Video Image Processing System for RT-Middleware

Gábor Sziebig, Andor Gaudia, Péter Korondi

Budapest University of Technology and Economics, Hungary
gabor.sziebig@gmail.com

Noriaki Ando

National Institute of Advanced Industrial Science and Technology, Japan
n-ando@aist.go.jp

Abstract: Video cameras have always been worthy sensors in robots. Many methods exists to retrieve information from video camera images, but they lack of robustness and too expensive to use them in real-time applications. Integration of video cameras in robot vision has many difficulties, till the last years it was unprofitable to develop video image processing system. There was no standardized framework for robot programming, and there was no need for fully customizable robot vision system. Time has changed and the demand for non industrial robots rose. The main contribution of this paper is that we created a modularized; middleware based video image processing system for robot vision, called Distributed Image Analyzer (DIMAN).

Keywords: RT-middleware, OpenRTM-aist, Intelligent Space, image processing, modularized robots, robot vision, Distributed Image Analyzer

1 Introduction

In the 21st century rapid progress in computer communication and technologies, robot systems are becoming larger and more complicated. The demand of low-cost mass production of robots, the rising number of elderly people, who needs support of their everyday life, called forth of middleware technology in robot technologies. In the beginning of the middleware developments it was doubtful, that the technology causes retardation in speed, but after benchmarks of Christopher D. Gill and William D. Smart [1] it showed that it has more advantages then disadvantages. They strongly believe that CORBA-based[1]

[1] CORBA: Common Object Request Broker Architecture

middleware offers great advantages in robotics and embed sensor applications. To manage the rapidly growing sensor data the single robot systems transformed to networked systems. The urgent need of middleware forced the robot programmers create their own middleware's that meets the best their need. Unfortunately, most of the pioneering initiatives are developed independently of the others, driven by specific applications and objectives [2].

Till now there are four significant middleware frameworks, they can be divided into two basic groups. The first group uses CORBA for communication and object managing between framework components, the second group uses socket programming. The details of the groups can be seen in Table 1.

	Name	Middleware tecnology	Technology extension
Group A	Miro (Middleware for robots)	CORBA	ACE+TAO
	Orca robotics	ICE	
	RT (Robot Technology) middleware	CORBA	ACE+omniORB
Group B	Player/Stage project	Socket programming	Boost

Table 1
Grouping of the middleware frameworks

In order to settle the state of chaos OMG[2] had created a Domain Task Force in 2005 December. They would like to contribute to the promotion of the standardization in field of robotics.

The remainder part of the paper is organized as follows: The next section introduces the different kinds of middleware frameworks. The DIMAN system is introduced in Section 3.

2 Middleware Frameworks

2.1 Miro (Middleware for Robots)

Miro is a distributed object oriented framework for mobile robot control, based on CORBA technology. The Miro core components have been developed in C++ for Linux. But due to the programming language independency of CORBA further components can be written in any language and on any platform that provides CORBA implementations.

[2] OMG: Object Managed Group, http://robotics.omg.org/

The Miro core components have been developed under the aid of ACE[3], an object oriented multi-platform framework for OS-independent inter process, network and real time communication. They use TAO[4] as their ORB[5], a CORBA implementation designed for high performance and real time applications.

In order to demonstrate the usability of the Miro framework a video image processing (VIP) framework has been created [3]. Robot vision applications require extensive testing and tuning of filter configurations. The VIP framework utilizes the infrastructure available in Miro for extensive support of the development process. VIP utilizes the parameter and configuration management toolkit provided by Miro for various purposes within the framework architecture.

2.2 Orca Robotics

Orca is an open-source framework for developing component-based robotic systems [4]. It provides the means for defining and developing the building-blocks which can be pieced together to form arbitrarily complex robotic systems. The types of systems they are targeting range from single vehicles to distributed sensor networks. Importantly, they envision usage patterns typical for both academic and commercial environments. The project's main goal is to promote software reuse in robotics. To implement a distributed component-based system, one must be able to define interfaces and make a choice of communication mechanism. In the case of cross-platform operation involving different operating systems, the software which provides such functionality is typically referred to as middleware. Given the realities of robotic software development they consider support for C/C++ on Linux to be essential. While CORBA is sufficiently flexible for Orca's middleware requirements, it is also large and complex. Experience with CORBA in earlier versions of Orca, showed this complexity to be problematic. In comparison, Ice offers a much smaller and more consistent API, superior feature set and similar performance.

2.3 Player / Stage Project

Player is a network server for robot control. Running on robot, Player provides a clean and simple interface to the robot's sensors and actuators over the IP network. The client program talks to Player over a TCP socket, reads data from sensors, writes commands to actuators, and configures devices on the fly. Player supports a variety of robot hardware. The original Player platform is the ActivMedia Pioneer 2 family, but several other robots and many common sensors are supported. Player's modular architecture makes it easy to add support for new hardware, and

3 ACE: Adaptive Communication Enviroments, http://www.cs.wustl.edu/~schmidt/
4 TAO: The ACE ORB, http://www.theaceorb.com/
5 ORB: Object Request Broker

an active user/developer community contributes new drivers. Player runs on Linux (PC and embedded), Solaris and *BSD [5].

2.4 RT (Robot Technology)-Middleware [6]

The Japanese Ministry of Economy, Trade and Industry (METI) in collaboration with the Japan Robot Association (JARA) and National Institute of Advanced Industrial Science and Technology (AIST) started a 3 year-national project 'Consolidation of Software Infrastructure for Robot Development' in 2002. With the intention of implementing robot systems to meet diversified users' needs, this project has pursued R&D of technologies to make up robots and their functional parts in modular structure at the software level, and to allow system designers or integrators building versatile robots or systems with relative ease by simply combining selected modular parts.

The robot technology middleware having been developed as infrastructure software for implementing the proposed robot architecture, named 'OpenRTM-aist'[6].

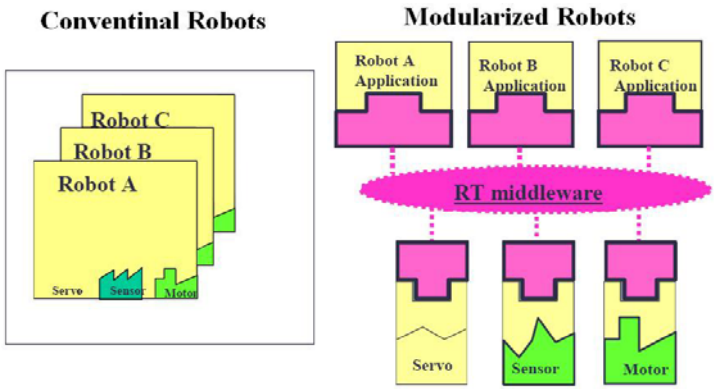

Figure 1
Difference between the conventional and modularized robot concepts

The answer of why we need RT-middleware can be seen in Figure 1. Not only thousands of hours of robot programming could be saved, but even more the interoperability between simulation and real applications in robots is solved.

Two prototype systems have been made to ascertain the effectiveness of the developed RT-middleware [7].

- A robot arm control system based on real time control.
- A life supporting robot system (RT space, also known as Intelligent Space), one of promising applications.

6 OpenRTM-aist: Open Robot Technology Middleware

2.5 What is RT Space (Intelligent Space)?

Is a space (room, corridor or street) with distributed robotic elements utilizing fixed or wireless network and middleware techniques. These robotic elements can be viewed as network nodes, using RT-middleware to communicate. In Intelligent Space these nodes are called Distributed Intelligent Networked Devices (DIND). They are composed of three basic elements: sensor (actuator), processor (computer) and communication elements.

Until now, many examples have been reported, which enable build an intelligent space in human living environments using ubiquitous devices and sensor networks, while there have been no examples which enable build a ubiquitous space from the robotic viewpoint. Moreover, up to now, distributed devices have been connected with wires, and the programs for controlling them were created, resulting in a large expense of labor and money.

In the network connection of robotic devices such as security and information devices, the OpenRTM-aist technique enables cost reduction, and thus can facilitate the introduction of robotic devices into general homes.

2.6 Concept of OpenRTM-aist

In the last few years, market, technology and industry trends have changed in dramatic way. The needs of customers expand fast, not only for industrial, but also for domestic applications of robots. This means that IT has to react fast to meet customers satisfaction. But how can IT suite customers personalized requests, requirements in short time? The only way is standardization. Till now there was no common platform for developing robot applications, this inchoate AIST to develop OpenRTM-aist.

RT-middleware, so thus OpenRTM-aist provides standardized communication between RT-middleware components (nodes), but it exceeds this, it provides complete framework for flexible robot programming.

2.7 Structure of OpenRTM-aist

It's a framework for programming robots. The framework is build from components. Each component is supervised by one RTC[7] Manager. The manager communicates with other RTC Managers and with the RTC Base component, and manages the life-cycle of the base component. The life-cycle of the components is shown in Figure 2.

The Base components represent the functional part of the framework. This level is responsible for data processing, or for example moving MHI PA10 manipulator.

[7] RTC: RT-Component

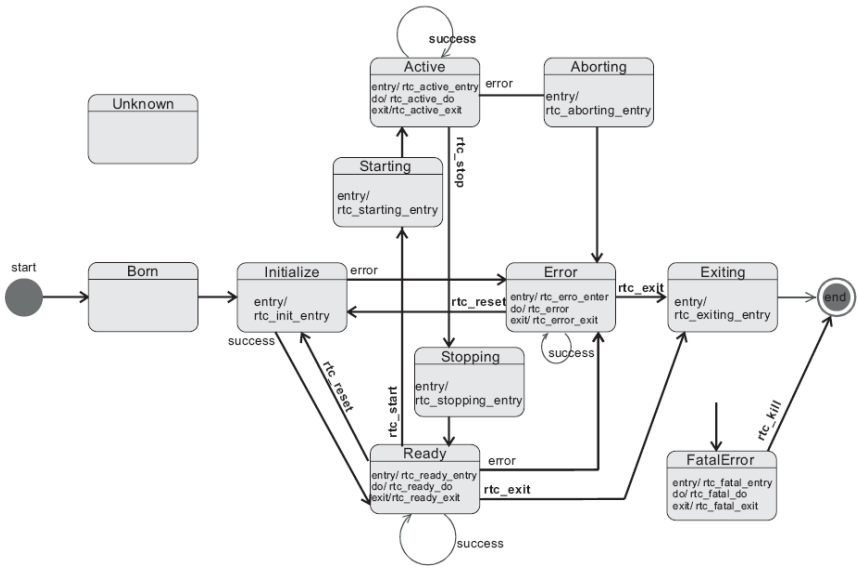

Figure 2
Life-cycle of an OpenRTM-aist component

The communication between RTC Managers is realized with CORBA objects that are called InPort/OutPort. This hierarchical structure and the CORBA system provide the power of OpenRTM-aist.

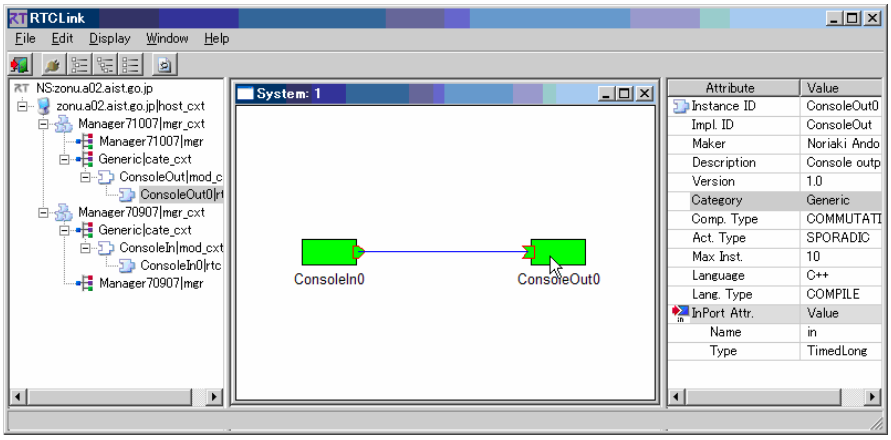

Figure 3
RTC-Link window

In the conventional robot system, a slight modification has required an extensive alteration of software. With RT-middleware, it has been made possible to provide new services by creating a module of new functional part (RTC Manager) for

providing necessary functions and posting it in the network, even when user-requested services cannot be implemented with the existing functional parts.

To create an easily manageable system, a simulating window has been created. This window is shown in Figure 3.

2.8 What is CORBA in OpenRTM-aist?

It's a specific type of distributed system, what introduces Interface Definition Language. CORBA provides transparency of all levels of programming, from object location to protocol transparency. This means that the client doesn't need to know where an object is located; he only knows that it is reachable for the program at any time. The use of the CORBA components in RT-Components is shown in Figure 4.

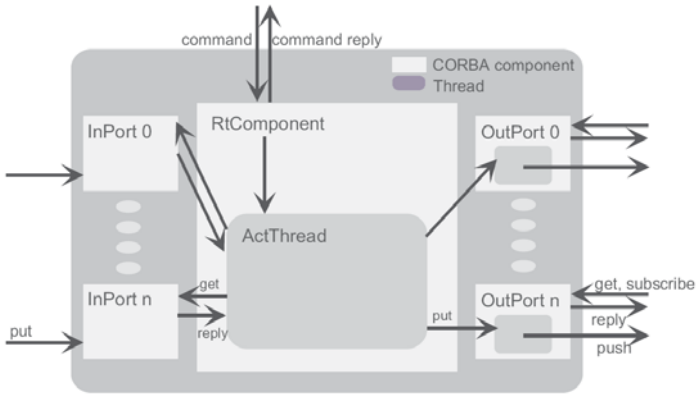

Figure 4
Use of CORBA components in RT-Components

3 General Description

OpenRTM-aist provides many features, but suffers from its early appearance; for example, limited usage of RT-Components, no support for 'on-the-fly' modification.

We adopted OpenRTM-aist framework to Windows platform, so thus programming of robots became simpler. New framework monitoring window was created, and bounds were expanded, which makes changes in robot programs remotely. In this stage of application OpenRTM-aist framework is used for image processing. What explains the name of the application: Distributed Image Analyzer.

3.1 Distributed Image Analyzer (DIMAN)

DIMAN builds on OpenRTM-aist framework, and expands its usability. The system has two types of components: Control Unit (CU) and Processing Unit (PU). In a system there can be one Control Unit and infinite Processing Units. In the view of OpenRTM-aist framework we can explain the Processing Unit's role as RTC Manager's role, but a Processing Unit has the capability of holding many RTC Managers, which are called simply 'modules'.

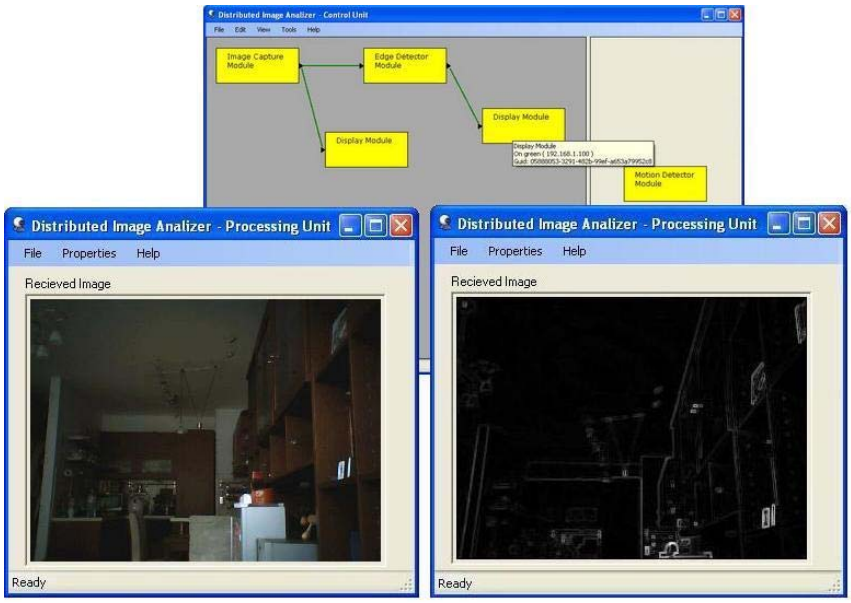

Figure 5
Windows of framework

The Control Unit discovers the Processing Units all over the network. Every Processing Unit answers for the discovery beam, and sends back its module's descriptors (name, attributes, attribute values, description and capability). The windows of the framework are shown in Figure 5.

The discovered modules in Processing Units are listed on the right side. We simply Drag & Drop the modules from the right panel to the left panel (Workspace). After the modules are on the workspace we can connect the input stubs with output stubs, and create a system. After we are finished with the setting up of the system, we can start the simulation with one click, and the flow of data starts immediately. The data process will be done as we specified it earlier on workspace. It is allowed to modify the attribute values of modules before simulation, so we can manipulate the module without physical interaction. The interaction is done remotely, without knowing exact location of module.

The Processing Units are building up from modules (up to infinite). A PU can show any kind of message to user interface, and runs in background. The modules are supervised by PU. Data flow between the Processing Units and Control Unit is shown in Figure 6.

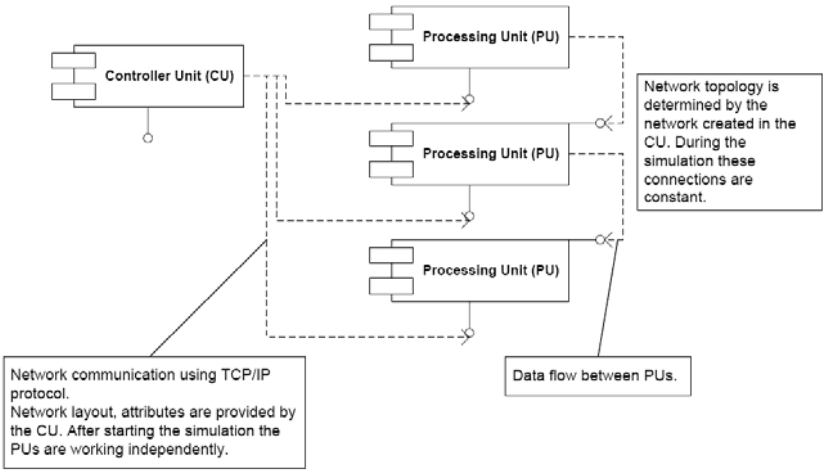

Figure 6
Data flow between framework elements

In modules only the data processing has to be solved (programmed), everything else is solved by the DIMAN framework. This removes the load of programmers, and creates interoperability between modules.

Interoperability is solved by inheriting Core Module Structure, which is a representation of a basic module. In this stage of DIMAN framework some sample basic modules were created, to demonstrate the usability and life-readiness.

The created sample modules are:

- Camera Module, to simulate an eye of a robot
- Edge Detector Module
- Motion Detector Module, to demonstrate an application of image processing
- Color Mode Converter Module
- 2D-3D Converter Module
- Display Module, to display the result of image process

The usability of the modules and the mathematical background (if necessary) is listed below.

3.1.1 Camera Module

As we wanted to use the framework for processing images, it was indispensable to create a module that provides images. Every parameter of a camera (image color depth, image size, image color, sampling rate, saving of captured image, video source) can me modified threw Control Unit. The source of the video can be a local camera (can be on any port), JPEG stream, MPEG stream, TV-TUNER card. The sampling rate defines the rate of capturing images from video stream. The output of the module is a pure image, with properties defined before. This image can be used in many modules, can be an input of any of the listed modules, or can be used in any of robot modules.

3.1.2 Edge Detector Module

To meet your satisfaction you can choose your best fit edge detector algorithm:

- Canny algorithm
- Difference algorithm
- Homogeneity algorithm
- Sobel algorithm

The module will create an 8 bit depth image.

3.1.3 Motion Detector Module

Much kind of motion detection algorithms were implemented. The type of the algorithm can be remotely modified. All of these algorithms are based on common approach of motion detection: comparing previous images to current one.

The result of the module is an image where motion is signed somehow (depends on algorithm).

3.1.4 Color Mode Converter Module

This module provides conversion between color spaces. The supported color spaces are RGB, HSV and YUV. The conversion between colors spaces are done pixel by pixel. The input and the output is a 24 bit depth color image.

3.1.5 2D-3D Converter Module

This is an experimental module, filters edges that are not in the same plain as the picture, and these edges are ordered into 3. dimension. The module creates from a 2D picture a 3D binary tensor or its projection.

3.1.6 Display Module

To see the result of the image processing we must display the synthesized picture. The input of the display module is an 8 or 24 bit depth image, and it is shown in its original size.

3.2 Modularized Structure of Framework

Fully modularized, by program running dynamically loadable libraries were created. As in Linux platform the CORBA is widely used for this, in Windows platform COM[8] is used for the same. The programmer simply puts his newly developed robot controller in the Processing Units Module directory and it is discovered automatically. The module is ready to use immediately.

Not only module development is made so easy, it is possible to create new window environment for both Control Unit and Processing Unit. This capability is given by the view of application development called layer logics.

These layers are in hierarchically:

- Data layer, responsible of data flow
- Business layer, represents business logic
- User Interface layer, responsible for user interaction

Data layer is for communication which is absolutely separated from the other layers, which creates easily developable User Interfaces.

The potential use of the framework will be to support research by allowing fast and modular development of different cognitive functions as modules. Furthermore, cognitive modules can be connected with other modules, cognitive or non-cognitive. Combining different modules may result in a system endowed with the advantages of both conventional and cognitive systems, yielding a hybrid system.

Conclusions

The created framework assures a programmer that he only has to focus only on the data processing (robot manipulation) in modules, and everything else is done by framework. Whatever he creates in DIMAN framework and tests it in laboratory environment with simulation, it will cope with real application in life. Modularity and robustness provides the power of the framework.

Acknowledgement

The authors wish to thank the National Science Research Fund (OTKA K62836), Control Research Group and János Bolyai Research Scholarship of Hungarian Academy of Science for their financial support and the support stemming from the Intergovernmental S & T Cooperation Program.

References

[1] Christopher D. Gill, William D. Smart: Middleware for Robots?, American Association for Artificial Intelligence (www.aaai.org), 2002

8 COM: Component Object Model Technologies

[2] Tetsuo Kotoku: Robot Middleware and its Standardization in OMG, in International Conference on Intelligent Robots and Systems (IROS'06) Workshop on Robotic Standardization (Submitted), Beijing, China, Oct. 11-13, 2006

[3] VIP - A Framework-based Approach to Robot Vision, International Journal of Advanced Robotic Systems, Vol. 3, No. 1 (2006), ISSN 1729-8806, pp. 067-072

[4] Makarenko, A., Brooks, A., Kaupp, T.: Orca: Components for Robotics, in International Conference on Intelligent Robots and Systems (IROS'06) Workshop on Robotic Standardization (Submitted), Beijing, China, Oct. 11-13, 2006

[5] Matthias Kranz, Radu Bogdan Rusu, Alexis Maldonado, Michael Beetz, Albrecht Schmidt: A Player/Stage System for Context-Aware Intelligent Environments, in Proceedings of the System Support for Ubiquitous Computing Workshop (UbiSys 2006), Orange County, California, September 2006

[6] Noriaki ando, Takashi suehiro, Kosei kitagaki, Tetsuo kotoku, Woo-Keun Yoon: RT-Middleware: Distributed Component Middleware for RT (Robot Technology), 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2005), August 2005, pp. 3555-3560

[7] National Institute of Advanced Industrial Science and Technology (AIST), Intelligent Systems Research Institute, Task-Intelligence R,G, Noriaki Ando, http://www.is.aist.go.jp/rt/