Integration of Mobile Robot Navigation on a Control Kernel Middleware based system

Eduardo Munera Sánchez, Manuel Muñoz Alcobendas, Juan L. Posadas Yagüe, Jose-Luis Poza-Luján, J. Francisco Blanes Noguera

Institute of Control Systems and Industrial Computing
Polytechnic City of Innovation
Polytechnic University of Valencia, Spain,
`emunera@ai2.upv.es`, `mmunoz@ai2.upv.es`, `jposadas@disca.upv.es`, `jopolu@disca.upv.es`, `pblanes@ai2.upv.es`
`www.ai2.upv.es`

Abstract. This paper introduces how a mobile robot can perform navigation tasks by taking the advantages of implementing a control kernel middleware (CKM) based system. Smart resources are also included into the topology of the system for improving the distribution of computational load of the needed tasks. The CKM and the smart resources are both highly reconfigurable, even on execution time, and they also implement fault detection mechanisms and QoS policies. By combining of these capabilities, the system can be dinamically adapted to the requirements of its tasks. Furthermore, this solution is suitable for most type of robots, including those which are provided of a low computational power because of the distribution of load, the benefits of exploiting the smart resources capabilities, and the dynamic performance of the system.

Keywords: Distributed Control Systems, Control Kernel, Robot Navigation, Limited Resources Management, Embedded Systems

1 Introduction

A navigation system is a must in every kind of robot which has to perform its tasks in an autonomous way and deal with an uncertain dynamic environments [14]. Although navigation is a well known topic in researches with mobile robots, it is always associated to a high computational load in comparison with other tasks that the robot usually performs. Thus many researches have been focused on how to deal with this load or the way to reduce it, but in every case, computational capabilities of the robot have to be always well dimensioned for being able to perform it properly.

Besides, navigation system also implies a strong requirement of data acquisition, even more in those cases which involve visual information [4]. Therefore, the type of sensor, the reliability of the provided data, and its supplying rate will affect on the performance of the navigation system. So, should be considered the proper acquisition and management of the perceptual data required for nourishing the navigation system.

Finally, a middleware-based implementation improve the performance and the reliability of the system. It also offers the possibility of working with high level abstraction and produces portable and reusable code. Therefore, this kind of implementations offers a great support for developing on mobile robots [6].

1.1 Related works

There are many middleware solutions focused on how to deal with sensor management(data acquisition) and navigation system. One of the more used framework in robotics is Robot Operating System (ROS) [13], which offers high level capabilities and works properly for collaborative robot networks, and consequently improves in many aspects the robot communication and data management. But shows a lack of generality on low level robot configuration and does not provide a real-time core with a highly fault tolerance.

Another example is Open RObot COntrol Software (OROCOS) [3] where mainly features are compiled in two libraries, one for kinematics and dynamics, and other for bayesian filtering methods, and a toolchain for code generation. It is usually extended by frameworks like Robot Constrution Toolkit (Rock) [1]. OROCOS is distinguished for offering hard real-time framework for control systems by providing tools for data exchange and event-driven services. However it has a lack of capabilities on behavior management for mobile robot operations.

It also can be introduced Yet Another Robotic Platfform (YARP) [5] as a middleware which offers a set of libraries and tools for establishing a decoupled configuration of the robotic platform isolating its devices, in a similar way that is managed in the architecture here described. But YARP excludes system control management which depends on a underlying operating system.

Some robotic-specific frameworks that offers more specific capabilities can be found like the case of CArnegie MEllon Navigation (CARMEN) [7], which is focused on navigation. It include a full support for navigation tasks like sensor control, obstacle avoidance, localization or path planning. Despite of this CARMEN disregard low level control, behaviour or real-time management. In contrast, there can be found behaviour-specific framework for robotic platforms just as Integrated Behaviour-Based Control (IB2C) [12] that is used for generation fusion and management of robot behaviours, and supplies graphical tools for behaviour design and its validation.

As a conclusion, there is no framework with support for the navigation process from the lowest level real-time system to the highest behaviour management. A full support is need in order to adapt of the requirements of the navigation process, and behaviour tasks.This adaptability is bounded by the capacity of reconfiguration offered by the devices involved in the system.

1.2 Outline

This paper is structured as follows: Section 2 shows a brief description of the structure of the used control kernel middleware (CKM) and the integration of smart resources into the CKM topology. The main contribution is introduced along section 3, introducing the advantages of using the CKM and the smart resources as the support of the navigation method. The paper ends with some conclusions about the work in section 4, and the future lines are collected in section.

2 Framework

In this section is depicted the current implementation of the CKM evloved form the proposal described in [2]. This middleware is responsible of core tasks, and offers mechanisms to suppor the navigation process.

2.1 Control Kernel Middleware

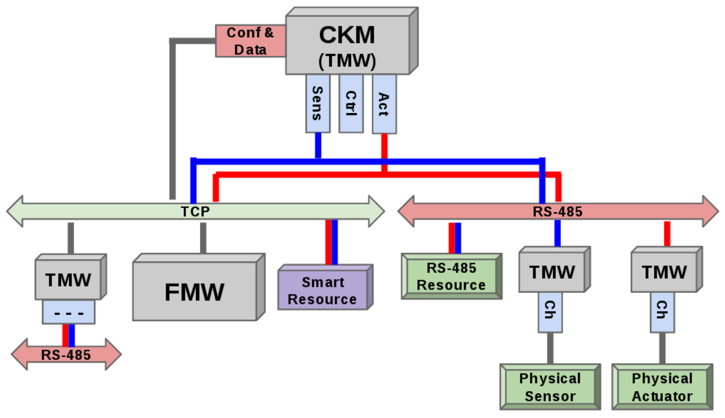

Fig. 1. Topology of a CKM based system .

This topology, as is is shown on Fig. 1, is characterized as a distributed control system just as is defined in [8], with slightly changes. The main elements depicted in this system are:

- Full Middleware (FMW): This is a full version of the CKM that implements all the services that are available in its definition.
- Tiny Middleware (TMW): This is reduced version of the CKM which implements control and communication services disregarding all the high level interfaces and configuration capabilities. A detailed description of its differences can be found on [8].
- Physical sensors and actuator: They are physical elements that are connected to a system that implements CKM, usually the TMW leaving the FMW only for configuration tasks.

- TCP & RS-485 connections: These protocols are implemented in order to provide communication capabilities. The RS-485 protocol is used for control data while the TCP is employed for configuration data and interconnection between different RS-485 networks. Both ways of communication must offer certain Quality of service (QoS) capabilities.
- RS-485 resources: These devices can be communicated by using RS-485 without the need of implementing a CKM version. More devices are usually sensors or actuators.
- Smart resources: Are introduced in next section.

2.2 Smart Resources

Smart resources are devices with specific computational capabilities that offer a TCP interface in order to access to provide services. This services are usually related to sensorization or actuation tasks that works with big amount of data and requires advance processing.

In the case of navigation tasks, smart sensors will be only considered. Navigation implies the acquisition and management of several data, usually provided by different kind of sensors. The processing of all this information is a highly resource consuming task in both, memory and computational power. An smart sensor can reduce this situation by a simple TCP interface which offers preprocessed information about the environment leaving only to the CKM the data fusion step.This sensors will take profit of the QoS advantages.

3 Robot navigation

Once the framework and the concept of smart resource have been introduced, the main contribution of this work is detailed: the Integration of Mobile Robot Navigation by using the CKM capabilities. Next a use case in order to validate this proposal is described .

3.1 Middleware support

The presented CKM is highly suitable for robotic platforms, such as has been detailed on [9], where it is introduced how the CKM can be used for establishing a mission-based control for several robotics platforms. Mission oriented tasks are in most cases extremely related with navigation, which has been defined as a "non-goal oriented tasks". In this case the system allow the navigation tasks influence the robot behaviours during the fusion process. The use of a proper configuration of the middleware in addition with one or several smart sensors can distribute the computational load, and consequently the accuracy of the system obtaining a better performance. Furthermore the use of QoS mechanisms allows to improve the reliability of the system, offering the capability of adapt the behaviour of the robot (or the smart sensor function) to the requirements of the system at anytime.

This implementation is oriented for offering a future support to a localization method derived from the one presented in [10]. This method is characterized as a reliability-based particle filter, where a reliability factor offers a statistical computation of how accurate is the position estimation according to the environment information. The value of the reliability factor must affect the configuration of smart sensors forcing a switch between their function modes for adapting the dynamic required for a proper execution of the navigation algorithm. In eq. 1 and 2 is reflected how the reliability factor (R) affects the coefficient value (f_{mode}) between 0 and 1. Both, the reliability factor and the coefficient (f_{mode}), are influenced by its corresponding weights: (w_R) and (w_{mode}). Bounding the function modes between threshold values allow to select the most appropriate on each case according to the f_{mode} coefficient.

$$f_{mode}(t) = \frac{R.w_R + f_{mode}(t-1).w_{mode}}{w_R + w_{mode}} \tag{1}$$

$$\begin{cases} 0 \leq f_{mode}(t) < thres_1 & \rightarrow MODE = 0 \\ \vdots & \\ thres_x < f_{mode}(t) < thres_{x+1} & \rightarrow MODE = X \\ \vdots & \\ thres_n < f_{mode}(t) \leq 1 & \rightarrow MODE = N \end{cases} \tag{2}$$

3.2 Use Case

It has been chosen the KertrolBot as the main robotic platform of this study. KertrolBot is a two wheeled mobile robot endowed with an array of IR sensors, its detailed characteristics may be reviewed in [15].

This platform is improved by the addition of a depth camera (which is one of the most essential sensor for robot navigation) as a Smart Sensor. Several commercial options are available, in this case an Asus Xtion camera is employed. This camera in combination of a Raspberry Pi provides a TCP interface, that permit to apply for concrete information about the environment, just like information about the closest object or distance to a certain colour object. Consequently is avoided to process raw camera data in the main CKM device, running on the core of the KertrolBot.

The proposed configuration is illustrated in Fig 2 where the following elements can be distinguished:

– KertrolBot on Board:
  - Core: Main unit, that implements a CKM version which attends to behaviours control.
  - Infrared Sensors (IR): Reduced CKM implementation which acquires raw data form IR sensors and offer the core unit a processed value of it.
  - Motors: Reduced CKM which interprets control signals from the core unit and executes a low level control on each wheel motor.

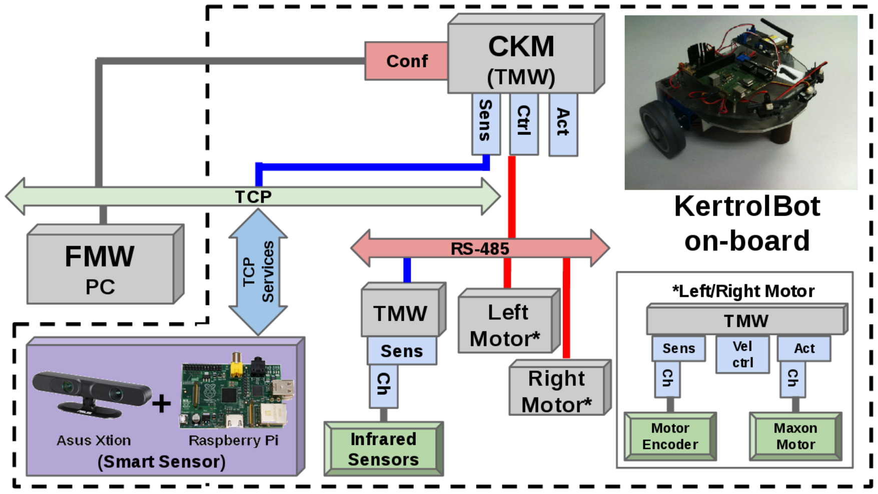

Fig. 2. Diagram of the use case.

  - Smart Sensor: Offers high-level services about sensorial information concerning the depth camera.
- FMW: Full middleware implementation running on an external PC that manages the configuration of the system.

The smart sensor here presented is the main provider of environment data used for the navigation process. Therefore it has been configured to perform up to four different modes depending on the navigation reliability factor. In Fig 3 can be check how the reliability factor influences the mode selection. The available modes are:

- Mode 1: This mode is used on lost robot situations. The camera is configured for working with maximum, resolution (VGA - 640x480), for allowing to spot more landmarks in the environment, and 10 frames per second (FPS), providing more time between frames for processing.
- Mode 2: In this case the camera uses the same resolution but improves the frame rate to 20 FPS.
- Mode 3: The most common mode in the robot. It deals with a smaller resolution (QVGA - 320x240) at 20 FPS, improving the processing time. The reliability on the robot position is good enough for helping to spot previously detected landmarks in a lower resolution.
- Mode 4: In this last mode resolution remains at QVGA, but uses a 33 FPS frame rate. This mode only takes place when navigation is fully reliable. As consequence, this system bring more reactivity to the system thanks to the data acquisition rate.

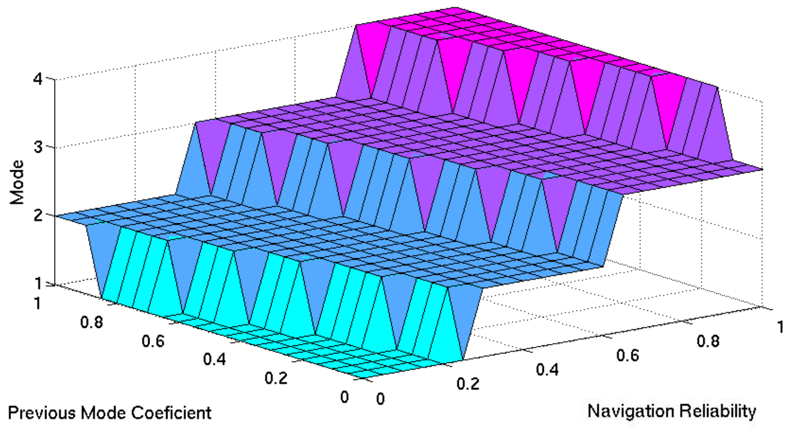

Fig. 3. Mode selection on Xtion smart sensor.

4 Conclusions

As a result of the work previously exposed, it can be concluded that developed proposal can exploit the real time control capabilities offered by the CKM and improve its sensorial capabilities by the use of smart sensors, which in turn is reflected on the navigation capabilities of the robot. The distribution of the computational load increases the capabilities of the system and the application of QoS mechanisms allows to detect and manage erroneous situations through a reliability factor.

5 Future lines of work

As future lines must be implemented a of reliability-based particle filter like the one presented on [10]. One of the main subject of future studies is to manage how the reliability factor and mode switching can influence the dynamic of the system in order to satisfy navigation needs. It also must be studied how QoS can help to detect system malfunction, and consequently be reflected in the reliability of the robot position [11].

Acknowledgments

This work has been partially supported by the Spanish Ministry of Economy and Competitiveness under the CICYT project Mission Based Control (COBAMI): DPI2011-28507-C02-02.

References

1. Rock (Robot Constrution Toolkit) http://www.rock-robotics.org/.
2. P Albertos, A Crespo, and J Simó. Control kernel: A key concept in embedded control systems. In *4th IFAC Symposium on Mechatronic Systems*, 2006.
3. Bob Bruyninckx, Herman and Soetens, Peter and Koninckx. The Real-Time Motion Control Core of the Orocos Project. In *IEEE International Conference on Robotics and Automation*, pages 2766—-2771, 2003.
4. Guilherme N DeSouza and Avinash C Kak. Vision for mobile robot navigation: A survey. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(2):237–267, 2002.
5. P Fitzpatrick, G Metta, and L Natale. Towards long-lived robot genes. *Robotics and Autonomous systems*, 2008.
6. Nader Mohamed, Jameela Al-Jaroodi, and Imad Jawhar. Middleware for robotics: A survey. In *Robotics, Automation and Mechatronics, 2008 IEEE Conference on*, pages 736–742. IEEE, 2008.
7. Michael Montemerlo, Nicholas Roy, and Sebastian Thrun. Perspectives on standardization in mobile robot programming: The carnegie mellon navigation (carmen) toolkit. In *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 3, pages 2436–2441. IEEE, 2003.
8. M. Muñoz, E. Munera, J. Francisco Blanes, José E. Simo, and G. Benet. Event driven middleware for distributed system control. *XXXIV Jornadas de Automatica*, page 8, 2013.
9. Manuel Muñoz, Eduardo Munera, J. Francisco Blanes, and Jose E. Simó. A hierarchical hybrid architecture for mission-oriented robot control. In Manuel A. Armada, Alberto Sanfeliu, and Manuel Ferre, editors, *ROBOT2013: First Iberian Robotics Conference*, volume 252 of *Advances in Intelligent Systems and Computing*, pages 363–380. Springer, 2014.
10. Eduardo Munera Sánchez, Manuel Muñoz Alcobendas, Juan Fco Blanes Noguera, Ginés Benet Gilabert, and José E Simó Ten. A reliability-based particle filter for humanoid robot self-localization in RoboCup Standard Platform League. *Sensors (Basel, Switzerland)*, 13(11):14954–83, January 2013.
11. J.Luis Poza Lujan. Relationship between Quality of Control and Quality of Service in Mobile Robot Navigation. *Distributed Computing and Artificial Intelligence*, pages 557–564, 2012.
12. M Proetzsch, T Luksch, and K Berns. Development of complex robotic systems using the behavior-based control architecture iB2C. *Robotics and Autonomous Systems*, 58(1):46–67, 2010.
13. Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, 2009.
14. Nicholas Roy, Wolfram Burgard, Dieter Fox, and Sebastian Thrun. Coastal navigation-mobile robot navigation with uncertainty in dynamic environments. In *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, volume 1, pages 35–40. IEEE, 1999.
15. Vicente Nicolau, Manuel Muñoz, and Jose Simó. KertrolBot Platform. SiDiReLi: Distributed System with Limited Resources. Technical report, Institute of Control Systems and Industrial Computing - Polytechnic University of Valencia, Valencia, Spain, 2011.