Defence Research and Development Canada
Recherche et développement pour la défense Canada

Staged Experiments in Mobile Vehicle Autonomy II

Autonomous Land Systems Project Demonstration Results

S. Monckton, J. Collier, J. Giesbrecht, G. Broten, D. Mackay, D. Erickson, I. Vincent, and S. Verret
DRDC Suffield

Technical Report
DRDC Suffield TR 2006-243
December 2006

Canada

Staged Experiments in Mobile Vehicle Autonomy II

Autonomous Land Systems Project Demonstration Results.

S. Monckton, J. Collier, J. Giesbrecht, G. Broten, D. Mackay, D. Erickson, I. Vincent, and S. Verret
Defence R&D Canada – Suffield

Defence R&D Canada – Suffield

Technical Report

DRDC Suffield TR 2006-243

December 2006

Principal Author

Original signed by S. Monckton

S. Monckton

Approved by

Original signed by D. Hanna

D. Hanna

Head/Autonomous Intelligent Systems Section

Approved for release by

Original signed by Dr Paul A. D'Agostino

Dr. Paul A. D'Agostino

Chairman/Document Review Panel

Abstract

Defence R&D Canada (DRDC) is extending its considerable experience in teleoperated air, land and seaborne systems to the development of autonomous systems for the Canadian Forces. The first project, Autonomous Land Systems (ALS), sought to establish personnel and technical foundations through the demonstration of basic autonomous multivehicle land capabilities. The outcome of a 2 year development effort, this paper summarizes the properties and methods of a basic multivehicle control system founded upon a plausible military software architecture.

Résumé

R & D pour la défense Canada (RDDC) a étendu sa considérable expérience concernant les systèmes télé-opérés aériens, terrestres et maritimes à la mise au point de systèmes autonomes, au service des Forces canadiennes. Le premier projet, les Systèmes terrestres autonomes (STA) visait à établir les fondements dans le domaine de la technique et du personnel, en démontrant les capacités terrestres fondamentales multi-véhicules autonomes. Cet article résume le résultat dun effort de mise au point qui dure depuis 2 ans, concernant les propriétés et les méthodes dun système de contrle multi-véhicule de base fondé sur une architecture de logiciels militaire plausible.

This page intentionally left blank.

Executive summary

Staged Experiments in Mobile Vehicle Autonomy II

S. Monckton, J. Collier, J. Giesbrecht, G. Broten, D. MacKay, D. Erickson, I. Vincent, S. Verret; DRDC Suffield TR 2006-243; Defence R&D Canada – Suffield; December 2006.

Background: The Autonomous Land Systems Project, spanning the years 1999 to 2005, was designed to demonstrate limited multivehicle autonomy in a 'low complexity' environment. With the surge of new staff over the 2002-2003 time period, the project would come to mix capability demonstration with training and teambuilding.

The ALS project attacked difficult challenges in sensing, modeling, and control faced by unmanned ground vehicles. Unlike the Segway RMP *Spiral 1* trial [11], the ALS *Spiral 2* trial converted an ATV class ackerman-steered vehicle to networked, multivehicle autonomous operation over the Suffield prairie. By project close, *these vehicles could autonomously follow waypoints, avoid obstacles, and adopt leader or follower roles.* This report provides an integrated overview of the resulting system.

Principle Results: The project created an important software infrastructure founded on industrial techniques and common standards where possible to exploit published, testable methods. Approximately 50000 lines of code were written including: networked laser, stereo, GPS, and IMU sensor drivers; advanced mapping and control software; and logging, visualization, and analysis tools. Each vehicle sensed its surroundings in three dimensions and built maps to recognize and avoid obstacles while navigating to waypoints.

Perhaps most significantly, the project generated an experienced core group of scientists, each capable of developing networked, multirobot systems. Though five of the group's eight core developers had postgraduate training and two of these had professional robotics experience, none had experience with such complex systems. By project close, each had contributed major elements to the project, understood its operation, and could install, run and add to the system.

Significance of Results: This project generated crucial software infrastructure, intellectual property (captured in software), and staff skills required to develop unmanned autonomous systems. Leveraging inter-platform communication infrastructure, and controller extensibility, DRDC is now positioned to explore the best mix of manned and unmanned solutions across platform types – a key capability at the early stages of unmanned system deployment and operational concept development.

Future Plans: Immediate follow on plans include technology transfer to other DRDC laboratories, expansion to include fixed and rotor wing aircraft, Shape Shifting Tracked Vehicle, and the Microhydraulic Toolkit (MHT) platforms. This software infrastructure provides a flexible 'container' for existing and new software, including new modeling, trafficability and target following software.

Sommaire

Staged Experiments in Mobile Vehicle Autonomy II

S. Monckton, J. Collier, J. Giesbrecht, G. Broten, D. MacKay, D. Erickson, I. Vincent, S. Verret; DRDC Suffield TR 2006-243; Défense Recherche et Développment Canada - Suffield; décembre 2006.

Contexte : Le Projet des systèmes terrestres autonomes a été conçu, durant la période allant de 1999 à 2005, pour démontrer l'autonomie limitée de la technologie multi-véhicule dans un milieu peu complexe . Avec un apport de personnel nouveau, de 2002 à 2003, les capacités de démonstration avec celle de formation et de promotion de travail d'équipe se sont naturellement combinées.

Le projet STA a relevé les défis difficiles concernant la détection, la modélisation et le contrôle que présentent les véhicules terrestres. Contrairement à l'essai sur le Spiral 1 Segway RMP [11], l'essai sur le STA Spiral 2 a converti un véhicule VTT de classe tractée par le système ackerman en une opération réseautée et autonome multi-véhicule, dans la prairie de Suffield. Le projet terminé, ces véhicules étaient en mesure de poursuivre des points de cheminement, d'éviter des obstacles et d'adopter des rôles de leader ou de suiveur, ceci de manière autonome. Ce rapport fournit un aperçu global du système résultant.

Résultats principaux : Le projet a créé une infrastructure de logiciels importante fondée sur les techniques industrielles et les normes générales là o il a été possible d'exploiter les méthodes publiées et testables. 50 000 lignes de programmation ont été écrites environ dont le laser, la stéréo, le GPS et les pilotes de capteurs UMI, tous réseautés ; les logiciels de pointe de cartographie et de contrôle ainsi que les outils d'enregistrement chronologique de données, de visualisation et d'analyse. Chaque véhicule captait son environnement en trois dimensions et construisait des cartes pour reconnaître et éviter des obstacles tout en navigant les points de cheminement.

Le plus important semble tre que le projet a généré un groupe cadre de scientifiques expérimentés, chacun capable de développer des systèmes réseautés multi-robots. Parmi les huit développeurs cadres, cinq d'entre eux possédaient une formation de niveau supérieur et deux d'entre eux possédaient une expérience de travail en robotique mais aucun d'entre eux n'avait d'expérience avec des systèmes d'une telle complexité. la fin du projet, chacun avait contribué des éléments majeurs au projet, compris son opération et était capable de l'installer, de l'opérer et d'effectuer des ajouts.

Portée des résultats : Ce projet a généré une infrastructure capitale de logiciels, de propriété intellectuelle (capturée par les logiciels) et d'habiletés requises par le personnel pour développer des systèmes autonomes sans pilote. L'optimisation de l'infrastructure de communication entre les plates-formes et l'extensibilité des contrôleurs font que RDDC est maintenant en mesure d'explorer la meilleure sélection de solutions avec et sans pilote, pour tous les types de plates-formes une capacité clé, à ce stage précoce du déploiement de système sans pilote et de mise au point de concept opérationnel.

Plans futurs : Le suivi immédiat de ces plans inclut le transfert de technologies à d'autres laboratoires RDDC ; l'expansion visant à inclure la voilure fixe et rotor d'aéronefs, le véhicule chenillé de forme changeante et les plates-formes avec boîte à outils microhydraulique (MHT). Cette infrastructure de logiciels fournit un cadre flexible pour les logiciels existants nouveaux dont les nouveaux logiciels de modélisation, de traficabilité et de poursuite de cibles.

This page intentionally left blank.

Table of contents

List of figures

List of tables

1 Introduction

With a long history of teleoperation research, Defence R&D Canada (DRDC) Autonomous Intelligent Systems Section (AISS) has embarked on autonomous systems development projects for the Canadian Forces. The first project, Autonomous Land Systems (ALS), sought to demonstrate basic autonomous multivehicle capabilities and establish the personnel base and technical foundations for future projects. The outcome of a 2 year development effort, this paper summarizes the properties and methods of a basic multivehicle control system founded upon a plausible military software architecture.

Through a review [7] of communications "middleware" DRDC concluded that the ACE-TAO-MIRO toolchain [29], with powerful CORBA interprocess communication, provided the most open, proven, real-time, and portable code base for a military research program. Consequently, AISS can now explore "Army of the Future" concepts such as net-enabled warfare and effects-based joint operations using efficient publish-subscribe and client-server networking paradigms.

Scale, modularity, distribution, platform independence, and flexibility govern military applications of autonomous unmanned vehicle control. The software environment must scale to tolerate different team sizes and must be modular to support variation in payloads. It must accommodate distributed computing capabilities through networking and must separate hardware from software to achieve platform independence. Finally, the software must not limit engineers to any single method of vehicle control.

Figure 1 reveals the final software structure of the ALS Project vehicle, loosely depicting the flow of information through the system. This hybrid controller mixes traditional model based methods with reactive elements combined within an arbitrator. Significantly, each module can run in any location on the network and each data flow publishes to any subscriber, local or remote. The following sections describe Figure 1 in detail, reviewing DRDC's vehicle and sensor suite, the vehicles' world representation, control, and multivehicle services. Each section also provides details about the associated CORBA interfaces that enable data exchanges between independent processes. Finally, the paper discusses DRDC's experience and field results with this architecture.

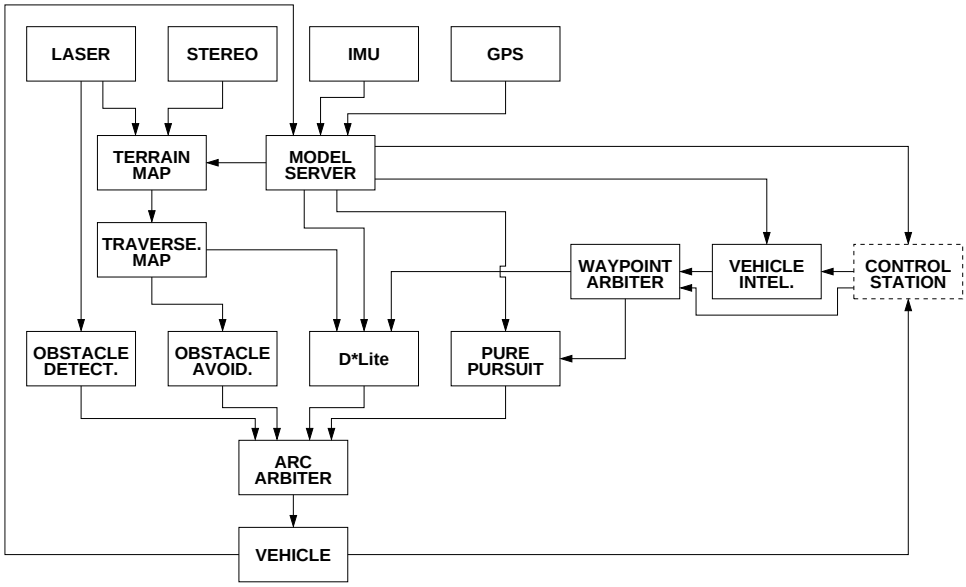

Figure 1: *ALS architecture flow diagram. Each box represents a CORBA service that can reside anywhere on a network and each connector represents data flows accessible to any subscriber service.*

Figure 2: *One of two modified Koyker Raptors used in DRDC's ALS Project. Each Raptor used one or more roof mounted SICK lasers and stereo cameras; Differential GPS and IMU; and wireless mesh networking routers.*

2 Hardware

2.1 Platform

DRDC selected the Koyker Raptor as its Unmanned Ground Vehicle (UGV) demonstration platform, based on payload and drivetrain requirements. In each, a 25Hp gasoline engine powered a 4x4 hydrostatic drivetrain while generating an additional 1.5 kW of on-board power. The vehicles on-board intelligence enclosure housed power inverters, quad and dual Pentium servers, ethernet and USB hubs. The SpeedLan mesh networking router implements an 802.11b class wireless communications network. XJ Design of Ottawa, Canada converted the vehicles to drive-by-wire using an MPC555 microcontroller.

2.2 Proprioceptive Sensing

The sensing system collected raw position and orientation data from a GPS, an IMU, and odometry. The Sokkia GSR2600 GPS, combined with a Pacific Crest PDL RVR radio, supplied differentially corrected GPS positions with an accuracy of 2-5cm at an update rate of 4 Hz. Equipped with two Honeywell 1GT101DC hall effect sensors, on-board software decoded quadrature pulse trains into the displacement and direction of each front wheel, while a frequency-to-voltage divider transformed this signal into wheel speed. Using magnetometers, gyros, and accelerometers, the Microstrain 3DM-GX1 produced orientation and angular rates with respect to gravity and magnetic north.

2.3 Exteroceptive Sensing

2.3.0.1 Laser Range Finders

To overcome the bulk and expense of traditional 3-D LRFs, investigators [1, 14] turned to inexpensive, light weight 2-D LRFs such as the SICK and Acuity lasers.

DRDC and Scientific Instrumentation Ltd. developed a nodding mechanism for a 2-D SICK, creating a system that returns 3-D data. Communicating through an ethernet interface, the embedded RTEMS controller nods the laser from 2-90 degrees/sec with 0.072 degree resolution and 4cm accuracy over 1 - 30 metres [4].

2.3.0.2 Stereo Vision

DRDC adopted Point Grey's Digiclops system to provide high speed range image streams. The Digiclops develops a disparity map between three camera image streams, publishing the resulting 3D range image stream over an IEEE-1394 Firewire digital connection.

2.4 Miro Services

DRDC developed MIRO-based services that allowed all hardware interfaces to publish and subscribe to CORBA events, as well as to respond to polling requests. The Raptor vehicle published *odometry* and *vehicle status data* and received vehicle control commands such as

the forward velocity and the steering angle. The proprioceptive sensing published *GPS* and *Imu*, while the exteroceptive sensing published *Stereo* and *Laser* data as generic 3D range data events [6].

3 World Representation

To navigate purposefully through outdoor environments, a mobile robot must represent the world in order to detect and avoid obstacles, and plan paths. Building an adequate world representation requires high precision extero- and proprioceptive sensors distributed over known vehicle geometry relative to an accurate vehicle world position. Therefore, the following section details a 2.5D world representation system for long range multivehicle navigation.

3.1 Vehicle Model and Pose Estimation

Large vehicle navigation services rely on widely distributed components, particularly sensors, positioned both in local and global coordinates. To establish these positions, DRDC combined an internal geometry database and vehicle pose estimation into a Model Server that publishes both fixed local geometric transforms and global vehicle position in UTM coordinates.

3.1.1 ModelServer: Vehicle Geometry

To avoid repeated whole-vehicle surveys, DRDC conducted modular "component" surveys that identify important frames (e.g. mounting points) in local component coordinates. In effect, DRDC simplified model maintenance by exchanging vehicle surveys for component surveys and on-line computation.

DRDC adopted dynamic modeling conventions [27] within a geometry database that manages three data types: a *Body*, $\mathcal{B}$, a *BodyFrame*, $\mathcal{F}$, and a *Constraint*, $\mathcal{C}$, accessible through a *BodyList*, $\mathbf{L_B} = [\mathcal{B}_1, \mathcal{B}_i, ..., \mathcal{B}_m]$ a *ConstraintList*, $\mathbf{L_C} = [\mathcal{C}_1, \mathcal{C}_i, ..., \mathcal{C}_q]$, and a directed graph *Model*, $\mathcal{M}$.

A *Body* contains a unique string name identifier, s_b, and a list of *BodyFrames*, $\mathbf{L}_{i\mathcal{F}} = [\mathcal{F}_1, \mathcal{F}_j, ..., \mathcal{F}_n]$ or for the ith body: $\mathcal{B}_i = \langle s_b, \mathbf{L}_{i\mathcal{F}} \rangle$.

A *BodyFrame* contains a unique string identifier, s_f, a homogeneous transform from the ith body's origin to the jth frame, $\mathbf{A}_{ij}$, and pointers to the parent Body, $\mathcal{B}_i$, and to a constraint, $\mathcal{C}_{ij}$. For the jth bodyframe: $\mathcal{B}_i : \mathcal{F}_j = \langle s_f, \mathcal{B}_i, \mathcal{C}_{ij}, \mathbf{A}_{ij} \rangle$.

Constraints bind distinct body-bodyframe pairs through a (currently) time invariant homogeneous transform, $\mathbf{T}_k$. For the kth constraint:$\mathcal{C}_k = \langle s_c, \mathcal{F}_{\mathbf{fr}}, \mathcal{F}_{\mathbf{to}}, \mathbf{T}_k \rangle : \mathcal{F}_{\mathbf{fr}} \neq \mathcal{F}_{\mathbf{to}}$. To capture the transformation 'direction', a *Constraint* contains pointers to *From* and *To* bodyframes, $\mathcal{F}_{\mathbf{fr}}$ and $\mathcal{F}_{\mathbf{to}}$ respectively.

The *Model* $\mathcal{M}$, $\mathbf{L}_{\mathcal{B}}$ constrained by $\mathbf{L}_{\mathcal{C}}$, resembles a cyclic directed graph of $\mathcal{B}_i$:$\mathcal{F}_j$ vertices with $\mathbf{A}_{ij}$ and $\mathbf{T}_k$ transformation connectors. The following simple rules govern the construction of $\mathcal{M}$:

1. There must be $m \geq 2$ bodies.

Figure 3: *A hierarchical rigid body model.*

2. A Body must have $n \geq 1$ bodyframes.
3. $\forall \mathcal{C}_k \in \mathbf{L}_\mathcal{C}$, $\mathcal{F}_{\mathbf{fr}}$ and $\mathcal{F}_{\mathbf{to}}$ must exist.
4. $\forall \mathcal{B}_i{:}\mathcal{F}_j$, $\mathcal{F}_j$ if constrained, may have only one constraint which must exist.

Not surprisingly, such trees can become quite complex as depicted in figure 3.

3.1.2 ModelServer: Operation

ModelServer loads a model XML file, culling bad bodyframe and constraint references (rules 3 and 4). Once loaded, the Model accepts client queries through a CORBA IDL interface to retrieve $\mathbf{L_B}$, $\mathbf{L}_{i\mathcal{F}}$ or a relative transformation, ${}^{\mathcal{F}_R}T_{\mathcal{F}_I}$, between a *frame of reference*, $\mathcal{B}_R{:}\mathcal{F}_R$, and a target *frame of interest* $\mathcal{B}_I{:}\mathcal{F}_I$.

A simple depth-first search recursively constructs a transformation between reference and target vertices. Starting at the $\mathcal{B}_R{:}\mathcal{F}_R$ vertex, the search compares $\mathcal{B}_i{:}\mathcal{F}_j$ in $\mathbf{L}_{i\mathcal{F}}$ against the $\mathcal{B}_I{:}\mathcal{F}_I$. Finding no match, the search will examine the next frame unless the frame is constrained. In this case it pushes $\mathcal{B}_i{:}\mathcal{F}_j$ onto a *Path* stack, "crosses" the constraint, and examines the attached frame. If a $\mathcal{B}_i{:}\mathcal{F}_j$ matches any in *Path*, a cycle exists and the search examines the next branch, popping *Path* as necessary. The search continues recursively until $\mathcal{B}_I{:}\mathcal{F}_I$ is found. The search then unwinds the recursion, building the transform product, ${}^{\mathcal{F}_R}T_{\mathcal{F}_I}$, according to the directed graph.

To the client, ModelServer provides the location of any $\mathcal{B}_i{:}\mathcal{F}_j$ vertex in the coordinates of a reference bodyframe. This construction, a simplified version of those found in CAD and CAE systems, reduces the impact of sensor position changes and provides relative geometry on demand to dependent services. Typical results could be viewed through the qtBodyViewer utility as shown in figure 4.

Figure 4: *BodyViewer representation of the Raptor Vehicle in a pose similar to figure 2 .*

More detail on ModelServer and the body geometry file format can be found in [23].

3.1.3 Vehicle Pose

A position 3-vector and a quaternion 4-vector represent the global and local pose respectively. The global pose is thus a 7-vector : $\mathbf{p}_W \equiv [\varphi_W, \lambda_W, z_W, q_s, \mathbf{q}]^T$ where W represents the global frame of reference, φ_W is the longitude displacement from the Prime Meridian in *radians*, λ_W represents the latitude displacement from the Equator in *radians,* and z_W represents the translation above/below Median Sea Level (MSL) in *metres*. The quaternion q defined by $(q_s, \mathbf{q}) \equiv [q_s, q_x, q_y, q_z]^T$is dimensionless. UTM was provided in addition to latitude and longitude, represented as x_W and y_W, measured in *metres* and substituted for φ_W and λ_W respectively. The quaternion$= [1.0, 0.0, 0.0, 0.0]^T$ defines the absolute orientation origin, representing True North about the Z axis, while having zero pitch and roll about the X and Y-axes respectively. Similarly, *local* pose is $\mathbf{p}_\ell \equiv [x_\ell, y_\ell, z_\ell, q_s, \mathbf{q}]^T$, where ℓ identifies the local ego-centric frame fixed to the Raptor's front axle. The local quaternion is identical in orientation to the absolute global orientation vector.

3.1.3.1 Kinematic Model

DRDC's localization service estimates state change through a simple "space craft" kinematic model composed of a rigid body undergoing uniform accelerations in all three axes and ignoring the vehicle's dynamic constraints. Simplifying estimation, this vehicle independent approach recognizes that while constrained to nonholonomic motion, the Raptor will also

experience wheel slip, rough ground transitions, pitch and roll vibrations in three dimensions due to the terrain. This model applies velocities in the local fixed frame and transforms them into state change in the global frame of reference. The velocity vector in the egocentric local frame of reference $\mathbf{v}_\ell \equiv [v_x, v_y, v_z, \omega_x, \omega_y, \omega_z]^T$where v_x , v_y and v_z are the translational velocities along the axes of ℓ, and ω_x ,ω_y ,ω_z are the rotational velocities about the axes of ℓ.

The transformation matrix $\mathbf{J}(\mathbf{q})$ relates the frame of reference ℓ to the global frame of reference W using the current filtered quaternion state. Given $\xi = [\dot{x}_W, \dot{y}_W, \dot{z}_W, \dot{q}_s, \dot{q}_x, \dot{q}_y, \dot{q}_z]^T$ then assuming uniform acceleration.:

$$\dot{\xi} = \mathbf{J}(q)\mathbf{v} = \begin{bmatrix} \mathbf{R}(\mathbf{q}) & \mathbf{0}_{3x3} \\ \mathbf{0_{4x3}} & \frac{1}{2}\mathbf{U}(\mathbf{q}) \end{bmatrix} \quad (1)$$

$$\mathbf{p}_W(t_{n+1}) = \mathbf{p}_W(t_n) + \Delta t\dot{\xi} \quad (2)$$

where $\mathbf{R}(\mathbf{q}) \in \mathrm{SO}(3)$ and $\mathbf{U}(\mathbf{q}) \in \mathbb{R}^{4x3}$ and are defined as:

$$\mathbf{R}(\mathbf{q}) = \mathbf{I_{3x3}} + \mathbf{2q}_s\mathbf{S}(q) + 2\mathbf{S}(\mathbf{q})^2 \quad (3)$$

$$\mathbf{U}(\mathbf{q}) = \begin{bmatrix} -\mathbf{q^T} \\ q_s\mathbf{I_{3x3}} + \mathbf{S}(\mathbf{q}) \end{bmatrix} = \begin{bmatrix} -\mathbf{q^T} \\ \mathbf{T}(\mathbf{q}) \end{bmatrix} \quad (4)$$

$\mathbf{S}(\mathbf{q})$ is the skew-symmetric matrix operator on **q.** The Jacobian represents a deterministic state space model for the behaviour of the vehicle independent of the sensed stochastic variables.

3.1.3.2 Kalman Filter Formulation

Three direct state Kalman filter observers filtered input from the GPS, IMU, and odometry sensors. A discrete-time, recursive *a priori* formulation [21] with state vector augmentation was selected to improve the factory filtering internal to the individual sensors. Data for each sensor was filtered individually and fused together to arrive at the estimate velocities and state at each instant. The fused estimate of velocity at each timestep was fed back to individual Kalman filters.

Process and system noise estimates were derived from experiments with additional tuning during system integration.

To update the state vector, the localizer timestamped and filtered incoming data, fused absolute and current pose estimates, integrated accelerations and fused velocities to estimate current velocity, transformed the local frame velocities into a global state change, and generated the absolute pose. This algorithm built pose estimates in a single step without a relative intermediate pose unlike dead-reckoning absolute (DRABS).

3.1.3.3 Sensor Fusion Techniques

The localizer adjusts the estimated state vector and fused velocity vector when data from sensors arrives and at a fixed timestep to achieve a update rate of at least 40Hz. Filtered rel-

ative orientation data was converted to quaternions prior to linear blending fusion. Filtered translational and rotational velocities were fused directly using weights.

3.1.4 Miro Services

As show in Figure 1 the ModelServer consumes *GPS*, *IMU* and *odometry* events and publishes *pose* events to numerous subscribers.

3.2 Terrain Maps

The data received from the exteroceptive sensors, such as the SICK laser and Digiclops stereo camera, is fused into a grid map, formed by rectangular array of regions, and this map servers as the world representation for the ALS project. The simplest, the *occupancy grid*, represents regions as occupied, empty, or unknown [3, 10, 32], suitable for flat indoor environments. In unstructured outdoor environments a *terrain map* [2, 13, 15, 19, 20], captures the terrain elevation, $\bar{z}$, and the elevation variance, σ^2 of each grid in the map. Figure 5 illustrates the structure of DRDC's scrolling ego-centric *Terrain Map* component. For a grid index (i, j), the element, $\mathcal{T}_{ij}$, has the familiar structure:

$$\mathcal{T}_{ij} \equiv \left\langle \bar{z}_{ij}, \sigma^2_{ij} \right\rangle \tag{5}$$

3.2.1 Terrain Map Size

Sensor characteristics limit terrain map size. Assume a vehicle has a set of range sensors, $S = [S_1 \dots S_n]$, each with a maximum range S_R, and a horizontal field of view S_α. The maximum size of the terrain map can be determined using:

$$X_{max} = \max(S_{R_i}), \quad Y_{max} = \max(S_{R_i} \sin(\frac{S_{\alpha_i}}{2})) \tag{6}$$

assuming all sensors scan to the horizon. Figure 5 shows a terrain map, with a grid size G_l, yields a map with dimensions of

$$N_x = \frac{X_{max}}{G_l}, \quad N_y = \frac{Y_{max}}{G_l} \tag{7}$$

where N_x and N_y are the number of grid elements along the X_{max} and Y_{max} axes respectively.

3.2.2 Wrappable Map Representation

The terrain map is *ego-centric*, fixing the map in the vehicle local coordinate system. Using the fixed dimensions of $X_{max} \times Y_{max}$, the map translates and rotates the observed world using actual vehicle motion. For this map the $\mathbf{x}$ coordinate is always parallel to the vehicle's direction of movement. Thus, in the $\mathbf{x}$ direction of motion, the map recycles departing

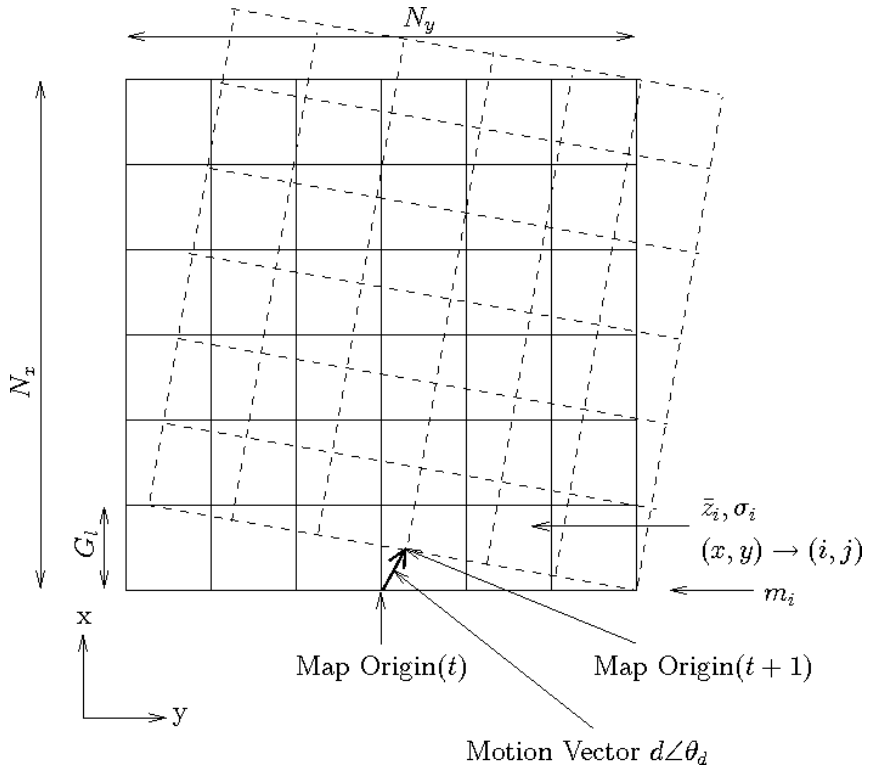

Figure 5: *The structure of DRDC's Scrolling Terrain Map.*

elements on map boundaries into new elements on entering across the opposite boundary, "wrapping the map" as the vehicle moves. The variable m_i, shown in Figure 5, tracks the current index into the wrappable map.

For rotational motion the map is mathematically transformed by the magnitude of the vehicle's yaw angle and the map data is shuffled into its corresponding grid location.

3.2.3 Indexing the Map

To improve map accuracy, the map module fuses new range data along with previous data into the map's grid elements.

The service first transforms a scan point's (x, y) position in vehicle local co-ordinates to an map grid element index, (i, j) using:

$$\hat{i} = \text{int}\left(\frac{\text{mod}(\frac{x}{x_{max}})}{G_l}\right) + m_i \tag{8}$$

$$i = \begin{cases} \hat{i} & \text{if } \hat{i} < N_x \\ \hat{i} - N_x & \text{if } \hat{i} \geq N_x \end{cases} \tag{9}$$

$$j = \text{int}\left(\frac{\text{mod}(\frac{y}{Y_{max}})}{G_l}\right) \tag{10}$$

$$m_i = \begin{cases} m_{i-1} + 1 & \text{if } x(t) - x(t-1) > G_l \\ m_{i-1} & \text{otherwise} \end{cases} \tag{11}$$

where $m_i =$ is the grid map index in the X direction.Then using the index pair (i, j), the map's elevation and variance, σ^2, are updated using the technique given in section 3.2.5

3.2.4 Coordinate Transformation due to Rotation

As shown in Figure 5, the vehicle yaw, θ, is about the **z**-axis. When the total yaw, $\sum_{i=1}^{n} \theta_i$, exceeds the maximum yaw angle, θ_d, a map yaw transformation occurs. The first step of the transformation uses the map indices (i, j) to determine the grid element's position coordinate.

$$x_{el}(t) = i_{el}(t) \cdot G_l + \frac{G_l}{2} + x_{re}(t-1) \tag{12}$$

$$y_{el}(t) = j_{el}(t) \cdot G_l + \frac{G_l}{2} + y_{re}(t-1) \tag{13}$$

The remainders from the previous rotation, given by $x_{re}(t-1)$ and $y_{re}(t-1)$, negate the quantization effects resulting from binning the (x, y) location into the grid.

The next step determines the vector, $\mathbf{R_{el}}$, from the map origin to the grid element position.

$$R_{el}(t) = \sqrt{x_{el}(t)^2 + y_{el}(t)^2} \;\; \phi_{el}(t) = \tan^{-1}(\frac{y_{el}(t)}{x_{el}(t)}) \tag{14}$$

Rotating the vector, $\mathbf{R_{el}}$, by the yaw angle, θ_d, yields the new, (x_{new}, y_{new}), grid element position, as well as the remainders.

$$\begin{aligned} x_{new}(t) &= R_{el}(t)\cos(\phi_{el}(t)+\theta_d) && (15)\\ y_{new}(t) &= R_{el}(t)\sin(\phi_{el}(t)+\theta_d) && (16)\\ x_{re}(t) &= \mathrm{mod}(x_{new}(t), G_l) && (17)\\ y_{re}(t) &= \mathrm{mod}(y_{new}(t), G_l) && (18) \end{aligned}$$

As described in Section 3.2.3, the $(x_{new}(t), y_{new}(t))$ coordinate pair converts into a map index pair, (i_{new}, j_{new}), which becomes the holder for the map data previously indexed by the map pair (i_{el}, j_{el}).

3.2.5 Fusion of Data

The map module optimally fuses transformed raw range data with existing terrain map data using the estimated 3-D orientation error of the scan point[1]. The map uses computationally efficient forms of the variance weighted mean [2] that avoids storing all data points:

$$\overline{\tau}_{i+1} = \frac{\sum_{i=1}^{N} \frac{\tau_i}{\sigma_i^2} + \frac{\tau_{N+1}}{\sigma_{N+1}^2}}{\sum_{i=1}^{N} \frac{1}{\sigma_i^2} + \frac{1}{\sigma_{N+1}^2}} \tag{19}$$

and variance weighted variance:

$$\sigma^2 = \frac{\sum_{i=1}^{N} \frac{1}{\sigma_i^2} \sum_{i=1}^{N} \frac{\tau_i^2}{\sigma_i^2} - \left(\sum_{i=1}^{N} \frac{\tau_i}{\sigma_i^2}\right)^2}{\left(\sum_{i=1}^{N} \frac{1}{\sigma_i^2}\right)^2} \tag{20}$$

to combine multiple data sets in real-time.

3.2.6 Miro Services

The terrain map consumes *Laser*, *Stereo* and *Pose* events and produces *Terrain Map* events on 500 ms intervals. Interested readers should consult [5] for more information.

3.3 Traversability Mapping

DRDC's traversability analysis maps a fine resolution terrain map, $\mathcal{T}$, to a coarser traversability map, $\mathcal{V}$. Similar to [26] and [13], the traversability map is composed of a grid where each element contains measures of *Traversability*, the ease with which a UGV can navigate the given cell, and *Goodness*, the accuracy of the data used to produce traversability. So for an element $\mathcal{V}_{pq}$:

$$\mathcal{V}_{pq} \equiv \left\langle \mathbf{V}_{pq}, \sigma_{pq}^2 \right\rangle \tag{21}$$

[1] The orientation error can be interpreted as the variance associated with the data point.

[2] Variance weighted statistics to optimally combine data with different variances - the underlying premise of Kalman Filtering.

This information can then be used by path planning algorithms to determine an optimal route through the map.

The traversability map is *ego-centric* and non-cumulative. Each traversability cell is constructed from a subset of terrain cells $\mathcal{S}_{pq}$ whose Cartesian coordinates lie within the bounding box of cell $\mathcal{V}_{pq}$:

$$\mathcal{T}_{ij}(x_t, y_t) \in S_{pq} : \begin{cases} x_v - 0.5w \leq x_t \leq x_v + 0.5w \\ y_v - 0.5d \leq y_t \leq y_v + 0.5d \end{cases} \tag{22}$$

where (x_v, y_v), (x_t, y_t) are the centroids of $\mathcal{V}_{pq}$ and $\mathcal{T}_{ij}$ respectively, w is the width of $\mathcal{V}_{pq}$ and d is the depth of $\mathcal{V}_{pq}$.

The subset $\mathcal{S}_{pq}$ calculates the traversability measure, $\mathbf{V}_{pq}$ ranging from 0, or fully traversable, to 1, indicating non-traversable terrain. In the event that $\mathcal{S}_{pq}$ has less than n_{valid} members containing valid terrain data, the traversability is marked as unknown, otherwise the following calculations are used to determine $\mathbf{V}_{pq}$:

3.3.0.1 Calculate Step Hazard

The step hazard, $\mathbf{H}_{pq}$, is defined as a change in elevation which the UGV cannot safely navigate and is calculated as follows:

$$\mathbf{H}_{pq} = \begin{cases} 0 & : & s_{max} < 0.5h_{obst} \\ \frac{s_{max}}{h_{obst}} & : & 0.5h_{obst} \leq s_{max} < h_{obst} \\ +1 & : & s_{max} \geq h_{obst} \end{cases} \tag{23}$$

Where s_{max} is the maximum elevation difference between any of the members of $\mathcal{S}_{pq}$ which contain valid data, and h_{obst} is the minimum step height the UGV can safely traverse.

3.3.0.2 Calculate Slope Hazard

The first step in determining a slope hazard, $\mathbf{D}_{pq}$, fits a plane to the elements of $\mathcal{S}_{pq}$. If the plane's pitch p_{pq} or roll r_{pq} relative to vehicles local frame is greater than the pitch or roll thresholds, p_{max} and r_{max} respectively, than slope hazard, $\mathbf{D}_{pq}$, assumes a value of 1, otherwise $\mathbf{D}_{pq} = 0$,.

$$\mathbf{D}_{pq} = \begin{cases} 1 & : & p_{pq} > p_{max} \text{ or } r_{pq} > r_{max} \\ 0 & : & \text{otherwise} \end{cases} \tag{24}$$

3.3.0.3 Calculate Traversability

The traversability of a grid cell $\mathbf{H}_{ij}$ is determined as follows:

$$\mathbf{V}_{pq} = \begin{cases} 1 & : & \mathbf{H}_{pq} = 1 \text{ or } \mathbf{D}_{pq} = 1 \\ \mathbf{H}_{pq} & : & \text{otherwise} \end{cases} \tag{25}$$

The Goodness of cell, $\mathcal{V}_{pq}$, is calculated as the sum of variances of elevation data σ_z^2 in $\mathcal{S}_{pq}$.

The traversability map is separated into two different regions, a near zone $\mathcal{Z}_1$ and a far zone $\mathcal{Z}_2$. $\mathcal{Z}_1$ includes any traversability cells which lie within a radius r_1 from the UGV and $\mathcal{Z}_2$ is the remainder. $\mathcal{Z}_1$ typically contains large amounts of valid terrain data and is of greater accuracy than $\mathcal{Z}_2$. To reduce false obstacles caused by inconsistencies in the terrain map, $h_{\mathcal{Z}_2 obst} = 2h_{\mathcal{Z}_1 obst}$.

Figure 9 shows a typical traversability map generated by the Raptor UGV. Traversable and non-traversable regions are represented by red and blue respectively, while white cells represent unknowns regions.

As shown in Figure 1, the traversability map consumes *Terrain Map* events and produces *Traverse Map* events. Further detail can be found in [8].

4 Vehicle Control

The modular vehicle controller exploits the AFA architecture to run several behaviour components such as Goal Seeking, Obstacle Detection, Obstacle Avoidance, and Path Planning, independently on one or more CPUs, distributed across a TCP/IP network to achieve varying levels of autonomy in a network transparent manner. The Obstacle Detection module examines raw *Laser* events to ensure safe teleoperation or autonomous goal seeking by ensuring obstacles are avoided enroute. Obstacle Avoidance and Path Planning modules permit greater autonomy, but require terrain mapping and accurate vehicle localization. Similar to DAMN [25], the Arc Arbiter module combines the output of each of these behaviours into steering angle and velocity commands.

4.1 Control Structure

A standardized CORBA IDL interface allows each behaviour to contribute to vehicle control through the Arc Arbiter. The interface passes a vote for each of a set of candidate steering angles, α_i distributed evenly between the maximum left and right steering angles shown in Figure 9. The vote for each of the candidate arcs is $\mathbf{v}_i \equiv \langle d, p, q, s_{max} \rangle$. Where $d \in [0, 1]$ is the desirability of a steering arc, $p \in [0, 1]$ is the certainty of this decision, $q = 0 \mid 1$ is a veto, allowing any behaviour to disallow a steering command, and finally s_{max} is the maximum allowable speed the vehicle may travel along this arc. With this structure, DRDC has considerable flexibility in behaviour design, evident in the behaviours described below.

4.2 Goal Seeking

Based upon the Pure Pursuit algorithm [9], a proven method of path following, the Goal Seeking module attempts to follow straight line segments between waypoints. The algorithm steers toward a point a fixed distance ahead of the current setpoint on the path, continuously recomputing this point and the desired steering angle, α_{ideal}, to produce smooth vehicle trajectories. The behaviour then generates the desirability d_i by convolving a Gaussian curve around α_{ideal} sets all certainties to $p_i = 1$, vetoes to $q_i = 0$ and maximum speeds s_i to a user prescribed maximum value. The vehicle will pursue the path more aggressively the farther away it is from the path.

4.3 Obstacle Detection

Drawing from one or more range sensors, the simple Obstacle Detection module halts the vehicle if it finds any obstacles in the vehicle's immediate path. The algorithm classifies any surface exceeding a tolerance height either above or below the ground as an obstacle. To compensate for noisy sensors, an obstacle sample threshold must be exceeded to trigger a vehicle halt. The module does not avoid obstacles, but safeguards teleoperation or waypoint following operations. Only obstacles within a prescribed obstacle zone, a rectangular area directly in front of the vehicle, will trigger a halt. This zone allows the driver, either human or autonomous, every opportunity to avoid the obstacle before the vehicle is halted.

Normally, the Obstacle Detection module generates a blank vote to the Arc Arbiter by setting all $d_i = 0$, $p_c = 1$, and maximum speed s_i to a large value to not inhibit other behaviours. Having detected an obstacle, the algorithm vetoes all of the candidate arcs (i.e. setting all $q_i = 1$), halting the vehicle.

4.4 Obstacle Avoidance

Similar to other candidate arc systems [13, 26], the *Obstacle Avoidance* Module estimates the cost of driving candidate angles based on the traversability map described in Section 3, as shown in Figure 9.

A circular arc, $\mathbf{a}_i$, approximates the path travelled for each of the candidate steering angles, α_i. The curve's thickness approximates vehicle width by including adjacent traversability cells in the cost. Intersecting cells which intersect the ith arc form a cost vector $\mathbf{c}_i^{'}$ of size N_i. The algorithm discounts these costs through a parameter $F_{disc} = [0, 1]$, the euclidean distance to the current cell e_{curr}, and the furthest cell in the Traversability Map, e_{max}:

$$\mathbf{c}_i = \mathbf{c}_i^{'}(1 - F_{disc} e_{curr}/e_{max}) \tag{26}$$

effectively reducing the cost of distant terrain. It finds the desirability, d_i, of a given arc vote v_i, from the average of the vector of discounted costs:

$$d_i = 1 - \sum_{j=1}^{N} \mathbf{c}_i \tag{27}$$

The obstacle avoidance algorithm vetoes arcs if any intersected cell has a cost of 1:

$$q_i = \begin{cases} 1 & : \quad \mathbf{c}_i[j] = 1 : j = 1 \rightarrow N_i \\ 0 & : \quad \text{otherwise} \end{cases} \tag{28}$$

Finally, the maximum allowed speed of each arc is based upon that arc's vote and a user-defined maximum speed s_{max} and minimum speed s_{min}:

$$s_i = d_i(s_{max} - s_{min}) + s_{min} \tag{29}$$

For a detailed review of path tracking performance see [12].

4.5 Path Planning

For this project DRDC adopted D^*Lite [18], an incremental heuristic search method implementing goal-directed robot navigation in unknown terrain. Incremental search methods reuse information to solve a series of similar searches faster than by solving each search from scratch. Heuristic methods, such as A^* [24], use task-specific information, such as approximated goal distances, to focus and solve search problems often more rapidly than uninformed search methods. In LPA^* [17] incremental and heuristic searches are combined to reduce replanning times. Conventional graph searches, such as repeated A^*, lengthen

prohibitively when forced to replan through previously unknown obstacles. In D^* [28], clever heuristics speed replanning one or two orders over repeated A^* searches. Easier to understand and extend, D^*Lite, builds on LPA^*, implementing the same navigation strategy as D^* and demonstrating similar or better efficiency [18].

D^*Lite finds the minimum cost path from a starting (current) vertex, s_{start}, to a goal vertex, s_{goal}, on a graph, composed of a finite set of vertices S, with changing edge costs. The traversal cost from node s to node s' is denoted by $0 < c(s, s') < \infty$. For each node s, D^*Lite maintains an estimate of the cost, g, to reach a node s from the goal vertex, a one step lookahead cost, rhs, and an estimate of the cost, h, to procede from there to the start vertex. From these estimates, the key values are obtained

$$k(s) = \left[\begin{array}{l} k_1(s) = \min(g(s), rhs(s) + h(s_{\text{start}}, s)) \\ k_2(s) = \min(g(s), rhs(s)) \end{array} \right]. \tag{30}$$

D^*Lite maintains a priority queue U of vertices sorted lexicographically in ascending order by key value. At each step, the vertex with the smallest key value is popped off the top of the priority queue, thus focusing the search in the most promising direction. The D^*Lite algorithm is shown in Figure 6. See [18] for complete details of the algorithm.

In C++, DRDC's D^*Lite algorithm is a template class, DStarLiteSearch. The template parameter, class State, encapsulates the state variables of an element in the search space. Internal to DStarLiteSearch, class Node instance encapsulates a search space element:

- the g, rhs, and h values,
- the key values, computed from the previous three, used to order the priority queue,
- a pointer to the State class instance to which this Node corresponds,
- a backpointer to the predecessor of this instance, and
- a pointer to a drawable representation so that the developing path can be displayed.

In this way, there is a clear separation between the workings of the D^*Lite algorithm and the representation of the planning space. The DStarLiteSearch class needs to know very little about its template parameter, class State, in order to update path costs and expand the search. In order for DStarLiteSearch to update path costs and expand the search, the State class must only provide the functions to: compute the heuristic estimate of the distance to the goal, determine the neighbours of this State, and return the *a priori* and sensed costs of moving to an adjacent State. Currently, the State class gathers traversal costs from a a global traversability map maintained by the Path Planner and updated regularly by the Traversability Map.

DRDC's D^*Lite departs from Koenig's in a number of respects. Backpointers simplify walking the current path. Once the starting node, the robot's current location, has been reached, backpointers identify the optimal path to the goal. In addition to the open list, D^*Lite maintains a closed node list, as in A^*. The open and closed lists are implemented

as std::vector<Node*> instances. D^*Lite maintains the open list as a priority queue (a minimum heap) using the Standard Template Library functions push_heap, pop_heap, and a binary functor, HeapCompare(), for pairwise comparison the key values. Examined nodes that are no longer expansion candidates are removed to the closed list. D^*Lite dynamically allocates Node instances using fixed block sizes to minimize memory allocation expense. Every Node* pointer is either a member of the open list and a candidate for further expansion or on the closed list and may be placed on the open list if its costs change. This minimizes the number of allocations to find a path to the goal and simplifies deallocation.

Separating the D^*Lite algorithm from the planning representation simplifies experimentation with alternate planning spaces. This abstract planning space also permits more sophisticated planners incorporating either additional dimensions or more complex node traversal costs. For example, using vehicle dynamics to build a dynamics-aware path planner.

The path planner generates arc votes for the Arc Arbiter to intercept the planned path at an interim target through an Adaptive Pure Pursuit algorithm similar to that described in [16] and in section 4.2. As a local obstacle avoidance grows the tracking error, the lookahead distance increases as well. Ideally, this advances the interim target beyond near field obstacles so that the robot can return to the desired path. If the tracking error grows beyond the maximum permissible value, then the planner is forced to replan. An in depth discussion of DRDC's implementation of D^*Lite is found in [22]

4.6 Decision Making

The Arc Arbiter uses votes from each available behaviour to make decisions and receives votes from each behaviour asynchronously. With each vote event, the Arc Arbiter asynchronously re-evaluates its decision and is, therefore, highly reactive to incoming data.

The Arc Arbiter chooses the candidate arc with the highest combined desireability and certainty, setting the speed to the arc's lowest s_{max} and disallowing any vetoed arcs. If all arcs are vetoed, the vehicle is stopped. The algorithm selects an arc by compiling the votes into a combined vote set $\mathbf{v}_{ci}$. For the ith arc, the combined vote is $\mathbf{v}_c =< d_c, s_c >$, where for a number of behaviours b it consists of:

$$d_{ci} = \begin{cases} 0 & : \quad q_j = 1 : j = 1 \rightarrow b \\ \sum_{j=0}^{b} d_j p_j & : \quad \text{otherwise} \end{cases} \tag{31}$$

$$s_{ci} = \min(s_j) : j = 1 \rightarrow b \tag{32}$$

At this point, it chooses the final steering angle and velocity sent to the vehicle (α_f, s_f) from $\mathbf{v}_c$:

$$\alpha_f = \alpha_i, \quad s_f = s_i \tag{33}$$

where i is the candidate arc with the highest value d_c. If $d_c = 0$ for all i, then the vehicle is stopped.

The Arc Arbiter produces *steering and velocity* events that, as shown in Figure 1, are consumed by the vehicle process which commands the vehicle actions.

```
CalculateKey(s)
return [min(g(s), rhs(s))+h(s_start, s)+k_m; min(g(s), rhs(s))]

Initialize()
U ← 0
k_m ← 0
for (s ∈ S) do rhs(s)← g(s)← ∞
rhs(s_goal)← 0
U.Insert(s_goal, CalculateKey(s_goal))

UpdateVertex(u)
if (u ≠ s_goal) then
    s'← arg min_{s'∈Succ(u)}(c(u, s')+g(s'))
    rhs(u)←c(u, s')+g(s')
    Next(u)← s'
if (u ∈ U) then U.Remove(u)
if (g(u)≠rhs(u)) then U.Insert(u, CalculateKey(u))

ComputeShortestPath()
while (U.TopKey < CalculateKey(s_start) or rhs(s_start)≠g(s_start)) do
    k_old ← U.TopKey()
    u ← U.Pop()
    if (k_old < CalculateKey(u)) then
        U.Insert(u, CalculateKey(u))
    else
        if (g(u)>rhs(u)) then
            g(u)←rhs(u)
            for (s ∈ Pred(u)) do UpdateVertex(s)
        else
            g(u)← ∞
            for (s ∈ Pred(u) ∪ {u}) do UpdateVertex(s)

Main()
s_last = s_start
Initialize()
ComputeShortestPath()
while (s_start ≠ s_goal) do
    // if (g(s_start)= ∞) then there is no known path
    s_start ← arg min_{s'∈Succ(s_start)}(c(s_start, s')+g(s'))
    Move to s_start
    Scan the graph for changed edge costs
    if (any edge costs changed) then
        k_m = k_m+h(s_last, s_start)
        s_last = s_start
        for (all directed edges (u, v) with changed costs) do
            Update the edge cost c(u, v)
            UpdateVertex(u)
        ComputeShortestPath()
```

Figure 6: *D*Lite:*TopKey() *returns the smallest key value from the priority queue.* Pop() *removes the vertex with the smallest key value from the queue.* Insert(u) *places node u on the queue, resorting it.* Remove(u) *removes node u from the queue.* Succu(u) *and* Pred(u) *are the successors and predecessors of u, respectively.*

5 Vehicle Intelligence

The Vehicle Intelligence module and the Waypoint Arbiter receive information, including pose, from multiple vehicles. Three different mission variables determine the operational state and reactivity of each vehicle: mission mode mM (teleoperation, autonomous and monitored), originator mD (control station and vehicle intelligence), and task mT (waypoint following, formation keeping and collective search). Figure 7 details the switching of control between these three modes.

5.1 Multicast Adaptor

The multicast adaptor is the element enabling the exchange of state information between vehicles in a seemless manner across the network. The multicast adaptor allows different naming contexts (domains) to talk to each other and share information.

Similar to [30], the multicast adaptor uses a message-based, high-level, connection-less, communications protocol transferred via IP-multicast across a wireless LAN and, therefore, is immune to both network breakdown and latencies.

DRDC modified the notify multicast module (NMC) of the MIRO [29] framework to work with their wireless system by changing the time to live levels. The NMC module exchanges events between robots transparently using IP-multicast. Figure 8 illustrates multicast adaptor operation.

Multivehicle tasks can use this transparent event exchange to implement team behaviours such as formation keeping or task distribution. DRDC developed two team behaviours that exploit this transparency: *follow-the-leader* and *search*. Both operate on the same principle of explicit inter-vehicle information exchange. For example: In follow-the-leader, the follower subscribes to leader pose information and, using Pure Pursuit, tracks positions a fixed distance behind the leader.

```
Initialize()
mM ← TELEOP
mD ← CSTATION
mT ← A2B
viInfo ← default()
csInfo ← default()

ControlStation()→return csInfo

ControlStation.getInfo(&mM,&mT,&wpl )
csInfo ← ControlStation.data(mD,mM,mT,wpl )

VehicleIntel(csInfo,viInfo)→return viInfo

if r.mM = TELEOP then
    viInfo ← NULL
else
    if r.mT = A2B then
        viInfo ← r
    else if r.mT = LEADER/FOLLOWER then
        if r.pose.distTo(r.target) < m.pose.distTo(r.target) then
            viInfo ← r
        else
            viInfo ← r.distBehind(m,DIST )
    else if r.mT = SEARCH then
        if r.pose.distTo(r.target) < m.pose.distTo(r.target) then
            viInfo ← r.searchInside()
        else
            viInfo ← r.searchOutside()
    else
        viInfo ← NULL

waypointArbiter(csInfo,viInfo)→return wps

if r.mM = TELEOP then
    if r.mD = CSTATION then
        mM ← csInfo.mM
        wps ← csInfo.wps
    else
        wps ← NULL
else
    if r.mD = CSTATION then
        if csInfo.mM = TELEOP then
            mM ← csInfo.mM
            wps ← csInfo.wps
        else
            wps ← NULL
    else
        mM ← viInfo.mM
        wps ← viInfo.wps

Main()
Initialize()
while (Command() ≠ STOP) do
    csInfo ← ControlStation()
    viInfo ← VehicleIntel(csInfo, viInfo )
    wpaCommand ← waypointArbiter(csInfo, viInfo )
    Command().Send(wpaCommand)
```

Figure 7: *The vehicle intelligence flow diagram uses the variables: robot r, teammate m, mission mode mM, mission designator mD, mission task mT, vehicle intelligence information $viInfo$, control station information $csInfo$, waypoint list wpl, and waypoints wps.*

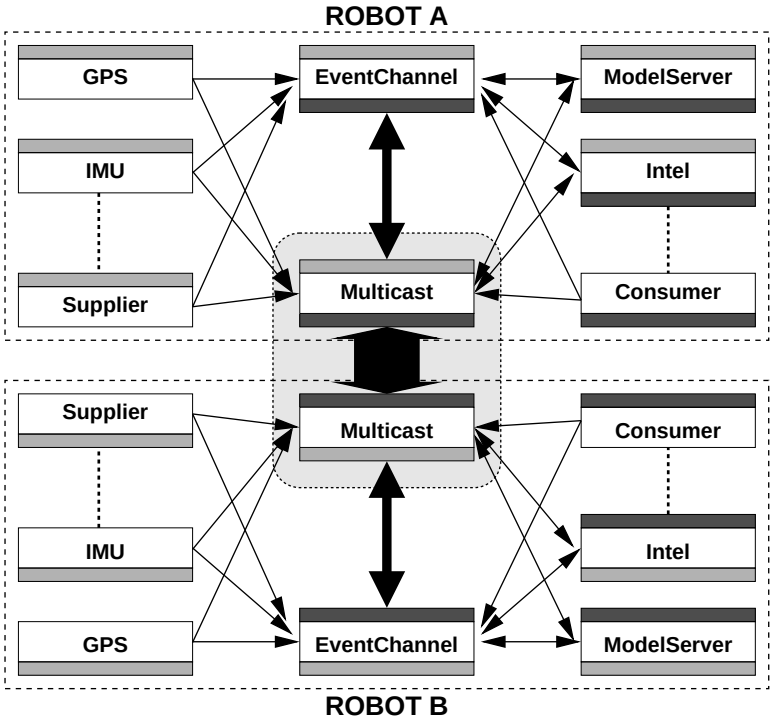

Figure 8: *Sharing information with interested clients via IP-multicast, the multicast adaptor and is unaffected by either network breakdown or latencies All inputs to the adaptor are supplied events, all outputs are subscribed events. Robot A's events are passed through the event channel to other processes requesting those events on Robot A. Robot A's events requested by Robot B are passed through the multicast adaptors.*

6 Results

In fall 2005, the ALS Project culminated in field trials designed to reveal the system's strengths and weaknesses while travelling through obstacle fields and over unimproved grassland prairie. Project results fall into three categories: hardware assessment, software utility, and algorithmic suitability.

6.1 Hardware Assessment

For the moderate grassland prairie environment, the Raptor's sensing proved adequate for low speed manoeuvres. The nodding laser and stereo imagery provides sufficient detail for traversing the selected environment at 5-10 km/h. With excellent range, field of view, and resolution, the system relied heavily on the nodding laser. Though Stereo imagery filled the map more effectively, limited field of view and edge effects undermined its performance. Augmented with DGPS corrections, the absolute position and orientation data from the GPS was collected at a rate of 4 Hz. IMU linear accelerations, rotational velocities, and absolute heading were sampled at a rate of approximately 16 Hz. The Raptor vehicle returned data about the Ackerman steering angle, the steered velocity of the front wheels, and the odometry from the left and right front wheels at a rate of up to 20 Hz.Surprisingly, COTS Differential GPS and IMU together could, and often did, replace centrally fused pose estimates in DRDC's field trials.

6.2 Software Utility

Though complex, the AfA toolchain greatly simplified software development. The mature publish/subscribe architecture, arbitrary data logging and playback, and IDL interface discipline, proved pivotal in the system's rapid evolution. With experience, developers could quickly develop and test new modules, often in parallel with earlier versions. Through Miro's event system, a single developer could quickly log arbitrary IDL events in the field, such as device or module output, for instant playback in the lab. This feature allowed multiple developers to independently, yet simultaneously, build and test their modules using real field data.

The software modularity promoted rapid algorithm development and easy reconfiguration while accommodating an emerging architecture. However, the system's complexity revealed the need for better run-time management. Depending on the sensors present and level of autonomy required, the user would have to uniquely configure and launch nearly 20 different processes, a significant barrier to usability. The group is currently working to improve both Miro-based XML configuration and process launch, monitor, and control facilities.

6.3 Algorithm Assessment

6.3.1 World Representation Services

DRDC's Model Service provides both internal geometry and vehicle pose to consuming services through an event driven MIRO process. Any one of *GPS*, *IMU*, *WheelEvent*, and

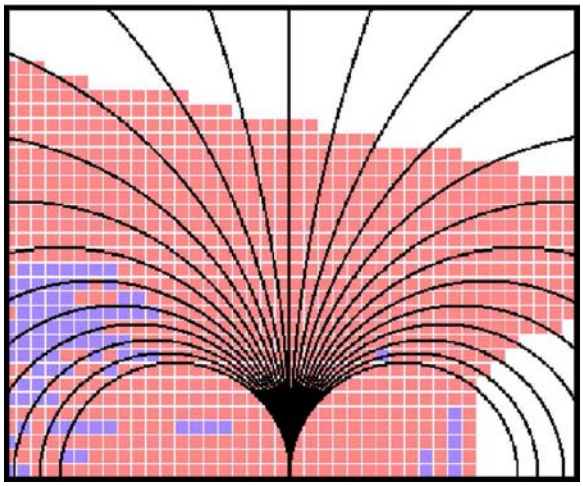

Figure 9: *A traversability map overlaid with 25 candidate arcs. Obstacles are blue and traversable areas are red.*

timer events could drive the recalculation of vehicle pose estimate and the subsequent 'Pose' event publication. Though DGPS and IMU data could substitute for a filtered pose estimate, filtered solutions provide the only practical option in the uncertain radio environment of the battlefield.

DRDC implemented the wrappable, variance weighted terrain map on the Raptor UGV. Taking *Pose*, *Laser*, and *Stereo* events, the Terrain Map service inserted range data into $0.2\text{m} \times 0.2\text{m}$ grid elements within a $20\text{m} \times 20\text{m}$ patch centered on the vehicle bumper. With 3D elevation error growing with range, small pose errors became significant only at long range. Variance weighted statistics resolved such errors for high data density regions but, understandably, had less success in lower density regions.

The Terrain Map's variance weighted algorithm also removed ghost or false obstacles from the map as new data arrived. Moving obstacles would remain in the map for a time after moving on, producing a ghost image in the map. Similarly sudden vehicle motion or changes in perspective near obscured views could produce large patches of unknown terrain — all treated as obstacles by the traversability map. Both effects were removed over time through the maps variance weighted approach.

DRDC's Traversability Service translated terrain map data into a $20\text{m} \times 20\text{m}$ traversability map with $1\text{m} \times 1\text{m}$ grid. Experimentation proved that the traversability map was sufficient for UGV planning over the 2.5D outdoor environment at low speeds.

6.3.2 Vehicle Control Services

Testing at DRDC demonstrated the combination of discrete behaviour modules in a reactive, arbitration system to create autonomous behaviour. In pure pursuit, the system performed without fault, even using unfiltered low resolution GPS modes. In obstacle avoidance, the system performed reasonably well. However, with only a bumper-forward terrain map,

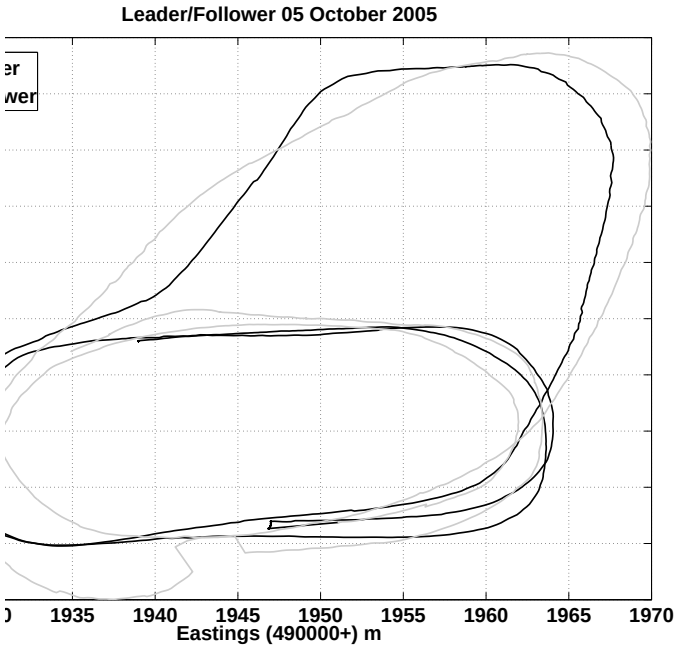

Figure 10: *Paths of two raptors, one following the other. Note the man driven Raptor trajectory appears smoother than the autonomous Raptor — evidence of short term trajectory planning typical of Pure Pursuit.*

obstacles disappeared as they fell behind the vehicle bumper. Passing within $\approx$ 20cm of an obstacle on the inside of an avoidance turn, the vehicle could collide with obstacles, now off the map, on the return to the path.

While the sensors rarely, if ever failed to detect real obstacles,transient obstacles and unknown terrain frequently made the vehicle appear hesitant and the steering uncertain. Both of these types of ghost obstacles would halt the vehicle until the forward view was clarified. For the most part, this resulted in sensible, if cautious, motion. However, if vehicle motion scrolled one of these ghost obstacles below the sensor horizon, the system would remain stopped, unable to clarify the near-field view.

6.3.3 Multivehicle Control Service

Unfortunately, the multivehicle control services were not ready for the 2005 September demonstrations. However, in October of 2005 DRDC experimented with the multicast adaptor. In these trials a *follower* tries to follow a *leader*'s waypoint breadcrumb path to implement a follow-the-leader behaviour. Figure 10 depicts logged results from one follow-the-leader run for two vehicles, where the the follower drove autonomously in convoy chasing a man-driven leader vehicle. Differences between follower and leader tracks can be attributed to shortsighted planning by Pure pursuit in comparison to human drivers.

	Robotics		Software Dev.		AfA Expertise		% FTE
	pre	post	pre	post	pre	post	
1	4	5	2	3	0	4	80
2	5	5	2	2	0	1	10
3	3	5	3	5	0	5	80
4	4	5	5	5	0	4	80
5	3	5	2	3	0	3	80
6	4	5	2	3	0	4	40
7	4	4	1	1	0	1	20
8	1	3	1	1	0	0	20
9	1	4	0	3	0	5	80
10	1	4	2	3	0	5	80
11	1	3	0	1	0	2	40
12	1	1	2	2	0	2	10
Ave.	2.69	4.15	1.69	2.62	0	2.77	52.5

Table 1: *Deputy Project Manager skill estimates of volunteers pre and post(0-no experience, 5 -expert) demonstration.*

6.4 Training and Teambuilding

The ALS Demonstration fell into two major categories: Mobility Demonstrations and the Raptor Demonstration. Training and teambuilding were major goals of the ALS Project and relied on individuals to volunteer time towards either category. In a section of some 25 scientists and engineers, 12 volunteers some time to the Raptor Demonstration. By the end of the Raptor demo, 6 possessed intimate understanding of the system's operation, 2 contributed significant time, but could not run the system alone, and 4 made peripheral or indirect contributions.

Robotics is a team-based a multi-disciplinary practice that benefits from committed team members with varied backgrounds but sharing software development skills. The DRDC research team included members who had in-depth software experience though most did not. Most of the experienced programmers had no formal software background. Virtually all the team were unfamiliar or uncomfortable with complicated software development and software integration issues. Finally, many had only a brief introduction to mobile robotics, a discipline that often lacks the scientific rigour common to manipulation or factory robotics.

Table 1 presents a highly subjective and qualitative list of volunteer skills pre and post demonstration scaled from 0 (no experience) to 5 (expert) from the deputy project manager's standpoint. The final column indicates an estimated FTE commitment by the volunteer, noting that 80% contribution is the practical maximum for most institutions.

Based on the estimates contained in table 1, approximately 12 man years were committed to the demonstration alone between September 2003 and September 2005. This correlates well with the more precise SLOCcount [31] code counting estimates of approximately 13

man years. Not surprisingly, there is a clear correlation between FTE and AFA expertise. Again, not surprisingly robotics and software expertise rose in proportion to participation.

With some notable exceptions, this table highlights the correlation between existing software skills and the successful mastery of the AfA system. Plainly stated, volunteers who did not already possess software experience did not easily acquire these skills without commitment, regardless of their prior training. This split along software skill lines is problematic, since members without the necessary software skills and familiarity will find experimentation difficult (usually in the form of software testing) on an operational robot. Although less concrete issues may have influenced this split (such as enthusiasm– a difficult property to estimate), software complexity and the lack of documentation were certainly significant factors and must be rectified to lower the necessary time to acquire these skills.

7 Conclusion

This project successfully demonstrated autonomous single and, later, multi-vehicle behaviour using off-the-shelf hardware, a plausible military software architecture, and an arbitrated behaviour control structure. A mixture of commercial hardware and services proved to meet the needs of basic vehicle autonomy with little in-house manufacturing. Similarly, the open source AfA toolchain provides all the foreseeable infrastructure for future autonomous unmanned systems, encouraging flexibility and modularity without sacrificing performance. The arbitrated behaviour structure clearly demonstrated the advantages of MIRO and arbitrated control, including modularity, extensibility, and networked multivehicle operations. Both traversability and terrain maps demonstrated the power of ego-centric mapping for local , while D*-lite reinforced the necessity of accurate consistent pose in path planning. Together the system proved complex and challenging, while promising considerable future potential.

This project also demonstrated the clear correlation between volunteer skill acquisition and time commitment. While software complexity and poor documentation were significant barriers to the Raptor demonstration, most developmental autonomous vehicle systems possess similarly complex and inscrutable software and will do so for the foreseeable future. Though software and hardware will eventually lower some of these barriers, there is no escape from the fundamental complexities of hardware, process timing, networks, signal analysis, vehicle control, kinematics and dynamics.

References

[1] D. S. Apostolopoulos. Technology and field demonstration of robotic search for antarctic meteorites. *International Journal of Robotics Research*, 19(11):1015–1032, 2000.

[2] P. Bellutta, R. Manduchi, L. Matthies, K. Owens, and A. Rankin. Terrain perception for demo iii. In *Proceedings of the 2000 Intelligent Vehicles Conference*, 2000.

[3] R. Brooks and T. Lozano-Perez. A subdivision algorithm in configuration space for findpath with rotation. In *Proceedings of the 8th International Conference on AI*, pages 799–806, 1983.

[4] G. Broten and J. Collier. The characterization of an inexpensive nodding laser. DRDC Suffield TR 2005-232, Defense R&D Canada – Suffield, Medicine Hat, Alberta, December 2005.

[5] G. Broten, J. Giesbrecht, and S. Monckton. World representation using terrain maps. DRDC Suffield TR 2005-248, Defence R&D Canada – Suffield, Medicine Hat, Alberta, December 2005.

[6] G. Broten, S. Monckton, J. Giesbrecht, and J. Collier. *Software Engineering for Experimental Robotics*, drdc suffield sl UxV Software Systems, An Applied Research Perspective. Number 2005-227. Springer Tracts in Advanced Robotics, 2006.

[7] G. Broten, S. Monckton, J. Giesbrecht, S. Verret, J. Collier, and B. Digney. Towards distributed intelligence - a high level definition. DRDC Suffield TR 2004-287, Defence R&D Canada – Suffield, December 2004.

[8] J Collier, G. Broten, and J. Giesbrecht. Traversibility analysis for unmanned ground vehicles. DRDC Suffield TM 2006-175, Defence R&D Canada – Suffield, P.O. Box 4000, Station Main, Medicine Hat, Albe, October 2006.

[9] R. Craig Coulter. Implementation of the pure pursuit path tracking algorithm. Tech Report CMU-RI-TR-92-01, Carnegie Mellon University, 1992.

[10] J. Giesbrecht. Global path planning for unmanned ground vehicles. DRDC Suffield TM 2004-272, Defence R&D Canada – Suffield, 2004.

[11] J. Giesbrecht, J. Collier, and S. Monckton. Staged experiments in mobile vehicle autonomy: Procedures and results with the SegwayRMP. DRDC Suffield TM 2004-288, Defense R&D Canada – Suffield, Medicine Hat, Alberta, December 2004.

[12] J. Giesbrecht, D. Mackay, J. Collier, and S. Verret. Path tracking for unmanned ground vehicle navigation. DRDC Suffield TM 2005-224, Defence R&D Canada – Suffield, December 2005.

[13] S. Goldberg, M. Maimone, and L. Matthies. Stereo vision and rover navigation software for planetary exploration. *IEEE Aerospace Conference Proceedings*, 2002.

[14] L. Henriksen and E. Krotkov. Natural terrain hazard detection with a laser rangefinder. In *Proceedings IEEE International Conference On Robotics and Automation*, pages 968–973, Albuquerque, New Mexico, April 1997.

[15] M. Herbert and E. Krotkov. Local perception for mobile robot navigation in natural terrain: Two approaches. In *Workshop on Computer Vision for Space Applications*, pages 24–31, Sept. 1993.

[16] A. J. Kelly. An approach to rough terrain autonomous mobility. In *International Conference on Mobile Planetary Robots*, Santa Monica, January 1997.

[17] S. Koenig and M. Likhachev. Lifelong planning a*. Technical Report GIT-COGSCI-2002/2, Georgia Institute of Technology, 2001.

[18] S. Koenig and M. Likhachev. D* lite. *Proceedings of the National Conference on Artificial Intelligence*, pages 476–483, 2002.

[19] S. Kweon and T. Kanade. High-resolution terrain map from multiple sensor data. *IEEE Transactions on Pattern Analysis and Machine Vision*, 14(2):278–292, Feb. 1992.

[20] S. Lacroix, A. Mallet, and D. Bonnafous. Autonomous rover navigation on unknown terrains demonstrations in the space museum "cite de l'espace" at toulouse automation, albuquerque, usa, 1997. In *7th International Symp. on Experimental Robotics*, pages 669–683, Honolulu, HI, 2000.

[21] Frank L. Lewis. *Applied Optimal Control and Estimation.* Digital Signal Processing Series. Prentice Hall, 1992.

[22] David Mackay. Path planning with d*-lite. DRDC Suffield TM 2005-242, DRDC Suffield, December 2005.

[23] S. Monckton, I. Vincent, and G. Broten. A prototype vehicle geometry server: Design and development of the modelserver corba service. DRDC Suffield TR 2005-240, Defence R&D Canada – Suffield, Medicine Hat, Alberta, December 2005.

[24] N. Nilsson. *Principles of Artificial Intelligence.* Tioga Publishing Company, 1980.

[25] J.K. Rosenblatt. DAMN: A distributed architecture for mobile navigation. In *Proceedings of the 1995 AAAI Spring Symposium on Lessons Learned from Implemented Software Architectures for Physical Agents , H. Hexmoor and D. Kortenkamp (Eds.).* AAAI Press, Menlo Park, CA., March 1995.

[26] S. Singh, K Schwehr, R. Simmons, T. Smith, A. Stenz, V. Verma, and A. Yahja. Recent progress in local and global traversability for planetary rovers. *International Conference on Robotics and Automation*, 2000.

[27] Russell Smith. *Open Dynamics Engine v0.5 Users Guide*, May 2004.

[28] A. Stentz. The focussed d algorithm for real-time planning. *Proceedings of the InternationalJoint Conference on Artificial Intelligence*, 1995.

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

[30] Hans Utz, Freek Stulup, and Arndt Mühlenfeld. Sharing belief in teams of heterogeneous robots. In D. Nardi, Riedmiller, Sammut M., and J. C., Santos-Victor, editors, *RoboCup 2004: Robot Soccer World Cup VIII*, pages 508–515, Berlin, Heidelberg, Germany, 2005. Springer-Verlag.

[31] D. Wheeler. Sloccount: Source lines of code count. Webpage, 2004. Version 2.26.

[32] D. Zhu and J. Latombe. Constraint reformulation and graph searching techniques in hierarchical path planning. Technical Report CS-89-1279, Dept. of Computer Science, Stanford University, 1989.

This page intentionally left blank.

DOCUMENT CONTROL DATA

(Security classification of title, body of abstract and indexing annotation must be entered when document is classified)

1. ORIGINATOR (the name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Centre sponsoring a contractor's report, or tasking agency, are entered in section 8.)

   Defence R&D Canada – Suffield
   PO Box 4000, Medicine Hat, AB, Canada T1A 8K6

2. SECURITY CLASSIFICATION (overall security classification of the document including special warning terms if applicable).

   UNCLASSIFIED

3. TITLE (the complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S,C,R or U) in parentheses after the title).

   Staged Experiments in Mobile Vehicle Autonomy II

4. AUTHORS (last name, first name, middle initial)

   Monckton, S.; Collier, J.; Giesbrecht, J.; Broten, G.; MacKay, D.; Erickson, D.; Vincent, I.; Verret, S.

5. DATE OF PUBLICATION (month and year of publication of document)	6a. NO. OF PAGES (total containing information. Include Annexes, Appendices, etc).	6b. NO. OF REFS (total cited in document)
December 2006	44	32

7. DESCRIPTIVE NOTES (the category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered).

   Technical Report

8. SPONSORING ACTIVITY (the name of the department project office or laboratory sponsoring the research and development. Include address).

   Defence R&D Canada – Suffield
   PO Box 4000, Medicine Hat, AB, Canada T1A 8K6

9a. PROJECT NO. (the applicable research and development project number under which the document was written. Specify whether project).	9b. GRANT OR CONTRACT NO. (if appropriate, the applicable number under which the document was written).

10a. ORIGINATOR'S DOCUMENT NUMBER (the official document number by which the document is identified by the originating activity. This number must be unique.)	10b. OTHER DOCUMENT NOs. (Any other numbers which may be assigned this document either by the originator or by the sponsor.)
DRDC Suffield TR 2006-243	

11. DOCUMENT AVAILABILITY (any limitations on further dissemination of the document, other than those imposed by security classification)

    (X) Unlimited distribution
    ( ) Defence departments and defence contractors; further distribution only as approved
    ( ) Defence departments and Canadian defence contractors; further distribution only as approved
    ( ) Government departments and agencies; further distribution only as approved
    ( ) Defence departments; further distribution only as approved
    ( ) Other (please specify):

12. DOCUMENT ANNOUNCEMENT (any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution beyond the audience specified in (11) is possible, a wider announcement audience may be selected).

    Unlimited

13. ABSTRACT (a brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), (R), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual).

Defence R&D Canada (DRDC) is extending its considerable experience in teleoperated air, land and seaborne systems to the development of autonomous systems for the Canadian Forces. The first project, Autonomous Land Systems (ALS), sought to establish personnel and technical foundations through the demonstration of basic autonomous multivehicle land capabilities. The outcome of a 2 year development effort, this paper summarizes the properties and methods of a basic multivehicle control system founded upon a plausible military software architecture.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus. e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus-identified. If it not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title).

autonomous robotics, multivehicle systems, autonomous intelligent systems