AAS 06-096

TRINITY, AN OMNIDIRECTIONAL ROBOTIC DEMONSTRATOR FOR RECONFIGURABLE COMPUTING

Samuel A. Miller,[*] Arthur T. Bradley,[*]
Nathanael A. Miller,[*] Robert F. Hodson[*]

In this paper, we present the results of an unmanned ground vehicle robot development effort completed at NASA LaRC's Robotics and Intelligent Machines Laboratory. The robot is capable of conducting both tele-operation and autonomous activities. Novel omnidirectional mobility is achieved using three Mecanum wheels operating in a configuration not previously seen by this team. The robot is equipped with a suite of sensors, including a stereoscopic camera, FLIR, omnidirectional camera, ultrasonic range finders, 3-degree-of-freedom gyroscope, experimental acoustic eye, and ex-ray/visible spectrum fluoroscopes. The robot's architecture is designed to support reconfigurable scalable computing resources that can be dynamically adapted to changing mission requirements. This effort was funded as part of NASA's Office of Exploration Systems and is meant to demonstrate the utility of reconfigurable modular electronics.

INTRODUCTION

NASA's Exploration Systems Architecture Study (ESAS) has identified reconfigurable computing as a required technology to meet processing requirements for future missions to the Moon and Mars. Reconfigurable computing uses Field Programmable Gate Arrays (FPGAs) as the primary computing element to implement electronic functionality. Reconfigurable computing has been shown to both increase performance and reduce power consumption of embedded applications.

In addition to performance and power benefits, reconfigurable computing offers cost savings, as the hardware elements of a reconfigurable computer can be modified without re-qualification for space. There is a broad range of applications that can be accommodated without redesign, thus eliminating engineering costs in future missions. A reduction in spare-parts inventory is also a cost-saving byproduct of the common hardware approach.

The Reconfigurable Scalable Computing (RSC) project is working toward building the first space-qualified reconfigurable computer. As part of that effort, a demonstration system is needed to establish the real-world benefits of a reconfigurable architecture. The team has therefore developed a 3-wheel omnidirectional robot, called *Trinity*, to demonstrate the applicability of reconfigurable computing for real-time control and data processing. A multi-sensor robotics demonstration application was chosen because it provides a challenging real-time environment and has practical application to NASA's future exploration missions. This development leveraged previous robotics work

[*] NASA Langley Research Center's Electronic Systems Branch, 5 North Dryden Street, Hampton, VA 23681. E-mail: arthur.t.bradley@nasa.gov. Phone: (757) 864-7343. Fax: (757) 864-7944.

at NASA Langley's Robotics and Intelligent Machines Laboratory in which a 4-wheeled omnidirectional system was constructed [1]. *Trinity* further advances our investigation into novel robotic platforms for control-algorithm development and multi-sensor data processing.

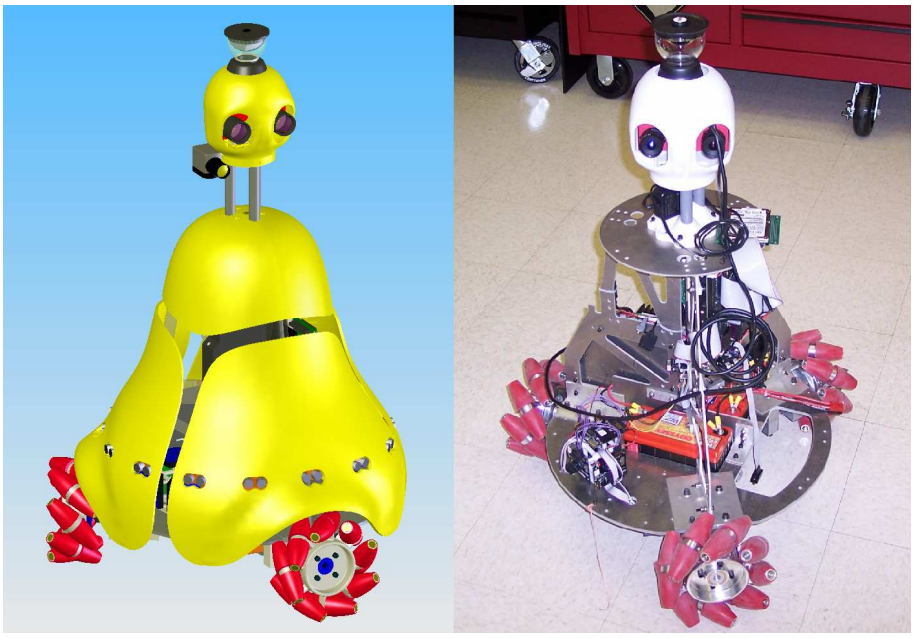
This paper provides an overview of the *Trinity* robot and the RSC system. It describes the current electronics architecture, sensor suite, the system kinematics, and the control algorithm. The paper also discusses the future mapping of the control and data-processing functions onto the RSC system and describes the benefits of a reconfigurable approach for real-time control of robotics applications.

ROBOT CONCEPT

Trinity's primary purpose is to provide a mobile platform for demonstrating computing devices, instruments, and algorithms. It is a rolling product showcase – a device suitable for robotics research, algorithm development, sensor technology demonstrations, outreach events, and computer capability testing.

As with many robots, high-level functionality is the driving motivation for its creation. This robot is designed to support three high-level goals:

I. Provide a mobile platform to demonstrate a functional implementation of Langley's Reconfigurable Scalable Computing (RSC) architecture;
II. Facilitate mobile demonstrations of scientific/research instruments; and
III. Provide sensors, computational resources, and a full-featured development environment for implementing and testing autonomous robotic algorithms.



a) Concept **b) Physical System**

Figure 1 *Trinity*, a 3-wheeled omnidirectional robot

The goals are far-reaching, but we are confident that the combination of high-performance space-ready RSC modules, scientific instrumentation, and omnidirectional mobility represents a unique technology tool for overcoming challenges associated with NASA's vision of creating a sustained space-exploration program.

Trinity has a three-wheeled omnidirectional mobile chassis. As shown in Figure 1, its holonomic mobility comes from the novel orientation of its Mecanum wheels. Although there are many methods to achieve holonomic motion [2-8], the three Mecanum wheels serve as an original method of locomotion, both functional and previously unseen in robotics literature.

Although the robot serves primarily as a computer-system/algorithm test-bed, it was also used to explore non-traditional paths to mechanical implementation. The hardware design is a frame constructed of water-cut aluminum plates, with ancillary components attached with adaptors "grown" in a continuous-fused-deposition machine.

The protective panels forming the robots skin are "grown" using a similar process. Water-cutting and RPM (rapid prototyping and manufacturing) are two rapid manufacturing processes that significantly reduce the amount of time required to build high-precision complex geometries. Creating the robot's structure in this way resonates with the high-level purpose of demonstrating a flexible architecture for complex computing tasks.

Currently, space-exploration efforts are severely hampered by the lack of high-performance computing systems. The RSC architecture proposes to remedy the situation by providing a system that:

1. Uses reconfigurable logic resources that can be optimized for specific applications without re-qualifying the hardware;
2. Is modular and can be scaled to meet the processing requirements of the user application;
3. Uses components that offer higher performance-growth rates than current space processors, thus providing a sustainable long-term path for future space-computing needs; and
4. Has smaller volume, less mass, and requires less power, than current systems.

Pending the arrival of the full RSC system (illustrated in Figure 2), the robot's computational needs are met with standard stackable computers. Currently there are two such computers on the robot, both running general-purpose Linux operating systems. One computer is devoted to low-latency processing and handles real-time motion control and operator feedback. The computer (a 700 MHz PIII) runs the RTLinux operating system, which is well-suited to feedback-loop control problems.

With a 2.0 GHz P4M processor and a gigabyte of RAM, the second computer is also quite capable. This processor handles computationally intensive data-manipulation operations, including processing the video data from the robot's multiple cameras.

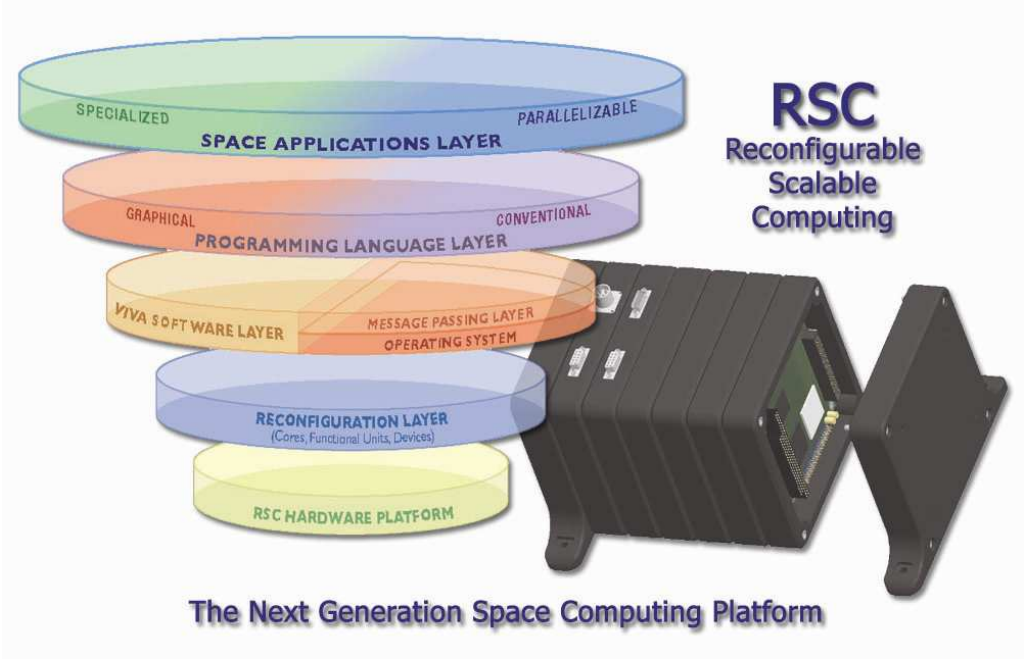


Figure 2 The RSC system

KINEMATICS

The kinematics of the three-wheeled omnidirectional robot are described in this section. Simple trigonometry and geometry combined with a few simplifying assumptions lead to a very intuitive and easy-to-use control solution.

The relation between joystick position and angular wheel velocity for the three-wheeled robot is determined by first studying the conditions required for the vehicle to be free of rotation. This separation of rotation from translation is a fundamental assumption to achieving a simple kinematic solution. From Figure 3, we see that for the system to move only in translation, the rotational forces caused by tangential components of the wheel velocity vectors must sum to zero.

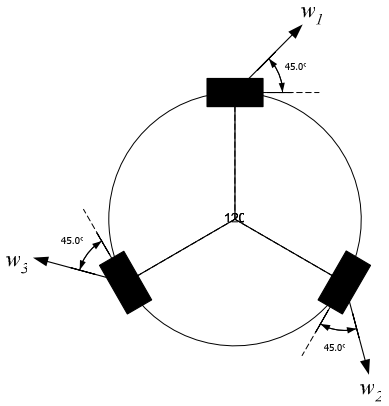


Figure 3 Tangential velocity vector components.

Assuming a rigid body, and ignoring constants (i.e., robot radius and cos(45°)) we see that there exists a simple condition for rotation-free translational movement – namely,

$$w_1 + w_2 + w_3 = 0. \tag{1}$$

We use the following relation to convert from wheel-translation velocities to angular wheel velocities:

$$w_i = R \cdot \omega_i, \tag{2}$$

where w_i denotes the wheel-translation velocity (m/s), ω_i denotes the angular wheel velocity in rad/sec, and R denotes the wheel radius in meters. The wheel-translation velocities (w_i's) can also be thought of as components of the net translational velocity vector, all of which combined to determine the robot's translational motion.

To include rotational movement, we set the term in (1) proportional to the joystick z-axis term, z_j. Note also that this solution is independent of any rotation in axes (something we will take advantage of later).

$$K_R(\omega_1 + \omega_2 + \omega_3) = z_j \tag{3}$$

Note that all the constants have been combined into K_R, a user-defined rotational sensitivity constant.

When considering the conditions required for translation, it is convenient to use a coordinate system rotated by 45°. If we assume the robot acts as a rigid body, we can arrive at the free body diagram given in Figure 4.

Writing the robot's total velocity vector, simple trigonometry yields

$$\vec{v} = K_T\left\{\frac{\sqrt{3}}{2}(\omega_2 - \omega_3)\,\hat{x} + \left(\omega_1 - \frac{1}{2}(\omega_2 + \omega_3)\right)\hat{y}\right\}. \tag{4}$$

If we define the robot's forward direction to be along the y-axis, we can directly extract the relationship between joystick movement and angular wheel velocity.

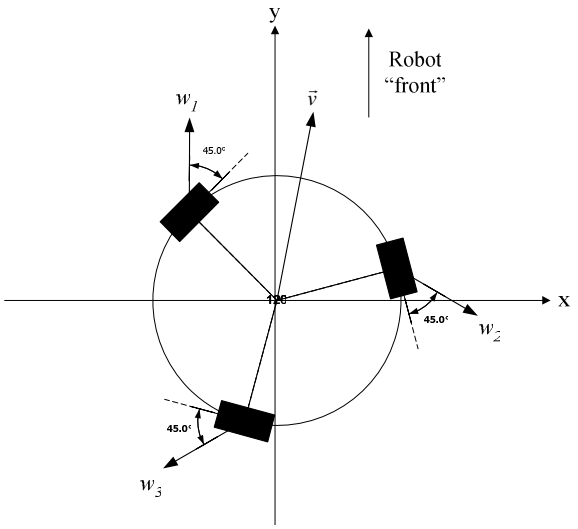


Figure 4 Convenient coordinate system

$$x_j = \frac{\sqrt{3}}{2} K_T (\omega_2 - \omega_3)$$
$$y_j = K_T \left[\omega_1 - \frac{1}{2}(\omega_2 + \omega_3) \right] \quad (5)$$
$$z_j = K_R [\omega_1 + \omega_2 + \omega_3]$$

where K_T is another user-defined translational sensitivity constant.

Solving the set of linear equations for angular wheel velocities, we arrive at the relation between individual wheel speeds and the desired translation and rotational movement provided by the joystick.

$$\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} = \begin{bmatrix} 0 & \frac{2}{3K_T} & \frac{1}{3K_R} \\ \frac{\sqrt{3}}{3K_T} & -\frac{1}{3K_T} & \frac{1}{3K_R} \\ -\frac{\sqrt{3}}{3K_T} & -\frac{1}{3K_T} & \frac{1}{3K_R} \end{bmatrix} \begin{bmatrix} x_j \\ y_j \\ z_j \end{bmatrix} \quad (6)$$

SYSTEM ARCHITECTURE

Electrical Hardware

The central elements of *Trinity*'s electrical-hardware architecture (shown in Figure 5) are the two high-performance embedded computers. Functionally, these two computers facilitate all three goals of the project. First, their PC/104-PLUS stackable form-factor allows the addition of individual Reconfigurable Processor Modules (RPMs) from the RSC architecture. As will be discussed later, adding RPMs facilitates replacing some low-level microcontrollers and some software-bound computing tasks – currently implemented in the high-level processors – to their own dedicated hardware components. Such future hardware additions would convert some of Figure 6's software-process blocks into much higher-performance hardware modules that encapsulate the same functionality.

The electrical hardware architecture supports the project's second goal (mobile demonstration platform for science/research instruments) by facilitating advanced data processing on the vehicle. This onboard processing reduces bandwidth dependence for processing scientific data, and allows the robot's higher-level functions access to advanced data products on which it can base navigation decisions. Finally, the architecture, with its wide array of sensors, microcontrollers, computers, and software development environments provides a self-contained unit ideally suited to autonomous-algorithm development.

Computers/Microcontrollers

The fastest computer on *Trinity* is the 2GHz Pentium M, Lippert, GmbH. CoolRoadRunner IV, dubbed *Oculus*. Its primary purpose is processing real-time video streams from the three cameras described in the sensors section below. Currently there are two PC/104-PLUS cards attached to *Oculus*: an Advanced Micro Peripherals, Ltd. FireSpeed 2000 IEEE-1394 (FireWire) controller, and a WinSystems, Inc. PPM-CardBus adaptor. The FireWire controller transfers raw data from the three cameras directly to the

processor using a direct memory interface, and the CardBus adapter is the attachment point for a NETGEAR, Inc. WG511 IEEE-802.11b wireless card.

The second computer's primary role is low-latency data processing for the less bandwidth-hungry sensors and indicators. The computer itself (*Lapsus*) is a 700MHz Pentium III, Diamond Systems, Inc. Hercules. The only PC/104-PLUS card currently attached is the same PPM-CardBus as on *Oculus*, but mounts a D-Link, Corp. DWL-G650 IEEE-802.11b/g wireless card.

The other devices attached to *Lapsus*' RS-232 ports are a Matrix Orbital, Corp. LCD display, a MicroStrain, Inc. 3DM orientation sensor, and an Acroname, Inc. Brainstem GP microcontroller. The sensors and PC/104-PLUS card indicated in Figure 5 with dashed lines are future additions.

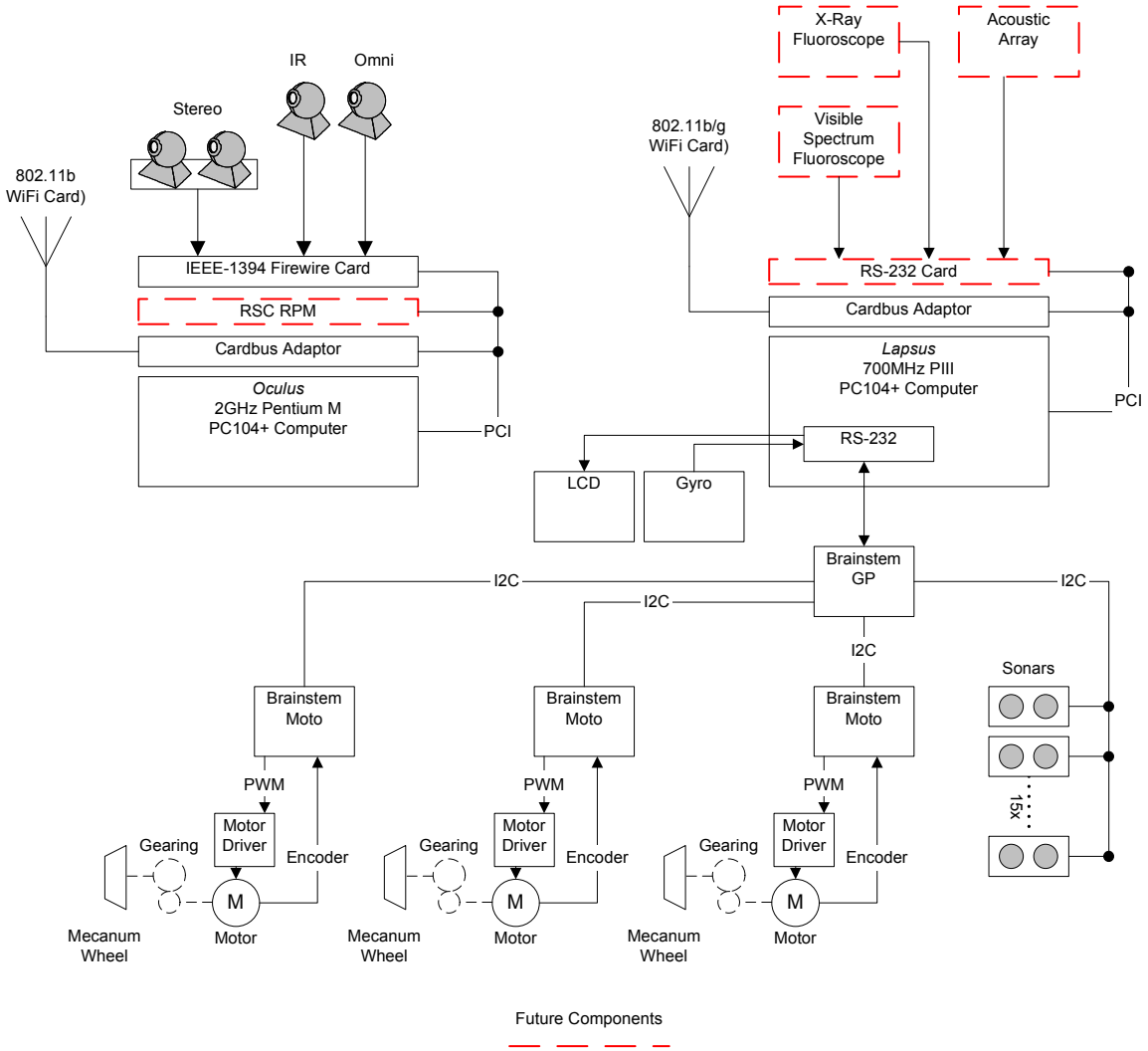


Figure 5 Electrical-hardware architecture

The Brainstem GP microcontroller acts as a router to send and receive low-level motion data from Brainstem Moto modules. The GP also coordinates data-capture and data-relay from 15 Devantech, Ltd. SRF08 sonar range-finders. (Although their main function is acquiring range information, each SRF08 also collects light-level data from its built-in light sensors.) The GP communicates with the Motos and SRF08s using a high-speed I^2C serial communications network. In its current configuration the I^2C alternates between 400kbs, when talking to the sonars, and 1Mbs, when communicating with the Motos.

Based on velocity set-point commands routed to it through the GP, each Moto generates a pulse-width-modulated (PWM) signal for its respective Devantech, Ltd. MD03 motor driver. Each motor driver controls a Maxon, AG. RE 30 motor with a paired HEDL 55 300-count, two-channel, optical encoder. Each Moto's internal PD control loop maintains the current velocity set-point based on this two-channel encoder feedback.

The hardware architecture is distributed in nature – designed so the high-level processors are minimally involved with the low-level operations. The Moto modules handle motor velocity feedback control, the sonars each handle their own time-of-flight monitoring for distance calculation, and the GP handles overall sonar timing and general data routing. Thus, *Lapsus*' software has no critical timing issues related to regulating the minute details of motor and sonar operation.

Sensors

Trinity boasts a diverse sensor suite. The following are currently implemented sensors.

- A VidereDesign stereoscopic camera: resolutions from 640x480 at 30 fps to 1280x1024 at 7.5 fps
- A FLIR Systems infrared camera: 160x120 at 30 fps
- An Eizoh omnidirectional camera: captures 640x480 360° pictures at 30 fps
- 15 Acroname SRF08 ultrasonic sonar modules
- A Microstrain 3-DOF inertial gyroscope

The following instruments await integration.

- An experimental "acoustic eye" from the Air Force Research Lab
- An x-ray fluoroscope and data-processing module
- A visible-spectrum fluoroscope and data-recording module

For the scope of this project, the sensor suite is classified into three primary groups: high-resolution environment sensors, low-resolution/limited-scope environment sensors, and scientific instruments.

The first group, high-resolution environment sensors, includes the stereo, infrared, and omnidirectional cameras. These sensors have general application to high-level autonomy functions (e.g., target acquisition, localization, and navigation).

The second group – the ultrasonic sonars, acoustic eye, and inertial gyroscope – are more closely related to low-level autonomy functions, such as obstacle avoidance (sonars), camera retargeting (acoustic eye), and motion stability (gyroscope).

Finally, as their group name suggests, the scientific instruments fill the classical role of obtaining detailed measurements of particular phenomena. In this setting, the x-

ray fluoroscope characterizes the elemental composition of ground surfaces, and the visible-spectrum fluoroscope performs an analogous function for the environment surrounding the robot.

Algorithms

For input to a mobile algorithm platform, the sensor suite described above provides a diverse and rich data set. Although there are many directions we could take our algorithm-development efforts, our initial objective is to focus on three areas:

1. Vision algorithms for navigation and localization,
2. Control algorithms for vehicle-stabilization/obstacle-avoidance, and
3. Algorithms for scientific data processing.

In each of these areas it is important to note that we do not intend to develop, for instance, vision algorithms like a university might. Instead, we see our role as synthesizing algorithms others have already formulated and perfected. Our interest is in the functionality of the *system*, rather than the perfection of the *parts*. We have therefore chosen algorithmic objectives that tie back to the three motivating goals of the project.

The initial objective for the vision system is to combine the diverse capabilities of the various cameras to provide the robot with room navigation and localization capabilities. The base-line localization system will use the omnidirectional camera for location identification. Developed by researchers at Carnegie Mellon [14], the system identifies locations (e.g., rooms, hallways, or outdoor spaces) by the histogram profiles of the omnidirectional images. For location-to-location navigation, we will implement an algorithm that correlates edges detected from a single image of the stereo camera with high-spatial-frequency changes in the depth readings from the stereo camera. Armed with these two baseline capabilities (finding doors and ascertaining its own location), the robot is poised for many high-level robotic tasks – such as "night watchman," "errand boy," or "find science target." Another algorithmic challenge we intend to pursue is fusing imagery from the visible spectrum cameras with the IR images from the FLIR camera. The combined images hold promise as a diverse data product useful for many low-light scenarios, and could also greatly benefit tracking warm people in typically cluttered indoor spaces.

The high-level vision system requirements correlate nicely with the overall project goal of providing a mobile demonstration platform for the RSC architecture. Initially the algorithms will run on the high-performance stackable computer mentioned earlier, but when the requisite hardware becomes available, portions of the high-level vision-processing will run on a dedicated RSC hardware module configured for high-bandwidth/computationally expensive operations.

Our focus for the second algorithmic objective (vehicle stabilization/control and low-level obstacle avoidance) stems from roots in standard control theory, and the more relaxed ideas of behavior-based autonomy. We plan on developing an autonomous stabilization algorithm that uses a standard set-point feedback loop to maintain a desired speed and heading: the algorithm will use vehicle-direction inputs from either a joystick or an autonomous guidance system, encoder counts from the wheels, and heading information from the gyroscope. This method will provide the foundation for high-

performance "drive-by-wire" control for human-guided operation, as well as a dynamically stable motion foundation for autonomous operations.

The other side of our low-level control challenge is obstacle avoidance. The first-order sensors for obstacle avoidance are the sonar arrays. From an algorithmic standpoint, behavior-based control ideas are a promising approach to base-line obstacle avoidance using the sonar modules. Additionally, for nearly any desired functionality, knowledge about navigation *goals* must pass between the high-level vision system and the obstacle-avoidance algorithms; the behavior-based approach easily facilitates loose data interactions between the high-level goals and the low-level data products, furthermore it accommodates data-source additions later in the robot's development cycle. Our first-order objectives in this realm are behavioral algorithms allowing *Trinity* to autonomously navigate typical indoor spaces, avoiding people and objects using low-level animal-like instincts and behaviors.

The robot's final algorithmic objective is scientific-data processing. Our approach is to analyze the data using algorithms supplied by the instrument designers. Initially, the robot will have no autonomous functionality for acquiring science targets, but will rather rely on human intervention for specific data-acquisition instructions. Obviously, a future goal of the project is to implement autonomous algorithms for science-target discovery.

Algorithm Implementation

Integrating the algorithms mentioned above is a key challenge for this robotic system. Particularly, the tight interconnection that must exist among the high-level vision algorithms and the low-level navigation/obstacle avoidance behaviors requires a system-integration tool that is easily reconfigurable.

Further motivating this need for configuration flexibility is the nature of the algorithms themselves. Because their inspiration and origins stem from studies of human/animal cognition, tuning the algorithms will require the trial-and-error method found in natural learning-systems.

From our observations of algorithm implementation tools, the methodology most amenable to trial-and-error reconfiguration is that employed in the icon-based, data-flow graphical toolkits such as National Instrument's LabVIEW and MathWorks Simulink. Our overarching approach to algorithm integration is thus to "wrap" the various sensor algorithms (for vision, obstacle avoidance, etc.) into one of these toolkits, where we can easily change the macro-scale inter-algorithm data-passing options and their interconnection topologies. Our work on this goal thus far has produced sets of objects that act as "glue" between core software functions in C/C++ and LabVIEW's graphical programming language. The libraries currently being integrated are Intel's OpenCV image processing library [13] and the real-time hardware interface code outlined in the next section.

Particularly in LabVIEW's case, the object-oriented nature of the graphical coding blocks facilitates easy reconfiguration of the algorithm options. This graphical, object-oriented reconfiguration capability promises great labor-saving dividends at all levels – from the low-level algorithms to the high-level system interconnections. With the eventual integration of the RSC modules, Starbridge Systems, Inc. VIVA programming language can easily extend the object-oriented graphical approach to FPGA algorithm implementations.

Software

The previous two sections outline *Trinity*'s hardware architecture and algorithmic methodology. Figure 6 graphically represents its software topology and current state of realization. The software spans multiple coding languages and the full spectrum of hardware implemented.

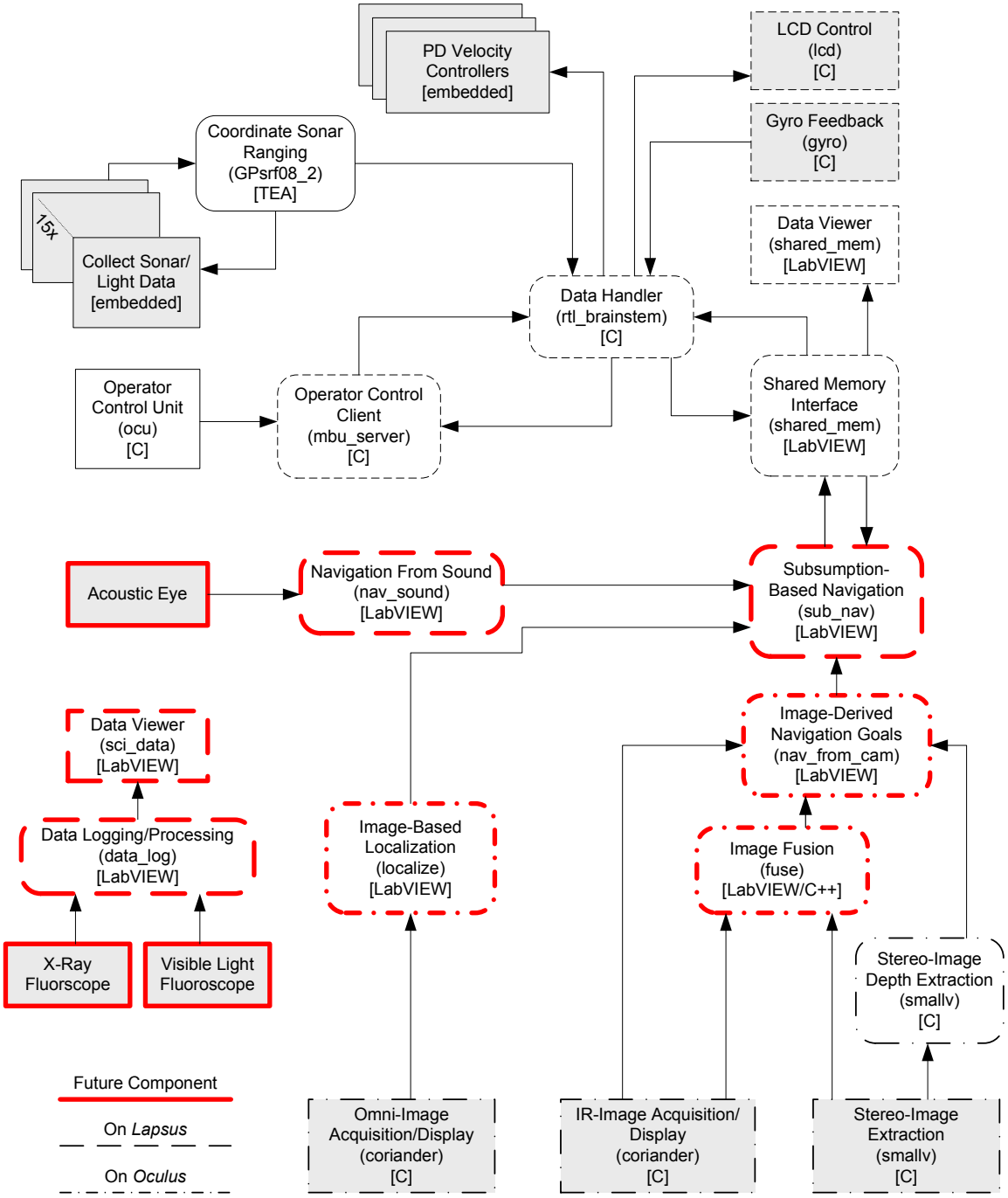


Figure 6 Software architecture

Each grey box in Figure 6 represents a hardware device, or an interface to one. Some devices, such as the sonars and PD velocity controllers, are equipped with embedded software-interfaces (the sonars calculate distance values internally, and the velocity controllers' embedded code maintain wheel-speed set-points). The other hardware interfaces currently implemented are the LCD and gyro, and the three image-extraction blocks. While the LCD and gyro software modules provide access to the hardware, the camera-interface blocks not only capture camera data, but also provide display functionality. The remaining hardware interfaces – both fluoroscopes and the acoustic eye – await integration.

Central to the upper half of the software interaction diagram is the "Data Handler" – a real-time Linux application that coordinates the acquisition of data from the sonar and gyro interfaces, sends and receives commands from the motor controllers and user commands from the operator control client, and posts the system status to the LCD module. All these programs communicate and share their data values though a shared-memory data structure on *Lapsus*.

The operator-control client, also resident on *Lapsus*, receives user commands from the operator control unit (usually on a remote computer) through TCP and UDP sockets. Critical data such as emergency-stop and heartbeat signals are handled over the connection-oriented TCP link, while less critical higher-frequency commands, such as joystick values, are transmitted through the open-ended UDP link. These values are stored in the shared-memory data structure upon arrival.

All the software discussed thus far is written in C or an embedded microcontroller language. Transitioning the information collected by these various programs (which are functionally hardware drivers) to the high-level, graphical algorithm framework discussed in the previous section is handled by the LabVIEW block labeled "Shared Memory Interface." This interface reads the C shared-memory data-structure and makes that information available – for reading and writing – within the LabVIEW programming environment.

With the exception of the operator control unit (OCU) located on a remote computer, all the software interfaces discussed thus far reside on *Lapsus*. In *Trinity*'s architecture, the LabVIEW "Shared Memory Interface" is the critical component that links the low-bandwidth data and control software with the much higher-bandwidth vision components. As Figure 6 indicates, the "Subsumption-Based Navigation" block receives the processed images from the robot's vision systems and generates high-level navigation goals that it sends to the shared-memory interface.

Already implemented are the hardware interfaces to the various cameras, enabling video acquisition and display, as well as the core functionality for visual distance extraction from the stereo camera. Future tasks include developing the core features of *Trinity*'s high-level intelligence: the image-fusion and doorway-extraction algorithms for recognizing paths between rooms, the algorithms that will enable image-based localization, and the visible-IR image fusion. While the latter two have been experimentally verified in other systems [14,16], the doorway-extraction algorithm must be developed for *Trinity*.

Further down the development path is the integration of the acoustic eye and the fluoroscopes. The first will play a role in *Trinity*'s navigation analogous to one way our

hearing affects our interactions with the world: by redirecting attention to potentially interesting situations. The fluoroscopes on the other hand are currently planned to act independently of *Trinity*'s autonomy system because the frequency and nature of their data products do not readily lend themselves to dynamic navigation tasks.

As mentioned earlier, a future autonomous algorithm could attempt to locate potentially interesting sites to explore with the instruments. In this case, the data from the instruments would provide feedback on the veracity of the autonomous science-target identifications.

With the implementation of the full RSC system, some components of this software architecture, along with various aspects of the hardware implementation, will be reconfigured. Specifically, some image-processing functions will become a dedicated hardware module, and the majority of the low-level motor-control hardware/software system will transition to the RSC architecture.

RECONFIGURABLE COMPUTING

The availability of reconfigurable radiation-tolerant FPGAs for space applications creates an opportunity to explore and potentially exploit new processing paradigms for increased computing performance. Studies have shown that custom FPGA-based implementations, or soft-core processors, coupled with custom co-processors, can greatly improve the performance of some applications. Imaging applications have been shown to achieve speedups of eight to 800 times over an 800 MHz Pentium III processor [9]. Real-time feature extraction can run at up to 100 frames per second on reconfigurable logic [10]. Other embedded benchmarking programs have shown an average performance speedup of 5.8 when soft-core processors are combined with custom co-processors. Reduced power consumption (on average 57%) has also been demonstrated [11].

Capitalizing on the availability of radiation-tolerant FPGAs, soft-core processors (that run in the FPGA's fabric), operating systems, and networking protocols create an opportunity to develop a new class of space-avionics that is modular and reconfigurable. If properly designed to survive the space environment, such processors would find broad use in the space industry.

RSC Architecture

The high-level architecture of the RSC system is shown in Figure 7. The three primary architectural features of the system are 1) soft-core processors that can support an operating system, (2) custom logic (cores) that are user-definable, and (3) a scalable network.

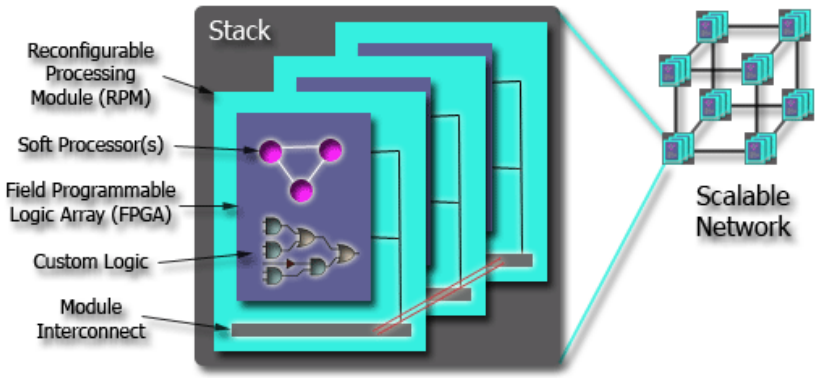


Figure 7 High-level RSC architecture

To implement this functionality, the RSC defines four basic modules that may be organized into processing stacks. The modules are:

1) *Reconfigurable Processor Module (RPM)*
   The RPM provides the reconfigurable resources for application developers. This module is where the soft-core processor(s) and custom cores for a specific application will reside.
2) *Command and Control Module (CCM)*
   The CCM provides the command interface for a RSC stack. It also manages the system boot process and system state-of-health reporting
3) *Network Module (NM)*
   The NM provides a mechanism to scale to multi-stack RSC systems. The routing and data-link layer functionality exists in the NM.
4) *Power Module (PM)*
   The PM down-converts the system input voltage to appropriate lower voltage levels.

The stack uses a PC/104-PLUS compatible form factor that has been enhanced for conduction cooling, launch-load survivability and radiation shielding. A typical stack is shown in Figure 8.

In addition to the hardware architecture, RSC supports a robust software environment. The soft-core processor(s) support the uCLinux operating system and the GNU toolchain for development support. The network architecture supports the Internet Protocol and high-level messaging via MPI when used in a multi-processor configuration. Traditional hardware description languages may be used for custom core development, as well as other commercially available high-level tools. More details on the RSC's architecture may be found in [12].

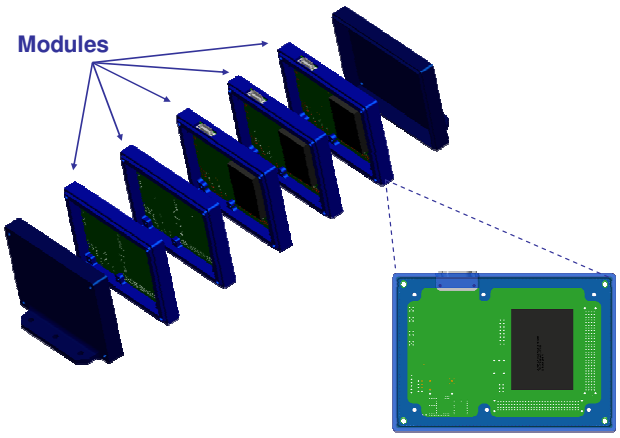


Figure 8 Exploded view of RSC stack

RSC for Robot Control and Data Processing

The computing architecture of the *Trinity* robot is depicted in Figure 5. The architecture's initial implementation was done with commercial-of-the-self (COTS) electronics. As the development of the RSC system matures however, several COTS subsystems will be replaced, demonstrating the viability of the space-qualified RSC system in challenging real-time applications. Specifically, the two areas of focus are the real-time motor control and the data-intensive image processing.

The motor-control hardware, described earlier, is implemented with Brainstem modules. The Brainstem modules each have a 40 MHz RISC processor with on-board memory (EEPROM) and IO support. Both analog and digital IO are supported, along with several standard interfaces, including RS-232 and I^2C. The three Brainstem Moto modules function as PD controllers for the robot's wheel drive motors. The Brainstem GP module's primary function is to interface with the PC/104-PLUS based *Lapsu*s computer, and route commands via the I^2C bus to the motor controllers.

For demonstrating real-time motor-control functionality with reconfigurable logic, the three Brainstem Moto modules will be replaced with RPMs: each RPM having a soft-core processor to implement the low-level PD control function. The RPMs will plug directly into the PC/104-PLUS stack, with *Lapsus* serving as the host controller for the bus. The routing function previously performed by the Brainstem GP module will be moved to *Lapsus*, and the PCI bus will communicate velocity set-point information to the RPMs.

The second RSC capability to be demonstrated is real-time data-intensive image processing. *Trinity*'s camera systems are tied into *Oculus*' PC/104-PLUS stack via a FireWire interface card. In the COTS implementation, all the image processing must be performed in software on *Oculus*. The addition of an RPM to this processing stack facilitates replacement of those functions already in software, or allows additional image-processing functions to be implemented in the RPM's reconfigurable logic. The FireWire card will transfer image data from the visible and IR cameras to the RPM's memory; image enhancement, registration, and fusion will then be performed using reconfigurable

logic. The resulting processed images will then be transferred to *Oculus* for relay back to the operator's console or undergo further processing for autonomous functionality.

As other modules of the RSC system are developed and tested, additional functions of *Trinity's* computing architecture can be replaced with reconfigurable logic. Candidate functions include stereo image processing, omni-cam imagery transformation, data compression, and command/control functions.

CONCLUSION

To date the result of this project is a unique omnidirectional robotic system poised for a large spectrum of interesting tasks. The unique hardware structure readily lends itself to indoor motion and navigation, and the diverse sensor suit provides requisite variety for exploring a large swath of navigation-based autonomous motion challenges. Additionally, the science instruments provide an interesting resource for future explorations in autonomous science-target identification/acquisition algorithms.

From the start, *Trinity*'s electrical-hardware architecture was designed for the addition of reconfigurable computing components, and the algorithmic goals of the project partition well for the useful integration of more advanced space-ready computing hardware. The algorithm frameworks being developed also support this transition in that they are designed modularly, with loose data interactions between mostly independent blocks, rather than a tightly integrated monolithic system. Additionally, the fast computers mounted directly on the robot allow local algorithm development and execution, thus greatly simplifying implementation efforts.

Key results learned through the course of our research include the benefits of hardware modularity, the benefits of partitioning robotic systems – both software and hardware – into easily separable modules, and the intuitive nature of omni-directional mobility for tele-operation and autonomous control.

The primary thrust of our future efforts includes the completion of the algorithm development frameworks, the implementation of the autonomous algorithms for sonar- and image-based navigation, image-histogram-based localization, and the integration of the full RSC system.

The initial hardware design of RSC's reconfigurable processing module is complete and printed circuit boards have been fabricated and are now being populated. Additionally, VHDL core development is underway to implement the required functionality for supporting soft-core processors with operating system support. Initial applications are planned for third quarter '06, with integration into *Trinity*'s computing system planned for fourth quarter '06.

ACKNOWLEDGMENT

This effort was funded in part by NASA's Office of Exploration. We would also like to thank Omnix Technology Systems, Inc. for the omnidirectional wheels and their encouragement in support of this development.

REFERENCES

1. A. Bradley, S. Miller, et al., "Mobius, An Omnidirectional Robot Utilizing Mecanum Wheels and Fuzzy Logic Control, *Proc. of the 28th Annual AAS Rocky Mountain Guidance & Control Conference*, Feb. 2005, pp. 251-266.
2. M. West and H. Asada, "Design of Ball Wheel Mechanisms for Omnidirectional Vehicles with Full Mobility and Invariant Kinematics," *Journal of Mechanical Design*, Vol. 119, 1997, pp. 153-161.
3. H. Yu, M. Spenko, and S. Dubowsky, "Omnidirectional Mobility Using Active Split Offset Castors," *Journal of Mechanical Design*, Vol. 126, No. 5, Sept. 2004, pp. 822-829.
4. J. Blumrich, "Omnidirectional Vehicle," U.S. Patent 3,789,947, 1974.
5. B. Ilon, "Wheels for a Course Stable Self-Propelling Vehicle Movable in Any Desired Direction on the Ground or Some Other Base," U.S. Patent 3,876,255, 1975.
6. J. Song and K. Byun, "Design and Control of Four-Wheeled Omnidirectional Mobile Robot with Steerable Omnidirectional Wheels," *Journal of Robotic Systems*, Vol. 21, No. 4, Wiley Interscience, Dec. 2004, pp. 193-208.
7. L. Ferriere, B. Raucent, and J. Samin, "Rollmobs, A New Omnimobile Robot," *Proc. of International Conference on Intelligent Robots and Systems (IROS)*, Vol. 2, Sept. 1997, pp. 913-918.
8. K. Buyn and J. Song, "Design and Construction of Continuous Alternate Wheels for an Omnidirectional Mobile Robot," *Journal of Robotic Systems*, Vol. 20, No. 9, Wiley Interscience, Aug. 2003.
9. B. Draper, R. Beveridge, W. Böhm, C. Ross, and M. Chawathe, "Accelerated Image Processing on FPGAs," *IEEE Transactions on Image Processing*, Vol. 12, No. 12, Dec. 2003, pp. 1543-1551
10. P. Giacon, S. Saggin, G. Tommasi, and M. Busti, "Implementing DSP Algorithms using Spartan-3 FPGAs," *Xcell Journal*, No. 53, 2005.
11. R. Lysecky, and F. Vahid, "A Study of the Speedups and Competitiveness of FPGA Soft Processor Cores Using Dynamic Hardware/Software Partitioning," *Design Automation and Test in Europe (DATE)*, March 2005.
12. R. Hodson, K. Somervill, J. Williams, N. Bergman, and R. Jones, "An Architecture for Reconfigurable Computing in Space," *8th Military & Aerospace Programmable Logic Device International Conference*, Washington, D.C., Sept. 2005.
13. Intel Open Source Computer Vision Library, Feb 1, 2006, Available: http://www.intel.com/technology/computing/opencv/index.htm
14. I. Ulrich, and I. Nourbakhsh, "Appearance-Based Place Recognition for Topological Localization," *Proceedings of IEEE International Conference on Robotics and Automation*, April 2000, pp. 1023-1029.
15. A. Bradley, S. Miller, and G. Creary, "Omnidirectional Robots for Off-Planet Applications," *Robomech 2005*, Tokyo, Japan, Dec. 2005.
16. L. Toa, H. Ngo, et al., "Multi-Sensor Image Fusion and Enhancement System for Assisting Drivers in Poor Lighting Conditions," *IEEE International Workshop on Applied Imagery and Pattern Recognition, AIPR – 2005*, Washington, D.C., Oct 2005.