Proceedings of the 2009 IEEE
International Conference on Robotics and Biomimetics
December 19 -23, 2009, Guilin, China

Robot application development framework based on database

Subhash.S and K Madhava Krishna

Abstract—**In this paper we describe the robot software development framework with database at its core. In this framework the features of database is harnessed to make various components modular, and robot data handling and processing is made much more flexible. This empowers the researchers to design the system quickly, access the data and allow multiple collaborators work to be integrated with ease. The advantage of database based framework is explained case by case for various scenarios with example robot application developments.**

I. Introduction

Unlike computers, robots work in highly dynamic environment, the sensors are far from perfect. Robot system consists of wide array of sensors and actuators which are noise prone. This makes software development for robot far more complicated than developing software for computers. The features, capabilities and performance of different robot platforms are different, making it difficult to port software between different types of robot. Majority of the robotics software development is happening in universities and research labs, whose primary goal is to achieve excellence in particular area of robotics. In these places they have immediate priority for better algorithm to solve particular problem rather than long term goal of having modular software.

But we are fast approaching saturation in the areas of basic level control of robot, like localization, mapping, exploration, navigation, path planning and execution etc. Now the research is focused on the next level like higher level task planning, learning to interact with environment, decision process, etc. In order to pursue research in these areas, we need to have basic level control programs integrated and running reliably on the robot. But unfortunately these are developed in adhoc fashion with scant support for integrating universally with other modules of the robot software system. Without this basic level robot control generic software modules, we have almost hit a roadblock to reach the next level quickly.

With the advent of sensors with higher data rate, the data to be processed has increased by order of several magnitudes. We have moved from the days of single laser, single camera robotic system to multiple laser, multiple camera system with high frame rate and higher resolution. It has become a technical challenge to process all this data and integrate the information from multiple sensors. This problem is mostly being tackled deep down at the algorithm level. The same can also be tackled while processing the sensor data for use by algorithm. The way data is presented for the algorithm can influence the algorithm design to large extent.

Manuscript received July 31st, 2009. Subhash.S and Madhava Krishna are with the Robotics Research Center, IIIT-H, Hyderabad, AP, India
e-mail: subhash_s@ research.iiit.ac.in).

II. Review of existing robot software development frameworks

ROS [1], which is being developed by Willow Garage Inc has good modular framework uses TCP/IP or UDP for data transfer which results in sequential data access.
In Carmen [4] developed by CMU is open-source collection of software for mobile robot control. CARMEN is modular software designed to provide basic navigation primitives including: base and sensor control, logging, obstacle avoidance, localization, path planning, and mapping. It uses IPC for communication between processes.
In STAIR [2] also has modular functional units which can run over multiple computers and OS. Communication between processes is done over TCP connections.
Player [5] is control interface to wide array of robots and sensors

After studying the features of the existing robotics development frameworks, to the best of our knowledge we have not come across any development frameworks which used database to modularize and transfer data between modules. Other frameworks have good support for modular framework but data can be accessed only sequentially.

III. Design goals

The system has modular software development framework which is simple to implement and provides support for rapid system prototyping and development. The system also provides support for continuous change in robot software structure in dynamic development environment by enabling, adding and removing the modules with little coding. The system was designed from the limitations faced while using conventional data processing approaches, and difficulties faced in developing broader application set for single/multiple robots. While database makes the system inherently modular, the system design is also universal and can be used across wide range of robot platforms for different applications, as database is OS and programming language independent. This framework is just one of the ways robot software can be developed.

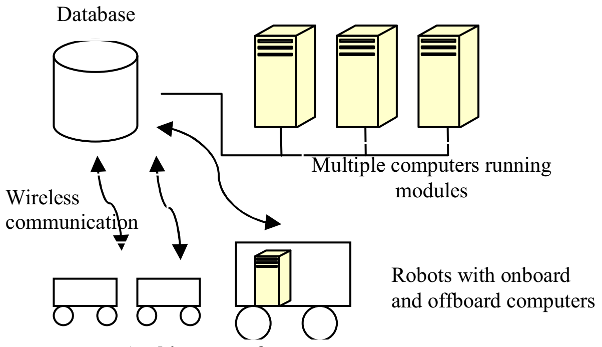
In our lab we have wide array of robots ranging from AmigoBots, Pioneer and in house developed Microrobot.

We have combination of robots which have onboard computing and off board computing. In order to support heterogeneous group of multiple robot, and varying needs of the researchers, we came with following architectural features.

- Modular architecture
- Database based communication and data exchange system
- Multiple programming language support within same robot system
- Data logging and support for offline algorithm testing
- Multiple OS and multiple computer support
- Data management
- Support for multiple heterogeneous robots to be controlled within a single robot software system

A. *Modular Architecture*

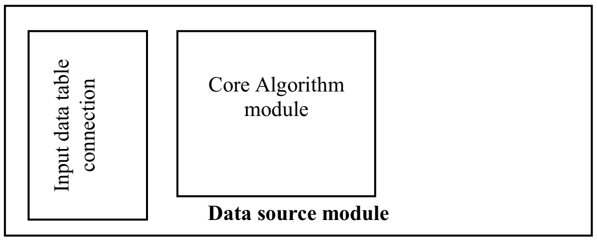
Modules in the robot software can be divided into 3 broad categories


Database
Multiple computers running modules
Wireless communication
Robots with onboard and offboard computers

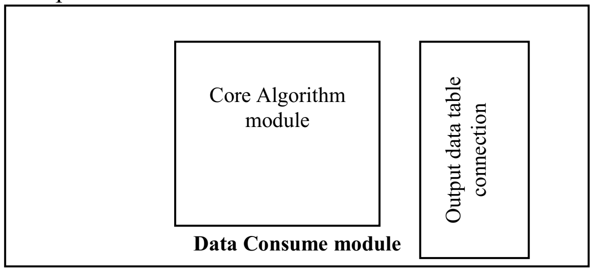
Architecture of system

- Data source module
- Data consume module
- Data process module

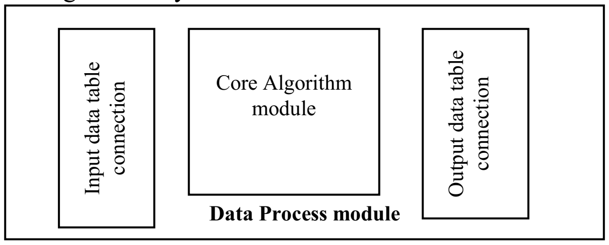
Data source module:
This type of modules only generates data which is to be consumed by other modules. They do not consume data generated by other modules. Sensors come under this category. Any number of sensors can be added as source module, independent of the other modules. Data from any data source is available as read only.
Any number of data consume modules can be interfaced to data source modules independent of each other. The data is written into table in the database with computer generated table name. This way any other module will be aware of the table name of all the sensors interfaced the robot in concern.



Data consume module:
This type of modules only consumes the data and does not generate any data. Graphical data display module, robot action module, debug data display module, etc come under this category. Any number of modules can connect to single data source independent of other modules as they only read the data. All the data consume modules have their own pointers, based on timestamp, to keep track of previously read position from data source.



Data process module:
This module consumes as well as generates data. Most of the algorithms come under this category. Any number of modules can run in parallel in separate threads. Algorithms which do not have any dependencies can run independent of each other. For modules having dependencies, the input table of module is the output table of other module. This modular architecture enables the robot researcher to add and remove algorithms as if it were building blocks without having to modify the whole structure.



B. *Modularity at data level:*

In most of the algorithms data is processed sequentially and discarded. Even when reading out the logged data it is processed sequentially. For example laser data is read and processed scan by scan. Better way of extracting information from sensor data which is stored in database table is to query for data based on required criteria like robot pose, time stamp, sensor range points within the grid, etc. This feature will give flexibility at the algorithm level and influences how the algorithm is written.

C. *Multiple programming languages:*

Different users want to program in different languages due to various reasons like ease of programming, availability of libraries, fast prototyping etc. Within single robot application there should be support to run modules built using different programming languages.

In our development environment, as database server is the core of the system and all modules access data from the tables in the database, any programming language which has support for database connectivity and sql can be used to write independent module which runs as separate thread

D. Modules running on different computers:

It might be required to run some programs in different OS for various reasons like availability of library, tool, etc. As data exchange point is a table, programs can run from any OS. This gives the flexibility for programmers to choose platform of their choice irrespective of what platform rest of the system is running in.

E. Data logging and offline algorithm testing:

By design all the data source module and data process module data is logged. This will also enable testing and debugging some of the modules offline without needing to run the robot. Here entire dataset can be queried for data of interest.

F. Heterogeneous multi robot system integration:

Multiple robots can be run simultaneously, with each connected to a common or a separate database. Multi robot control algorithms can synchronize the data from different robots by accessing the data from database.

IV. ADVANTAGE OF DATABASE BASED FRAMEWORK

Typically data is available as streams, which has to be consumed as it is generated, otherwise data is probably lost. Or some part of the program must be constantly listening for data. Also data has to be processed sequentially in the order it arrives. If large set of data is being processed simultaneously then the process must wait and store until all required data is available. Also lots of code needs to be written if the data has to be selectively picked up. For example instead of processing the laser data scan by scan we might want to process laser points only within certain grid in the map. In order to this by a sequential data processing algorithm, it has to store all the laser scans in the memory and then run a search logic to select only points of interest. Whereas, if the same thing had been executed in our architecture, all the laser scans would be saved in a table and not in memory (RAM). Database can be queried by giving the required selection criteria (in this instance points within a given grid)

Data is processed only once and stored forever, so same data can be reused at later point of time by different or same algorithm with different selection criteria. It is possible to segregate data based on timestamp as well. So it is possible to query for points within a given grid and compare it across different instances. This could be helpful in SLAM and localization.

Database has immense data management tools, which reduces the burden on algorithms by fetching different dataset. In the age of data explosion with robot having more sensors generating data at much faster rate, database goes a long way to manage all data. It will also be much easier to integrate various sensor data. Say for example there are 3 different IR range sensors on Microrobot. Instead of processing all 3 of them separately, data points within a given grid generated by all 3 of them within given time interval and within given robot pose range can be queried. This is simple few lines of codes. We are not aware any simpler methods to achieve the same goal.

While it can be said that, writing and reading database will clog the bandwidth of local area network or for the matter any network, with proper placement of high data traffic modules together these issues can be resolved. Also with faster networks and efficient databases these issues won't cause noticeable bottleneck in the performance.

Continuous learning:
Database helps in continuous learning. Raw and processed data from previous runs are available. Current data and previous data can be fused together to get more info. When we are using only high level data, due to nature of algorithms some vital information might get lost in the process of dimension reduction. With database to manage data, it is never a problem to manage huge data set and store them. With the advent of hard disk technology it is no longer concern of hard disk space, it is concern of quality of data. With robot application software still in research and evolving stage, we will need all the data that we can have. Once the design has matured only required data can be retained and data which is not needed can be pruned.

V. IMPLEMENTATION CASES

All the modules of the robot system can be started as separate threads within the same application or as a separate application. The modules will set up connections with designated databases and read/write data from their respective input-output tables.

Automatic training data generation:
For any learning algorithm we need large amount of labeled data. The process of generating labeled data is time consuming, labor intensive and expensive job. Instead if we set up a known environment and have other robots to act as additional sensors/observers, the process of generating labeled data can be automated to a large extent by considering data from various sensors.

Example cases:

A. Generation of radio signal strength indicator (RSSI) vs distance data:

The microrobot which has poor odometry and has ZigBee radio transceiver was placed on Amigobot which has much better odometry. The odometry data generated from Amigobot and RSSI data from microrobot was logged simultaneously and was used to generate the training data of

RSSI vs distance. Several such amigobot + microrobot were used for various kinds of environment and large set of training data of RSSI vs distance between multiple robots were generated, with hardly any manual labor. Training dataset was generated by querying for RSSI for a given distance and within the time interval of that run, for a given pair of robots. This data was later used to train an artificial neural network to learn the relation between RSSI and distance.

B. Exploration and mapping using Pioneer robot with Laser :

There are 2 ways to process laser data.

- Keep waiting for new laser scan to arrive, process the scan and update the map and any other data structure. Discard the scan readings.
- Write all scan readings to database. Query the database for all laser scan points within certain rectangle, process all the points within that rectangle from different scans taken at different robot pose. With much denser data more consistent result can be obtained, compared to considering scan by scan. The objects can be more reliably clustered and classified.

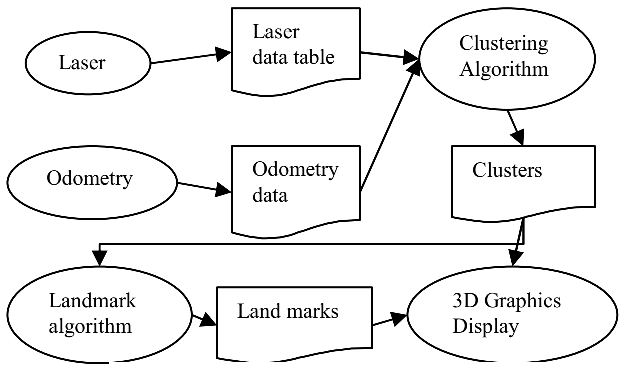
  Query the frames based on robot pose, timestamp, and process multiple frames at one go to extract more meaningful information like, change in environment over time, structure from motion, etc. only the imagination of researcher is the limit.

Description of all the modules in the case of exploration and mapping using Pioneer robot with SICK Laser
Data source modules:

- Laser data source module which writes all the incoming laser data into predefined table as it arrives along with timestamp and pose of the robot when scan was taken.
- Odometry data source module which writes all the incoming odometry data along with timestamp.
- Camera data module writes all the camera frames into database along with robot pose and timestamp

Data consume modules:
OpenGL based 3D graphical display system. This system will display the data from any table it has been pointed in suitable format depending on nature of data. There can be several rules to display data. For real time algorithm output, the display module periodically checks for new output data from the algorithm and displays. For offline algorithm testing it displays entire output after algorithm has completed running.

As we are debugging various algorithms we can enable the output of those algorithms to displayed, test and debug the algorithm. It becomes a lot easier to test the algorithm with easy 3D graphical display. It speeds up the whole design process by allowing the researcher to visualize nature of data, effect of algorithm on data, and come up with better solutions faster.



Clustering and landmark identification modules

Data process modules:
Clustering module:
Laser points within the given grid area is queried for and clustered based on minimum distance. This is repeated for multiple grids covering entire map, and then all clusters are merged. All the laser points are assigned the cluster number it belongs to and written into output cluster table. It would be have been very difficult to obtain this data if database was not used. If entire map data is considered in one go, due to large data set, the clustering algorithm will take more time to compute the same result

Landmark extraction module
This module just reads out the clusters from the output table of clustering algorithm. Multiple lines are fitted based on least mean square error. Again database tables help to group the laser points into different clusters. So each time new cluster is to be read out, query criteria would be cluster number.

VI. CONCLUSION

More information can be extracted from the data when required information is selectively queried from entire logged data available rather than considering the instantaneous data. By harnessing the power of database for constructing modular software architecture and selective data handling, algorithm development can be taken to new dimension, by giving better insight into the problem than ever before. Faster development time is achieved by getting the ground reality at all stages of development. While database based software framework is not new, we have not found robotic software architectures exploiting these advantages based on our survey.

REFERENCES

[1] Morgan Quigley, Brian Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, Andrew Ng, “ROS: an open-source Robot Operating System” in International Conference on Robotics and Automation, (2009).

[2] M. Quigley, E. Berger, and A. Y. Ng, “STAIR: Hardware and Software Architecture,” in AAAI 2007 Robotics Workshop, Vancouver, B.C, August, 2007.

[3] Davide Brugali and Erwin Prassier, “Software Engineering for Robotics” in IEEE Robotics and Automation Magazine, March 2009

[4] M. Montemerlo, N. Roy, and S. Thrun, “Perspectives on standardization in mobile robot programming: The Carnegie Mellon Navigation (CARMEN) Toolkit,” in Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS), Las Vegas, Nevada, Oct. 2003, pp. 2436–2441.

[5] R. T. Vaughan and B. P. Gerkey, “Reusable robot code and the Player/Stage Project,” in Software Engineering for Experimental Robotics, ser. Springer Tracts on Advanced Robotics, D. Brugali, Ed. Springer, 2007, pp. 267–289.

[6] [R. Alami, R Chatila, S. Fleury, M. Ghallab, F. Ingrand, “An architecture for autonomy” in The International Journal of Robotics Research, Vol. 17, No. 4, 315-337 (1998).

[7] O. Michel, “Webots: a powerful realistic mobile robots simulator,” in Proc. of the Second Intl. Workshop on RoboCup. Springer-Verlag, 1998.C. J. Kaufman, Rocky Mountain Research Lab., Boulder, CO, private communication, May 1995.

[8] J. Jackson, “Microsoft robotics studio: A technical introduction,” in IEEE Robotics and Automation Magazine, Dec. 2007, http://msdn.microsoft.com/en-us/robotics.M. Young, *The Techincal Writers Handbook.* Mill Valley, CA: University Science, 1989.

[9] Lois E. Navarro-Serment, Robert Grabowski, Christiaan J.J. Paredis and Pradeep K.Khosla, “Millibots: The Development of Framework and Algorithms for a Distributed Heterogeneous Robot Team”, in IEEE Robotics and Automation Magazine, December 2002.

[10] Saddek Bensalem, Matthieu Gallien, Felix Ingrand, Imen Kahloul and Nguyen Thanh-Hung, “Designing autonomous robots: towards more dependable software architecture”, in IEEE Robotics and Automation Magazinge, March 2009.