Perspectives on Standardization in Mobile Robot Programming : The Carnegie Mellon Navigation (CARMEN) Toolkit

Michael Montemerlo
mmde@ri.cmu.edu

Nicholas Roy
nickr@ri.cmu.edu

Sebastian Thrun
thrun@cs.cmu.edu

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
USA

***Abstract*— In this paper we describe our open-source robot control software, the Carnegie Mellon Navigation (CARMEN) Toolkit. The ultimate goals of CARMEN are to lower the barrier to implementing new algorithms on real and simulated robots and to facilitate sharing of research and algorithms between different institutions. In order for CARMEN to be as inclusive of various research approaches as possible, we have chosen not to adopt strict software standards, but to instead focus on good design practices. This paper will outline the lessons we have learned in developing these practices.**

I. Introduction

In 2002, we developed a open-source collection of robot control software called the Carnegie Mellon Navigation (CARMEN) Toolkit. CARMEN was designed to provide a consistent interface and a basic set of primitives for robotics research on a wide variety of commercial robot platforms. The ultimate goals of CARMEN are to lower the barrier to implementing new algorithms on real and simulated robots and to facilitate sharing of research and algorithms between different institutions. Robotics research covers a spectrum of different approaches and formalisms. We have adopted the philosophy of making CARMEN as inclusive as possible. For this reason, we have chosen not to adopt strict standards, but to instead focus on good design practices. This paper will outline the lessons we have learned in developing these practices.

CARMEN is modular software, organized as an approximate three-tier architecture, popularized by Bonasso et al. [1]. The base layer governs hardware interaction and control, by providing an abstract set of base and sensor interfaces. The base layer also provides low-level control loops for simple rotation and straight-line motion. Finally, the base layer integrates sensor and motion information to provide improved sensor odometry, and also allows for low-level collision detection (distinct from collision avoidance). CARMEN base modules can control a wide range of commercial bases, including but not limited to the Nomadic Technologies' Scout and XR4000, ActivMedia Pioneers, and iRobot's b21 and ATRV series. CARMEN also provides a uniform simulation platform for all commercial bases.

The navigation layer implements intermediate navigation primitives including localization, dynamic object tracking, and motion planning. Unlike many other navigation systems, motion control is not divided into high-level (strategic) planning and lower-level (tactical) collision avoidance; instead, all but the lowest-level motor control is integrated into a single module. This allows for more reliable motion planning, at a reasonable computational cost. Finally, the third tier is reserved for user-level tasks employing primitives from the second tier. CARMEN also contains a separate layer of non-autonomous components, including display modules, editors, etc.

II. Standards in CARMEN

The nature of research is to improve the state-of-the-art by developing novel approaches to open problems. As a consequence, the goals of robotics research and the goal of standardizing robot software are often in conflict. It is very difficult to impose standards on a still-maturing field, because as new techniques evolve, existing standards will inevitably preclude certain approaches. If developers ignore the standards, the standards lose their value.

Instead, CARMEN was designed with several driving principles in mind, namely ease-of-use, extensibility, and robustness. It is not reasonable to expect one standard to enforce these principles on any piece of software, however CARMEN provides a framework and tools that encourage practices that satisfy these principles. For instance, every CARMEN module includes a functional interface that abstracts away the communication mechanism between modules. As a result, the communication layer can be upgraded transparently without effect on individual modules. Similarly, every CARMEN module is robust to the presence or absence of other CARMEN modules – failure of a single module will not cause catastrophic failure of other modules.

CARMEN is freely available over the web and is currently being used by a number of research groups around the world in applications such as health care robotics, mine exploration and multi-robot collaboration. We hope to be able to leverage CARMEN's design principles in extending the state of the art of robotic research.

III. Good practices

CARMEN was designed with the following three goals in mind:

- Ease of use - The amount of time needed to learn how to use CARMEN should be small. The basic capabilities of CARMEN should be easy to get running on a new robot.
- Extensibility - New CARMEN programs should be easy to write. The core CARMEN programs should serve as a solid foundation for building higher level capabilities. It should also be easy to replace existing CARMEN modules.
- Robustness - CARMEN programs should be robust to a variety of failures, including failures in communications and failures of other programs.

The following sections will describe seven programming practices which encourage these principles in CARMEN.

A. Modularity

CARMEN was designed as a modular software architecture, with each major major capability built as a separate module. Modules communicate with each other over a communication protocol called IPC, developed by Reid Simmons at Carnegie Mellon University [9]. While modularity introduces some overhead, it has several important advantages over monolithic systems:

- Flexibility - Robots come in many configurations. A user can mix and match different robot bases and sensors by simply running the appropriate modules. This eliminates the need to accommodate all possible hardware configurations in a single process.
- Network support - All of the modules need not run on the same processor. Processor intensive tasks can even be run off-board the robot.
- Reliability - If a single module fails, it does not cause the remaining modules to fail.
- Extensibility - It is much easier to modify robot components, or develop new ones, if each component is self-contained.

Modularity encourages ease-of-use and extensibility, in that different modules that perform the same function (for example, the modules for each base type) must converge to a single abstract interface. For example, by developing each base controller as a separate module, the need for a single abstract base interface became clear, and easier to implement.

B. Simple core modules

We have isolated a set of core CARMEN modules that provide a simple set of navigation primitives; these primitives (base control, localization, tracking, and path planning) should serve as a strong foundation for building higher-level robot capabilities. Many existing robot software packages tend to bundle multiple features into single modules. Our approach is to constrain tightly the number of features in the core modules, and require additional features to be implemented in higher layers.

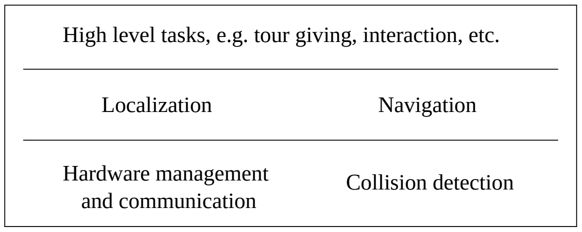

High level tasks, e.g. tour giving, interaction, etc.
Localization
Navigation
Hardware management and communication
Collision detection

Fig. 1. The approximate separation of modules within Carmen.

Providing a small set of core functionality addresses all of our stated design goals, in that simple modules are typically easier to understand, and are more easily made reliable. Tracking down bugs becomes increasingly time-consuming for developers as the size of the modules (and the size of distributions) balloon. Large software distributions can also be overwhelming for both developers and users alike.

C. Separation of control and display

One important design principle of CARMEN is the separation of robot control from graphical displays. No CARMEN core module that is required for autonomous control has embedded graphical displays; instead, displays run in separate processes. Information is communicated between the controlling modules and the displays using standard communication protocols.

This design principle is an example of the software paradigm known as the Model-View-Controller, and was chosen to encourage more transparent modules. Robot control software often employs graphical displays as debugging tools. Embedded displays can lead to the situation where the state of the software is visible to the user, but inaccessible to other programs. It is not possible, or even desirable, to require complete transparency (e.g., the contents of all internal variables and data structures) of every module. However, we have found that separating graphical displays from the implemented algorithms promotes an appropriate degree of transparency. This ensures that information in a module that could be useful is available to all other modules. Additionally using the Model-View-Controller paradigm allows for distributed displays. A single graphical display can be used to display the state of multiple robots, where the display is remote from all robots.

D. Abstract Communication

During the development of CARMEN, we have employed three different communications protocols for module-to-module communication. One lesson from this experience is that exposing the particular quirks of a given communication protocol to the developer can have unintended consequences on the design of the robot modules. Tight integration of the communication protocol and the core modules also makes upgrading communications packages extremely difficult if a better alternative becomes available. For these reasons, we encourage all communication functionality in CARMEN to be hidden behind abstract interfaces.

All functions generating outbound communication traffic from a module are placed in a separate source file. Calls to these functions make no assumptions about the particular communications protocol being used. All modules include a separate interface library that abstracts the subscription process to the modules' messages. In order to upgrade the communication protocol, only the code in the outbound and inbound interface libraries must be changed. All of the robot code will remain unchanged.

E. Abstract Hardware Interfaces

1) Base Interfaces: CARMEN supports a wide variety of commercial robot bases. Unfortunately, these bases differ not only in form factor, but also in their command interfaces. Some robots accept wheel velocities, others accept translational and rotational commands; some robots report the output of odometers, others perform odometer integration in the hardware and report (x,y,θ) from some arbitrary start position.

In order to ensure uniformity across platforms, CARMEN supports a standard set of interface functions, and performs the necessary transformations to the interface appropriate for the platform:

- Velocity commands: v_{trans}, v_{rot}
- Translational acceleration commands: $acceleration, deceleration$
- Vector commands: $\delta_{trans}, \delta_{rot}$
- Odometry reporting: x, y, θ

This interface functionality supports the criterion of ease-of-use, in that every base can be controlled in the same manner; higher level code does not need to have special cases for each kind of base. These interface functions are also easily implemented in a robust manner.

One disadvantage is that certain kinds of motion are not supported. For instance, the Nomadic Technologies' Nomad 200 supports a turret rotation, and the Nomad XR4000 support translation in arbitrary directions without rotation; both of these robots give the appearance of true holonomic motion, which is not directly supported by CARMEN's hardware interface. In a similar vein, the assumption is that all bases are capable of point turns; non-quasi-holonomic robots (e.g., Ackerman-steered robots) are not currently supported.

2) Sensor Interfaces: It is often tempting to convert sensor measurements into a more convenient form, for example, range measurements into (x,y) pairs that correspond to the obstacles at the end of each range, ignoring measurements at the end of the sensor range. However, in order to maintain reliability, CARMEN does not encourage or support this approach, because information is almost always lost during transformations between representations. Sensor information should always be preserved in the form that is closest to the raw data; individual modules and applications can transform data as they see fit.

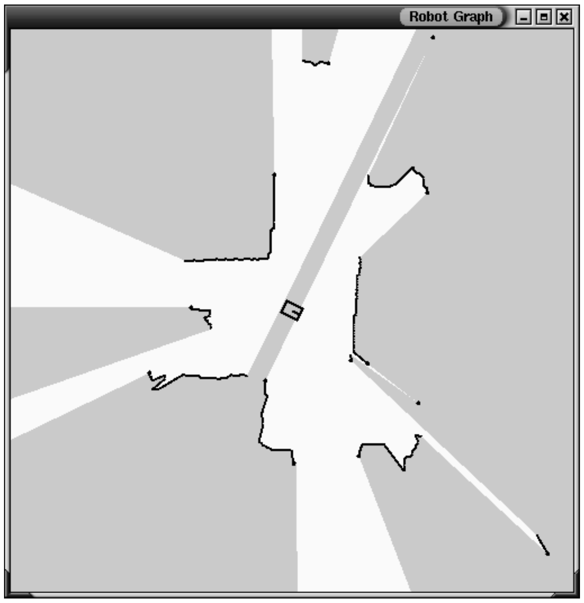


Fig. 2. The Robotgraph display, that shows the information available at the lowest level of the CARMEN hierarchy. This display shows the output of odometry and laser sensing information. The white areas are clear space, and the gray areas are unknown. The black dots correspond to obstacles detected by the laser range finder. The black square in the centre depicts the robot's current pose.

CARMEN currently only supports two basic kinds of sensors: position sensors (e.g., GPS, INS, etc.) and range sensors (e.g., laser range finders). While the raw data from every sensor is available to all modules, CARMEN provides a higher-level format that is typically much more useful. Each abstract sensor message has an interpolated odometry stamp attached, based on timestamps, knowledge of the robot's motion, and the sensor position on the robot. By providing position estimates for each sensor measurement, the robustness of localization (especially during fast motion) is improved substantially.

F. Standardized Coordinates and Units

One important but often ignored issue of robot architectures is the question of units and co-ordinate frames. CARMEN assumes that all units are in the International System of Units (SI) [1] [10], with all distances in metres and all angular measures in radians. [2] The advantage to using the international standard of SI is that it reduces confusion to users ("is this parameter in metres or centimetres?"), and also minimizes the likelihood of disagreement between collaborators, who wish to standardize on a different selection of units.

CARMEN allows exactly three co-ordinate frames:

1) The robot's frame of reference. Distances are in metres, and the robot always faces along the positive x axis.
2) The global frame of reference. Distances are in metres, and $\theta = 0$ lies along the x-axis of the map.
3) The map frame of reference. Distances are in grid cells, and $\theta = 0$ lies along the x-axis of the map. This is a meaningless frame of reference without a map.

It is tempting to occasionally represent information in left-handed co-ordinate systems (θ increasing clockwise), in order to allow easier interaction with display systems, such as X-windows. However, this practice can lead to interface confusion. There are, in general, many fewer co-ordinate systems appropriate to robots than often seems to be the case.

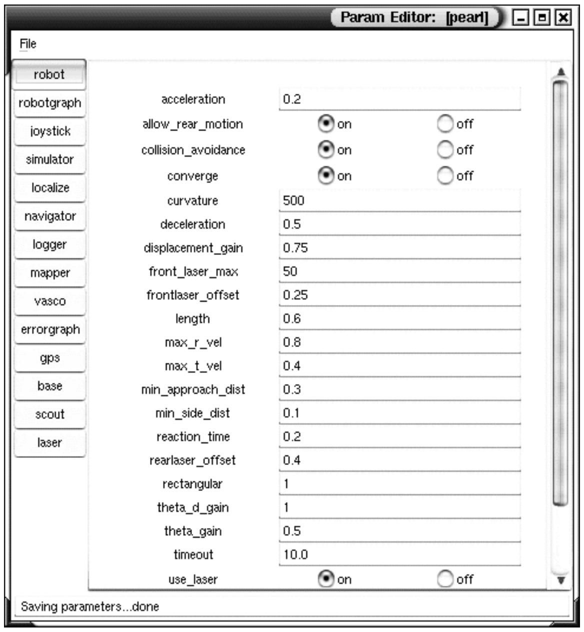
G. Centralized Model Repository

CARMEN has also addressed the robustness and ease-of-use principles by providing a central repository and interface for handling parameters. Mobile robot software is commonly distributed across multiple processors. A common failure point is the conflict between parameter values that are loaded from different local sources. By requiring modules to get their parameters from a single source (loaded from a single file), CARMEN ensures that parameters values are consistent across all modules and platforms. This approach of storing parameters in a single file, and distributing parameters programmatically, has the additional benefit that parameters can be updated during run-time, eliminating the need to restart (or worse, recompile) modules.

We have also extended the model repository to include maps. Instead of loading environment descriptions from local files, all modules also request maps from a central server. The same benefits of run-time updating of parameters can also be extended to maps. Distributing

[1]Also known as MKS.

[2]Many systems represent angles in degrees for debugging and printing ease; however, this practice will inevitable result in a line of code that passes an angle in degrees to a trigonometric function like sin or cos that accepts radians.



acceleration	0.2	
allow_rear_motion	[✓ on]	off
collision_avoidance	[✓ on]	off
converge	[✓ on]	off
curvature	500	
deceleration	0.5	
displacement_gain	0.75	
front_laser_max	50	
frontlaser_offset	0.25	
length	0.6	
max_r_vel	0.8	
max_t_vel	0.4	
min_approach_dist	0.3	
min_side_dist	0.1	
reaction_time	0.2	
rearlaser_offset	0.4	
rectangular	1	
theta_d_gain	1	
theta_gain	0.5	
timeout	10.0	
use_laser	[✓ on]	off

Saving parameters...done

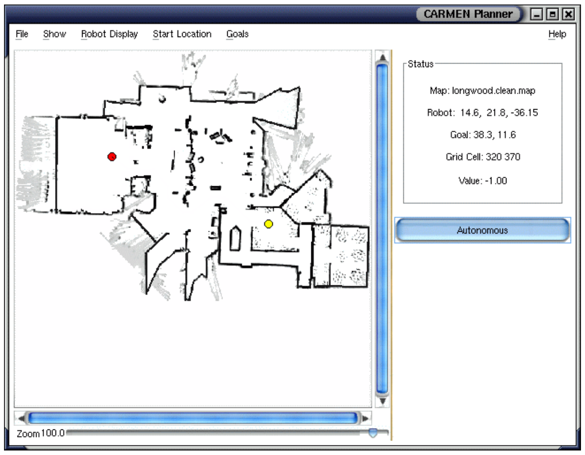
Fig. 3. The Parameter Editor, that shows the information available in the CARMEN registry. Parameters are organized according to modules in the Editor, and while no strict type system exists, the editor attempts to infer parameter types and display them appropriately.

maps programmatically does have one disadvantage, in that transferring maps across networks, especially wireless networks, consumes bandwidth. However, this cost is only incurred at startup, and when amortized over the operation of the robot, is negligible compared to the value of ensuring map consistency. We have also not hesitated to use freely available compression tools (e.g., zlib [5]) to reduce map transmission costs.

IV. Navigation

CARMEN also includes map-based people detection and tracking. This tracking is accomplished by tracking differences between actual and expected laser scans given the most likely position of the robot at every timestep. In the future, we plan to use this same map differencing technique to modify the map over time.

Navigation is a difficult concept to standardize, and therefore CARMEN has tried to make it possible to support many different styles of navigation. The current navigation implementation is an implementation of Konolige's local gradient descent planner [4], which on modern CPUs is fast for even the largest maps. Previous implementations [12] of mobile robot control software favoured a multi-level support, with coarse strategic planning providing intermediate goals to a lower-level collision avoidance module [2]. However, this approach becomes brittle in the limit of arbitrarily large deviations from the intended high-level path, and can lead to catastrophic navigation failures. The integration of collision avoidance and planning allows CARMEN to achieve a higher navigational reliability, especially when unmapped obstacles close routes through the environment.



Map: longwood.clean.map
Robot: 14.6, 21.8, -36.15
Goal: 38.3, 11.6
Grid Cell: 320 370
Value: -1.00

Fig. 4. The CARMEN planner, depicting the current map, the robot's current position in the map, and the expected trajectory.

V. Localization and Tracking

The localization module in CARMEN implements a variation of Monte-Carlo Localization [12] algorithm. The module accepts odometry readings and laser readings from the base modules and is able to estimate the position of the robot in a map at approximately 10 Hz. Monte-Carlo Localization for CARMEN because it has a number of nice properties. By estimating a posterior probability distribution of possible poses of the robot, the localizer is robust to error in the robot's odometry and sensors. This leads to reliable, long-term operation. Monte-Carlo Localization is also flexible in that it also supports global localization. During global localization, the algorithm can determine the position of the robot given no information about its starting point in the map.

CARMEN also includes map-based people detection and tracking. This tracking is accomplished by tracking differences between actual and expected laser scans given the most likely position of the robot at every timestep. In the future, we plan to use this same map differencing technique to modify the map over time. This procedure can be used to keep the map up-to-date as furniture is moved and new areas of the map are exposed. Preliminary results for lifelong map-learning are shown in Figure 5. The original map is shown in black. Modifications to the map are shown in red and yellow. Red cells are new occupied cells, and yellow cells are new empty cells. Trash cans and boxes in the main hallway have been removed from the map, and five new offices have been added to the map.

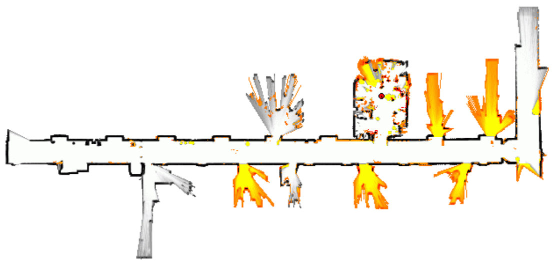


Fig. 5. An example of lifelong map learning. The original map is shown in black. Modifications to the map are shown in red (additions) and yellow (subtractions).

1) Automatic Map Building: The use of map-based navigation assumes the existence of an accurate map of the environment. CARMEN contains an implementation of Hähnel et al.'s map-builder [3], which builds high-quality metric maps from data recorded from the laser-range finder. CARMEN assumes a single metric map for all map-based navigation, as opposed to hierarchical map representations, topological representations, or feature-based representations. This map representation has been used by a number of research projects in a number of different arenas, such as the museum tour-guide project [11], multi-agent exploration and planning [6] and mine-mapping [13]. Discrete maps provide a natural extension to higher-level planning methodologies such as Partially Observable Markov Decision Processes [8], [7].

VI. Conclusion

We have described a set of design practices for CARMEN chosen that encourage the ease-of-use, extensibility, and reliability of robot control software. CARMEN is still a work-in-progress, and aspects of the design will inevitably change over time. However, we believe that adherence to the design principles described in this paper will help maintain the utility of CARMEN as the software continues to mature.

Acknowledgments

The authors would like to thank the developers of beeSoft, especially Wolfram Burgard, Dieter Fox and Dirk Hähnel for the inspiration for Carmen. Many people have contributed code, patches and test cases, in particular Greg Armstrong, Drew Bagnell, Curt Bererton, Jeff Biseda, Charlie Garrod, Jared Glover, Geoff Gordon, Nell Hoffman, Emilie Ann Philips, Joelle Pineau, Matt Rosencrantz, Andy Rubin, Alvaro Soto, Cyrill Stachniss and Rudy Triebel.

VII. References

[1] R. P. Bonasso, R. J. Firby, E. Gat, David Kortenkamp, D. Miller, and M. Slack. Experiences

with an architecture for intelligent, reactive agents. *Journal of Experimental and Theoretical Artificial Intelligence*, 9(1), 1997.

[2] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1), 1997.

[3] D. Hähnel, D. Schulz, and W. Burgard. Map building with mobile robots in populated environments. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2002.

[4] Kurt Konolige. A gradient method for realtime robot control. In *Proc. of the IEEE/RSJ Interational Conference on Intelligent Robotic Systems (IROS)*, 2000.

[5] Jean loup Gailly and Mark Adler. zlib compression library. http://www.gzip.org/zlib.

[6] Matt Rosencrantz, Geoff Gordon, and Sebastian Thrun. Locating moving entities in dynamic indoor environments with teams of mobile robots. In Tuomas Sandholm and Makoto Yokoo, editors, *Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, volume 2, 2003.

[7] Nicholas Roy and Geoffrey Gordon. Exponential family pca for belief compression in pomdps. In *Advances in Neural Processing Systems*, volume 15, 2002.

[8] Nicholas Roy and Sebastian Thrun. Coastal navigation with mobile robots. In *Advances in Neural Processing Systems*, volume 12, pages 1043–1049, 1999.

[9] Reid Simmons. The inter-process communication (IPC) system. http://www-2.cs.cmu.edu/afs/cs/project/TCA/www/ipc/ipc.html.

[10] Barry N. Taylor. *Guide for the Use of the International System of Units (SI)*, 1995 edition edition.

[11] S. Thrun, M. Beetz, M. Bennewitz, W. Burgard, A.B. Cremers, F. Dellaert, D. Fox, D. Haehnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz. Probabilistic algorithms and the interactive museum tour-guide robot minerva. *International Journal of Robotics Research*, 19(11):972–999, November 2000.

[12] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust monte carlo localization for mobile robots. *Artificial Intelligence*, 101:99–141, 2000.

[13] S. Thrun, D. Hähnel, D. Ferguson, M. Montemerlo, R. Triebel, W. Burgard, C. Baker, Z. Omohundro, S. Thayer, and W. Whittaker. A system for volumetric robotic mapping of abandoned mines. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2003.