Learning Advanced Robotics with TIAGo and its ROS Online Tutorials

Jordi Pages, Luca Marchionni, Francesco Ferro

PAL Robotics S.L.
c/ Pujades 77-79, 4-4, 08005 Barcelona, Spain
jordi.pages@pal-robotics.com, luca.marchionni@pal-robotics.com,
francesco.ferro@pal-robotics.com

Abstract. Learning Robotics in these recent years has become more easy thanks to ROS and its worldwide success [1]. ROS has managed to reduce the learning curve for newcomers as results can be obtained even before going into the theoretical basis like differential steering systems, forward and inverse kinematics and motion planning. This paper presents the comprehensive set of on-line tutorials featuring TIAGo, the mobile manipulator of PAL Robotics, which use Gazebo simulator and ROS to take a widespread audience to a trip to discover in a practical way a broad range of areas like control, motion planning, mapping and navigation and 2D/3D perception.

1 Introduction

Dynamic simulations of different robots are very common nowadays. Accurate simulation models provide an easy way for rapid prototyping and validation of robotic tasks. Focusing in ROS based simulations, the abstraction layer provided by frameworks like **ros_control** [1] [2], ensures that what you get in simulation is what you will get in the real robot, at least in terms of interfaces. Furthermore, working in simulation is very important when learning robotics, as programming a real robot may be dangerous for the robot itself, the environment or people around. Simulation then is an essential tool for learning robotics.

PAL Robotics has published a comprehensive set of on-line tutorials based on its mobile manipulator TIAGo [2]. The tutorials are organized in different blocks which are depicted in Figure 1. The first section explains in detail how to get a computer ready to follow the tutorials presented afterwards. The installation instructions for an Ubuntu computer and a ROS distribution are explained along with all the required ROS packages from PAL Robotics in order to have the dynamic TIAGo's simulation in Gazebo up and running.

All the tutorials are based on spawning the model of TIAGo in a given simulated world in order to perform a specific task. The tutorials are structured in a way that provides the basic ROS instructions in order to have the simulation

[1] http://wiki.ros.org/ros_control
[2] http://tiago.pal-robotics.com

running, how to run specific ROS nodes performing the target task and also providing some theoretical background of the robotic task when needed.

The first block of tutorials addresses basic control aspects of the robot. Afterwards tutorials about laser-based mapping, localization and autonomous navigation are presented. The next block teaches different ways to perform motion planning with TIAGo including a pick and place demo. Right after a set of tutorials showing different computer vision applications are included. Then, tutorials about processing point clouds show basic perception tasks based on 3D data obtained with the depth camera of the robot.

2 Control

These tutorials aim to teach how to command the different Degrees of Freedom (DoF) of TIAGo.

2.1 Differential Drive Base

The first two tutorials in this block show how to control the differential drive base of TIAGo in order to have the robot moving in the ground plane. The main teaching of these tutorials is that a differential drive system can perform a Twist [3], i.e. the composition of a linear velocity along the axis of the robot pointing forward and a rotation speed about the center of the robot base.

One of the tutorials instruct on how to send the appropriate message. For example, the message to have TIAGo moving at 0.5 m/s forward and at the same time rotating at 0.2 rad/s about its vertical axis can be seen in the following command line instruction:

```
rostopic pub /mobile_base_controller/cmd_vel \
geometry_msgs/Twist "linear:
  x: 0.5
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 0.2" -r 3
```

2.2 Upper body joints

The goal of these tutorials is to provide the reader with knowledge about the typical controllers in ROS based robots to control them in joint space and what can be achieved with these basic controllers.

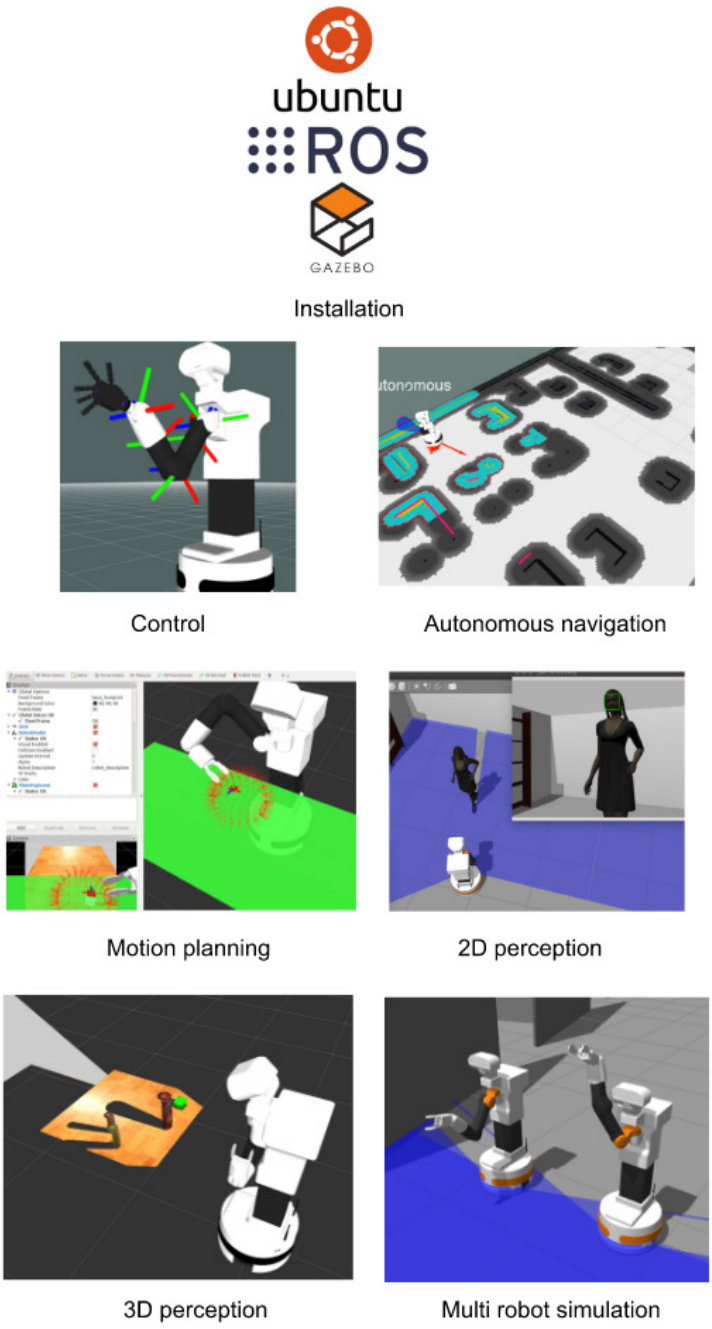

Fig. 1. Overview of the on-line tutorials of TIAGo

Joint Trajectory Controllers Two tutorials are provided so that the reader can have a grasp of the different ROS Joint Trajectory Controllers [3] defined in the upper body of TIAGo each one controlling the following groups of joints:

- **arm_torso**: group composed of the prismatic joint of the torso and the 7 joints of the arm
- **gripper** or **hand**: groups of the 2 joints of the gripper or the 3 actuated joints of the humanoid hand of TIAGo
- **head**: the 2 joints of the head composing the pan-tilt mechanism

The tutorials provide links to the official ROS documentation explaining how a Joint Trajectory Controller is useful to execute joint-space trajectories defined by a set of waypoints so that the trajectory results from interpolation using different strategies. The use of TIAGo's joint trajectory controllers is exemplified with a simple C++ code that the user can easily modify and produce complex motions with the robot in simulation.

This tutorial can be used when teaching robotics in order to quickly validate forward kinematics computations of the arm given a joint space configuration by comparing the pose of the end-effector obtained in simulation. For example, in order to obtain the cartesian coordinates of TIAGo's arm tip frame the following ROS instruction can be used

```
rosrun tf tf_echo /torso_lift_link /arm_1_link
```

which will print the transformation from the arm tip link, i.e. arm_7_link, with respect its parent link, i.e. torso_lift_link, which is presented in the following form

```
- Translation: [0.155, 0.014, -0.151]
- Rotation: in Quaternion [0.000, 0.000, 0.020, 1.000]
            in RPY (radian) [0.000, -0.000, 0.039]
            in RPY (degree) [0.000, -0.000, 2.242]
```

This is also a good point to introduce the ROS Transform Library [4] to the students and explain the two rotation representations provided, i.e. the Quaternion based and the Roll-Pitch-Yaw representation.

Head control This tutorial provides an example of a more sophisticated controller that can be build on top of the joint-space trajectory controller of the 2 joints of the head. This new controller is the Head Action controller [5]. The goal of this controller is to have the robot's head pointing to a given direction, i.e. looking at an specific cartesian point. Therefore, the input of the controller is not in joint-space but in cartesian space. The source code of the controller,

[3] http://wiki.ros.org/joint_trajectory_controller
[4] http://wiki.ros.org/tf2
[5] http://wiki.ros.org/head_action

available in [6], is also a good example on how to implement inverse kinematics using the Kinematics and Dynamics Library (KDL) [7].

The tutorial is based on a simple ROS node implemented in C++ which also shows how to use the actionlib interface of ROS [8]. The C++ example opens a window with the RGB image providing from TIAGo's head camera so that the user can click on any pixel and the robot in simulation then moves the head so that it looks at the given direction, see Figure 2.

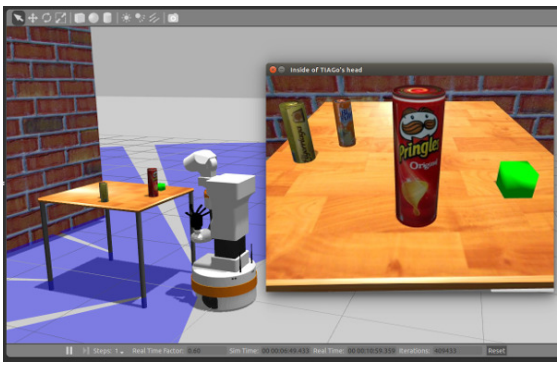

Fig. 2. Head control of TIAGo defining a sight direction

The C++ example also shows how to make use of the intrinsic parameters of the camera in order to compute a cartesian point in the line of sight defined by the pixel clicked by the user.

Playing back pre-defined motions The last tutorial in the Control block explains how to store in a yaml file upper body motions defined in joint space so that they can be played back at any time with the Play Motion library in ROS [9]. The tutorial also explains how to change the velocity of the motions by varying the time given to reach each waypoint of the trajectory.

3 Autonomous Navigation

Two tutorials are presented in order that lead to autonomous navigation of TIAGo in simulation. The first tutorial is devoted to explain how to generate a map with the laser range-finder of the robot. This tutorial can be used as demonstration of Simultaneous Localization and Mapping (SLAM) by using the gmapping algorithm wrapped in ROS [10], see Figure 3.

[6] https://github.com/pal-robotics/head_action
[7] http://www.orocos.org/kdl
[8] http://wiki.ros.org/actionlib
[9] http://wiki.ros.org/play_motion
[10] http://wiki.ros.org/gmapping

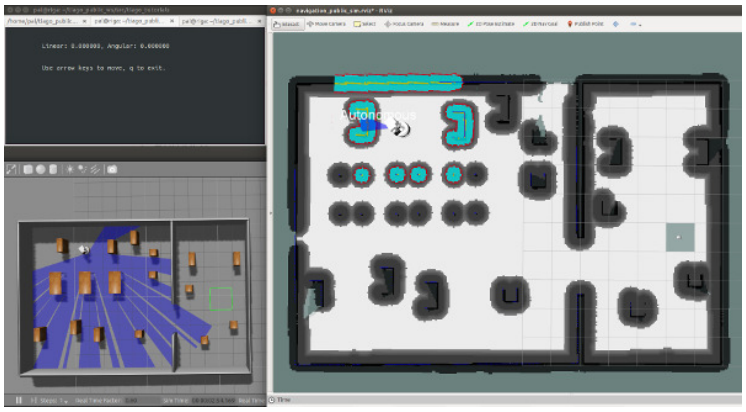

Fig. 3. Laser-based map generated by TIAGo

Once the user has created a map a tutorial showing the autonomous navigation of TIAGo is presented. This tutorial embraces theoretical concepts like Adaptive Monte Carlo Localization [4], particle filtering to track the pose of the robot in the map, the costmap concept, laser scan matching and obstacle avoidance with motion planning.

The navigation tutorial also introduces a more sophisticated approach to obstacle avoidance which consists in using not only the laser scan but also a virtual scan produced from the point cloud of the depth camera of TIAGo's head.

In Figure 4 a global trajectory that TIAGo is trying to follow to reach a destination point in the map is shown in blue. Furhtermore, the costamps appear overlayed in the map defining inflated areas that produce repulsion points taken into account by the local planner that reactively corrects the robot trajectory in order to avoid obstacles detected by the laser and the camera.

4 Motion Planning

The block of Motion Planning focues on MoveIt! [11] [5], which supports several motion planning libraries like OMPL [6] the state of the art software in motion planning for manipulation. The different tutorials are briefly explained hereafter.

4.1 Motion planning in joint space

This is the basic tutorial of motion planning. It shows how to define the kinematic configuration to reach in joint space and how to generate a plan and execute it in MoveIt!. This is a good example on how to obtain collision-free and joint limit avoidance when moving the robot upper body in joint space.

[11] http://moveit.ros.org

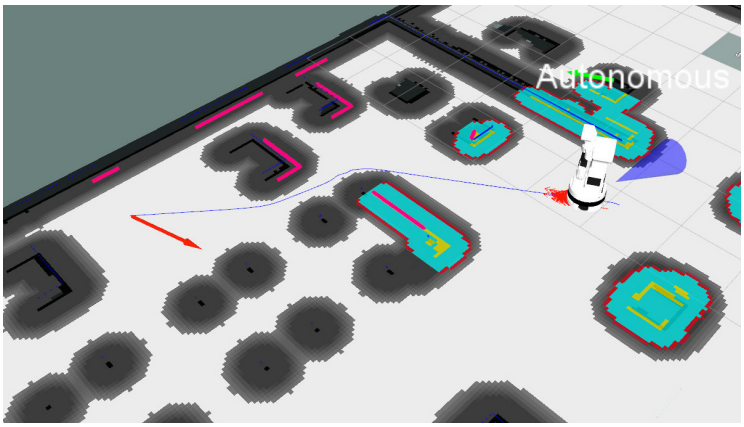

Fig. 4. AMCL-based localization of TIAGo and path planning

4.2 Motion planning in cartesian space

This tutorial allows the user to define the goal kinematic configuration in cartesian space, i.e. by defining the goal pose of the given frame of the robot's upper body. Then, it is an example on how to run inverse kinematics using MoveIt!

4.3 Motion planning taking Octomaps into account

This tutorial is an extension of the previous one in order to add vision-based collision avoidance with the environment. Indeed, it is an example on how MoveIt! can handle a 3D representation of the environment based on Octomap's occupancy grids [7], see 5.

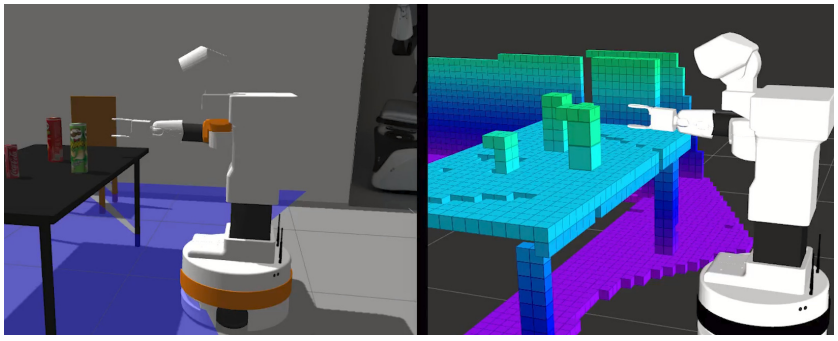

Fig. 5. Motion planning including Octomap occupancy grid

4.4 Demonstration of a pick & place application

The last tutorial of motion planning includes a full pipeline to have TIAGo picking an object from a table, raising it and then place it back to its initial position. The tutorial shows an example on how to use monocular model-based object pose estimation and how to use MoveIt! to perform the pick and place operations by avoiding collisions with the table. A snapshot of the demo is shown in Figure 6.

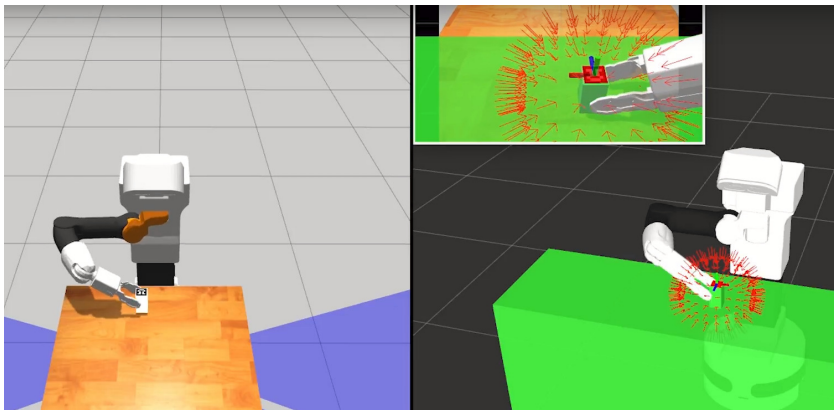

Fig. 6. Pick and place demo using MoveIt!

5 Perception

5.1 2D perception

A collection of tutorials based on OpenCV [8] are included in order to show typical tasks in robotics involving computer vision. These tutorials cover the following areas:

- Color-based tracking
- Keypoint detection and descriptors computation
- Fiducial markers detection and pose estimation
- Person and face detection
- Planar textured object detection based on homobraphy estimation

Figure 7 shows a couple of simulations. On top, the output of the tutorial showing how to perform person detection. On the bottom, the example of textured planar object detection and pose estimation.

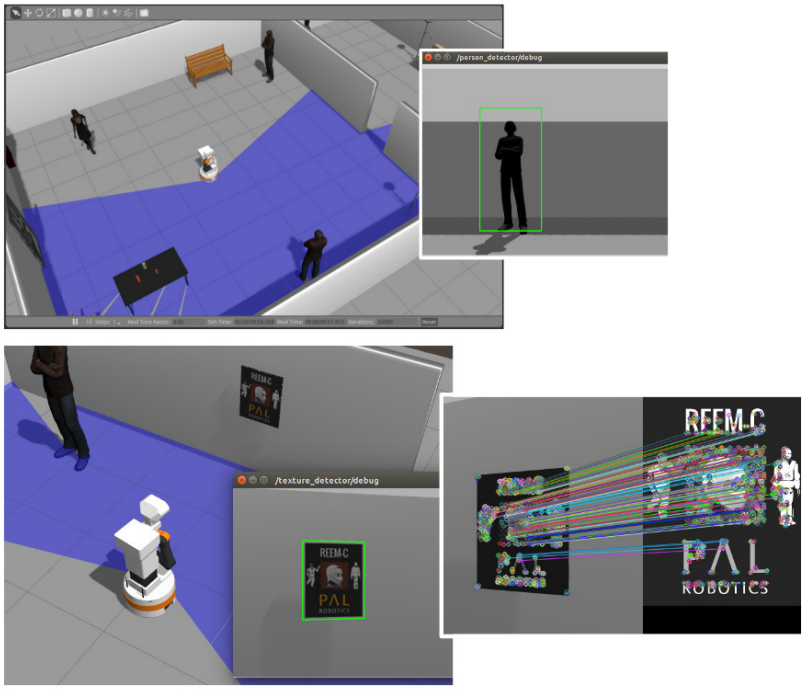

Fig. 7. Some examples of 2D perception in simulation

5.2 Point cloud perception

TIAGo is provided with a RGBD camera placed in its head. Nowadays this kind of low cost depth sensors are of common use in robotics applications. Furthermore, the existance of the Point Cloud Libray [9] provides a useful set of tools to process the data from the depth sensors and obtain semantic data out of it.

A set of tutorials showing point cloud processing are presented. The tutorials aim to show how to detect, segment and estimate the pose of and object on top of a table for robotic manipulation tasks. Figure 8 show the result of the tutorial providing cylindrical object detection and pose estimation.

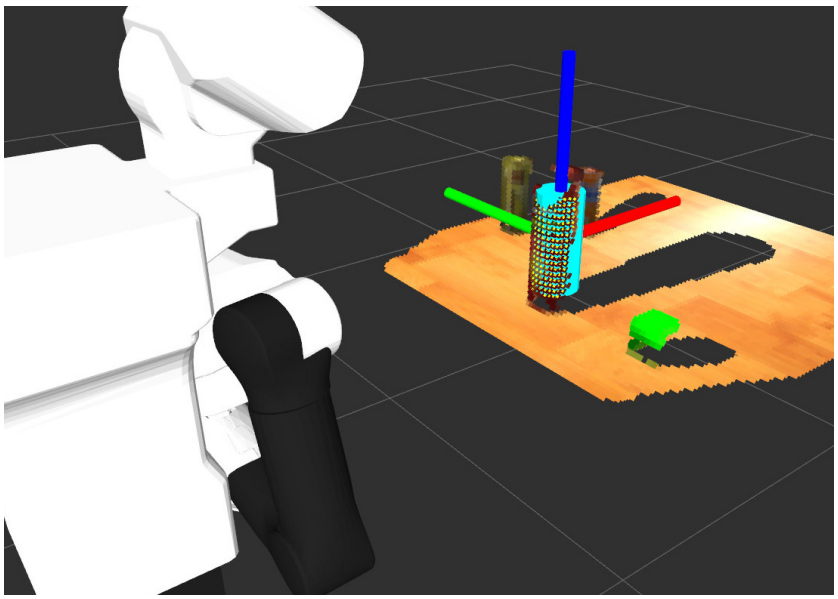

Fig. 8. Table top cylindrical object detection and pose estimation

The tutorials show step by step how to obtain this result starting from the detection of the plane of the table, following with the segmentation of the point clusters on top of such a plane, and finally identifying the clusters that fit on the cylindrical model.

The result of these tutorials can be then combined with the motion planning tutorials in order to implement grasping tasks with integrated perception.

6 Conclusions

This paper presents the on-line tutorials of ROS for the mobile manipulator TIAGo. The paper has shown how this tutorials can be used as practial examples

when teaching fundamental topics in Robotics and how they can be used as demonstrators of the different theoretical approaches involved.

The authors are convinced that these simulation-based tutorials can reduce the learning curve of robotics students as well as being powerful tools for teachers in order to engage easily their students by providing them with a framework where practical results are achieved very fast.

Since the publication of the tutorials, on early October 2016, a number of users, from teachers to PhD/under-graduate students, have adopted them to teach or learning some aspects of robotics. One important feedback provided by some of the users was that the installation part of the tutorials was a bit hard to meet in all existing Operating Systems and/or versions of ROS. In order to partially overcome this limitation the tutorials were also published in the one-line site of The Construct[12], so that the tutorials can be followed just by using a Web browser. The on-line course also offers exercises to practise the different topics taught in the tutorials.

References

1. Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y.: ROS: an open-source robot operating system. In: ICRA Workshop on Open Source Software. Volume 3., Kobe (2009) 5
2. Chitta, S., Marder-Eppstein, E., Meeussen, W., Pradeep, V., Rodríguez Tsouroukdissian, A., Bohren, J., Coleman, D., Magyar, B., Raiola, G., Lüdtke, M., Fernández Perdomo, E.: ros_control: A generic and simple control framework for ros. The Journal of Open Source Software (2017)
3. Davidson, J., Hunt, K.: Robots and Screw Theory: Applications of Kinematics and Statics to Robotics. Oxford University Press (2004)
4. Thrun, S., Fox, D., Burgard, W., Dellaert, F.: Monte carlo localization for mobile robots. In: In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA. (1999)
5. Chitta, S., Sucan, I., Cousins, S.: Moveit![ros topics]. IEEE Robotics & Automation Magazine **19**(1) (2012) 18–19
6. Şucan, I.A., Moll, M., Kavraki, L.E.: The Open Motion Planning Library. IEEE Robotics & Automation Magazine **19**(4) (December 2012) 72–82 http://ompl.kavrakilab.org.
7. Wurm, K.M., Hornung, A., Bennewitz, M., Stachniss, C., Burgard, W.: Octomap: A probabilistic, flexible, and compact 3d map representation for robotic systems. In: In Proc. of the ICRA 2010 workshop. (2010)
8. Bradski, G.: The OpenCV Library. Dr. Dobb's Journal of Software Tools (2000)
9. Rusu, R.B., Cousins, S.: 3d is here: Point cloud library (pcl. In: In Robotics and Automation (ICRA), 2011 IEEE International Conference on, IEEE 1–4

[12] http://www.theconstructsim.com