Middleware for Robotics: Applications on Real-time Systems

Gregory Broten, David Mackay and Rene Desgagnes
Defence Research and Development Canada – Suffield
{*firstname.lastname*}*@drdc-rddc.gc.ca*

***Abstract*— Defence R&D Canada (DRDC) has a long history of developing robotic applications, including tele-operated and autonomous ground vehicles. The autonomous ground vehicles come in various configurations, with differing sensor payloads and varying performance capabilities. Many of these applications occupy a grey zone where both soft and hard real-time capabilities co-exist. As a general rule, higher vehicle speeds drive an application from soft real-time towards harder real-time requirements. Under soft real-time conditions, DRDC uses the linux operating system and CORBA middleware, with the RTEMS real-time operating system used for certain time critical implementations. CORBA middleware yields portable, modular and extensible components that simplify the integration of multiple capabilities onto a single platform. This paper discusses DRDC's experiences with CORBA, its advantages and disadvantages, and it applicability to real-time autonomous systems.**

I. Introduction

Under idealized conditions, robotics software would build upon flexible, portable, modular and extensible components, while maintaining responsive real-time capabilities. Unfortunately these two goals are often at odds, where the extra overhead associated with components impairs real-time capabilities. Component based systems, through a middleware intermediary, pass information among various processes or applications. The middleware is key to this seamless information flow as it hides the data marshalling/unmarshaling, along with the specific means of transport. This approach leads to network transparent implementations that enables distributed computing. Frameworks such as Miro [1], [2], built upon CORBA [3], and ORCA [4], which uses ICE middleware [5], provide elegant tools for implementing distributed computing. While this high level of abstraction is desirable from the software reuse and design perspective, it is not without limitations. For robotic's researchers, the chief among these limitations is the real-time performance. CORBA and ICE, with their roots in office/internet applications, do not specifically address the real-time issues found in robotics applications. As a result, their adoption by the robotics community has been tentative. Robotics specific middleware, that attempts to address real-time issues, has been created; examples include IPC [6], IPT [7] and RTC [8]. These toolkits are light weight, efficient and have a small footprint, thus are applicable to a wide range of platforms, including embedded systems. Although these robotics specific middleware toolkits are more applicable to real-time implementations, they do not provide the software engineering tools that assist in the development and maintenance of components.

DRDC researchers have confronted this software development/reuse versus performance dilemma and as a solution propose a hybrid implementation that marries a Component Based Software Engineering (CBSE) [9] approach with real-time capabilities. This approach recognizes that only certain subsets of software demand hard real-time capabilities, while most robotics applications function under less stringent soft real-time requirements. Using this distinction, a large majority of robotics applications can be developed under soft real-time constraints, where high level middleware toolkits assist in the implementation of flexible and extensible components.

This paper describes DRDC's software implementation, where the Miro framework and CORBA middleware are used under soft real-time conditions, while RTEMS provides hard real-time capabilities. The paper is divided into 4 sections. Section II describes DRDC's operational environment and its requirements. In Section III DRDC's architecture for autonomy is covered, while CORBA's performance in field trials is given in Section IV. Finally, conclusions are given in Section V.

II. Operational Environment and Requirements

DRDC's unmanned ground vehicles (UGVs), similar to the vehicles that competed in the DARPA Grand Challenge [10], [11], operate in outdoor, unstructured terrain that features roads of various types, flat plains and semi-rural settings. Figure 1 is typical of DRDC's expected operating environment. The Raptor, shown in Figure 2, is a typical of 4 wheeled UGV. Given the operating environment and the vehicle style, a key research objective is high speed (20 - 60 Km/hr) traversal. This requires real-time sensing, world representation, obstacle detection and navigation, and this all must occur as the vehicle is moving.

Traversal speed is the key factor that drives computing performance and real-time requirements. Faster speeds translate into shortened decision times, which in turn moves the computing requirements from softer real-time towards hard real-time performance.

A. Soft versus Hard Real-time Performance

A system is said to be real-time if the total correctness of an operation depends not only upon its logical correctness, but also upon the time at which it is performed [12]. Additionally, real-time systems are classified as either hard real-

Fig. 1. Typical Operating Environment

Fig. 2. 4-Wheeled Raptor Unmanned Ground Vehicle

time or soft real-time. Hard real-time systems feature strict deadlines, which must not be violated. On the other hand, a soft real-time system can tolerate a certain degree of response *sloppiness* without causing a major failure. DRDC's software architecture uses a combination of both hard and soft real-time applications, in a hybrid implementation.

B. Sensing Limitations

The Raptor UGV uses the laser rangefinder as its primary sensor. Given the rangefinder's intrinsic range limitations and its mounting geometry, the maximum sensing distance is limited to approximately 20 to 25 *m* [13]. This range is important as it defines the maximum range at which obstacles can be detected. Using the maximum sensing range and the vehicle's speed, it is simple to calculate the time required to traverse this distance. The traversal time, t, is given by $t = \frac{d}{v}$, where d is the distance and vehicle's speed is given by v. Table I shows the traversal time versus vehicle speed, for selected speeds. This table clearly shows that as the speed increases, myopic sensing severely limits the amount of time available for computing purposes.

C. The Effects of Traversal Speeds

The upper bound on the time available to avoid an obstacle presented in the previous section, based solely on the vehicle

Max. Sensing Distance (m)	Speed (*m/s*)	Time (*s*)
20	2.78	7.2
20	5.56	3.6
20	8.33	2.4
20	11.1	1.8

TABLE I
TRAVERSAL TIME VS SPEED

speed and the maximum range of the laser rangefinder, is clearly an oversimplification. It is valid only if an obstacle can reliably be detected at the maximum range of the sensor and if the obstacle is stationary and can be avoided by steering around it in a dynamically stable manoeuvre. In a worst case scenario, the vehicle will encounter an obstacle of sufficient extent such that it's not possible to manoeuvre around it and a panic stop must be initiated to avoid a collision. For what range of vehicle speeds is this possible? To address this question, consider the force balance on the idealized vehicle, in Figure 3, travelling down the plane with speed v_0 when braking is initiated. Equating the sum of the forces in the x direction

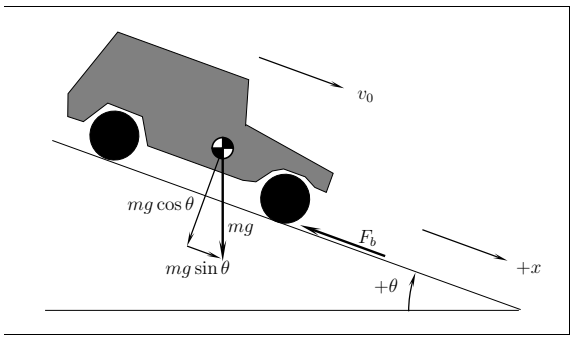

Fig. 3. Force balance on an idealized vehicle braking on sloped terrain.

to the mass of the vehicle, m, times the acceleration, a_x, in that direction, one can obtain an expression for the acceleration down the plane,

$$\sum F_x = mg\sin\theta - F_b = ma_x$$

$$a_x = \{\sin\theta - \frac{F_b}{mg}\}g$$

where F_b is the net braking force. Treating the vehicle as a rigid body, ignoring any suspension motion, and assuming one dimensional motion, one can obtain an estimate of the stopping time, t_s,

$$v(t) = v_0 + \int_0^t \ddot{x}(\tau)d\tau = v_0 + a_x t$$

$$\Rightarrow t_s = \frac{v(t_s) - v_0}{a_x} = \frac{-v_0}{a_x} \Leftarrow v(t_s) = 0.$$

In a similar fashion, the stopping distance, x_s, can be obtained,

$$x(t) = \int_0^t \dot{x}(\tau)d\tau = \int_0^t v_0 + a_x\tau d\tau = v_0 t + 1/2 a_x t^2$$

$$x_s = v_0 t_s + 1/2 a_x t_s^2 = -1/2 \frac{v_0^2}{a_x}.$$

Pivtoraiko et al. [14] have compiled normalized braking force, F_b/mg, data obtained for an off-road vehicle during the CMU PerceptOR program. These tests encompass vehicle speeds ranging from 1 to 4 *m/s* and a variety of terrain slopes and surfaces. The vehicle controller behaved akin to an anti-lock braking system, not allowing the wheels to lock and thus maintaining steering. The normalized braking force, obtained off-road in these trials, ranged in value from 0.15 to 0.45, much less than typical values reported for passenger vehicle tires on dry asphalt, 0.71 [15] and 0.85 [16]. Stopping times and distances, computed using the PerceptOR normalized braking force data, for a range of vehicle speeds and terrain slopes are presented in Table II.

θ (deg)	F_b/mg	v_0 (m/s)	a_x (m/s^2)	t_s (s)	x_s (m)
13	0.4	1	-1.72	0.58	0.29
13	0.4	4	-1.72	2.33	4.66
13	0.4	10	-1.72	5.82	29.12
4	0.28	1	-2.06	0.48	0.24
4	0.28	4	-2.06	1.94	3.88
4	0.28	10	-2.06	4.85	24.24
1	0.25	1	-2.28	0.44	0.22
1	0.25	4	-2.28	1.75	3.51
1	0.25	10	-2.28	4.38	21.92
-6	0.18	1	-2.79	0.36	0.18
-6	0.18	4	-2.79	1.43	2.87
-6	0.18	10	-2.79	3.58	19.91

TABLE II
BRAKING TIME AND DISTANCE VS SPEED AND TERRAIN SLOPE

For the range of vehicle speeds used in the PerceptOR trials, 1 to 4 *m/s*, the stopping distances are well within the maximum range of the SICK laser rangefinder. This isn't at all surprising as the PerceptOR vehicle used the same SICK laser rangefinder as the DRDC Raptor. The margin of safety isn't really that large as *reaction time* and other system delays have not been included in the analysis. The data for the extrapolated 10 *m/s* vehicle speed are justifiably suspect as this is outside the range of speeds encountered in the PerceptOR trials. They are included to make the point that even for moderate vehicle speeds the stopping distance alone can easily exceed the maximum sensing range of the SICK laser rangefinder leading to certain collision.

III. DRDC'S ARCHITECTURE FOR AUTONOMY

DRDC has developed a hybrid architecture, interally called the *Architecture for Autonomy* , that marries both soft and hard real-time applications into a single system. Figure 4 illustrates this hybrid architecture and denotes each component's hard, soft or other real-time requirements.

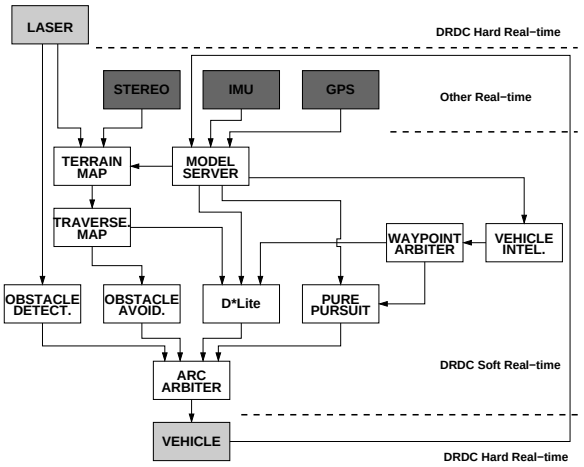

Fig. 4. Representative Flow Chart of DRDC's Hybrid Architecture

Two DRDC components, the laser and the vehicle, have hard real-time requirements[1], as they implement time critical functionality. The remaining components, implementing the majority of the autonomous capabilities, operate under soft real-time requirements.

A. Notification Services for Soft Real-time Applications

DRDC adapted the Miro framework [17] to support UGV applications, where Miro builds upon the CORBA notification services to implement a publish/subscribe architecture. Under this paradigm producers anonymously publish data, while anonymous consumers subscribe to specific event types. CORBA, via the IDL compiler, Naming Service and notification services, provides all the required facilities to implement this process, as shown in Figure 5.

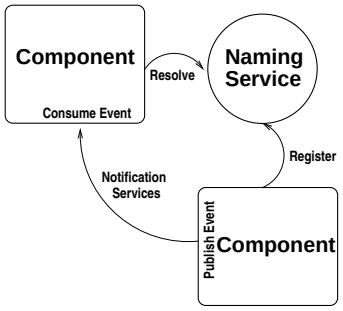

Fig. 5. The Publish/Subscribe Paradigm under CORBA Notification Services

The publish/subscribe approach has an elegance and practicality that makes it well suited for robotics implementations. Robotic applications, such as UGVs, tend to be event driven, where the reception and analysis of new data triggers a new or different mode of operation. Additionally, DRDC's UGV fleet varies significantly in size and capabilities. CORBA's network

[1]The Stereo and INS devices may/will have internal real-time requirements, that as proprietary devices, are beyond DRDC's control.

transparency seamlessly enables distributed computing on the larger platforms where multiple computers are available.

Unfortunately, the high level capabilities provided by notification services come with a cost; namely they impose a computational and resource burden. DRDC has evaluated notification service's performance under a variety of conditions and the results of these investigations are given in the following section.

IV. Performance of Notification Services

A. Experimental Setup

As was detailed in Section II, a UGV's operational environment defines the time window for sensing, modeling, planning and acting (SMPA). This time window is inversely proportional to the vehicle's speed; as the speed increases less time is available. When an obstacle appears within the sensing range, the total time from the data acquisition, through its transportation, processing, decision making and command execution, must be less than the available time window. This process is shown in Figure 6.

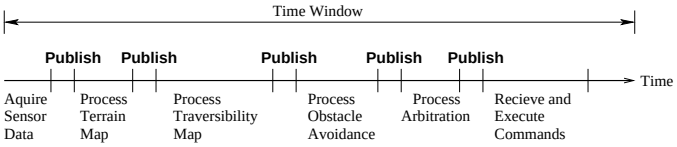

Fig. 6. Process Execution Time Line

DRDC implements each process as a component that uses notification services to publish events. The reception of sensor data triggers a cascade of subsequent processes leading ultimately to a steering or braking command. Given the dependence on notification services, it is crucial that each individual publication cycle consumes a minimal time slice. Research conducted at DRDC profiled publication times for typical UGV data structures including:

- Laser rangefinder data and stereo vision range data,
- Raw visual images,
- Terrain and traversibility maps, and,
- Various command and control structures.

The corresponding data structures sizes are given in Table III. Only the components that handle large data structures are given, as the demands placed on the notification services are relative to the data sizes. Note, stereo range data maybe published as a static array of doubles[2], or as a sequence of long integers.

The following sections provide the experimental details.

B. Event Publication under Notification Services

Experiments were conducted using data structures that represented typical UGV data transfers where each data structure is roughly an order of magnitude different in size. The data transfers, using event publications, are labelled small, medium, large and very large, where the laser range, terrain map, stereo range and stereo range/image were used as representative examples. For all experiments, the primary computer is a 3.2 GHz Intel Pentium 4, with 1 GB RAM, a 100 Mbit ethernet interface, that runs linux and utilizes the ACE/TAO 5.4.10 version of CORBA.

[2]When published as doubles ATLAS matrix multiplication can be directly used.

Component	Structure	Type	Size	Bytes
Laser Range	Sequence	Long	361x4	5,776
Stereo Range	Static	Long	320x240x4	1,228,800
Stereo Range	Sequence	Double	320x240x4	2,457,600
Rect. Image	Sequence	Short	320x240x3	460,800
Raw Image	Sequence	Short	1024x768x3	4,718,952
Terrain Map	Sequence	Double	150x150x3	540,000
Traverse Map	Sequence	Double	40x40x2	25,600

TABLE III
Typical UGV Data Structure Sizes

1) Publication of Small Data Structures: Laser range data of 5,776 bytes, provided by the SICK LMS laser, is representative of a relatively small data structure. Laser range events are published by the laser driver at a 26.6 ms update rate and the notification services implement one of three CORBA protocols for data delivery:

- Internet Inter-Orb Protocol (IIOP),
- Unix Inter-Orb Protocol (UIOP), and
- Shared Memory Inter-Orb Protocol (SHMIOP).

SHMIOP and UIOP are restricted to a single host, while IIOP is applicable to both local and remote configurations. Table IV shows the event publication times for these experiments.

Transport	Protocol	Time us	Std. us
Local	IIOP	487	37
Ethernet	IIOP	1317	34
Local	UIOP	477	15
Local	SHMIOP	540	18

TABLE IV
Publication Times for Laser Range Data, Sequence Structure, 3600 data samples, Publish by Reference

As can be seen in this table, the CORBA protocol has a negligible affect on the publication time. However, when the data was piped over the ethernet the publication time approximately doubled. In all instances the publication time was small and never exceeds 1.5 ms, which is significantly smaller than the 26.6 ms laser scan period.

2) Publication of Medium Data Structures: These experiments mirrored the previous experiment, with the exception that the data payload now represents a terrain map. The times required to publish 540,000 bytes are given in Table V.

The time required to publish the map event is relatively small with its magnitude is near 7 ms for all protocols, including publishing across an ethernet interface.

3) Publication of Large Data Structures: The Bumblebee stereo camera can publish range events using either a CORBA

Transport	Protocol	Time ms	Std. ms
Local	IIOP	7.25	0.54
Ethernet	IIOP	7.03	0.35
Local	UIOP	7.02	0.32
Local	SHMIOP	7.03	0.46

TABLE V
PUBLICATION TIMES FOR TERRAIN MAP DATA, SEQUENCE STRUCTURE, 1700 DATA SAMPLES, PUBLISH BY REFERENCE

sequence or a static array. Regardless of the structure used, this event represents a large data payload. For these experiments investigators profiled three different publication approaches:

- A CORBA sequence of 1,228,800 bytes,
- A statically defined array of 2,457,600 bytes, and,
- Publications using the notification service to signal the presence of data in linux shared memory (conforming to POSIX.1-2001).

In the first experiment, the range data is stored as a sequence and table VI shows the publication times.

Transport	Protocol	Time *ms*	Std. *ms*	Publish By
Local	IIOP	26.3	0.45	Reference
Ethernet	IIOP	28.0	1.6	Reference
Local	UIOP	24.7	0.55	Reference
Local	SHMIOP	24.7	0.48	Reference
Local	IIOP	13.6	0.09	Pointer
Ethernet	IIOP	13.1	0.22	Pointer

TABLE VI
PUBLICATION TIMES FOR STEREO DATA, SEQUENCES OF 1,228,800 BYTES, 1000 DATA SAMPLES

Although publishing via a pointer[3] reduces the apparent publication time, the cost associated with creating and initializing a new sequence is approximately 13.5 *ms*, so there is no overall savings accrued. Hence, publication time remains essentially constant, regardless of the protocol or invocation type.

In the second experiment, the storage array was changed to a static structure. Table VII reveals that this switch had a significant and variable impact on the publication times.

Transport	Protocol	Time *ms*	Std. *ms*	Publish By
Local	IIOP	10.2	0.27	Reference
Ethernet	IIOP	206	37	Reference
Local	UIOP	10.2	0.94	Reference
Local	SHMIOP	10.4	0.98	Reference
Local	IIOP	2.7	0.04	Pointer
Ethernet	IIOP	188	31	Pointer

TABLE VII
PUBLICATION TIMES FOR STEREO DATA, STATIC ARRAY OF 2,457,600 BYTES, 1000 DATA SAMPLES

[3]When CORBA receives a pointer to data it automatically deletes object after publication is completed.

Although this experiment doubles the memory requirements, for local delivery the publication time is roughly half that required by a sequence based structure. Publication across an ethernet interface is an exception to this trend as it is significantly slower. The invocation type also impacted the publication time, as publishing via a pointer was noticeably quicker. The cost associated with creating the new static structure is approximately 15 *us*, hence using this invocation type had a significant performance benefit.

For the next experiment notification services only signalled the arrive of new data, the actual transfer of range data between processes used linux shared memory. The results of this experiment are given in Table VIII and show that notifications services for signalling is extremely fast.

Storage	Protocol	Time *us*	Std. *us*
Static	Shared Memory	35	4

TABLE VIII
LINUX SHARED MEMORY FOR STEREO DATA, 1000 DATA SAMPLES

Although the shared memory approach is efficient it is not suitable for dynamic structures such as CORBA sequences, as they internally manage memory allocations that are not easily mapped into shared memory allocations. Additionally, this approach is not network transparent, thus limited to a single computer configuration.

4) Publication of Very Large Data Structures: For the final experiment a sequence based structure including: stereo range data, the rectified image and a raw image, was published at a 1 fps rate. The times required to publish this 6,408,552 bytes of data are given in Table IX.

Transport	Protocol	Time *ms*	Std. *ms*
Local	IIOP	299	5.9
Ethernet	IIOP	295	12.5
Local	UIOP	295	4.7
Local	SHMIOP	292	4.4

TABLE IX
PUBLICATION TIMES FOR STEREO DATA, SEQUENCES, 1000 DATA SAMPLES, PUBLISH BY REFERENCE

This table reveals that notification services limits the publication rate to a maximum of 3 fps, assuming that other factors, such as the available processing power, do not erect other barriers.

V. CONCLUSIONS

The field of robotics covers a wide spectrum of applications, which operate in an equally large range of environments with diverse requirements. Under the unmanned vehicle subset, an unmanned ground vehicle's operating environment and operational requirements could be suitable for a hybrid hard/soft real-time computing implementation. Thus, component based software engineering, using middleware toolkits, could be

utilized where soft real-time capabilities are required, while a hard real-time operating system is only used where required. For this approach to be successful the requisite middleware must not impose significant time delays or performance penalties.

DRDC researchers have implemented such a hybrid approach using linux and CORBA for soft real-time requirements, and RTEMS where hard real-time is required. Given that a majority of DRDC's autonomy applications have soft real-time requirements this approach should, in principle, provide acceptable results. In-depth investigations have shown that CORBA middleware, using notification services, can meet the DRDC's Raptor UGV requirements, under specified speed limitations. These investigations revealed that, for common UGV data structures, the event publication process does not impose a significant burden. For all data structure sizes, except those that are very large, the publication process commonly consumes only 10's of milliseconds or less. Under standard operating conditions the Raptor UGV doesn't exceed 2.8 *m/s* (10 km/hr), hence it has a time window in excess of 7 sec. for the SMPA cycle, while the stopping time is estimated at less than 2 seconds. Given that this cycle has a pipeline length limited to 5 or 6 stages, the summed publication time has an upper limit in the order of 100 *ms*, which represents roughly $\frac{1}{50}^{th}$ of the available time. Thus, it is evident that CORBA notification services are applicable under the Raptor's operating conditions. Even for a speed of 5.6 *m/s*, where the stopping time is greater than 2 sec. and the time window is 3.6 sec., the overhead associated with event publications are relatively insignificant.

The overhead associated with notification services becomes an issue for distributed computing applications, where very large data structures are shared between separate computers. Under such configurations, the marshalling time, TCP/IP overhead and limited bandwidth all conspire to limit performance. Improved middleware performance, without impacting the CBSE approach, is possible using a different publish-subscribe communication paradigm. Currently DRDC uses CORBA notification services to propagate events from publishers to subscribers. Data Distribution Service (DDS) is an alternative approach that provides similar capabilities, but features lower overhead and, thus, is better suited for higher performance systems. Future DRDC research will investigate adapting Miro to support DDS, and then investigate its performance on the Raptor UGV, along with other DRDC unmanned vehicles.

Finally, it is possible to dramatically improve a single computer's performance using a mixed approach, where notification services are used for signalling and shared memory transports the bulk of the data. Under this paradigm, the 10's of *ms* required to publish large data structures can be reduced by 3 orders of magnitude to 10's of *us*. The tradeoffs to achieve this efficiency are less scalability, modularity and portability, as shared memory breaks network transparency and it precludes using dynamic data structures such as sequences. Although it is possible to reduce publication times by orders of magnitude, the benefits are only marginal given that overall maximum time is already relatively small. Extending the sensor ranges, which increases the maximum obstacle detection distance, is a significantly more productive approach for enlarging the available time window

REFERENCES

[1] G. Broten, S. Monckton, J. Giesbrecht, and J. Collier, "Software sysetms for robotics, an applied research perspective," *International Journal of Advanced Robotic Systems*, vol. Volume 3, 1, no. 2005-204, pp. 11–17, March 2006.

[2] H. Utz, S. Sablatnog, S. Enderle, and G. Kraetzschmar, "Miro - middleware for mobile robot applications," *IEEE Transactions on Robotics and Automation*, June 2002.

[3] F. Bolton, *Pure CORBA: A code intensive premium reference*. SAMS, 2002.

[4] A. Brooks, T. Kaupp, A. Makarenko, A. Oreback, and S. Williams, "Towards component-based robotics," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, August 2005.

[5] M. Henning, "A new approach to object-oriented middleware," *IEEE Computer Society*, vol. 8, no. 1, pp. 66–75, January-Febuary 2004.

[6] D. J. Reid Simmons, *IPC - A Reference Manual*, Carnegie Mellon University - School of Computer Science / Robotics Institute, February 2001, iPC Version 3.4.

[7] J. Gowdy, "Ipt: An object oriented toolkit for interprocess communication," Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-96-07, March 1996.

[8] J. . Pedersen, "Robust communications for high bandwidth real-time systems," Carnegie Mellon University, Tech. Rep. CMU-RI-TR-98-13, 1998.

[9] C. Szyperski, *Component Software: Beyond Object-Oriented Programming*. Reading, MA: Addison-Wesley, 1998.

[10] C. Urmson, J. Anhalt, D. Bartz, M. Clark, T. Galatali, A. Gutierrez, S. Harbaugh, J. Johnston, H. Kato, P. L. Koon, W. Messner, N. Miller, A. Mosher, K. Peterson, C. Ragusa, D. Ray, B. K. Smith, J. M. Snider, S. Spiker, J. C. Struble, J. Ziglar, and W. R. L. Whittaker, "A robust approach to high-speed navigation for unrehearsed desert terrain," *Journal of Field Robotics*, vol. 23, no. 8, pp. 467–508, August 2006.

[11] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney, "Stanley: The robot that won the darpa grand challenge," *Journal of Field Robotics*, vol. 23, no. 9, pp. 661–692, September 2006.

[12] W. T. free encyclopedia, "Real-time computing," 2007, http://en.wikipedia.org/wiki/Real_time.

[13] G. Broten and J. Collier, "Continuous motion, outdoor, 2 1/2d grid map generation using an inexpensive nodding 2-d laser rangefinder," in *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, no. 2006-061, Orlando, Fl, May 2006, DRDC Suffield CP, pp. 4240–4245.

[14] M. Pivtoraiko, A. Kelly, P. Rander, and J. Bares, "Efficient braking model for off-road mobile robots," in *Field and Service Robotics*, Port Douglas, Australia, October 2005.

[15] L. Alvarez, J. Yi, R. Horowitz, and L. Olmos, "Dynamic friction model-based tire-road friction estimation and emergency braking control," *Transactions of the ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 127, pp. 22 – 32, March 2005.

[16] T. Day and S. Roberts, "A simulation model for vehicle braking systems fitted with abs," in *SAE 2002 World Congress*, ser. SAE Technical Paper Series 2002-01-0559, Detroit, Michigan, March 2002, pp. 1 – 19.

[17] S. Enderle, H. Utz, S. Sablatnog, S. Simon, G. Kraetzschmar, and G. Palm, "Miro - middleware for autonomous mobile robots," *International Federation of Automatic Control*, 2001.