We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

20

Robotic Localization Service Standard for Ubiquitous Network Robots

Shuichi Nishio[1], JaeYeong Lee and Wonpil Yu[2], Yeonho Kim[3], Takeshi Sakamoto[4], Itsuki Noda[5], Takashi Tsubouchi[6], Miwako Doi[7]
[1]*ATR Intelligent Robotics and Communication Laboratories, Japan*
[2]*Electronics and Telecommunications Research Institute, Korea*
[3]*Samsung Electronics Co., Ltd., Korea*
[4]*Technologic Arts Inc., Japan*
[5]*National Institute of Advanced Industrial Science and Technology, Japan*
[6]*University of Tsukuba, Japan*
[7]*Toshiba Research and Development Center, Japan*

1. Introduction

Location and relevant information is one of the essentials for every robotic actions. Navigation requires position and pose of robot itself and other positions such as goals or obstacles to be avoided. Manipulation requires to know where the target objects is located, and at the same time, to know positions and poses of robot arms. Robot-human interaction definitely requires where the person to interact with is located. For making interaction even effective, such by having eye contacts, further detailed information may be required. As such, robots need to acquire various location related information for its activity. This means that components of robots need to frequently exchange various types of location information. Thus, a generic framework for representing and exchanging location information that is independent to specific algorithms or sensing device are significant for decreasing manufacturing costs and accelerating the market growth of robotic industry. However, currently there exists no standard mean to represent and exchange location or related information for robots, nor any common interface for building localization related software modules. Although localization methods are still one of the main research topics in the field of robotics, the fundamental methodology and elements necessary are becoming established (17).

Although numbers of methods for representing and exchanging location data have been proposed, there exist no means suitable for robotic services that aim to serve people. One of the industrial robot standards defined in International Organization for Standardization (ISO) defines a method to define position and pose information of robots (6). Another example is the standards defined in Joint Architecture for Unmanned Systems (JAUS) (16) where data formats for exchanging position information are defined. However, these standards only define simple position and pose information on fixed Cartesian coordinate systems and are neither sufficient nor flexible enough for treating complex information required for service robots and modern estimation techniques.

Probably the most widespread standard on positioning is for the Global Positioning System (GPS) (12). GPS provides absolute position on the earth by giving 2D or 3D coordinate values in latitude, longitude and elevation. Although the GPS itself is a single system, the terminals that people use to receive the satellite signals and perform positioning varies. Thus, there are variations in how GPS receivers output the calculated position data. One of the most commonly used format is the NMEA-0183 format defined by National Marine Electronics Association (NMEA) (13). However, as NMEA format only supports simple absolute positions based on latitude and longitude, it is not sufficient for general robotics usage. Another related field is Geographic Information System (GIS). GIS is one of the most popular and established systems that treats location information. In the International Organization for Standardization, many location related specifications have been standardized (for example, (7)). There already exist versatile production services based on these standards such as road navigation systems or land surveying database. However, current GIS specifications are also not powerful enough to represent or treat information required in the field of robotics.

In this paper, we represent a new framework for representing and exchanging Robotic Localization (RoLo) results. Although the word "localization" is often used for the act of obtaining the position of robots, here we use for locating physical entities in general. This means that the target of localization is not just the robot itself, but also includes objects to be manipulated or people to interact with. For robotic activities, mere position is not sufficient. In combination with position, heading orientation, pose information or additional information such as target identity, measurement error or measurement time need to be treated. Thus, here the verb "locate" may imply not only measuring position in the spatio-temporal space.

Our framework not only targets the robotic technology available today but also concerns of some near future systems currently under research. These include such systems as environmental sensor network systems (14), network robot systems (4) or next-generation location based systems. Figure 1 illustrates a typical network robotic service situation where localization of various entities is required. Here, a robot in service needs to find out where a missing cellular phone is by utilizing information from various robotic entities (robots or sensors) in the environment. These robotic entities have the ability to estimate the location of entities within their sensing range. Thus, the problem here is to aggregate the location estimations from the robotic entities, and to localize the cellular phone in target.

Since 2007, the authors have been working on the standardization of Robotic Localization Service. This is done at an international standardization organization Object Management Group (OMG). OMG is an consortium widely known for software component standards such as CORBA and UML. As of May 2009, the standardization activity on Robotic Localization Service (RLS) (15) is still ongoing and is now on its final stage. The latest specification and accompanying documents can be found at: `http://www.omg.org/spec/RLS/`.

In this following sections, we will describe elements of the new framework for representing and exchanging localization results for robotic usage. We first present a new method for representing position and related information. Items specific to robotics use such as mobile coordinate systems and error information are described. We describe several functionalities required for exchanging and controlling localization data flow. Finally, some example usages are shown.

2. Data Architecture

In this section, we present a new method for representing location data and related information that is suitable for various usages in robotics, which forms the core part of the proposed

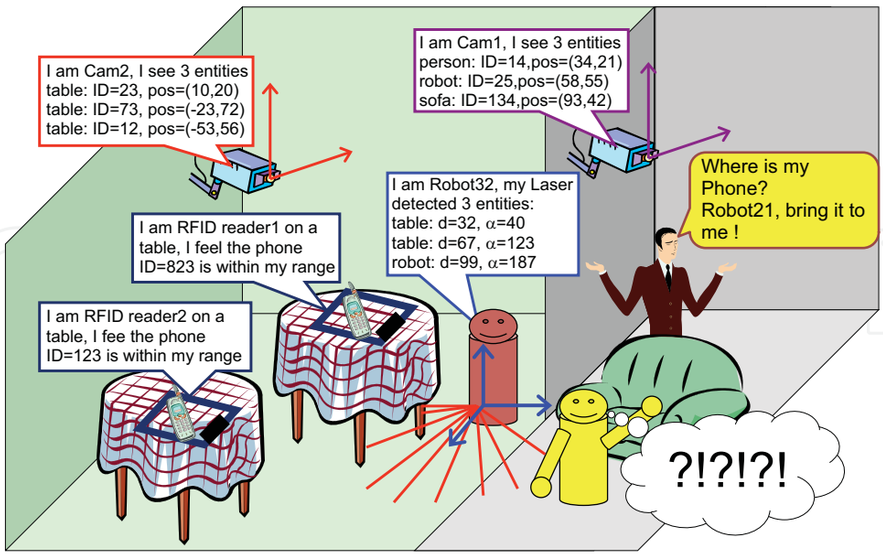


Fig. 1. Example of a typical robotic service situation requiring location information exchange (from (15))

framework. Modern robotic algorithms for position estimation or localization require more than simple spatial positioning. Various types of information related to the measurement performed are also required for precise and consistent results. One obvious example can be seen when combining outputs of laser range finder (LRF) and odometer installed in a mobile robot. When the robot turns around, LRF measurement values quickly change. If the two sensors are not temporally synchronized, the combined output will result in a complete mess (18). In order to obtain precise results, measurement time and error estimation is crucial for integrating measurements from multiple sensors. Pose information is also important. When grasping an object or talking to a person nearby, robots always need to obtain in which direction they or their body parts are heading. When sensors in use can perform measurements of multiple entities at once, target identity (ID) information is also required. For example, when vision systems are used to locate people in an environment, the system needs to track and distinguish people from each other. As such, there are numbers of information to be treated in combination with simple spatial location. In order to make various robotic services treat and process these versatile information easily and effectively, our idea is to represent these heterogeneous information within a common unified framework. As stated before, the proposed framework is defined by extending the existing GIS specifications such as ISO 19111(7). In the following sections, we describe three extensions required for robotics usage. And then we describe a generic framework for composing structured robotic localization results.

2.1 Relative and Mobile Coordinate Reference Systems

In general, spatio-temporal locations are represented as points in space. Cartesian coordinate is one typical example where location is defined by a set of numeric values that each represent the distance from the origin, when the location is projected to each axis that defines the coordinate system. As described in this example, locations are defined by a combination of information: a coordinate value and the coordinate system where the coordinate value is based on.

Before going further, let us clarify the terms used in the following sentences. A *coordinate system* (CS) is a system for assigning an n-tuple of scalar values to each point in an n-dimensional space. Mathematically, a scalar is in general an element of commutative ring, but we do not apply this restriction here. Instead, each of the tuple element is allowed to be taken from arbitrary set of symbols, as explained later. Normally this set consists of rational numbers. A *coordinate value* denotes the n-tuple of scalars assigned with respect to a CS. In this document, we will assume that every coordinate value is related to a CS, through which it was assigned. That is, every coordinate values are typed to be a member of an element set that belongs to a certain CS. Note that, there exists no uncertainty with coordinate values themselves. The uncertainty (or error) with the observation or estimation, if any, is represented by another accompanying value. We will call this an *error value* (or error in short).

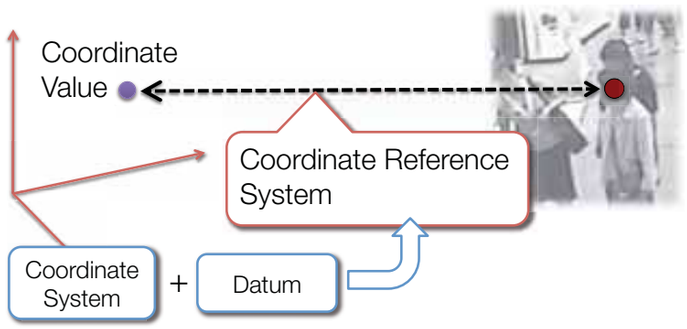


Fig. 2. Coordinate Reference System (CRS)

Figure 2 shows how a position in real world is mapped to a coordinate value defined over a certain CS. This mapping is typically done through some observation or sensing. However, in order to obtain the measurement values in a consistent manner, some rule must be defined on how to associate the coordinate values to the real world position. This mapping or grounding rule is called *datum*. With both the CS and datum defined, the mapping function from real world position to coordinate values can be defined. In other words, in order to define a perform an observation of real world phenomena, a combination of CS and datum is required. This combination is called *coordinate reference system* (CRS).

The basic idea in GIS specifications is that every CSs used for representing position data are fixed relative to the earth (i.e. referenced). There exists descriptions for relative coordinate systems in GIS standards (e.g. *Engineering CRS*), but they are hardly used and the usage is not clear. In robotics usage, however, CSs are not always fixed, and in many cases they are also mobile. That is, its relation to the earth changes by time. Although it may not be impossible to express every data in some global, fixed CSs, in most cases, it is much convenient to treat data in a relative form. There exists a GIS specification recently published (8) which specifies a method for describing moving entities. However this method is mainly aimed for car navigation that assumes the localized objects to move alongside some predefined roads and is not easy to use in robotics usage.

Especially in mobile robots, CRSs defined on a moving robot change its relation with other CRSs in time. For example, imagine that there are two rooms, room A and room B, and a mobile robot equipped with a 3-degree-of-freedom hand. When this robot grasp and move some objects from room A to room B, at least one CRS that represents the area including two rooms and one CRS that moves along as robot navigates are required. In some cases,

each room may also have an individually defined coordinate space, related to the 'global' coordinate space representing both rooms in common. Moreover, in order to represent the gripper location at the end of the robotic hand, several CSs must be defined over the robotic hand each related to other coordinate systems by some means such as Denavit-Hartenberg convention (1). The object to be gripped and moved by the robot may also hold some CRSs that indicate the position or the pose of the object. When the object is carried by the robot, these CSs also shift in space as the robot moves.
As can be seen from this example, not all the CRSs used need to be grounded to the earth or the 'global' CRS in use. Requiring users to strictly define datums for every CRS in use is not realistic. Also, some mechanism for easily process CRSs on a moving platform and for transforming coordinate values on it to some static CRSs on demand is required. In the proposed framework, a relative coordinate reference system is defined as a CRS where the relation with the fixed world may be not known at some instant or users have no interest in referencing it to other CRSs. A mobile CRS is defined as a relative CRS with an dynamic datum referring to output of different localization output (figure 3).

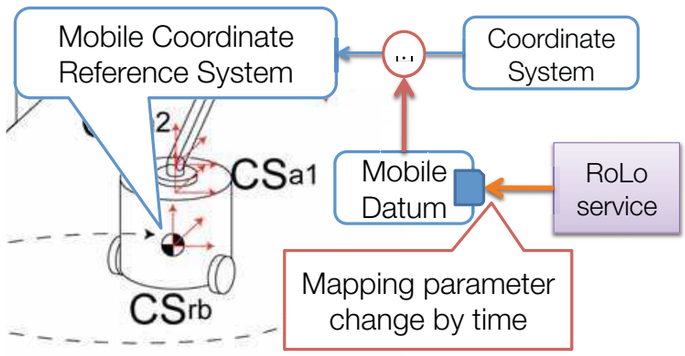


Fig. 3. Mobile CRS

2.2 Identity Representation

Identity (ID) information which is assigned for localized targets, can also be treated as a value on some CS. For example, MAC addresses used in Ethernet communication protocols can be represented as a coordinate value on a two-dimensional CS, vendor code and vendor-dependent code (5). Electric Product Code (EPC) (3) used for identifying RF tags is another example of identification systems defined by multi-dimensional coordinate system. There also exists some ID systems, such as family names, that is usually not explicitly defined over some mathematical structure.
In general, sensors hold their own ID system, and each observed entity are assigned an ID from this local ID system. This is because, at least on the initial stage, there are no means to identify an observed entity and assign it a global ID. Thus, when multiple sensors are in use, there exist multiple local ID systems independent to each other, and it becomes necessary to properly manage and integrate these ID systems (ID association problem). Also as previously described, ID assignments are probabilistic, just like other location information.
From these considerations, we can say that ID information requires representation or access methods similar to other types of location information. Thus, we propose to treat ID information in the same manner as spatial positions or other location information, as a value on a CS.

Since in GIS specifications CSs cannot handle axis defined over a set of symbols or discrete set of numbers, we extend this point. Note however, that some operations such as comparison is not always defined over this axis, as symbols in ID systems do not form an ordered set in general. Also, transformation between ID CSs will likely to be defined as a conversion table, not an mathematical operation.

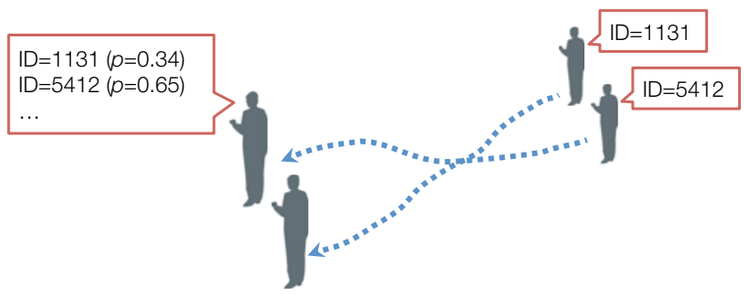


Fig. 4. Identity Association Problem

2.3 Error Representation

Error (or uncertainty, reliability) information plays a important role in robotic algorithms. This is especially important when localization results are used for further processing such as sensor fusion or hi-level estimation. Thus, measurement or estimation errors are one of the most essential features required for robotics usage. In GIS specifications, only static information of expected error concerning inter-coordinate transformation can be stated. Thus, here we extend the GIS specification in the following points:

- Localization results can be attributed an error information.
- Allow errors to be represented in various forms.

Just like every location information is associated with some coordinate (reference) system that defines what the information represents, every error information is associated with some error type. Modern measurement or estimation techniques require versatile forms of error representation depending on what computation methods are used or what devices are used. These include reliability, covariance or probability distribution. Distributions are typically approximated by finite combination of simple distributions, such as a single Gaussian distribution or mixture of Gaussians. Distributions may also be represented by random sampling such as by Monte Carlo method. Thus, rather than fixing how error information are represented to a single way, it is better to define a framework that allows multiple forms of error representation and allows users to extend necessary forms if necessary. Figure 5 shows some predefined error types that are commonly used in current localization techniques. We have designed the framework to be extendable so that users can extend their own error type if necessary.
In some cases, a single error information may be related to multiple position data. In such cases, a special structure for describing such relation is necessary. This structure is described in the next section.

2.4 Describing Complex Data Structure

Up to now, we have defined necessary elements for describing individual information required in robotics usage. The next step is to provide a mean to describe complex data struc-

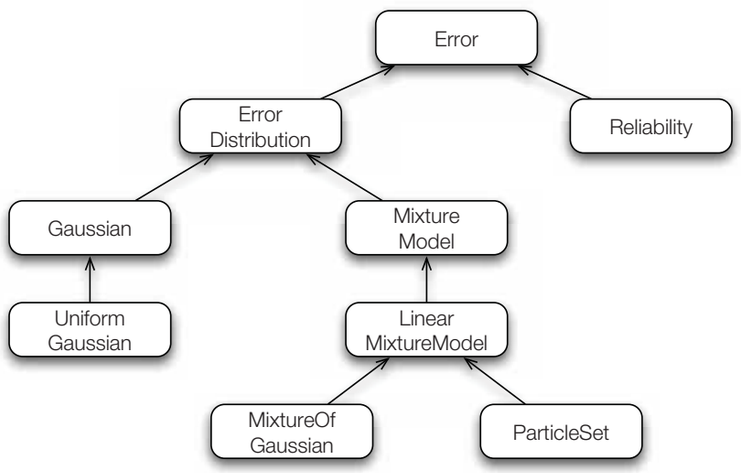


Fig. 5. Hierarchy of predefined error types

ture that consists of multiple measurements or estimation results. This combination is often required as it is quite rare that only a single type of measurement is obtained as a result of localization; in most practical cases, measurements such as position, velocity, ID and pose are given out. Also multiple sensing equipments are often used in order to increase robustness of the measurement or resulting output. This is also true if we assume a situation as described in Figure 1, where numbers of environmental sensors or robots are utilized and combined to perform a single task.

Figure 6 shows an example of an combined data definition. Here three types of information are combined: measurement time, target ID and spatial position. As such, users can define their own data structure that contains arbitrary numbers of necessary data elements. In this case, each element contains an error information. However note that errors are not mandatory, and if unnecessary, users can safely omit the error part both in definition and in the actual data set.

In defining a data structure, CSs that each values are based on, are kept independent to each other and individual values remain in its original form. This means that, definition of each values are kept in their original form, and the containing structure defines their relation with each other elements. In other words, multiple information are represented in combination to be suitable for certain usage, still remaining the 'meaning' of each individual values. Note that, this specification does neither oblige the users to specify information of some 'meaning' nor restrict the 'meaning' of information expressed by RoLo Data Specification. For example, the spatial coordinate in the above example may represent the centroid of the robotic body, or it may represent the position of a robotic arm. The meaning of each coordinate information contained in RoLo Data Specification definitions are out of the scope of this specification. Only the users and provider of the output module needs to agree in how each coordinate information will be interpreted.

Normally, error information is associated with one main location data. However in certain cases, there is a need to hold an integrated error among multiple data. For example, in a typical Kalman filter usage, multiple measurements such as spatial position and velocity are used to form a single state vector. When the elements of the state vector are not independent, which is the usual case, errors include cross effect among multiple measurements and

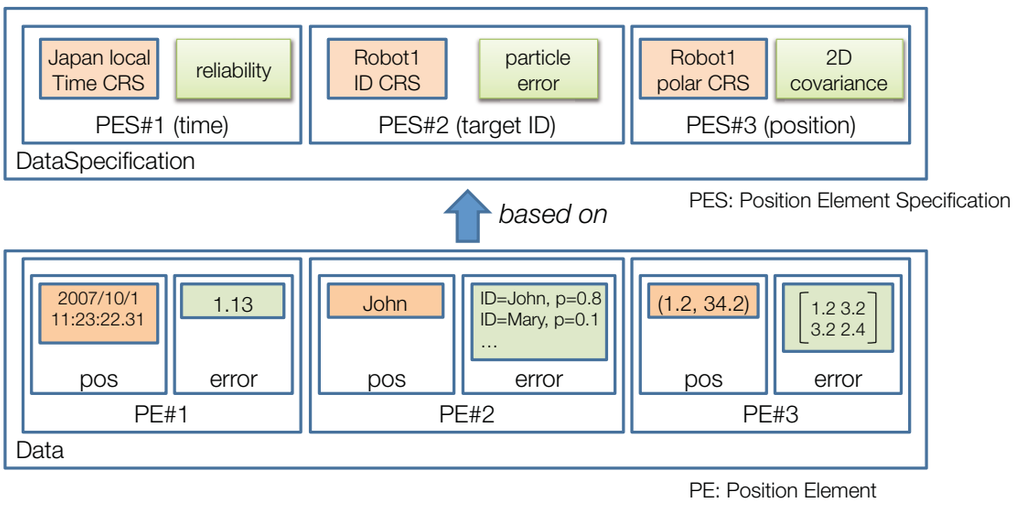


Fig. 6. Sample of RoLo Data Specification

are often represented as a covariance matrix. In such case, the Error Element Specification instance specifies which main information slot the error is related to, and the actual error data is contained by Error Element instances (Figure 7).

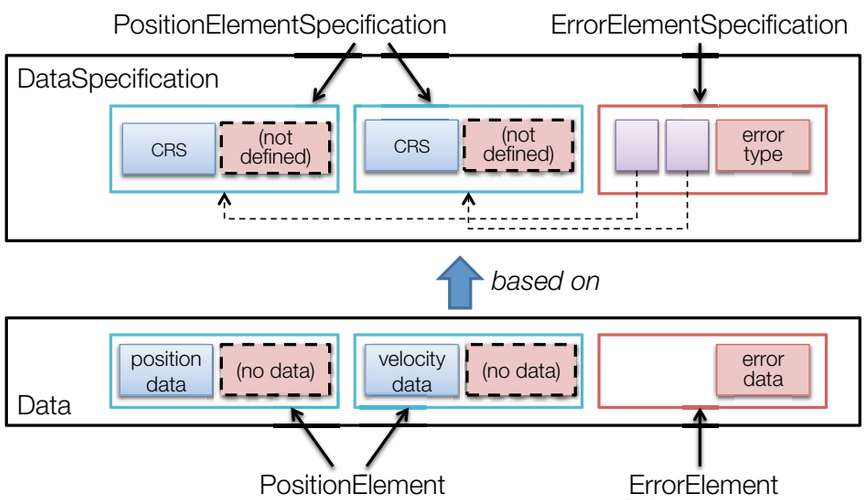


Fig. 7. Sample of Complex RLS Error definition

2.5 Don't-Cares

Consider that you want to develop a database system with RLS interface which accumulates results from numbers of people tracking modules. These modules are measurement modules corresponding to sensors of identical type installed in different locations (Figure 8). Being installed in different location means that each camera output is bound to a different CRS, *i.e.*, same coordinate system but different datum. As the sensor hardware are the same, and the DB system will not see the datum for each data, you may want to develop a generic interface that

accepts data from all of the sensor modules. However, as stated later, RLS interfaces can only be bound to a finite number of data specifications. Thus you cannot make a generic interface which may accept infinite variations of coordinate reference systems. Would you give up and go on with the tedious development of interfaces for every newly added sensor modules?

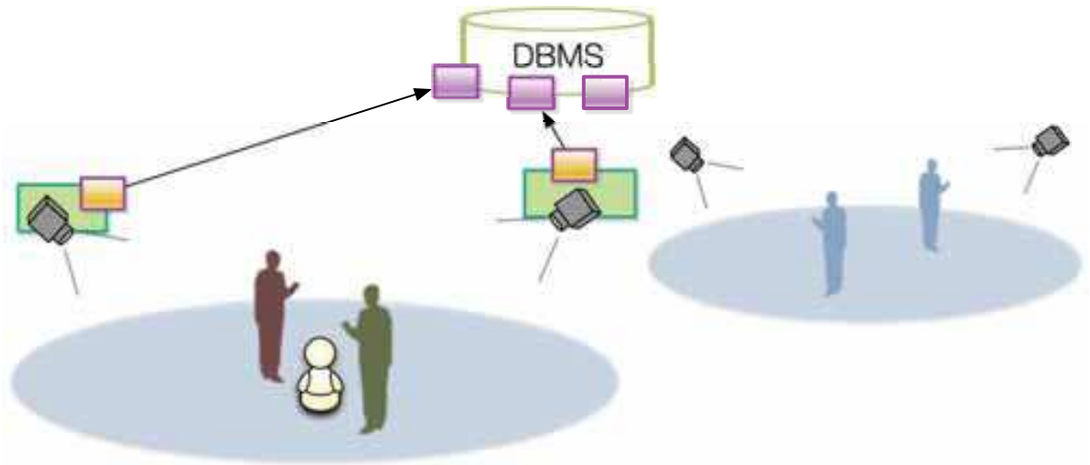


Fig. 8. Example situation where don't-care elements are required

As such, there are often cases that you are not interested in some part of the data specification. You want to just pass-through those parts. That is, you don't care whatever elements are specified in the uninterested parts of the data. *Don't-care elements*, similar to the usage in automata theory, are prepared for such usage. In the above example, you specify a data specification for the database's input stream ability with a coordinate reference system that contains a don't-care datum. This way you can specify only the specification parts you (the module) are interested, and leave the other parts unspecified.
Issues similar to the above example can quite often be seen, so the use of don't-cares will increase the flexibility and usability of the service. However, this use of don't-cares may require notice as they are quite likely to result in high computation requirements not suitable for systems with limited resources. Also, don't-cares may lead to ambiguous, useless specifications that break the idea of having specifications for data. Therefore in the specification, some rules are defined to prohibit misleading usages and to avoid unnecessary costs.

3. Data Format

In the previous section, we have showed a framework for defining and holding complex localization data. However, when exchanging information amongst modules, knowledge on data structures is not enough. For example, data may be exchanged in XML, in comma-separated values or in some binary format. Roughly speaking, the relation between data specification and data formats are similar to Saussurian relation between signifié and signifiant. The mapping from data specification to data format is basically arbitrary. That is, the same data may be represented in numbers of ways. Thus, in order to exchange data between different systems or modules, we need to specify how data is represented in addition to how it is structured and what it means. Here, data formats are means to indicate how data is represented in a machine readable form.

Generally, when defining a specification, there are two approaches about data formats. One is to fix the data format to be used, and another is to define a meta-specification for describing the variety of formats. Specifications such as NMEA-0183 (13) or JAUS (16) are example of the first form. Fixing the way data are represented can lead to a simple and rigid specification. However, the extendability is in general low. On the other hand, some specifications such as ASN.1 (10) allows several data formats to be used for data exchange. This provides flexibility and extendability, but often leads to difficulty in implementation.
On designing RLS specification, there were three requirements: 1) Make the standard to be extendable so that the standard can be used with forthcoming new technologies and devices. 2) Allow existing device/system developers and users to easily use the new standard. 3) Maintain minimum connectivity between various modules. As for 1) and 2), we decided to include both of the two approaches for data formats. As for 3), we prepared the 'common data format'.
For exchanging robotic localization data, we can think of roughly two categories of data formats. The first category is about data formats that are independent to data specifications. This category is for the first requirement given above. Data formats in this category includes those specified by some systematic rules. Encoding rules such as XER(9) or PER(11) are examples of such data formats. Comma Separated Values (CSV) is also another example of such that is widely used. These rules generally have the ability to map a wide range of data structures to data representations such as XML or bit sequences. Based on the defined rules, data formats specific to the data structure in use can be generated. In this sense, we can think that this category of data formats are independent to data specification. Another category, for the second requirement, is about formats bound to some specific data specification. That is, formats are defined over some target data specification. Most of the sensor device outputs in market these days uses formats of this type. Usually, reference manuals for these devices describe data structure and how they are given out in combination.
In the RLS specification, both categories of data format are supported. In order to clarify what data format is used, data format information is provided implicitly or explicitly in the access interface. Details are described in the next section. In some cases, users or developers do not need to care about data formats. For example, when sending messages with arguments between C++ objects, there is no need to think about formats. The compiler will take care of the actual data packing. The same can be said for distributed systems with IDL definitions. However in some cases such as receiving raw outputs from sensors or reading data files, data format information is essential.

3.1 Common Data Formats

Think of a situation where two foreigners meet together. When both of them cannot speak others' mother language, how shall they communicate with each other? Often in such case, they will choose to talk in a third language that both can speak and understand, such as English, even if not fluent. It is always better to have something than nothing.
Common data formats are aimed to be something like the third language in this example. They are for maintaining minimum connectivity between heterogeneous RLS modules. The RLS specification defines three common data formats each accompanied with two data specifications. These combinations were chosen from the most frequently used CSs in robotics, Cartesian CS, polar CS and geodetic (GPS) CS. Figure 9 shows an example of common data format definition.

Table 71 - Common data format type II-2 (Spherical Coordinate System, ZYX-Euler Angle Representation)

Parameter	Format of value	Value type	Unit
Position	[r, θ, φ]	Real, Real, Real	meter, radian, radian
Orientation	[yaw α, pitch β, roll γ]	Real, Real, Real	radian, radian, radian
Timestamp	POSIX time	Integer, Integer	second, nanosecond
ID	--	Integer	--

Fig. 9. Example of common data format definition (from (15))

The specification requires every RLS module to support at least one of these common data formats and accompanying data specification. Thus, even in cases where two modules has no common way to exchange data, they can always 'fall back' to these data formats. They will not be able to transmit detailed results, but can exchange the least amount of information. Such fall-backs are assumed to be especially important on near-future network robot usages as in figure 1, as robots need to get as much as possible from variations of modules situated in different environments. We can also see a similar example in today's home appliances. Recent appliances are equipped with advanced plugs such as HDMI and so on for transmitting hi-definition videos or digital audios, but older equipments are not. When connecting newer appliances to older ones, you use traditional connectors such as the yellow RCA plug. You may not get the best out of it, but at least there's something on the screen.

4. Interface

In this section, we will describe how RLS modules can be accessed. As stated in the introduction, one of our goal was to make a scalable specification. That is, a specification that can be used not only for complex, large-scaled systems but also for small embedded systems where available resources are limited. However, as our original purpose was to compile a specification which can be used in near-future network robot systems, module interfaces must be able to handle intelligent requests.

4.1 Module Structure

In general, several types of modules are commonly used for treating location data in robotic systems. The simplest form of module is which receives data from sensors, calculates location and outputs the results. However this type of interface strongly depends on sensor interfaces or sensor output formats. Strong dependency on specific products or vendors is not suitable for standardization. Moreover, when a location is calculated, many kinds of resources such as map data, specific to each sensing system, are required. It is impractical to include each of these resources into the standard specification. Thus, we decided to embed and hide the individual device or localization algorithm details inside the module structure.
On the other hand, if we focus on functionalities required to localization modules, we can classify them into roughly three classes (figure 10):

- Calculate localization results based on sensor outputs (measurement)
- Aggregate or integrate multiple localization results (aggregation)
- Transform localization results into different coordinate reference systems (transformation)

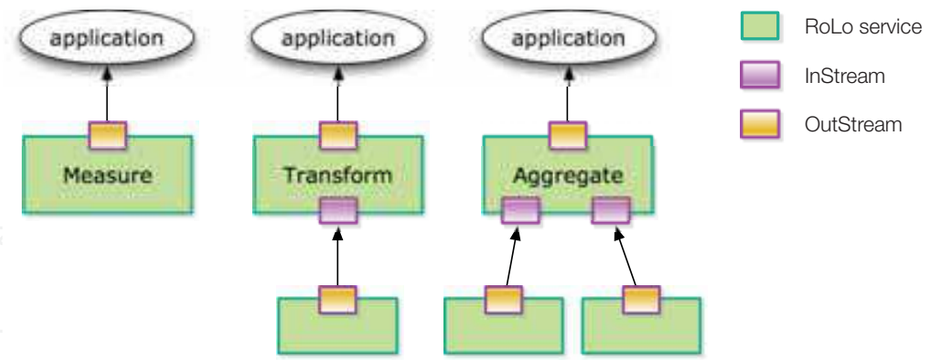


Fig. 10. Three forms of typical RLS module usage

These functionalities differ in their internal algorithms or the number of input / output streams. However, in all of these, the main data to be exchanged is localization results. As we are focusing on the interface of RLS modules, and not on their functionalities, we decided to abstract these different types of modules into a single form of module. This abstract module holds n input streams and a single output stream. By abstracting various types of modules and assuming a uniform interface, complex module compositions such as hierarchical or recursive module connections can be easily realized.

4.2 Module Ability

If each module can represent what it can perform, or provide information on available configurable parameters, a large amount of development efforts can be reduced. By defining the "meaning" of parameters, the ambiguity in functional definition or parameters can be eliminated, resulting in increase of developing efficiency. Moreover, advanced features can be implemented such as verification of inter-module connection, automatic search of specific modules or semi-automatic parameter negotiation between modules. In future network robot environments where sensors or robots are distributed in the environment and cooperate with each other, it becomes essential to register each module's capabilities in repositories and make them searchable.

As a basis for realizing such functionality, we have also defined a facility to describe *ability* of each modules or streams. In definition, each stream owns an ability description, which shows how this stream can perform and be configured. This includes the list of Data Specifications or Data Formats that this stream can handle. Also it describes configurable parameters specific to the module. Each service module also owns an ability description for the service it provides, besides the ability description for its streams. The configurable parameters defined in the ability description can be specified values via the module interface, restricted by the ability description held by the service module or the belonging stream modules.

5. Using Robotic Localization Service

In this section, we will introduce how RLS modules can be used. Typical steps of using RLS Services are as following:

1. (optional) Obtain ability description
   This can be performed by calling the 'getAbility' method toward RLS service or stream.

This step can be omitted if users already have sufficient information such by reading reference manuals.

2. (optional) Set up parameters
   This is done by calling the 'setParameterValues' method. If the default settings are sufficient or if there exists no parameter to be configured, this step can be omitted.
3. Establish connection
   Connection establishment can be initiated from both side; either from the service that outputs data (OUT service) or from the service that accepts data inputs (IN service).
4. (optional) Set initial data
   Robots often require initial data setting. For example, when you bring a mobile robot to a room and power it on, the first thing you need to do is to let the robot know where it is located and to which direction it is heading. By obtaining these, the robot can establish reference from its own mobile CS to the coordinates in the room. Initialization is performed by calling the 'adjust' method.
5. Perform data passing
   Data passing is the main aim for RLS modules. Two types of data passing are defined, *PUSH mode* and *PULL mode*. In PUSH mode, data passing is triggered by OUT service and in PULL mode, IN service triggeres data passing.
6. (optional) Perform adjustment
   Occasionally while passing data, perform adjustment if necessary. Adjustment is an act to provide auxiliary information to the target module for improving localization result.
7. Disconnect from service
   Shut down the connection and disconnect from the target module.

5.1 Scenario 1: Simple Usage

The first scenario is about a very simple navigation robot whose purpose is to reach a predefined goal. The robot repeatedly estimates its position in the space, performs path planning and move. This repetition of seek-and-act is necessary as both estimation and motion imply errors. Figure 11 shows a sample block diagram for this robot and the RoLo data specifications used.

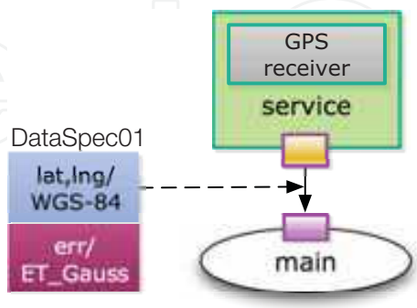


Fig. 11. [Scenario 1] Block diagram and data specification

The robot is equipped with a single a GPS receiver which is embedded in the 'service' module. This module outputs geodetic position (latitude and longitude) and an error modeled as an uniform normal distribution, based on the measurements from the GPS receiver. Note that

here the GPS receiver is embedded in the module, as its interface is not based on RLS specification. However, we can also think of a RLS module which outputs GPS measurement results in NMEA format or some vendor-specific binary data format. These can be realized by coding a wrapper for RLS interface API and by defining some data specifications and data formats describing necessary data structures and formats.
Figure 12 shows a sample code fragment for the 'main' process. In this example, the 'service' module

```
1  #include <RoLo.hpp>
2  extern Service *service;
3
4  class MyInStream: public InStream
5  {
6  public:
7    void setData(const Data& d){ calc_n_move(d); }
8    ...
9  };
10
11 int main(int argc, char **argv)
12 {
13   MyInStream inStream;
14   ...
15   OutStream *outStream;
16   try {
17     service->connect(inStream, outStream);
18   } catch (Returncode_t r){  }
19
20   outStream->activate();
21   ...
22   outStream->deactivate();
23   outStream->disconnect();
24 }
```

Fig. 12. [Scenario 1] Sample code

5.2 Scenario 2: Environmental Sensors

The second scenario describes a vacuum cleaning robot in a room that can return to a charging station when it finishes cleaning or go to the position that is located by a person as shown in Figure 13.
The robot is assumed to be able to calculate its position by combining a position information from an embedded odometer and from several landmarks in the environment. There is no restriction on the landmarks in this scenario and they can be obtained from any type of sensor such as cameras or laser range sensors. It is also assumed in this scenario that the position of a charging station and a pointer can be calculated by using a location sensing system in the environment such as RF beacons.
Figure 14 shows a block diagram that describes a structure of RoLo services and RoLo data specifications for each service modules. In this block diagram, each RoLo data specification shows time, identification, location and error parameters. In Figure 14, 'DataSpec01' is the RoLo data specification for the odometer where 't' represents the time-stamp in UTC format

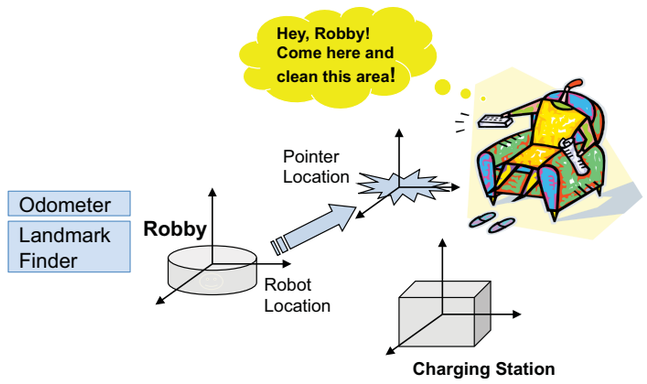


Fig. 13. Scenario 2

and 'x,y,α' the 2D position with a reference CS denoted by 'pCRS1' and 'err' the estimation errors with an error type denoted by 'ET01'. 'DataSpec02' is the RoLo data specification for the landmark finder where 'id' is the identification parameter with a reference coordinate system denoted by 'iCRS2' and 'x,y,z' the 3D position with a position reference CS 'pCRS2' and other parameters are the same as in 'DataSpec01'. 'DataSpec02' is for the robot and the type of parameters are the same as in 'DataSpec01'. 'DataSpec04' is for the beacons and 'id' is the identification parameter with a reference CS denoted by 'iCRS4' and 'd' is the distance between the beacon and the object that is to be localized using the beacons such as the charging station. 'DataSpec05' and 'DataSpec06' have parameters representing time-stamp, 3D position and position errors in the same manner with other RoLo data specifications.

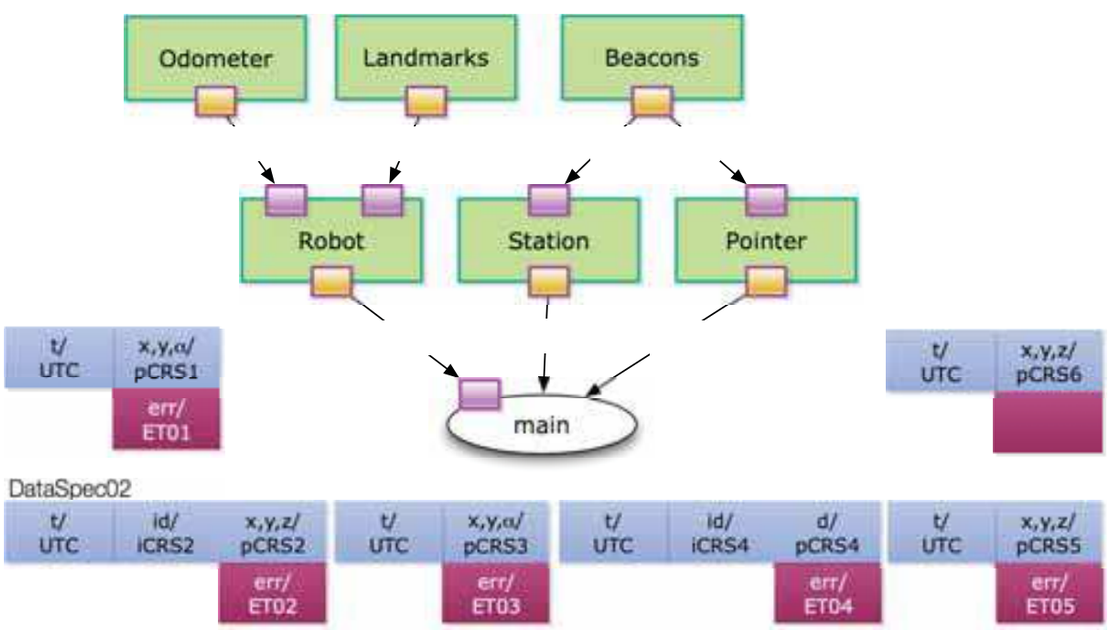


Fig. 14. [Scenario 2] Block diagram and data specifications

Figure 15 show an exemplary sequence diagram for the main part of this scenario. Here, the 'main' application uses PUSH mode to get the location of the robot and PULL mode to get the location of the charging station and the pointer.

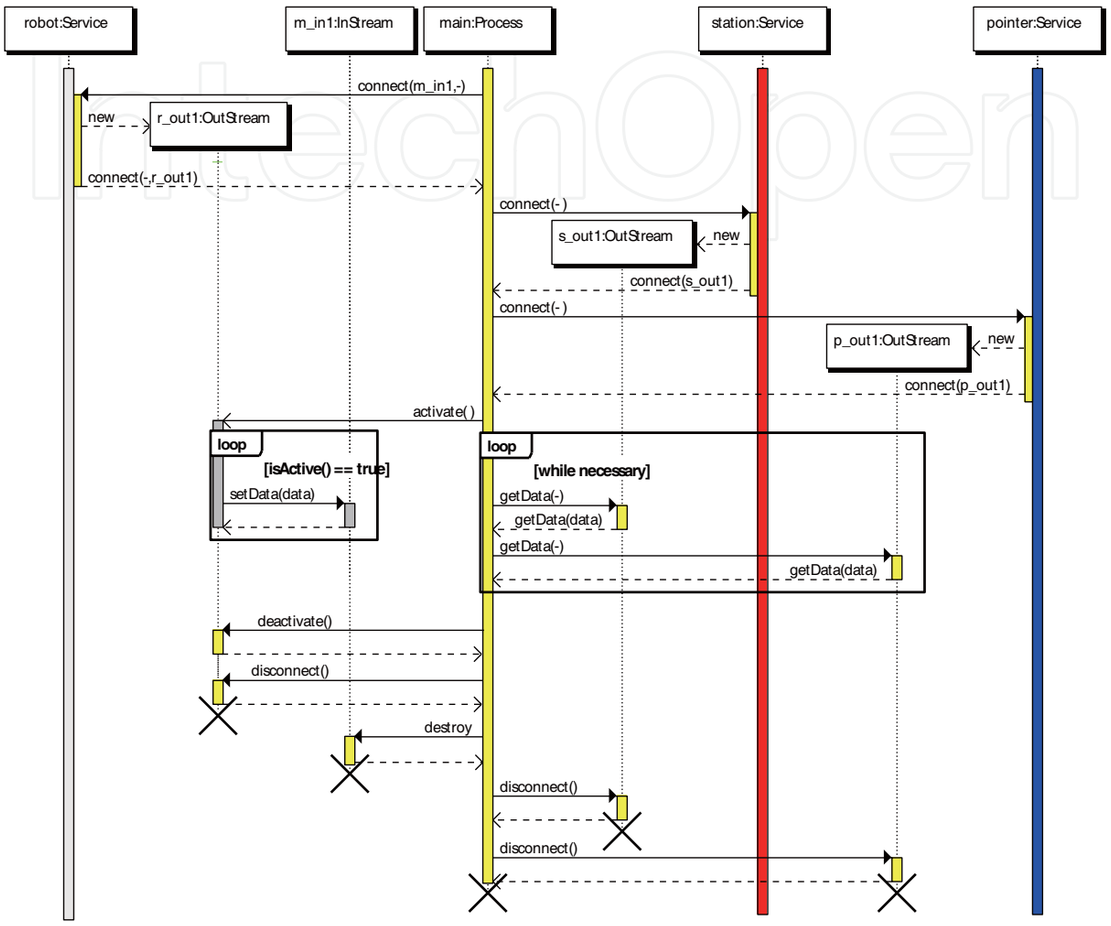


Fig. 15. [Scenario 2] Sequence Diagram (Main Part)

5.3 Scenario 3: Ubiquitous Network Robot

Figure 16 shows a situation where numbers of robots cooperate with each other to assist people over multiple locations. In this example, a robot navigates a person from one environment to another environment. The robots finds a person who needs help, approaches the person, searches for the goal location and starts navigating with the person. In order to perform these tasks, the robot needs to cooperate with sensors and information systems in the environment. Especially in this case, not just those in the surroundings, but also with those in different environments. Also, the robot needs to service the person seamlessly while navigating along two (or more) environments. Such wide-area robotic system is called *Ubiquitous Network Robot* system.

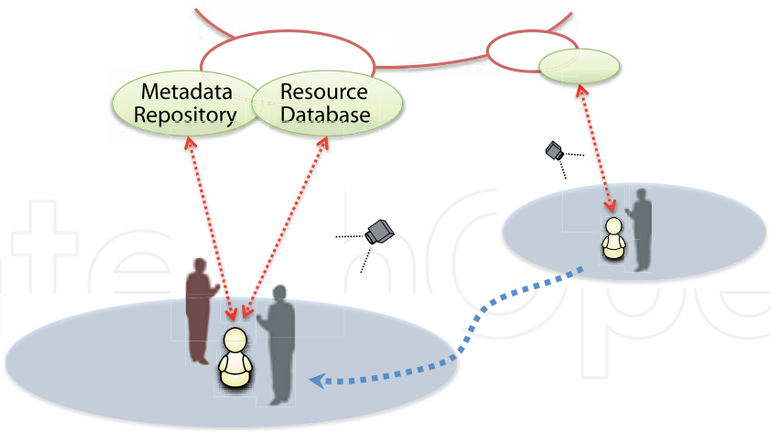


Fig. 16. Scenario 3

Upon entering a new environment, the robot needs to ask the management system there for a list of available RLS modules, and to find out which module is suitable for its use. This search and examination will be performed by using the ability descriptors of each module. Here, we need to assume for additional specifications for the management system: that is, registering and searching ability descriptors for the modules the system manages. Such specification is out of scope of the RLS specification and will be considered for future standardization. One such example is the one implemented in the *Kansai Environmental Platform* (14). Although this environment consisted of only a single environment, a resource management system was implemented experimentally. The steps for robots were as follows:

1. Robot (R) asks resource manager (RM) for a certain type of RLS modules
2. RM searches its database and returns a list of available modules
3. R obtains ability descriptor for each of the modules and examines which to use
4. After decision, R connects to target modules and starts setting up parameters
5. If parameter setup was successful, R activates the modules and start receiving (or sending) data

Figure 17 shows a sample block diagram for scenario 3. As robots move to different environment, the block diagram changes by connecting to modules specific to each environment. As can be seen from these flow and diagram, the robot and the environment need to have a common 'knowledge' for performing device search or parameter setting. For example, if the search request contains devices or CS definitions unknown to the robot, will the robot be able to 'understand' what or how the device can give out? Generally speaking, this requires exchange and bridging of ontologies from heterogeneous systems and currently there are no firm method for performing these. However, if we can limit the target domain, such as to robotic localization, knowledge exchange between heterogeneous systems may become possible. As every data following our specification is related with its formal definition, implicitly or explicitly, our idea is that through accessing a repository of definitions, it will become possible to exchange 'meanings' alongside with data. Such functionality is essential for ubiquitous network robots that provide services seamlessly and continuously in multiple locations. This means that wherever you bring robots, the robots will obtain information necessary from the

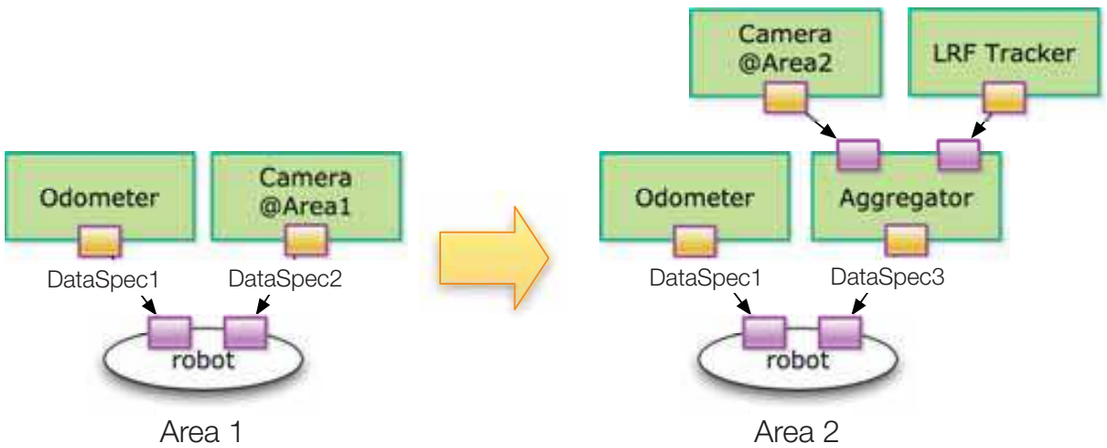


Fig. 17. [Scenario 3] Block diagrams in different areas

environment, automatically adjust to the place and start working. That is, the *Robot Plug-and-Play* (2).
Also note that, although the example here describes the usage for robots, this is not limited to robots only. Applications such as people navigation have similar requirements. Ability search and exchange facilities can thus be said to be one of the key requirements for next-generation location based applications where sensors and information sources local to each environments are actively utilized.

6. Conclusion and Future Works

In this paper, we have presented the concept behind a new framework for treating and exchanging location information in network robotic systems. One of the issues regarding frameworks of this kind is how vendor-specific functionalities or raw data can be represented. Although in this paper we have focused on describing the representation of localization data, our framework includes elements such as data format or vendor-specific parameter specification that can be utilized for treating vendor specific data or interfaces. However note that most of the existing outputs (including raw sensor data) follows some rule. This means these data is defined over some kind of system, and this allows the data to be represented by means of CSs in most of the realistic cases. Although the proposed framework has no ability to treat grammar-based complex data flows, in realistic cases this limitation can be avoided by defining symbolic axes based on codebooks such as the symbolic information described above.
Although the formal standard document has not been published yet, several companies and research projects have started to implement and use this specification. One such system is the *Kansai Environmental Platform* (14). The Kansai Environment is a field-trial environment for robotic services where sensor network is installed in a real world location such as shopping mall. Various types of sensors are installed in the environment to produce information on the environment such as where people is, how people are behaving or how a certain area (space) is used. This is done by performing people tracking and real-time pattern recognition based on statistical processing of the measured trajectories. These results are sent to servicing robots in the environment on request. By utilizing these information, robots can easily provide useful services to people in the environment. Here, RLS specification is extended for passing these

position information and, at the same time, also for passing symbolic recognition results to robots.
As can be seen in this implementation, the framework described here can be easily extended to treat and exchange versatile information, such as what a person is doing or what kind of place a person is staying at. Thus, the proposed specification has the possibility to be extended to further generic data exchange for robotics use. This generalization of localization service specification is one of the future issues.
The localization standard is only one of the fundamental frameworks required to build robots with common interfaced modules. Profiling on robotic devices, or descriptions on robotic service flows may be other candidates for standardization. Also as we stated briefly with the module ability description, by defining and exchanging module description metadata, much flexible and dynamic configuration of network robots composed of distributed modules over networks will be available. This will also make the interconnection with heterogeneous network systems such as ubiquitous sensing systems or GIS systems much easier. In the future, we will consider common frameworks for such architecture and metadata repository.

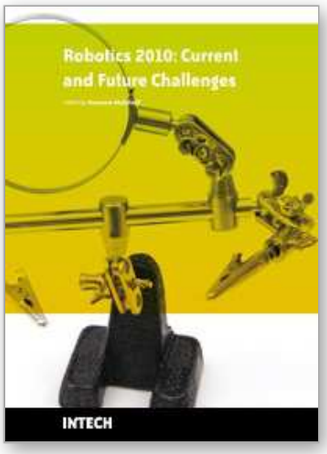
Acknowledgements

Standardization of the Robotic Localization Service has been initially worked by the Robotic Functional Services working group in Robotics Domain Task Force and later by Robotic Localization Service Finalization Task Force, OMG. The authors would like to thank the members of the working groups, especially the ex-co-chairs Olivier Lemaire and Kyuseo Han. The authors would also like to thank the members of joint localization working in Japan Robot Association and Network Robot Forum for their discussion and continuous support. Part of this work was supported by the Ministry of Internal Affairs and Communications of Japan, New Energy and Industrial Technology Development Organization (NEDO) and the Ministry of Economy, Trade and Industry of Japan.

7. References

[1] Denavit, J. & Hartenberg, R. S. (1955) A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices, *J. Applied Mechanics*, ASME 22, pp. 215-221
[2] Doi, M. (2007) Network Robots' Plug and Play, In: *Proc. ICRA 2007 Workshop on Network Robot System: Ubiquitous, Cooperative, Interactive Robots for Human Robot Symbiosis*
[3] EPC global (2008) Tag Data Standard, ver. 1.4
[4] Hagita, N. (2006) Communication Robots in the Network Robot Framework, In: *Proc. Int. Conf. Control, Automation, Robotics and Vision*, pp. 1–6
[5] Institute of Electrical and Electronics Engineers, Inc., Standard for local and metropolitan area networks: overview and architecture. Amendment 2: registration of object identifiers, IEEE Std. 802-1990, 1990.
[6] International Organization for Standardization (1999) Manipulating industrial robots - coordinate systems and motion nomenclatures. 9787:1999
[7] International Organization for Standardization (2007) Geographic information - Spatial Referencing by Coordinates, 19111:2007
[8] International Organization for Standardization (2008) Geographic information – Schema for moving features, 19141:2008
[9] International Telecommunication Union Telecommunication Standardization Sector (2001) XML Encoding Rules (XER), ITU-T Rec. X.693

[10] International Telecommunication Union Telecommunication Standardization Sector (2002) Abstract Syntax Notation One (ASN.1): Specification of basic notation, ITU-T Rec. X.680
[11] International Telecommunication Union Telecommunication Standardization Sector (2002) Specification of Packed Encoding Rules (PER), ITU-T Rec. X.691
[12] Misra, P. & Enge, P. (2001) Global Positioning System: Signals, Measurements, and Performance, Ganga-Jamuna Press
[13] National Marine Electronics Association (2002) NMEA 0183 Standard for Interfacing Marine Electronic Navigational Devices, Version 3.01
[14] Nishio, S.; Hagita, N.; Miyashita, T.; Kanda, T.; Mitsunaga, N.; Shiomi, M. & Yamazaki, T. (2009) Sensor network for structuring people and environmental information, In: Cutting Edge Robotics 2009, Kordic, V., Lazinica, A. and Merdan, M. (Eds.), 383-402, IN-TECH, ISBN 978-3-902613-46-2, Vienna
[15] Object Management Group (2009) Robotic Localization Service specification (Beta3), document no. dtc/2009-06-04
[16] SAE International (2009) JAUS/SDP Transport Specification, AS5669, rev. A
[17] Thrun, S., Burgard, W. and Fox, D. (2005) Probabilistic Robotics, MIT Press, Cambridge, MA
[18] Ueda, T., Kawata, H., Tomizawa, T., Ohya A. and Yuta, S. (2006) Mobile SOKUIKI Sensor System – Accurate Range Data Mapping System with Sensor Motion –, In: *Proceedings of Third International Conference on Autonomous Robots and Agents*, pp.309–314



Robotics 2010 Current and Future Challenges

Edited by Houssem Abdellatif

ISBN 978-953-7619-78-7
Hard cover, 494 pages
Publisher InTech
Published online 01, February, 2010
Published in print edition February, 2010

Without a doubt, robotics has made an incredible progress over the last decades. The vision of developing, designing and creating technical systems that help humans to achieve hard and complex tasks, has intelligently led to an incredible variety of solutions. There are barely technical fields that could exhibit more interdisciplinary interconnections like robotics. This fact is generated by highly complex challenges imposed by robotic systems, especially the requirement on intelligent and autonomous operation. This book tries to give an insight into the evolutionary process that takes place in robotics. It provides articles covering a wide range of this exciting area. The progress of technical challenges and concepts may illuminate the relationship between developments that seem to be completely different at first sight. The robotics remains an exciting scientific and engineering field. The community looks optimistically ahead and also looks forward for the future challenges and new development.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen