Framework Using ROS and SimTwo Simulator for Realistic Test of Mobile Robot Controllers

Tatiana Pinho[1], António Paulo Moreira[2], and José Boaventura-Cunha[1]

[1] INESC TEC (formerly INESC Porto) and Universidade de Trás-os-Montes e Alto Douro, UTAD, Escola de Ciências e Tecnologia, Quinta de Prados, 5000-801 Vila Real, Portugal
[2] INESC TEC (formerly INESC Porto) and Faculty of Engineering, University of Porto, Porto, Portugal
{al30888,jboavent}@utad.pt, amoreira@fe.up.pt

Abstract. In robotics, a reliable simulation tool is an important design and test resource because the performance of algorithms is evaluated before being implemented in real mobile robots. The virtual environment makes it possible to conduct extensive experiments in controlled scenarios, without the dependence of a physical platform, in a faster and inexpensive way. Although, simulators should be able to represent all the relevant characteristics that are present in the real environment, like dynamic (shape, mass, surface friction, etc.), impact simulation, realistic noise, among other factors, in order to guarantee the accuracy and reliability of the results.

This paper presents a ROS (Robot Operating System) framework for the SimTwo simulator. ROS is an open-source library that is commonly used for the development of robotic applications since it provides standard services and promotes large-scale integrative robotic research. SimTwo is a realistic simulation software suitable for test and design of several types of robots. This simulator conducts realistic navigation procedures, since the driving systems, the sensors, the mechanical and physical properties of the bodies are precisely modeled.

The framework presented in this research provides the integration of ROS-based systems with the SimTwo simulator. Therefore, this framework reduces the risk of damage of expensive robotic platforms and it can be used for the development of new mobile robot controllers, as well as for educational purposes.

Keywords: simulation, SimTwo, ROS, framework.

1 Introduction

Robots development is a process that involves several engineering areas such as programming, electronics and mechanics, among others. In this sense technological advanced materials and design methods are needed, which implies that the development of new robotic solutions is often an expensive practice. However, nowadays, the increasing of processing capacity allows the development of efficient simulation tools [1].

Considering the fact that a simulator doesn´t take into account hardware aspects of the robot, it provides an ideal and efficient way to fulfill preliminary tests of the

© Springer International Publishing Switzerland 2015

A.P. Moreira et al. (eds.), *CONTROLO'2014 - Proc. of the 11th Port. Conf. on Autom. Control*, Lecture Notes in Electrical Engineering 321, DOI: 10.1007/978-3-319-10380-8_72

developed algorithms, enabling the introduction of different configurations, without materials and/or personal risks [2], leading to better decisions and economic savings. Furthermore, it offers the possibility of test and direct debug in the robots programs [3]. Therefore, a simulator is used to reduce the time and costs involved in the development and validation of a robot model [4].

In addition to the modeling of the robot dynamics, simulators must allow interactions between the robot and the environment by means of sensors and actuators representations [5]. In this sense, the simulation environment can be used in cases in which such real environment cannot be available [2]. The simulation also allows computing variables that are difficult to access in real operation enabling a more exact monitoring of the robot behavior [6].

Another simulation dimension concerns to its educational purpose. Namely, as the robotics teaching requires equipment resources, specialized people for support, laboratorial space, etc., simulation tools provide simple and accessible ways to achieve these requirements [5].

In other words, simulators can be used to research robots technical features, teaching, control manipulation qualities, model a virtual robot with simplified configuration, develop the robot itself, etc. [4]. It should be noted that a simulator must have validated sensor noise models [2].

In conclusion, a robotic simulator can be defined as a software tool which simulates the real world and creates a virtual environment to robots [4]. So, it should incorporate important features of the real world, where the importance of an aspect is different for each case [7]. A simulator will be more reliable, as closer to reality are their results. In the ideal situation, for equal reference inputs, the same results are achieved in real and simulated robots, as represented in Figure 1.

In this way, the control algorithms developed and tested in simulated environments, can be directly transferred to the real operating conditions, with time benefits and avoiding robots damaging [1].

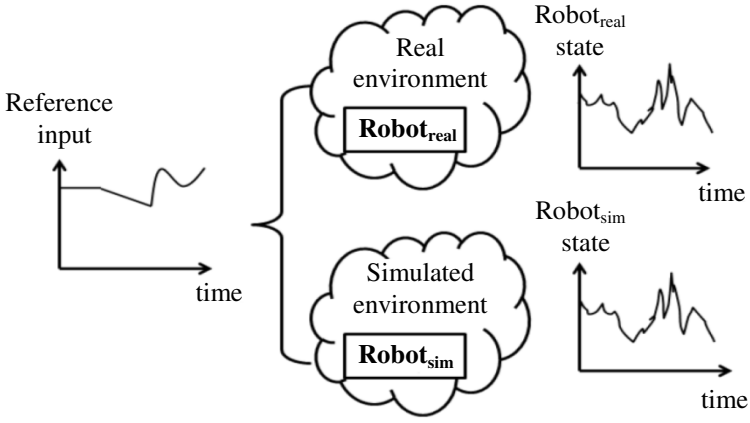

Fig. 1. Real and simulated robots comparison (Source: adapted from [1], p. 30)

Nowadays there are several available simulators for robotic systems, such as Über-Sim [7, 8], SimRobot [3], Webots [9], UCHILSIM [10], USARSim [5], Stage [11], Gazebo [12], ARGoS [13], V-REP [14], Delta3D, X-Plane, Microsoft Flight Simulator, Actin, Microsoft Robotics Developer Studio (MRDS), OpenSimulator, Simbad, FlightGear, Breve, Simspark, MURoSimF e OpenHRP3 [4, 5, 15], among others.

In this work a framework that integrates ROS systems and SimTwo simulation software is proposed, in order to promote the design and test of robot controllers, mainly in the educational context. This simulator choice was based on criteria such as simplicity, installation easiness and cost. This work is organized in 5 sections. In section 1 was made a brief introduction to the theme. Section 2 is dedicated to ROS (Robot Operating System). Section 3 presents the main features of the used simulator, SimTwo. Section 4 describes the proposed framework architecture and the last section (section 5), summarizes the main conclusions.

2 The ROS Framework

ROS is an open-source framework that provides an abstraction level to the complex hardware and software configurations in robotic area [2]. It is composed by a wide range of services and tools, available to users and developers, being included in its concept criteria such as: peer-to-peer communication for data transfer, tools-based, multi-lingual, thin and free and open-source [2, 16, 17]. In this way, ROS emerged as a way to facilitate the development in the robotic area and incorporate common solutions, once that this area involves a high level of complexity and constant update.

One of the ROS advantages is its high capacity of generalization to access to external hardware, either as sensors and actuators. Furthermore, through its modular character, it is possible to incorporate into the framework, new functionalities in a simple way [18]. As ROS is a middleware system, supported by a large community, it possesses shared libraries by the community itself, accessible to all users [16].

ROS uses a proper nomenclature, in which can be defined nodes, messages, topics, services, packages and stacks. Namely, nodes are processes that perform computation, messages are strictly defined structures and a topic consists in a string such as “odometry” or “map”. On the other hand, a service is defined by a string name and by two strictly typed messages, one for request and the other for response. A ROS package is a directory that contains a XML file with the package description and stating any dependencies. An organized packages set is defined as stack [2, 17].

In ROS, messages are sent by nodes, as pictured in Figure 2. Publishing nodes send their messages to a specific topic and the subscriber nodes receive that same message. A master node exposes the two types of nodes by a service [19]. It should be noted that publishing/subscribing and client/server are different communication types. The first one is used in general cases, and the second one only in specific cases, particularly when synchronism is needed [17]. It is also important to denote that several nodes can be connected to a robot with different purposes [19]. ROS is a publishing/subscribing system [16], or a client/server in the services case.

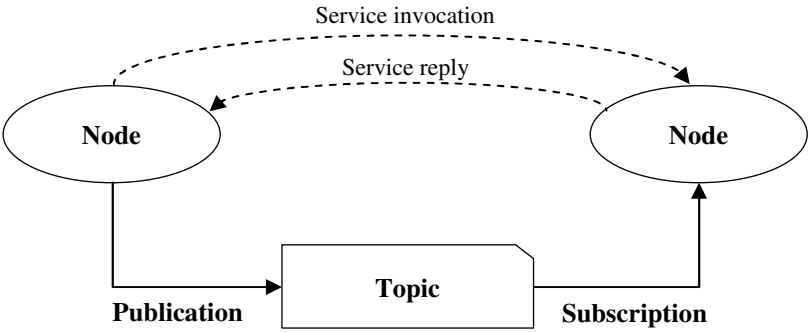

Fig. 2. ROS messaging mechanism (Source: adapted from [19], p. 3087)

3 SimTwo Simulator

SimTwo is a simulation tool developed in Object Pascal that allows the fast test and the construction of several robots types, namely, differential, omnidirectional, industrial, humanoids, among others, defined by the user in a 3D space [15, 20]. In a generic way, it can be said that any terrestrial robot type with rotating joints and/or wheels can be simulated with SimTwo.

For the dynamic simulation of rigid bodies, the Open Dynamics Engine is used, being the robot design and behavior are defined in XML files and the virtual world is represented by GLScene components. The robot control can be made by a script in the simulator itself or by a remote client that communicates by UDP or serial port.

SimTwo simulator has a high level of realism. In terms of dynamic this realism is conferred by robot separation into rigid bodies and electric motors. For each body, the behavior is simulated using physical features as shape, mass, moments of inertia, surface friction and elasticity [20]. Besides that, it possesses non-linear features that are important to the representation of robot real behaviors [21].

The SimTwo graphical interface is a *multiple document interface* (MDI), as represented in Figure 3, where all windows are under the "world view" window control.

In more detail, the "code editor" presents an *integrated development environment* (IDE) for a high-level programming in Pascal; the "configuration" window allows the control of several elements in the virtual scene; in "spreadsheet" can be defined "edit cells" and "button cells" with different purposes; in "chart" window all available variables can be graphically represented for each robot; and in the "scene editor" the scene is defined by several XML files [20].

Concerning to the control, SimTwo possesses two levels. Namely a high-level controller defined by the user, executed with a 40 ms periodicity, and a low-level controller, related to motors control, invocated with a 10 ms period. The model output is updated at a 1 ms rate [15].

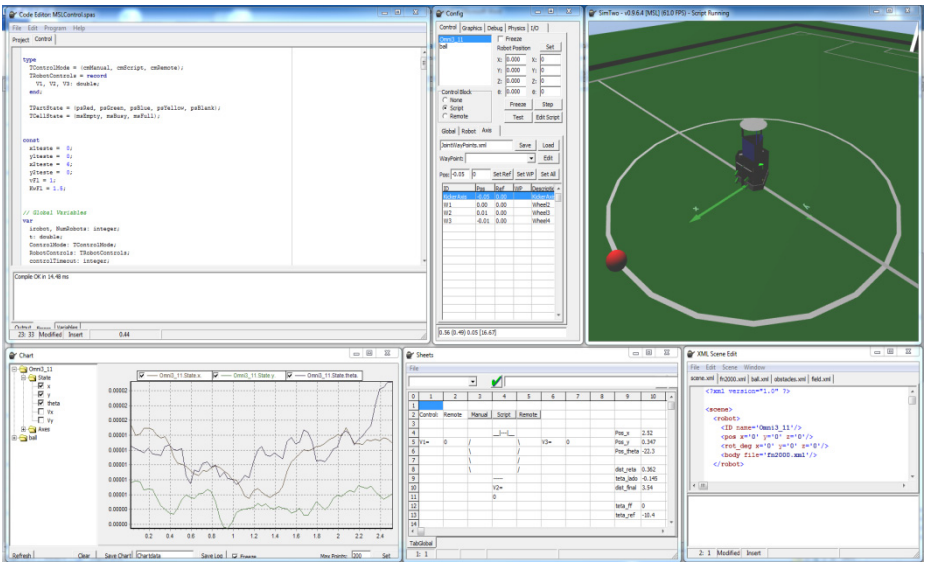

Fig. 3. SimTwo graphical interface

4 ROS/SimTwo Framework

This work proposes a framework development that allows the ROS and SimTwo systems integration. This will facilitate the transition to a real robot, since the developed ROS communicates with the simulator in the same way that communicates with the real robot. Furthermore, this system equally promotes the teaching and test of controllers in academic situation. Figure 4 represents the proposed framework architecture. The mobile robot trajectory control is described in section 4.1.

Considering the reference trajectory to be executed by the robot, together with the characteristic parameters of the model, the optimization is made to determinate which are the controls regarding linear (V, Vn) and angular (ω) velocities of the robot in order to follow the pre-defined trajectory.

Once calculated these controls, they are sent to the real robot by a ROS node and/or to the simulated robot, for which the SimTwo communication is made by UDP protocol, as shown in Figure 5. The synchronism between remote client and SimTwo was guaranteed in order to assure a proper operation of the controller. At the same time the real/simulated robot returns its localization data (x, y and θ), which are feedback into the controller to determine the following controls.

It should be noted that robot velocities V, Vn and ω conversion to each wheel velocity is made only after controls are sent. In other words, this conversion occurs independently of the remote client, giving it a generic character that allows its application to any type of robot.

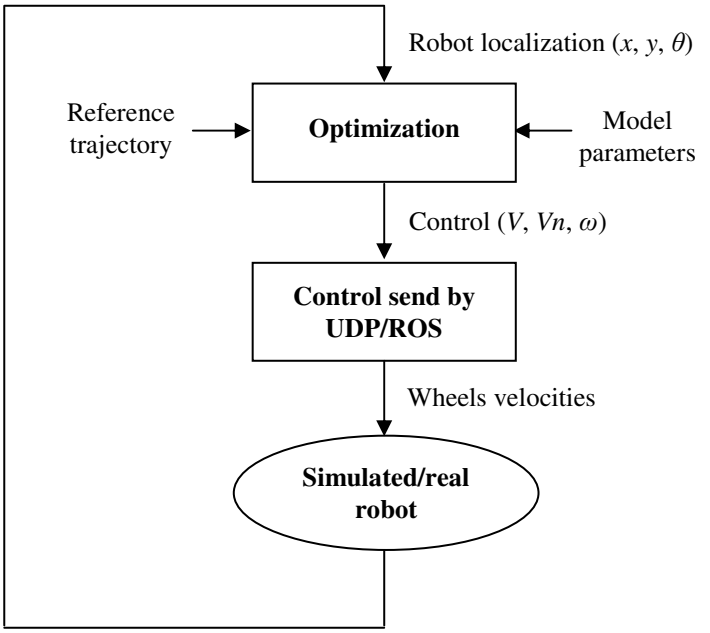

Fig. 4. Generic framework architecture

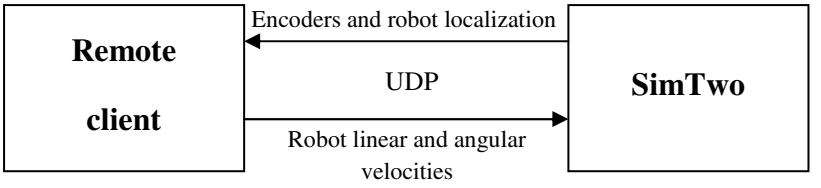

Fig. 5. Remote client – SimTwo communication

4.1 Framework Validation

For test and validation of the developed framework it were used MSL (Middle Size League) robots from the robotic soccer team of the University of Porto (Figure 3). The characteristic parameters of these robots implemented in SimTwo are described in Table 1.

In this work, a proportional controller of the distance and orientation errors related to a straight trajectory following was applied. The objective was to track a straight line coincident with the abscissa axis. To test the controller performance the robot was initially placed in a pose displaced 3 meters in *y*, and orientated in parallel to the line. Robot evolution in *x* and *y* poses is represented in Figure 6. In the same way,

Table 1. MSL robot constraints

Constraint	Measure	Unity
Robot mass	26	kg
Wheel mass	0.660	kg
Wheel radius	0.051	m
Wheel width	0.042	m
Encoder	12288	PPR
Gear ratio	12	
Resistance of the motor	0.316	Ω
Electric constant of the motor	0.0302	Nm/A
Maximum voltage on motor	24	V
Maximum allowed current	12	A
Moment of inertia (*x* axis)	0.388	kgm^2
Moment of inertia (*y* axis)	0.388	kgm^2
Moment of inertia (*z* axis)	0.705	kgm^2

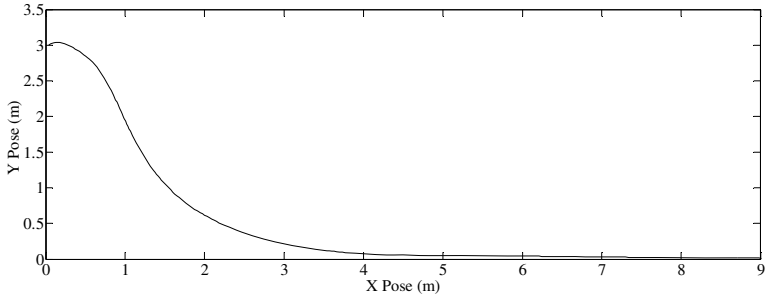

Fig. 6. Robot position, *x, y*, evolution, in m

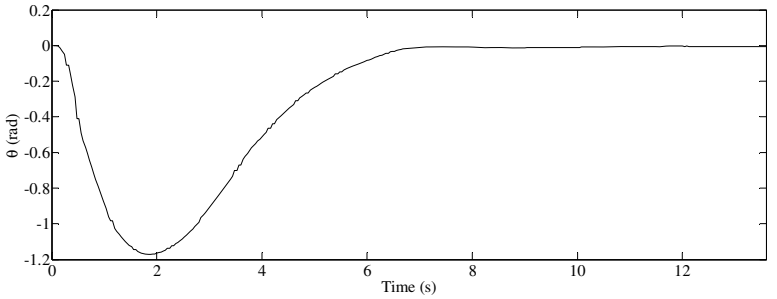

Fig. 7. Robot orientation, θ, evolution, in rad

its orientation evolution is presented in Figure 7. It should be noted that controller generic character was validated by the implementation of a dimensioned version for a differential traction robot in an omnidirectional robot.

According to Figures 6 and 7, it is possible to observe that the robot effectively corrects its localization in a relatively fast way, in order to be closer to the pretended trajectory, keeping afterwards coincident to the line. In this way it will be expected that after the transition to the real robot, its behavior will be similar to the simulated robot. This fact occurs because the ROS communicates in an identical way with the robot in the real environment.

5 Conclusions

This work proposes a framework to support the design and validation of robot controllers, mainly in educational context, integrating a ROS system and SimTwo simulation software. With this framework it is pretended to facilitate the passage of simulated to real situations, since ROS system communicates with the simulator in a similar way it communicates with the real robot. Furthermore, this work also aims to support the teaching and test of robotic controllers developed in academic environment.

In order to analyze the proposed framework performance, it was implemented a controller to coordinate a MSL soccer robot in way to follow a pre-defined trajectory. Its efficiency was confirmed by this application. Since the controls sent to the simulator refer to robot's linear and angular velocities, this work also possesses an abstraction character that provides the possibility of being implemented in any type of robot.

In the future, this work can be extended throughout the incorporation of other phenomena in the models, such as sensors noise, and the implementation of different control strategies aiming the evaluation of distinct control algorithms performances in simulations closer to the real environment robot operation.

Acknowledgements. The author also thanks the FCT for supporting this work through the project PTDC/EEI-AUT/1450/2012 – Optimal Control: Health, Energy and Robotics Applications.

References

1. de Lima, J.L.S.M.: Construção de um Modelo Realista e Controlo de um Robô Humanóide, Tese de Doutoramento em Engenharia Electrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto (2008)
2. Balakirski, S., Kootbally, Z.: USARSim/ROS: A combined framework for robotic control and simulation. In: Proceedings of the ASME 2012 International Symposium on Flexible Automation, St. Louis, Missouri, USA, pp. 1–8 (June 2012)
3. Laue, T., Spiess, K., Röfer, T.: SimRobot – A General Physical Robot Simulator and Its Application in RoboCup. In: Bredenfeld, A., Jacoff, A., Noda, I., Takahashi, Y. (eds.) RoboCup 2005. LNCS (LNAI), vol. 4020, pp. 173–183. Springer, Heidelberg (2006)
4. Kumar, K., Reel, P.S.: Analysis of Contemporary Robotics Simulators. In: Proceedings of ICETECT, pp. 661–665 (2011)
5. Carpin, S., Lewis, M., Wang, J., Balakirsky, S., Scrapper, C.: USARSim: a robot simulator for research and education. In: IEEE International Conference on Robotics and Automation, Roma, Italy, pp. 1400–1405 (April 2007)

6. Gonçalves, J., Lima, J., Malheiros, P., Costa, P.: Realistic simulation of a Lego Mindstorms NXT based robot. In: 18th IEEE International Conference on Control Applications, Part of IEEE Multi-Conference on Systems and Control, Saint Petersburg, Russia, pp. 1242–1247 (July 2009)
7. Go, J., Browning, B., Veloso, M.: Accurate and Flexible Simulation for Dynamic, Vision-Centric Robots, pp. 1–8. AAMAS, New York (2004)
8. Browning, B., Tryzelaar, E.: ÜberSim: A Multi-Robot Simulator for Robot Soccer, pp. 948–949. AAMAS, Melbourne (2003)
9. Michel, O.: Cyberbotics Ltd. Webots™: Professional Mobile Robot Simulation. International Journal of Advanced Robotic Systems 1(1), 39–42 (2004)
10. Zagal, J.C., Ruiz-del-Solar, J.: UCHILSIM: A Dynamically and Visually Realistic Simulator for the RoboCup Four Legged League. In: Nardi, D., Riedmiller, M., Sammut, C., Santos-Victor, J. (eds.) RoboCup 2004. LNCS (LNAI), vol. 3276, pp. 34–45. Springer, Heidelberg (2005)
11. Gerkey, B.P., Vaughan, R.T., Howard, A.: The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems. In: Proceedings of the International Conference on Advanced Robotics (ICAR), Coimbra, Portugal, pp. 317–323 (July 2003)
12. Koenig, N., Howard, A.: Design and Use Paradigms for Gazebo, An Open-Source Multi-Robot Simulator. In: Proceedings of 2004 IEEE/RSJ International Conference on Inteligent Robots and Systems, Sendai, Japan, pp. 2149–2154 (October 2004)
13. Pinciroli, C., Trianni, V., O'Grady, R., Pini, G., Brutschy, A., Brambilla, M., Mathews, N., Ferrante, E., Caro, G.D., Ducatelle, F., Birattari, M., Gambardella, L.M., Dorigo, M.: ARGoS: a modular, parallel, multi-engine simulatator for multi-robot systems. Swarm Intell. 6, 271–295 (2012)
14. Freese, M., Singh, S., Ozaki, F., Matsuhira, N.: Virtual Robot Experimentation Platform V-REP: A Versatile 3D Robot Simulator. In: Ando, N., Balakirsky, S., Hemker, T., Reggiani, M., von Stryk, O. (eds.) SIMPAR 2010. LNCS, vol. 6472, pp. 51–62. Springer, Heidelberg (2010)
15. Lima, J.L., Gonçalves, J.A., Costa, P.G., Moreira, A.P.: Humanoid Gait Optimization Resorting to an Improved Simulation Model. International Journal of Advanced Robotic Systems 10(67), 1–7 (2013)
16. De Marco, K., West, M.E., Collins, T.R.: An Implementation of ROS on the Yellowfin Autonomous Underwater Vehicle (AUV), pp. 1–7. IEEE, OCEAN (2011)
17. Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., Ng, A.: ROS: an open-source Robot Operating System. ICRA, 1–6 (2009)
18. Hax, V.A., Filho, N.L.D., Botelho, S.S.D.C., Mendizabal, O.M.: ROS as middleware to Internet of Things. Journal of Applied Computing Research 2(2), 91–97 (2012)
19. Arumugam, R., Enti, V.R., Bingbing, L., Xiaojun, W., Baskaran, K., Kong, F.F., Kumar, A.S., Meng, K.D., Kit, G.W.: DAvinCi: A Cloud Computing Framework for Service Robots. In: IEEE International Conference on Robotics and Automation, Anchorage, Alaska, USA, pp. 3084–3089 (May 2010)
20. Costa, P., Gonçalves, J., Lima, J., Malheiros, P.: SimTwo Realistic Simulator: A Tool for the Development and Validation of Robot Software. Theory and Applications of Mathematics & Computer Science 1, 17–33 (2011)
21. Nascimento, T.P., Moreira, A.P., Costa, P., Costa, P., Conceição, A.G.S.: Modeling omni-directional mobile robots: an approach using SimTwo. In: 10th Portuguese Conference on Automatic Control, Funchal, Portugal, pp. 117–122 (July 2012)