Integration of Mobile Robot Navigation on a Control Kernel Middleware Based System

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

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

Abstract. This paper introduces how mobile robots can perform navigation tasks by implementing a system based on the control kernel middleware (CKM), and how can take benefit of this. Smart resources are also included into the topology of the system, improving the distribution of the computational load required by the system tasks. The CKM and the smart resources are both highly reconfigurable, even on execution time, and they also implement fault detection mechanisms and Quality of Service (QoS) policies. By combining of these capabilities, the system can be dynamically adapted to the requirements of its tasks. Furthermore, this solution is designed to be implemented by almost every type of robot. The distribution of load make this system suitable even for those configurations which are provided with a low computational power. All these benefits are improved by 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 for every kind of robot which has to perform in an autonomous way and deal with an uncertain dynamic environments [14]. Although navigation is a well known topic, it is always associated to a high computational load in comparison with other tasks. Thus many researches have been focused on how to deal with this load, or the way to reduce it. In every case, computational capabilities of the robot have to be designed for being able to face its execution in a proper way.

Besides, navigation system also implies a strong requirement of data acquisition. Even more, these data may be particularly complex in those cases in which visual information is used[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 a proper acquisition and management of the perceptual data, which is required for nourishing the navigation system.

Finally, a middleware-based implementation will improve the performance and the reliability of the system. It also offers the possibility of working with high level abstraction, and produce portable and reusable code. Therefore, this implementations provide a great support for developing robot architectures [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. ROS is known to work properly with collaborative robot networks, and for improving in many aspects of communications between robots and data management. But shows a lack of generality on low level robot configuration, and does not provide a real-time core.

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

It also can be introduced the middleware Yet Another Robotic Platfform (YARP) [5]. YARP offers a set of libraries and tools for establishing a decoupled configuration of the robotic platform. For this purpose devices are isolated in a similar way that is done in the architecture proposed in this work. But YARP excludes the control system management, which relies on an underlying operating system.

Some robotic-specific frameworks offers more concrete capabilities, such as CArnegie MEllon Navigation (CARMEN) [7], which is focused on navigation. It offers a full support for navigation tasks like sensor control, obstacle avoidance, localization or path planning. Despite of this, it disregards low level control, behaviour, or real-time management. Instead, there can be found behaviour-specific framework for robotic platforms just as Integrated Behaviour-Based Control (IB2C) [12], used for generation, fusion, and management of behaviours.

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

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 evolved form the proposal described in [2]. The CKM is responsible of core tasks, and offers mechanisms to support the navigation process.

2.1 Control Kernel Middleware

This topology, as is is shown on Fig. 1, is characterized as a distributed control system such as is defined in [8]. Main elements of this system are:

– Full Middleware (FMW): A full version of the CKM. Implements full services support.
– Tiny Middleware (TMW): This reduced version of the CKM implements basic control and communication services. A detailed description of tiny and full midddleware can be found on [8].
– Physical sensors and actuator: Physical elements connected with a low profile TMW implementation for data management and communication.
– TCP & RS-485 connections: Provide communication capabilities to the system. RS-485 is used for control data, while the TCP is employed for system configuration and communication between RS-485 subnetworks. Both connections must offer Quality of Service (QoS) capabilities.

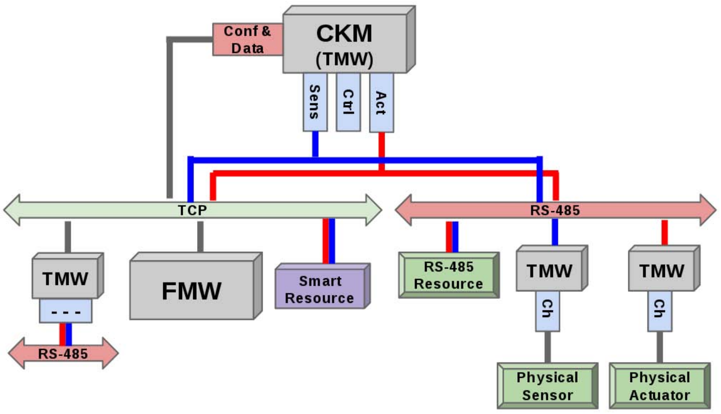

Fig. 1. Topology of a CKM based system

- RS-485 resources: Devices which can be communicated by RS-485 without implementing CKM. 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 has been introduced, the main contribution of this work is detailed: the Integration of a CKM-based support for mobile robot navigation. In order to validate this proposal is presented a use case. In this example is detailed how this integration can improve the performance of the system by its implementation on a certain robot.

3.1 Middleware Support

The CKM is a highly suitable solution for robotic platforms. As is detailed in [9], the CKM can be used for establishing a mission-based control on several robotic platforms. Mission oriented tasks are in most cases extremely related with the navigation process, which is defined as a "non-goal oriented tasks". This paradigm allow the robot behaviours to be influenced by the navigation task needs during the fusion process.

This implementation is oriented to offer a future support for a localization method derived from the one presented in [10]. This method is characterized as a reliability-based particle filter, where a reliability factor provides a statistical computation of how accurate is the position estimation. A proper configuration of the middleware, and the use of smart sensors leads to a distribution of the computational load, and consequently a better performance of the navigation tasks.

For improving execution of the navigation algorithm, the reliability factor (R) is designed to affect the operation mode of the smart sensors. That way, sensor is dynamically reconfigured to suit the localization requirements. In eq. 1 is computed the coefficient factor (f_{mode}) according to he reliability R.

Both values must be normalized between 0 and 1, and and weighted according to the values of (w_R) and (w_{mode}). The obtained f_{mode} is used to select the operation mode which is bounded by a threshold such as is shown on eq. 2.

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

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

3.2 Use Case

For this case of study it has been chosen a wheeled mobile robot called KertrolBot, which main characteristics can be reviewed in [15]. This platform is improved by the addition of a depth camera (a common device for robot navigation) integrated as a Smart Sensor. Smart Sensor is composed by an Asus Xtion camera is connected to a Raspberry Pi, which provides a TCP interface. According to this interface, it can be applied for some concrete information about the environment, just as 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.

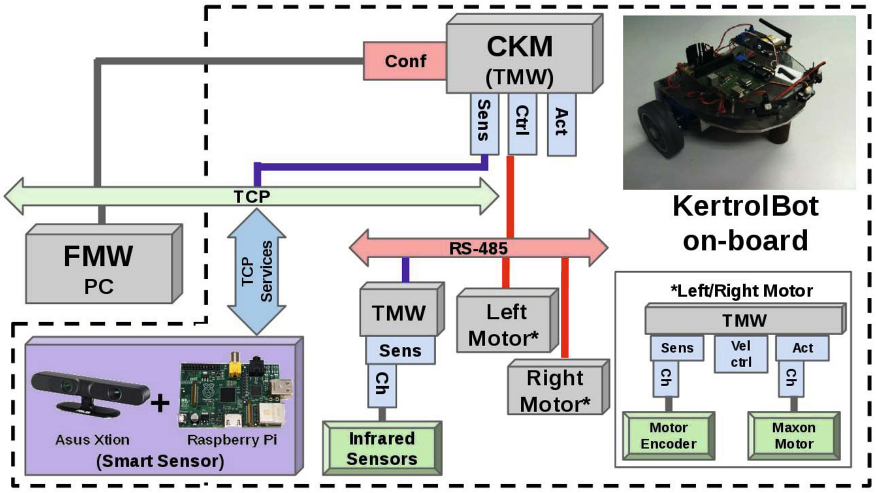

Fig. 2. Diagram of the use case

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

- KertrolBot on Board:
  - Core: This main unit implements a CKM verison for 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.
  - 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.

Main objective of this implementation is to bring the robot the capability of being localized in the environment, and performing an optimum management of the resources. This goal is achieved by the integration of the Smart Resources into the CKM topology. In this case, data acquisition is optimized by being adapted to the dynamic of the localization algorithm and its requirements. This adaptation is feasible thanks to the implementation of a CKM support. It manages the reconfiguration of system services, like the data acquisition tasks. Furthermore it offers a proper execution support for the localization process, being defined as a "non-goal oriented task".

Therefore the Smart Sensor (depth camera), as the main provider of environmental data, must be adapted to the localization performance. According to its different requirements, the camera can switch between these operation modes:

- Mode 1: Used on lost robot situation. Aims to obtain as much environmental information as is possible. For that purpose maximum resolution (VGA - 640x480) is used for a better landmarks detection, and 10 frames per second (FPS) for a longer processing between frames.
- Mode 2: Localization is not fully reliable. Offers same resolution, but improves the frame rate up to 20 FPS.
- Mode 3: Localization is reliable (most common mode). It deals with a smaller resolution (QVGA - 320x240) for improving the processing time, and 20 FPS.
- Mode 4: System reactivity is required regardless of the localization. QVGA resolution remains, but frame rate is increased to 33 FPS.

Mode switching is triggered by the value of the mode factor (f_{mode}) described on the equation 2 and according to the threshold value for each mode 1. As far as f_{mode} depends on the localization reliability factor, and its previous values, it reflects the quality of the system and its requirements.The relation between the reliability and the active mode is detailed in Fig 3.

The integration of QoS mechanisms will improve the reliability in this system. For a localization method, QoS helps to detects unexpected situations on data acquisition on Smart Sensor. This will affect to the computed reliability factor, and be reflected in the dynamic of the localization method. This variation will trigger mode switching on Smart Sensor trying to solve the acquisition problem.

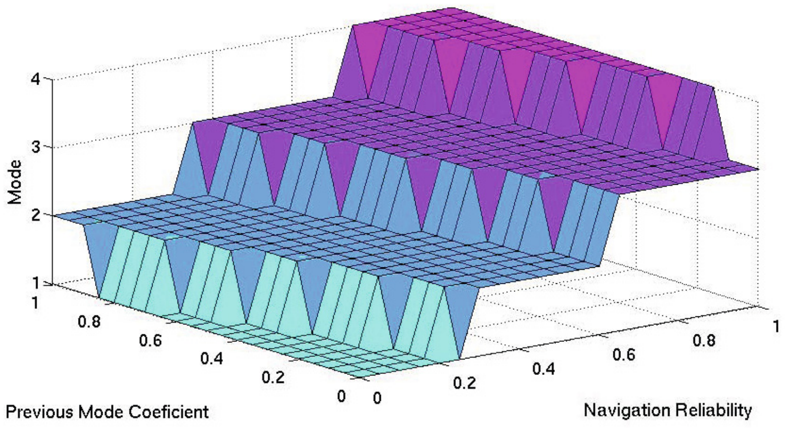

Fig. 3. Mode selection on Xtion smart sensor

4 Conclusions

According to the work previously exposed, the use of Smart Resources as a part of a CKM system helps to improve the optimization level of the localization process. A proper management of mode switching on the Smart Sensor based on the localization reliability provides a sensorial adaptation to the requirements of the localization algorithm. This integration also allow to distribute the computational load increasing the capacity of the system. Furthermore it offers a solid support for future QoS integration that will improve system reliability.

5 Future Lines of Work

As future work, it must be implemented a reliability-based particle filter, as an evolution of the one presented in[10]. This process will take profit of the support architecture here described. One of the main goals to achieve, is to proper management of the relation between the reliability factor and the mode switching on the Smart Sensor. It also must be characterized how this relation will affect the dynamic of the system, and the navigation needs. Finally, will be studied how QoS may help to detect malfunctions, and the way it can be managed in the localization [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 Construtión Toolkit), `http://www.rock-robotics.org/`
2. Albertos, P., Crespo, A., Simó, J.: Control kernel: A key concept in embedded control systems. In: 4th IFAC Symposium on Mechatronic Systems (2006)
3. Bruyninckx, H., Soetens, P., Koninckx, B.: The Real-Time Motion Control Core of the Orocos Project. In: IEEE International Conference on Robotics and Automation, pp. 2766–2771 (2003)
4. De Souza, G.N., Kak, A.C.: Vision for mobile robot navigation: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence 24(2), 237–267 (2002)
5. Fitzpatrick, P., Metta, G., Natale, L.: Towards long-lived robot genes. Robotics and Autonomous Systems (2008)
6. Mohamed, N., Al-Jaroodi, J., Jawhar, I.: Middleware for robotics: A survey. In: 2008 IEEE Conference on Robotics, Automation and Mechatronics, pp. 736–742. IEEE (2008)
7. Montemerlo, M., Roy, N., Thrun, S.: Perspectives on standardization in mobile robot programming: The carnegie mellon navigation (carmen) toolkit. In: Proceedings of 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003), vol. 3, pp. 2436–2441. IEEE (2003)
8. Muñoz, M., Munera, E., Blanes, J.F., Simo, J.E., Benet, G.: Event driven middleware for distributed system control. XXXIV Jornadas de Automatica, 8 (2013)
9. Muñoz, M., Munera, E., Blanes, J.F., Simó, J.E.: A hierarchical hybrid architecture for mission-oriented robot control. In: Armada, M.A., Sanfeliu, A., Ferre, M. (eds.) First Iberian Robotics Conference of ROBOT 2013. AISC, vol. 252, pp. 363–380. Springer, Heidelberg (2014)
10. Sánchez, E.M., Alcobendas, M.M., Noguera, J.F.B., Gilabert, G.B., Ten., J.E.S.: A reliability-based particle filter for humanoid robot self-localization in RoboCup Standard Platform League. Sensors (Basel, Switzerland) 13(11), 14954–14983 (2013)
11. Poza-Luján, J.-L., Posadas-Yagüe, J.-L., Simó-Ten, J.-E.: Relationship between Quality of Control and Quality of Service in Mobile Robot Navigation. In: Omatu, S., De Paz Santana, J.F., González, S.R., Molina, J.M., Bernardos, A.M., Rodríguez, J.M.C. (eds.) Distributed Computing and Artificial Intelligence. AISC, vol. 151, pp. 557–564. Springer, Heidelberg (2012)
12. Proetzsch, M., Luksch, T., Berns, K.: Development of complex robotic systems using the behavior-based control architecture iB2C. Robotics and Autonomous Systems 58(1), 46–67 (2010)
13. 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, vol. 3 (2009)
14. Roy, N., Burgard, W., Fox, D., Thrun, S.: Coastal navigation-mobile robot navigation with uncertainty in dynamic environments. In: Proceedings of the 1999 IEEE International Conference on Robotics and Automation, vol. 1, pp. 35–40. IEEE (1999)
15. Nicolau, V., Muñoz, M., Simó, J.: KertrolBot Platform: SiDiReLi: Distributed System with Limited Resources. Technical report, Institute of Control Systems and Industrial Computing - Polytechnic University of Valencia, Valencia, Spain (2011)