no sensors present for image processing but will be added in the future. Therefore only simulations are performed in order to finalize the type of sensors to be used along with the robot for accurate image processing. Upon start up, the motors have to be calibrated one after the other to take note of all the extremes and then the movement can be initiated. The motion of the leg can sometimes cause jitters due to several factors, hence a smoothing filter is applied to allow smoother translations.

5. *Functions and potential uses*

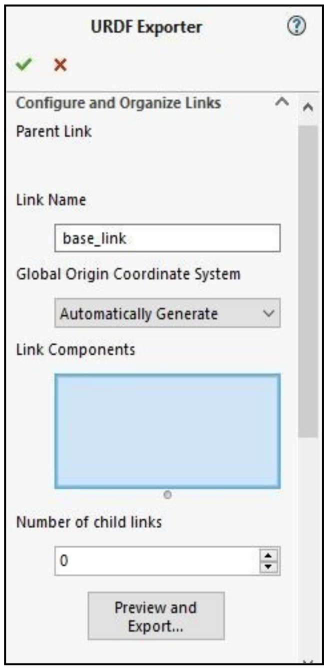
The robot consists of ability to perform 3 dof motion, consisting of pitch, roll and yaw, needed for optimal movement of the robot. It's core application lies in home security and surveillance, if the robot consists of stereo camera setup, then the blind spots in the surveillance system can be completely eliminated along with providing several levels of information based on analysis.

It can be installed with alarms and GSM facilities to inform the tenant of potential mishap. The possibilities can also be extended to disaster rescues by adding additional sensors and stabilization to enable robot to maneuver over difficult terrain and detect people trapped using heat sensors.

6. *Motion Analysis*

After the stress tests, the motion of the robot was to be simulated. But the primary requirement is for the model to be converted into a format compatible with Robotic Operating System. One of the most popular file extensions used to describe a model is URDF (Universal robot description format) form. No method to directly export the model into URDF currently exists but there is a community developed plugin which can be used to convert Solidworks models into URDF by following few simple steps.

For every joint which isn't fixed we have to specify a transmission, which is used to inform Gazebo what to do with the particular joint. Hence this will enable proper visualization in Rviz and Gazebo. We have to then position the controllers in the appropriate positons. Hence we have to consecutively add controllers for any additional joints. Hence we obtain a URDF file which will display all appropriate information when opened in ROS. The process to open a URDF file in ROS involves first placing the URDF file in the main catkin_ws folder in the system followed by running command to execute the file in Gazebo.


URDF Exporter

Configure and Organize Links

Parent Link

Link Name

base_link

Global Origin Coordinate System

Automatically Generate

Link Components

Number of child links

0

Preview and Export...

Fig. 4. URDF Exporter menu

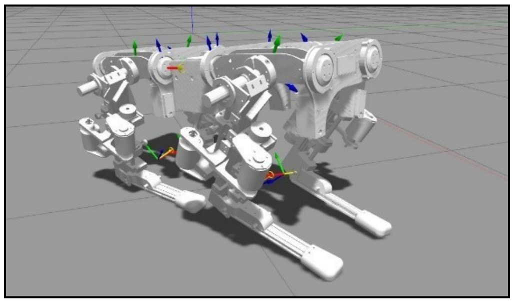


Fig. 5. Side view of OpenDog model with joint controls

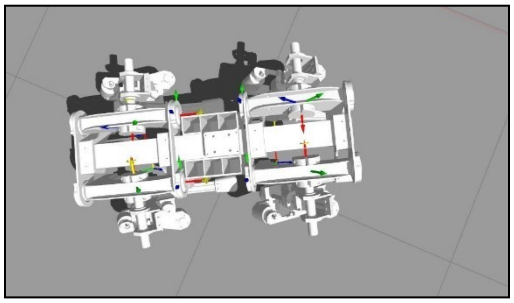


Fig. 6. Top view of OpenDog model with joint controls

Therefore the plugin exports in URDF by feeding it all the links and their relation to each other along with the type of joint they are connected with each other. The parent-child linkage is very important along with ensuring the co-ordinate system is set correctly. Hence we obtain the URDF model of the robot.

Therefore we port the URDF obtained from the conversion and open it in ROS. In order for ROS to completely understand the URDF file, we need to perform few steps, we have to start by attaching the gazeebo plugin in the code. Followed by spawning controllers by publishing based on the YAML file.

Therefore based on the export, we observe controllable joints at end of every link in order to accurately move the robot while observing its end effector angles. This doesn't provide very accurate movements as the center of each mass isn't properly defined yet. Therefore after applying the appropriate center of mass of each link, the model appears as follows.

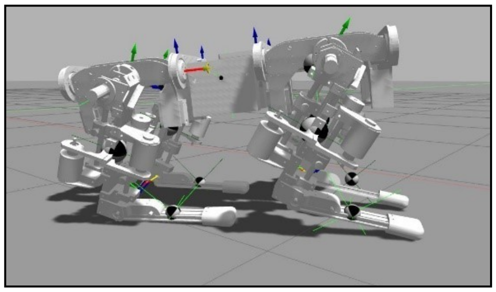


Fig. 7. Side view of OpenDog model with center of mass

After assigning the center of masses, we have to fix the inertia settings of the model. In its default state, it covers the entire model and even some of the surroundings hence there is a high requirement of modifying the inertia in order to only cover certain areas of the model and not the entire model, we edit the model file to achieve this.

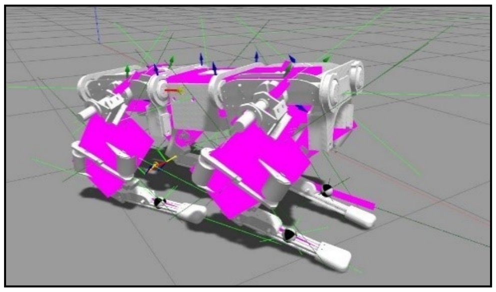
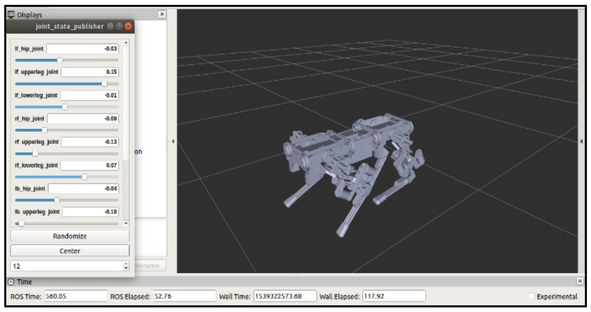


Fig. 8. Side view of OpenDog model with inertias configured

Therefore in this manner we obtain the dynamic model of the previously static model. This model can be used to perform simulations and test with collisions in different environment states. We created a respective launch file for this type of dynamic model to be able to be executed in gazebo. In order to further automate this control of different links, we created a ROS joint control launch file which works in parallel with the gazebo launch file previously constructed and hence the joints can be easily controlled once the Rostopic is correctly published to the relevant joint values control file.

In order to effectively visualize the model without the presence of any inertia or environment based factors, a simple visual display launch file was created to be opened in Rviz. First Roscore requires to be initialized followed by launching the display launch file which consists of a custom coded GUI for separate control of each joint using a simple slider along with adding the option to randomize the pose within certain parameters in order to visualize different poses and utilize the angle information to accurately calculate the required kinematics of the model.



lf_hip_joint -0.03
lf_upperleg_joint 0.15
lf_lowerleg_joint -0.01
rf_hip_joint -0.09
rf_upperleg_joint -0.13
rf_lowerleg_joint 0.07
lb_hip_joint -0.04
lb_upperleg_joint -0.19
12

ROS Time: 560.05 ROS Elapsed: 52.76 Wall Time: 1539322573.68 Wall Elapsed: 117.92

Fig. 9. Randomly assigned pose in RVIZ-1

Therefore we can use the gazebo launch file or the Rviz based display launch file based on whether we require to perform simulations or visualize data of different joint angles and their kinematics change. Therefore the next step in the project is to be able to perform a series of continuous steps for the robot, followed by eventually implementing optimized path planning algorithms in order to prepare the model for real life implementation.

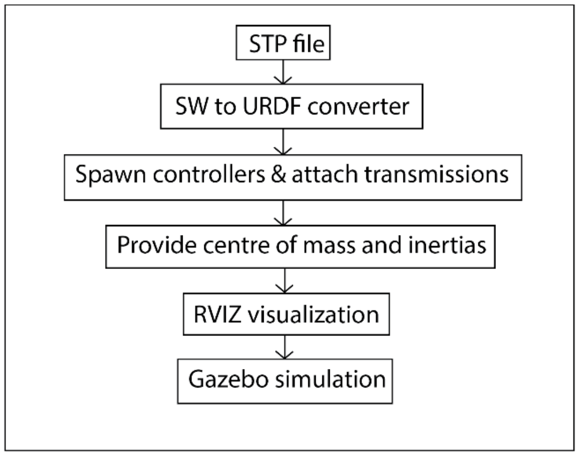

STP file
SW to URDF converter
Spawn controllers & attach transmissions
Provide centre of mass and inertias
RVIZ visualization
Gazebo simulation

Fig. 10. Flowchart of the model conversion process

7. *Future Work*

The future work consists of designing algorithms for obstacle detection and advanced path planning using advanced image processing sensors. There is also more research required in order to consider building a dog with very specific functionality such as high speed or high stability customized based on nature of applications in order to fully utilize the robots capabilities.

There can be several significant design changes and optimizations in the model to vary it for different purposes along with providing artificial intelligence in the robot to enable it to perform its own smart decisions while also learning over time and evolving its motion based on past experiences. The whole system currently works on Arduino Mega, but there are plans in the future to convert the brains of the whole system to a micro-computer which will result in much more efficient and quick computations.

8. *Conclusion*

Therefore complete simulation ready model is constructed using the CAD model as base. Based on pre-set restrictions over movements, dynamic joints are created and appropriate controllers are assigned for the same. Also, two separate launch files are created based on the URDF for the purpose of visualization in RVIZ and dynamic simulations in Gazebo. A custom UI is constructed to control joints individually based on publishing the appropriate files. A control launch file is created to control all joint motions together.

9. *References*

[1] David Wooden, Matthew Malchano, Kevin Blankespoor, Andrew Howard Alfred Rizzi and Marc Raibert, "Autonomous Navigation for BigDog" IEEE International Conference on Robotics and Automation, Anchorage, Alaska, USA (Conference paper)

[2] Kris Hauser, Timothy Bretl, Jean-Claude Latombe, Kensuke Harada and Brian Wilcox, "Motion planning for legged robotson varied terrain" The International Journal of Robotics Research, 2009 (Journal paper)

[3] Joel Chestnutt, "Navigation Planning for Legged Robots" Carnegie Mellon University. (Thesis submission)

[4] Thomas Linner, Alaguraj Shrikathiresan, Maxim Vetrenko and Thomas Bock, "Modeling and Operating Robotic Environments Using Gazebo/ROS" 28th International Symposium on Automation and Robotics in Construction. (Symposium paper)

[5] Dr. Asad Yousuf, Mr. William Lehman, Dr. Mohamad A. Mustafa and Dr. Mir M Hayder, "Introducing Kinematics with Robot Operating System (ROS)" 122nd ASEE Annual Conference & Exposition, Seattle, WA, USA (Conference paper)

[6] Kenta Takaya, Toshinori Asai, Valeri Kroumov and Florentin Smarandache, "Simulation Environment for Mobile Robots Testing Using ROS and Gazebo" 20th International Conference on System Theory, Control and Computing (ICSTCC), 2016, Sinaia, Romania (Conference paper)

[7] Ilya Shimchik, Artur Sagitov, Ilya Afanasyev and Fumitoshi Matsuno, "Golf cart prototype development and navigation simulation using ROS and Gazebo" International Conference on Mechanical, System and Control Engineering (ICMSC), Moscow, Russia (Conference paper)

[8] Wei Qian, Zeyang Xia, Yangzhou Gan, Yangchao Guo, Shaokui Weng, Hao Deng and Ying Hu "Manipulation task simulation using ROS and Gazebo" IEEE International Conference on Robotics and Biomimetics (Conference paper)

[9] Koenig Nathan, and Andrew Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator." in Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference, vol. 3, pp. 2149-2154. IEEE, 2004

[10] Quigley et al., "ROS: an open-source Robot Operating System." in ICRA workshop on open source software, 2009. (Workshop showcase)

[11] Ilya Afanasyev, Artur Sagitov and Evgeni Magid., "ROS-Based SLAM for a Gazebo-Simulated Mobile Robot in Image-Based 3D Model of Indoor Environment", International Conference on Advanced Concepts for Intelligent Vision Systems, 2015 (Conference paper)

[12] SW2URDF v1.3 software installer (2018, November 11th). Retrieved from wiki.ros.org/sw_urdf_exporter

[13] OpenDog 3-d CAD model (2018, July 6th). Retrieved from xrobots.co.uk/open-dog-the-open-source-robot/