Nao robot navigation system structure development in an agent-based architecture of the RAPP platform

Wojciech Dudek, Wojciech Szynkiewicz, Tomasz Winiarski

Warsaw University of Technology, Warsaw, Poland,
wojciechdudek.mail@gmail.com,
WWW home page: http://robotics.ia.pw.edu.pl

Abstract.

This paper focuses on development of a navigation system structure for the Nao humanoid robot in an agent-oriented distributed architecture. The proposed navigation system is a part of RAPP framework, a cloud robotics platform. The RAPP framework is an open-source software platform to support the creation and delivery of robotic applications, which are expected to increase the versatility and utility of robots. All navigation tasks are defined and divided into separate components. Our robot navigation system consists of a relative localisation based on Extended Kalman Filter (EKF) using both IMU and odometry measurements, visual QR-code based global localization, path planning, and motion control components. The proper allocation of navigation components, in the four-agent structure of the RAPP platform, is the main goal of this work. Navigation system components are implemented using Robot Operating System and Nao robot programming framework – NAOqi. Experimental results for the Nao robot are presented to show the validity of the proposed approach.

Keywords: Humanoid navigation, Nao robot, agent system

1 Introduction

Mobile robots as companions of human beings everyday life are very complex systems. Operating in the environment easily accessible by humans requests expanded perceptual, mental capabilities and human-like manual skills [1]. Robot servants can be specialized in a specific task (e.g. vacuum cleaners, robotic lawn mowers, ironing robots) or can be versatile and satisfy as many owner's requests as possible. It should be noted that the latter concept requires much more powerful computation capabilities then the former one. Additionally, one cannot predict all possible tasks that will be demanded of a robot companion. To answer these issues a concept of cloud infrastructure appears – a cloud possessing abilities to redistribute computations on the one hand, and store knowledge on

the other. Such an infrastructure, besides the ability to act as a data storage, should also contain robot skills, and distribute them among robots. Developing such a robot system, that will resolve above mentioned issues using the cloud infrastructure, is the goal of the RAPP (*Robotic Applications for Delivering Smart User Empowering Applications*) project [2]. The RAPP project will provide an open-source software platform to support the creation and delivery of Robotic Applications (RApps), which are expected to increase the versatility and utility of robots. The RAPP platform will provide computational capabilities and storing place for diverse RApps for different robots, in particular, for the affordable humanoid robot Nao used in this project. The general structure of the RAPP control system is presented in [3].

Humanoid robot autonomous navigation in domestic environments remains a challenging task. This paper describes the design, implementation and verification of the RAPP platform based on the structure of a navigation system of the Nao robot, in a social robot environment. The proper allocation of the navigation components, in the RAPP platform structure, is the main goal of this work. Our navigation system, alike the approach from [4], is divided into a local odometry based and global visual localizations. However, we additionally employed an EKF component to estimate the Nao pose using odometry and IMU (Inertia Measurement Unit) data. For the global localization purposes, the approach based on QR-code visual landmarks is used [5]. There are several known approaches to the Nao robot navigation using vision systems [6, 7]. An image-based visual servoing approach to humanoid robot guidance through corridors is proposed in [7]. In [8] a visual SLAM for Nao orientation estimation is presented. There are also several approaches that equipping the Nao with a laser range finder provide a SLAM algorithm [9]. The work [10] describes an interface between the Nao and the cloud service and demonstrates a use case in which several Nao robots download and execute abstract plans from RoboEarth cloud platform. Our robot navigation system is implemented using Robot Operating System middleware [11] and Nao robot programming framework (NAOqi) and satisfies all RAPP platform requirements.

2 Agent oriented distributed robot control system

Nowadays, most robot control systems are based on robot programming frameworks [12]. Frequently, these tools define just the middleware, i.e. providing inter-module communication structure, letting designer to define architectural structure. Most of those systems have a fixed structure [13, 14] – the modules of the system are not created or destroyed. RAPP platform approach enables to exchange its components and provides capabilities of the cloud.

Agent should be considered as anything that is aware of its environment and having an internal imperative to realize certain task acts upon that environment. Taking into consideration the RAPP platform, repository agent a_{rep} provides RApp store service, enabling robots to download RApps and use certain computational services. Core agent a_{core} is a robot specific subsystem that handles its

effectors and receptors. It behaves as a driver, enabling the RApps to use robot hardware layer. Virtual receptors aggregate variety of data from sensors and supply the system with an information about the surrounding environment. Virtual effectors execute developed instructions using devices and return robot current state. The control subsystem computes new control values and manages virtual receptors and effectors. Core agent, communicating with repository agent a_{rep}, is able to download RApps. RApps are either built with a single agent or two agents. The second case, thus RApp contains two agents, appears when additional computational capabilities are needed. Repository agent a_{rep} can provide simple services to the dynamic agent a_{dyn} and the core agent a_{core} directly. If more complex service or set of services is necessary to perform a task, a cloud agent a_{cloud} is invoked. In consequence, the RAPP application now contains two agents. The first one is created as a dynamic agent a_{dyn}, performing its work on the robot. The second one - cloud agent a_{cloud} - enables extra computational services in the cloud. The system establishes communication between dynamic agent a_{dyn} and cloud agent a_{cloud}, so the form of two agent RApp is created. Consequently to above, four agent system is established (Fig. 1).

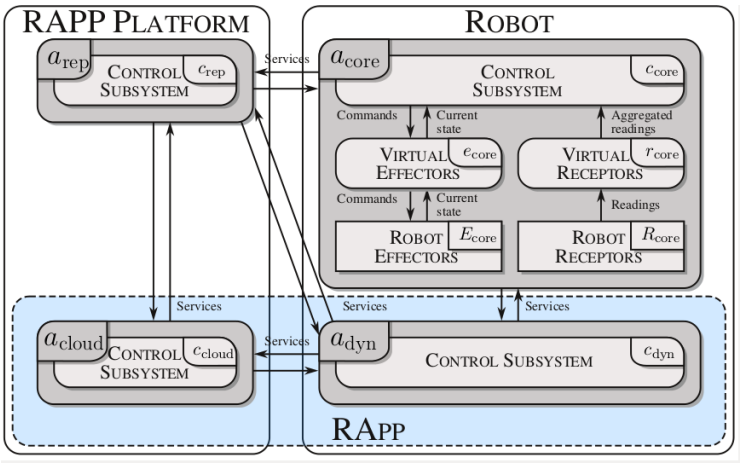


Fig. 1. General 4-agent system structure [3]

3 Navigation system structure and behaviour

Autonomous navigation is an essential capability of any service or personal robot that performs high level tasks involving mobility. There are variety of navigation systems. Usually they are based on a fixed structure. We propose a new approach to robot navigation using agent oriented distributed robot control system [3]. Specification of our navigation modules is based on System Modelling Language (SysML) [15]. It is an extension of popular UML, thus SysML is well defined and additionally supports specification, analysis, verification and validation processes of a wide range of devices and systems.

3.1 System functions

Taking into a consideration RAPP project requirements, Nao robot properties and limitations, the following basic navigation functions were established:

1. MoveTo[*callable*] – move to specified point,
2. MoveVel[*callable*] – move with specified velocity,
3. MoveStop[*callable*] – stop Nao walk,
4. MoveJoint[*callable*] – move specified joint to desired angle,
5. GetPosition[*callable*] – return current Nao position and orientation,
6. Global-path-planning[*callable*] – plan global path from start position to finish position,
7. MoveAlongPath[*callable*] – move along specified global path,
8. EKF-localization[*continuous*] – robot position estimation using EKF,
9. Obstacle-detection[*continuous*] – detect obstacle and stop robot motion,
10. QR-localization[*callable*] – estimate robot position using QR-code markers,
11. LookAtPoint[*callable*] – look at specified point in 3D coordinate frame,
12. TakePredefinedPosture[*callable*] – take specified, predefined posture,
13. Rest[*callable*] – take safe posture and remove motors stiffness.

The above functions form the navigation API. Most of them can be launched with arguments [*callable*], but two of them work continuously [continuous]. The system is designed to execute a single callable function in time. The execution of the functions implies an agents subsystems behaviours execution.

3.2 System components

Components of the navigation system have been chosen and designed to form navigation functionalities (Tab. 1).

	1	2	3	4	5	6	7	8	9	10	11	12	13
camera_server										+			
obstacle_detector	+	+	+				+		+				
state_server							+	+					
execution_server	+	+	+	+			+		+		+	+	+
robot_localization					+		+	+		+	+		
global_planner						+							
map_server						+							
QR_code_detection										+			
estimation_server					+		+	+		+	+		
move_server	+	+	+	+			+		+		+	+	+

Table 1. Components employment for specific navigation functions 1..13, as they are listed in Sec. 3.1

The components role is as follows:

1. *camera_ server* – responsible for image delivery,
2. *obstacle_ detector* – responsible for obstacles detection,
3. *state_ server* – delivers actual state of the Nao robot,
4. *execution_ server* – handles basic motion tasks,
5. *map_ server* – 2D environment map publisher,
6. *robot_ localization* – incremental localization handler, based on Extended Kalman Filter [16],
7. *QR_ code_ detection* – homogeneous transformation matrix from robot coordinate frame to QR-code coordinate frame,
8. *estimation_ server* – localization requests handler, driver for *robot_ localization* module,
9. *global_ planner* – global path planner,
10. *move_ server* – motion services handler

Basing on the navigation functions (Sec. 3.1), the component roles described above, and the use the components in the specific functions (as shown in Tab. 1), we determine communication links between these components (Fig. 2). Establishing these links is necessary for proper distribution of the components in the RAPP platform (Sec. 3.3). The figure 2 shows which components interact with delay-sensitive data or their algorithm is sensitive to Wi-Fi connection latency picks (black arrow), and which communication links can be realized using Wi-Fi (white arrow).

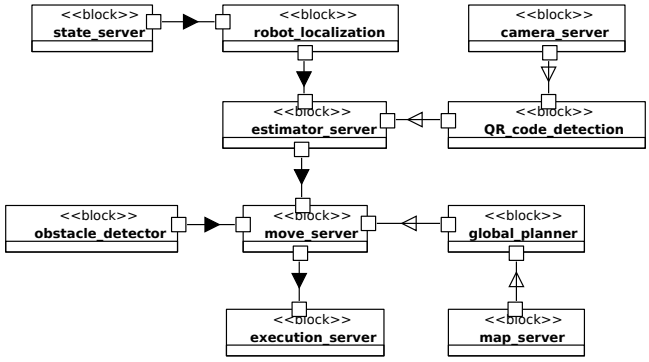
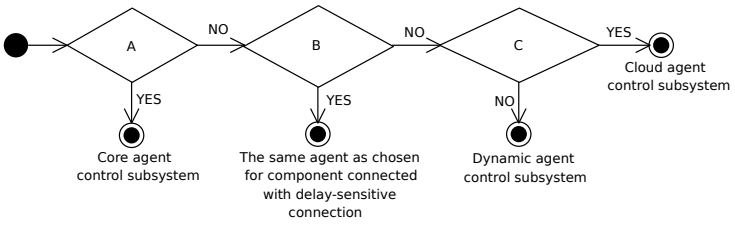


Fig. 2. Interactions between the navigation system components

3.3 Distribution of components in the RAPP platform

The RAPP platform structure (Fig. 1) is composed of four instances of the control subsystem – one in each agent, virtual effectors and virtual receptors. Therefore, all virtual effectors and receptors components were allocated in the core agent a_{core}. The components distribution is started with these virtual subsystems. Then, the rest of navigation components is distributed among the control

subsystems of four agents. We performed several tests with different structure configurations to determine criteria of proper components distribution. These criteria are formulated in three questions and are presented in the flow diagram (Fig. 3). This diagram is designed to guide developers how to allocate components in the four-agent RAPP platform structure. It should be noted



A) Does the component perform robot core functionality or is associated with a specific robot hardware/software?
B) Does the component interact with delay-sensitive data during intercomponental communication with its predecessor?
C) Is the algorithm computationally intensive?

Fig. 3. Allocation procedure of the control system components in the RAPP platform

that the criterion B depends on intercomponental communication parameters, thus the proper order of components distribution is required, starting from the components interacting with the components associated to the virtual receptors and effectors. Then, components connected with these components, etc.

The presented allocation procedure was verified during validation of our navigation system (Sec. 3.4). It is obvious that the procedure can fail, when navigation functions are decomposed into components in improper way. For example, let us assume that the first component, allocated in the cloud agent, is connected to the second component by sensitive link. Next, the second component following the distribution procedure (Fig. 3) is allocated in the core agent. According to the system structure (Fig. 1), there is no direct connection between a cloud agent and components directly associated with a specific robot – a core agent.

3.4 Distribution of navigation components of the Nao robot

The procedure starts with virtual effectors and virtual receptors of the core agent. There are two virtual receptors: *camera_server* and *obstacle_detector*. Virtual effector is composed of three modules: *state_server*, *execution_server* and *robot_localization*. Next, following the procedure, *move_server* (motion services handling) and *estimator_server* (localization services handling should be located in the core agent control subsystem. However, *global_planner*, requires

a lot of computational power, but it is not sensitive to WiFi communication delays and does not provide robot core functionality. Therefore, such a component should be placed in the cloud agent control subsystem. In Fig. 4 the distribution of navigation system components in the RAPP infrastructure is presented. Navigation modules provide basic robot services, so most of the components

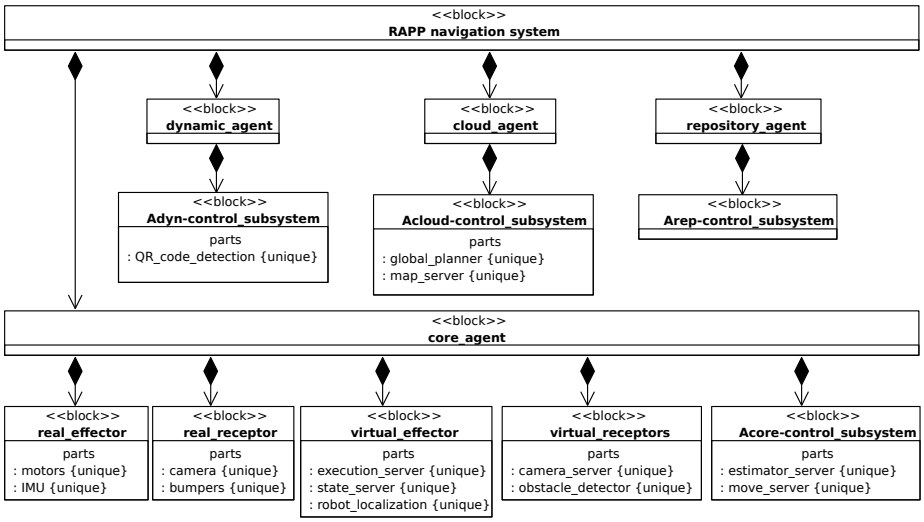


Fig. 4. Navigation system in the RAPP platform structure

are allocated in the core agent. The *QR_ code_ detection* component delivering common functionality, that is not directly associated with a specific robot, is allocated to the dynamic agent. It enables every robot, connected to the RAPP platform, to download this module and use it to localize itself in the map.

4 Navigation system structure implementation and verification

Implementation of the navigation system components is based on the Robot Operating System (ROS) middleware [11]. We use some ROS packages to perform navigation and localization services, i.e. the *robot_ localization* [16] as the EKF implementation, and the *global_ planner* [17] as the global path planning module. Virtual effector and receptor components communicate with the Nao robot using NAOqi framework. In Fig. 5 virtual effector interfaces structure is presented. In the *move_ server* component several motion control algorithms are implemented, e.g. *moveVel*, *moveAlongPath*, *moveJoint*, and *takePredefinedPosture*. Moreover, *estimation_ server* handles localization requests.

The navigation system was verified on an example of a hazard detection task. To accomplish this task the dynamic agent was created. The dynamic

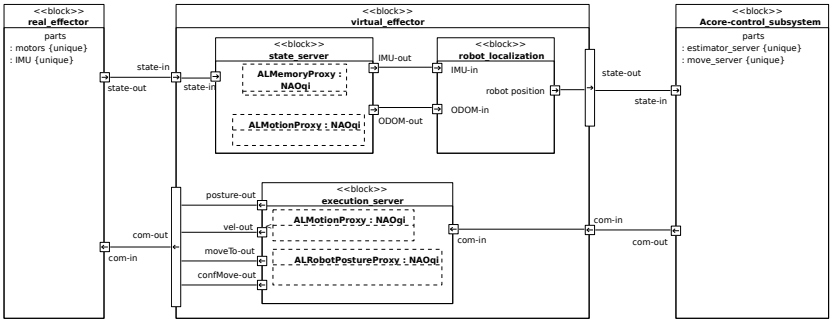


Fig. 5. Virtual effector interfaces

agent invokes several functions of the navigation system. The whole task was completed successfully. Performed experiments show that the Nao robot localizes itself with good accuracy, the path is collision-free and the EKF-based pose estimation enables the robot to precisely track the desired path. As shown in Fig. 6, illustrating the representative experiment execution, the Nao follows the desired path with good enough accuracy. The maximum path tracking error is 19 cm, and it is a satisfactory result for the Nao robot.

5 Conclusions

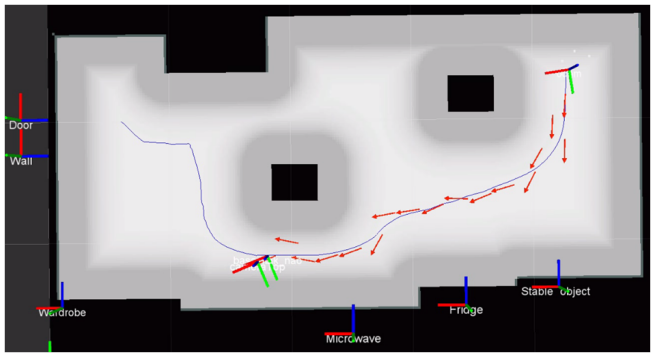
In this paper we presented a structure of an autonomous robot navigation system in an agent-based distributed architecture. Based on the performed experiments we proposed a procedure that enables appropriate allocation of the control subsystem components in the RAPP cloud robotics platform. Experiments demonstrated that the proper components distribution model must be carefully chosen. This paper provides an insight into partitioning of navigation system functions into those that should be assigned to core agent, to dynamic agent, and those that should be allocated to cloud agent. The capabilities of the distributed navigation system were presented on the example of hazard detection task. In the future work, we plan to improve visual localization technique and provide an efficient obstacle avoidance algorithm to avoid collisions with non-stationary objects, including people moving around.

Acknowledgment

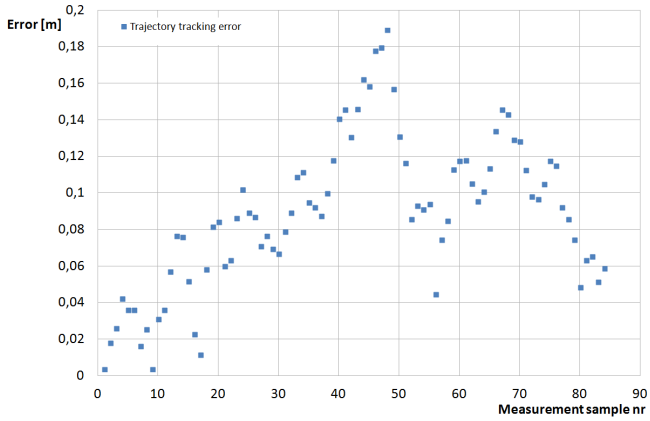
This work is funded by the FP7 Collaborative Project RAPP (Grant Agreement No. 610947), funded by the European Commission.

References

1. Winiarski, T., Banachowicz, K., Seredyński, D.: Multi-sensory feedback control in door approaching and opening. In: Intelligent Systems'2014. Volume 323 of



(a) Visualisation of the path tracking service



(b) Path tracking error

Fig. 6. Path tracking performance in the hazard detection task

Advances in Intelligent Systems and Computing. Springer International Publishing (2015) 57–70

2. Psomopoulos, F., Tsardoulias, E., Giokas, A., Zieliński, C., Prunet, V., Trochidis, I., Daney, D., Serrano, M., Courtes, L., Arampatzis, S., Mitkas, P.: Rapp system architecture. In: IROS 2014 – Assistance and Service Robotics in a Human Environment, Workshop in conjunction with IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, Illinois, September 14 (2014) 14–18
3. Zieliński, C., Szynkiewicz, W., Figat, M., Szlenk, M., Kornuta, T., Kasprzak, W., Stefańczyk, M., Zielińska, T., Figat, J.: Reconfigurable control architecture for exploratory robots. In Kozłowski, K., ed.: 10th International Workshop on Robot Motion and Control (RoMoCo), IEEE (2015) 130–135
4. Wei, C., Xu, J., Wang, C., Wiggers, P., Hindriks, K.: An approach to navigation for the humanoid robot nao in domestic environments. In: Towards Autonomous Robotic Systems. Volume 8069., Springer Berlin Heidelberg (2014) pp 298–310
5. Figat, J., Kasprzak, W.: NAO-mark vs QR-code Recognition by NAO Robot Vision. In Szewczyk, R., Zieliński, C., Kaliczyńska, M., eds.: Progress in Automation,

Robotics and Measuring Techniques. Vol. 2 Robotics. Volume 351 of Advances in Intelligent Systems and Computing (AISC)., Springer (2015) 55–64
6. Gouda, W., Gomaa, W., Ogawa, T.: Vision based slam for humanoid robots: A survey. In: Electronics, Communications and Computers (JEC-ECC), 2013 Japan-Egypt International Conference on, IEEE (2013) 170 – 175
7. Faragasso, A., Oriolo, G., Paolillo, A., Vendittelli, M.: Vision-based corridor navigation for humanoid robots. In: Robotics and Automation (ICRA), 2013 IEEE International Conference. (2013) 3190–3195
8. Wirbel, E., Bonnabel, S., Fortelle, A.D.L., Moutarde, F.: Humanoid robot navigation: getting localization information from vision. Journal of Intelligent Systems **23**(2) (2014) 113–132
9. Wen, S., Othman, K.K.M., Rad, A., Zhang, Y., Zhao, Y.: Indoor SLAM Using Laser and Camera with Closed-Loop Controller for NAO Humanoid Robot. Abstract and Applied Analysis **2014** (2014) 8 pages
10. Johannssen, F.: Nao in the Cloud. Knowledge Sharing for Robots via Cloud Services. In: Workshop on Software Development and Integration in Robotics ICRA 2013. (2013)
11. Foundatioin, O.S.R.: Robot Operating System. `http://ros.org/` [Online; accessed 21-May-2015].
12. Zieliński, C., Winiarski, T.: Motion generation in the MRROC++ robot programming framework. International Journal of Robotics Research **29**(4) (2010) 386–413
13. Zieliński, C., Kornuta, T., Trojanek, P., Winiarski, T., Walęcki, M.: Specification of a multi-agent robot-based reconfigurable fixture control system. Robot Motion & Control 2011 (Lecture Notes in Control & Information Sciences) **422** (2012) 171–182
14. Zieliński, C., Winiarski, T.: General specification of multi-robot control system structures. Bulletin of the Polish Academy of Sciences – Technical Sciences **58**(1) (2010) 15–28
15. SysML.org: SysML documentation. `http://sysml.org/` (2003) [Online; accessed 21-May-2015].
16. Moore, T., Stouch, D.: A Generalized Extended Kalman Filter Implementation for the Robot Operating System. `http://wiki.ros.org/robot_localization?action=AttachFile&do=view&target=robot_localization_ias13_revised.pdf` [Online; accessed 17-Sep-2015].
17. Marder-Eppstein, E., Berger, E., Foote, T., Gerkey, B., Konolige, K.: The office marathon: Robust navigation in an indoor office environment. In: Robotics and Automation (ICRA), 2010 IEEE International Conference on. (2010) 300–307