Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

A Tutorial Introduction to Autonomous Systems

Kevin L. Moore

*Division of Engineering, Colorado School of Mines, Golden, CO 80403
USA (Tel: 303-273-3898; e-mail: kmoore@mines.edu).*

Abstract: This paper presents a tutorial-level introduction to the technical aspects of unmanned autonomous systems. We emphasize a system engineering perspective on the conceptual design and integration of both the components used in unmanned systems, including the locomotion, sensors, and computing systems needed to provide inherent autonomy capability, and the algorithms and architectures needed to enable control and autonomy, including path-tracking control and high-level planning strategies. Concepts are illustrated using case study examples from robotic and unmanned system developed by the author and his colleagues.

1. INTRODUCTION

This paper presents a basic introduction to the technical aspects of unmanned and autonomous systems. Our perspective is that one must take a systems engineering approach to develop an unmanned, autonomous system and the paper is organized to emphasize our viewpoint. Thus we consider first application-driven conceptual design of autonomous systems. Next we consider the components needed to implement an unmanned system and how these components are integrated together, including: power and drive train components for locomotion, electro-mechanical and computational system elements, proprioceptive sensors, environmental sensors for self-sensing, localization, and perception. We then consider architectures to integrate and coordinate the algorithms in an unmanned system, including servo-level and path-tracking controllers, based on actuator modelling and system-level kinematics, as well as navigation, motion planning, and intelligent behaviour generation using high-level planning techniques based on searching. The ideas presented in the paper are generally applicable to all types of unmanned systems, but we will focus on autonomous unmanned ground systems, using case study examples of real systems developed by the author and his colleagues. We conclude the paper with comments on research opportunities and challenges in autonomous unmanned systems[1].

2. WHAT IS AN UNMANNED SYSTEM?

What is an unmanned system? Is it the same as an unmanned vehicle? Is an unmanned system a robot? Is a robot an unmanned system? Is an unmanned system an autonomous system? What is an autonomous system? What about unmanned sensors? What about mobile sensors? What about tele-presence or tele-operation? What about teams of unmanned vehicles, or swarms? All of these terms and phrases appear in the literature with a variety of meanings, but often used synonymously.

[1] Due to the tutorial nature of this paper, all reference citations are collected at the end of the paper.

To establish a framework, let us begin with the word “robot.” From Wikipedia™ we learn that the word “robot” was introduced in *Rossum's Universal Robots*, a 1921 science fiction play by Karl Čapek. In this play a factory makes artificial people that are called robots, which can be mistaken for humans. In more recent times, people think of “robots” as either manipulator arms or systems with human-like motion or features, such as the Kuka or Asimov robots shown in Fig. 1.

Fig. 1. Industrial manipulator made by KUKA (photo credited to) and Asimov robot made by Sony (photo from web).

Again referring to Wikipedia™, one definition is that “… a *robot* is a mechanical or virtual, artificial agent. A robot is usually an electro-mechanical system, which, by its appearance or movements, conveys a sense that it has intent or agency of its own. A robot may have the following properties: it is not natural/has been artificially created; it can sense its environment; it can manipulate things in its environment; it has some degree of intelligence, or ability to make choices based on the environment, or automatic control or pre-programmed sequence; it is programmable; it can move with one or more axes of rotation or translation; it can make dexterous coordinated movements; it appears to have intent or agency.”

These descriptors of robots are useful and provide some ideas about how to define an unmanned or autonomous system. The key distinction is that while people often associate the word robot with a manipulator arm or a humanoid-type system, we want to consider a broader class of systems that admit a variety of locomotion capabilities as well as autonomous behaviour capabilities. To this end, let us define:

> *Unmanned system*: any electro-mechanical system with the capability to carry out a prescribed task or portion of a prescribed task automatically, without human intervention.

Such systems can be *autonomous*, able to act alone to follow some plan or script while adapting to changes in the environment. However, we will also allow the descriptor "unmanned system" to be applied to *tele-operated* or *tele-manipulated* systems as well, meaning those systems that can be operated remotely by a human. A sub-class of unmanned systems are *unmanned vehicles*: a vehicle that does not contain a person. In the remainder of this paper we will use the phrases unmanned system, unmanned vehicle, autonomous system, and robotics system interchangeable, but will focus primarily on unmanned vehicles in our examples. It should also be noted that most unmanned systems are intended to deploy a payload (sensor or actuator).

Unmanned vehicles can come in several flavours:

- Unmanned land (or ground) vehicles (UGV), which come in all varieties, but which can often be categorised as either wheeled, tracked, or novel.
- Unmanned air vehicles (UAV), which may be fixed-wing, rotary-wing, or ducted fan.
- Unmanned maritime vehicles, which may be underwater vehicles (UUV) or surface vessels (USV).

Collectively we will refer to UxVs where "x" can mean ground, air, underwater, or surface. Examples of typical, currently- fielded UxVs are shown in Fig. 2. It is also common to identify so-called unattended ground sensors (UGS) as an unmanned system.

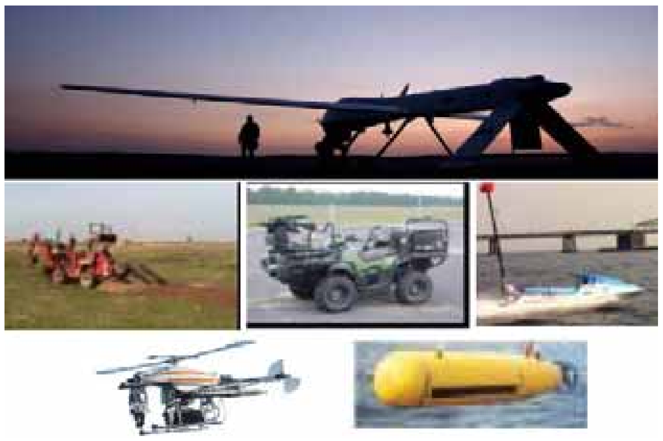


Fig. 2. Example UxVs (photos taken from the web).

We conclude this section by noting that recent research trends have extended beyond single-entity autonomy to address the cooperative autonomy that can emerge when collections of unmanned systems are deployed to work together. For example, the SwardBot™ developed by the company IRobot and studied at MIT (shown in Fig. 3), can demonstrate flocking behaviour typical of swarms found in nature when programmed with the right algorithms. In a related example, Fig. 4 depicts the idea of meta-swarms that can result when different types of UxVs work together inside the so-called global information grid (GiG). We will not address swarming or networked UxVs in the remainder of the paper, but will focus only on single-entity autonomy.

Fig. 3. Swarming (photo credit http://www.irobot.com).

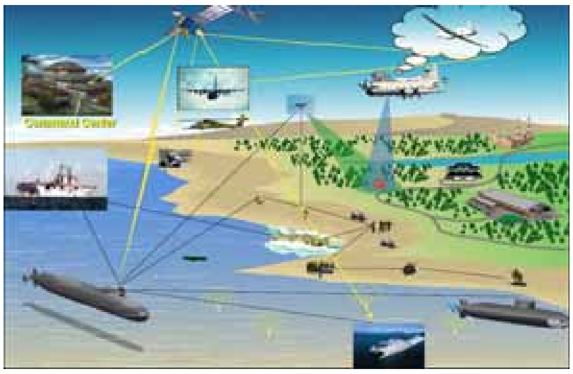


Fig. 4. Networked UxVs.

3. HOW DO YOU MAKE A UxV?

First, we can note that all UxVs have common elements: mechanical components (drive, power, chassis), electronics and computational resouces, sensing/mission payloads, communication systems, control systems, smart algorithms for perception and decision-making, and ways to interface to user.[2] These components are depicted in Fig. 5.

In considering the components of an unmanned autonomous system, we take the point of view that there are two key aspects of unmanned vehicles and autonomy:

1. Inherent physical capabilities built into the system.
2. Intelligent control to exploit these capabilities.

By *inherent physical capabilities built into the system*, we mean mechanisms for mobility and manipulation, power, sensors for perception, both proprioceptive (meaning self-

[2] While it may seem an oxymoron to talk about a user interface for an unmanned vehicle, practically speaking all unmanned vehicles or autonomous systems exist for a purpose defined by humans. Thus we must have a way to deploy and monitor autonomous systems.

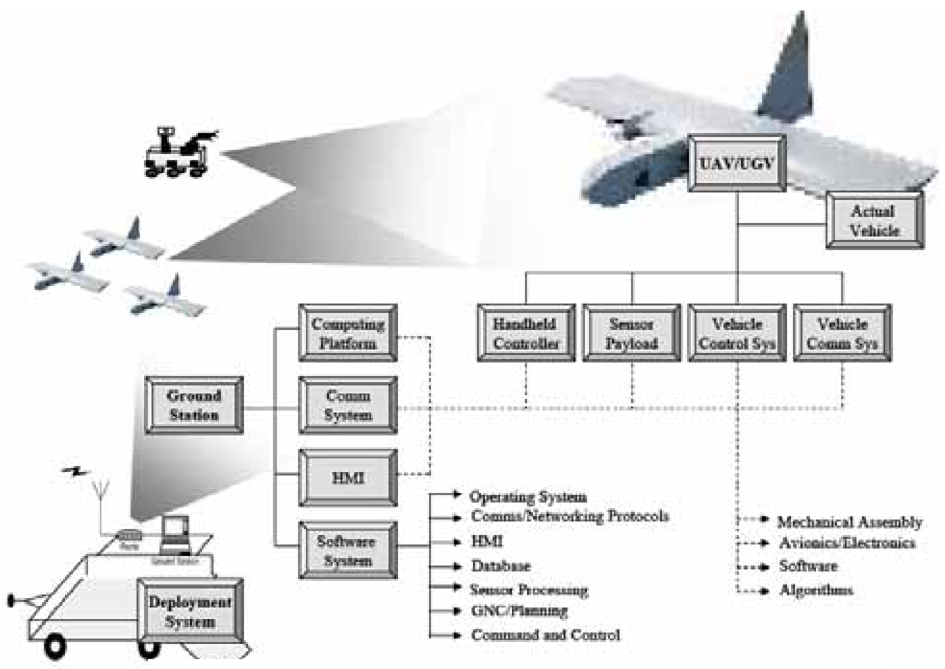


Fig. 5. Components of unmanned autonomous systems.

sensing) and external sensors, and computational resources. By *intelligent control to exploit these capabilities*, we refer to algorithms that include machine-level control, perception algorithms, reasoning, decision-making, learning, and human-machine interfaces.

In the remainder of this section we will discuss some of the individual components shown in Fig. 5. First, however, we note that all unmanned systems should be developed from the perspective of its concept of operations (CONOPS). For instance, Fig. 6 shows an unmanned vehicle intended for autonomous orchard spraying operations. Fig. 7 shows the OmniDirectional Inspection System (ODIS) intended for under-vehicle surveillance. Clearly the requirements for the individual subsystems of each vehicle are different. Once a CONOPS has been defined, then systems engineering is used to flow-down requirements for subsystems. A common mistake in robotic vehicle development is to not begin the process with a clearly stated CONOPS.

4. INHERENT CAPABILITIES OF UNMANNED SYSTEMS

In this section we discuss the inherent capabilities needed in unmanned systems and in the next section we discuss algorithms that exploit these inherent capabilities. Much of our discussion will use the ODIS robot as an illustrative case study. Thus we begin the section with an overview of ODIS.

4.1 ODIS CONOPS

The ODIS robot shown in Fig. 7 is only 3.75 inches tall and has three wheels, each with the same smart wheel ODV capability described in the next subsection. Using GPS, odometry, and on-board sensors, ODIS can navigate though a parking area either, a) going from stall to stall, inspecting any vehicles that it finds, or, b) going to a prescribed location and inspecting any vehicles it finds in that location. Once a vehicle is found to be in a parking stall some form of inspection may be carried out. After deciding to inspect a vehicle, ODIS characterizes the vehicle, finding bumper and tire locations, and then autonomously travels under the vehicle, sending streaming video back to the operator station for analysis.

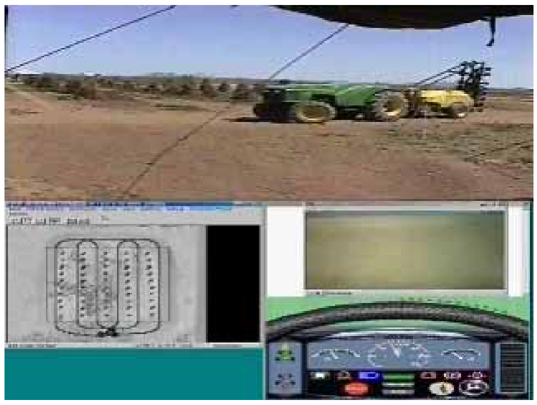


Fig. 6. Autonomous orchard spraying tractor.

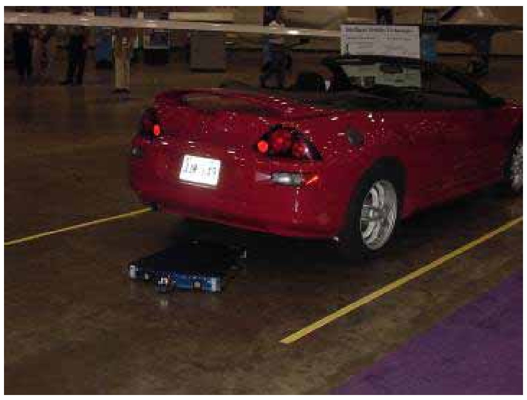


Fig. 7. ODIS robot.

4.2 System Engineering Concepts for UxVs

Based on the motivating UxVs CONOPS, a system engineering approach can be taken to define requirements for the subsystems of the system. In the case of the ODIS vehicle, the overall specifications shown in Fig. 8 were articulated based on the CONOPS.

• Weight:	approx. 40 lb.
• Height:	3.75 inches
• Footprint:	25" X 32"
• Velocity:	2.5 ft/sec
• Power Source:	Battery
• Number of Wheels:	3
• Number of Processors:	8
• Environmental Sensing:	Sonar, IR, Laser
• Position Sensing:	dGPS, FOG
• Vehicle Runtime:	1 Hour
• Number of Battery Packs:	2 (12 V and 24 V)
• Ground Clearance:	0.5 inches

Fig. 8 ODIS specifications.

From these specifications, the design process proceeded to consider the mechanical aspects of locomotion and power, vehicle electronics (also called vetronics), and sensors. Other subsystems included the chassis design. Fig. 9 depicts how theses subsystem can be viewed as "wrapping" around each other. We discuss each subsystem separately.

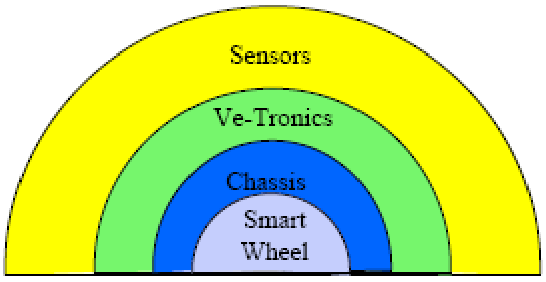


Fig. 9 ODIS components.

Locomotion: Except for unattended ground sensors, most unmanned systems must move. The primary locomotion systems used in most UxVs are:

- UGV: wheels and tracks. UGVs are further classified by their steering type (e.g., Ackerman, skid steer, unicycle, omni-directional, or novel).
- UAV: fixed wing, rotary wing, VTOL.
- USV/UUV: propeller based, jetted.

In general the motion and locomotion aspects of an unmanned vehicle are not remarkably different from those of their manned counterparts. Thus the design of motion and locomotion system becomes "only" an engineering task.

Sometime a novel mechanical concept for locomotion might become the basis of an unmanned system, rather than a motivating CONOPS. For example, Fig. 10 shows the original "smart wheel" concept developed at Utah State University. Also shown is the ODIS version of the smart wheel assembly. This USU-developed mobility capability has two or three independent degrees of freedom: drive, steering with infinite rotation, and height (in some robots). Multiple smart wheels on a chassis creates a "nearly-holonomic" or omni-directional (ODV) vehicle. Fig. 11 shows some of the ODV robots built using the smart wheel.

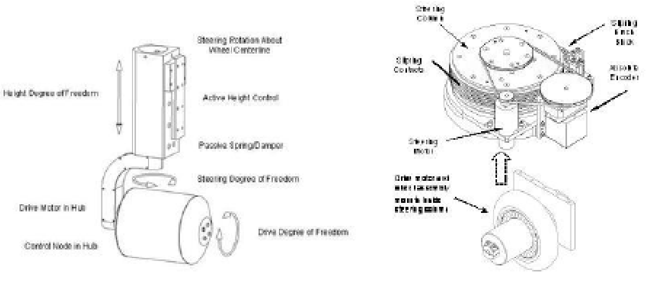


Fig. 10. USU smart wheel.

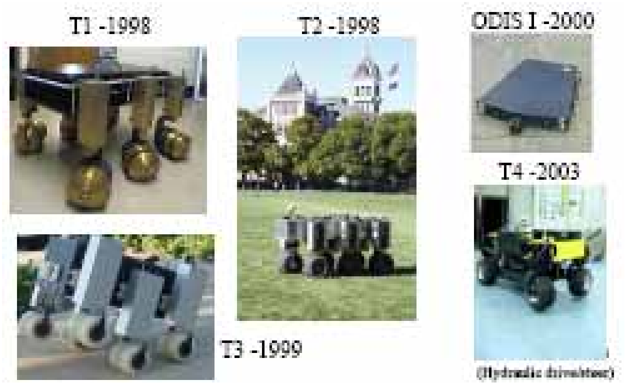


Fig. 11.ODV robots.

Power: Another important consideration in unmanned vehicle locomotion design is the system's power source. Power for UxVs is one of the technology's limiting issues. Many robots, especially in hobbyist or academic and research lab settings, are battery-based systems, using lead-acid, nickel-metal hydride, lithium-ion, or silver-zinc batteries. In larger, more "real-world" systems, however, it is more common to find combustion engine-based power sources, including both gasoline and diesel. Recently there have been examples of fuel cells being used in robots, as well as novel ideas such as the notion of energy scavenging. Another aspect to consider when addressing power is the so called "power budget," which can be obtained by analyzing a power distribution block diagram such as the one for ODIS shown in Fig. 12.

Vehicle electronics and computational capability: When designing a UxV one must identify all computational processes that are required and then allocate these requirements across multiple processors. A critical consideration will be the particular data sources that drive the computational algorithms and the interface requirements associated with those data sources (e.g., RS-232/485 serial,.

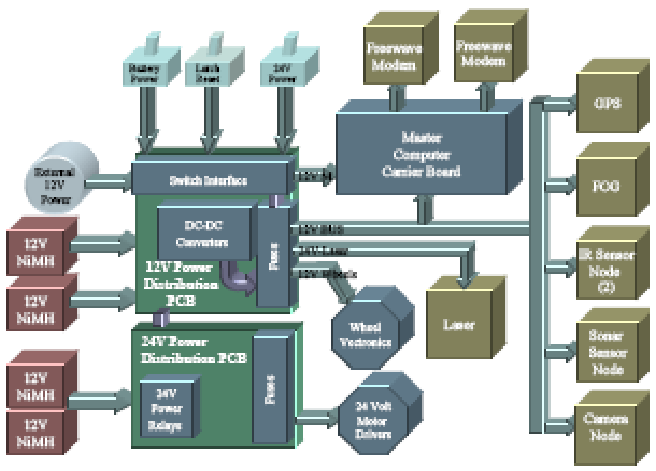


Fig. 12.ODIS power distribution diagram.

CAN, Ethernet, USB, Analog, Digital, etc.). Figs. 13-15 illustrate the vetronics design for the ODIS vehicle, with Fig. 13 showing the high-level block diagram, Fig. 14 showing how to describe the I/O of the master processor, and Fig. 15 showing similar detail for a lower-level wheel node processor

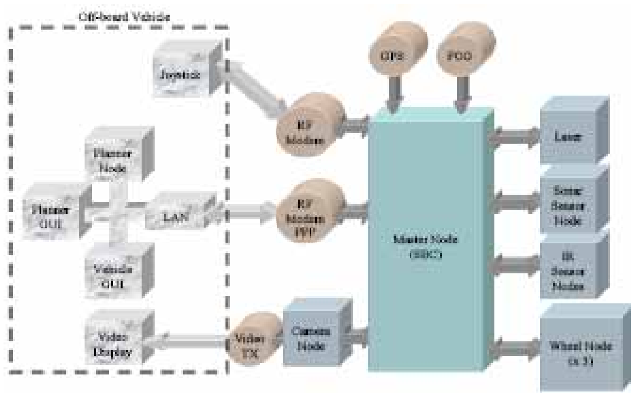


Fig. 13.Vetronics architecture.

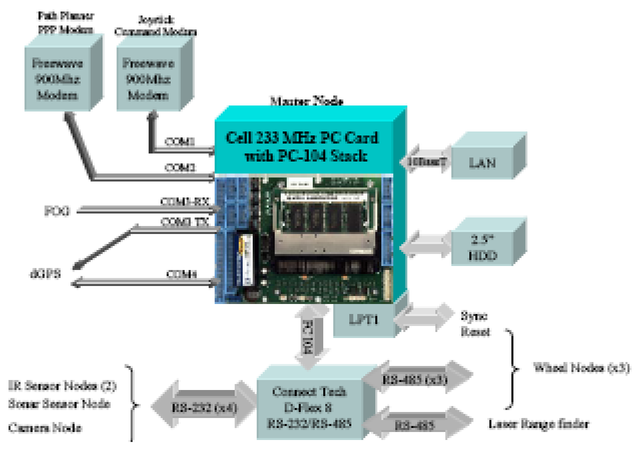


Fig. 14. Master node.

Chassis: UxV design can be a "chicken-and-egg" problem. While on the surface chassis development seems to be a standard mechanical engineering problem, the design cannot

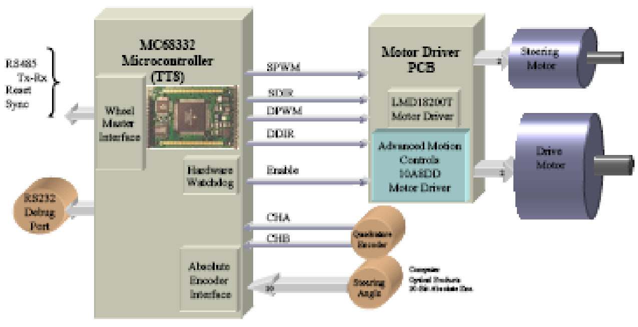


Fig. 15. Wheel node.

be done in a vacuum without consideration of other issues. In particular, the tradeoffs between layout (e.g., vehicle size), vehicle speed/torque, battery capacity, and weight become paramount. A bigger motor give more speed and torque, but takes up more space and requires more battery, which makes the vehicle heavier, which makes it need more power, which requires a bigger motor, which … Clearly all the teams working on the UxV design must collaborate. To this end it is useful to develop layout diagrams such as that shown in Fig. 16 and weight budgets calculations such as shown in Fig. 17. Another important aspect often overlooked is the design and layout of cables. Fig. 18 shows the final ODIS chassis layout.

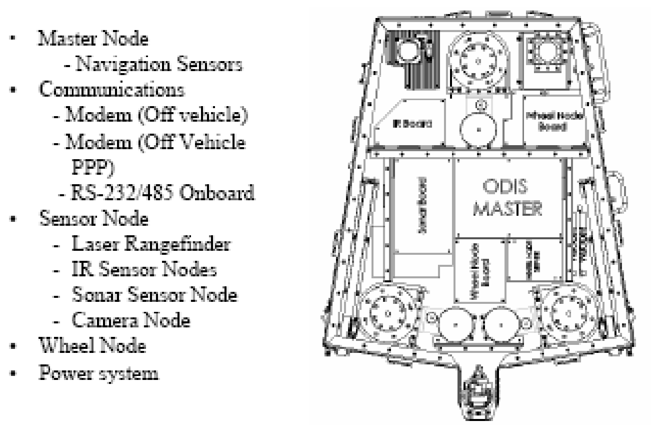
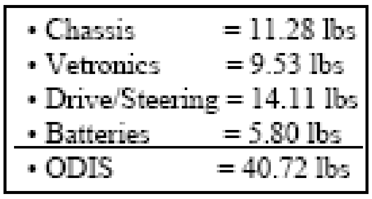

- Master Node
  - Navigation Sensors
- Communications
  - Modem (Off vehicle)
  - Modem (Off Vehicle PPP)
  - RS-232/485 Onboard
- Sensor Node
  - Laser Rangefinder
  - IR Sensor Nodes
  - Sonar Sensor Node
  - Camera Node
- Wheel Node
- Power system

Fig. 16. ODIS vetronics layout.



• Chassis	= 11.28 lbs
• Vetronics	= 9.53 lbs
• Drive/Steering	= 14.11 lbs
• Batteries	= 5.80 lbs
• ODIS	= 40.72 lbs

Fig. 17. ODIS weight budget.

Sensors: The ODIS robot was equipped with a full range of sensors. Fig. 19 depicts the external sensors that ODIS uses to understand its environment. We will not discuss the design and selection of these sensors in detail here. The interested reader can refer to the references in the bibliography. However, we do note two points. First, Fig. 19 and Fig 13 both emphasize that it is common to have separate processing boards for individual sensors. Such a "system-of-systems" approach makes it easy to quickly design new platforms. Second, a useful design technique for sensor selection is to use

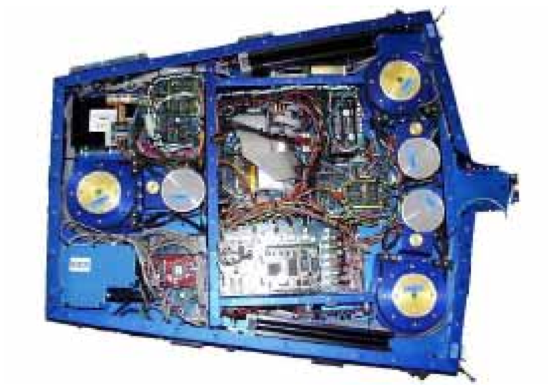


Fig. 18. Final ODIS chassis layout.

what we call a "sensor influence map," which shows the range and coverage of each sensor based on its placement on the robot. Fig. 20 shows the ODIS sensor placement map.

In addition to sensors used to perceive the external environment, UxVs need so-called "properioceptive sensors." These are sensors that allow the unmanned system to measure information about its own internal state, that is, self-status sensors. Such sensors can include:

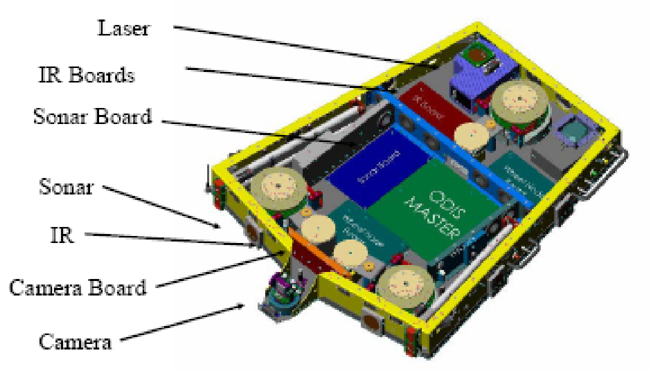


Fig. 19. ODIS external sensors.

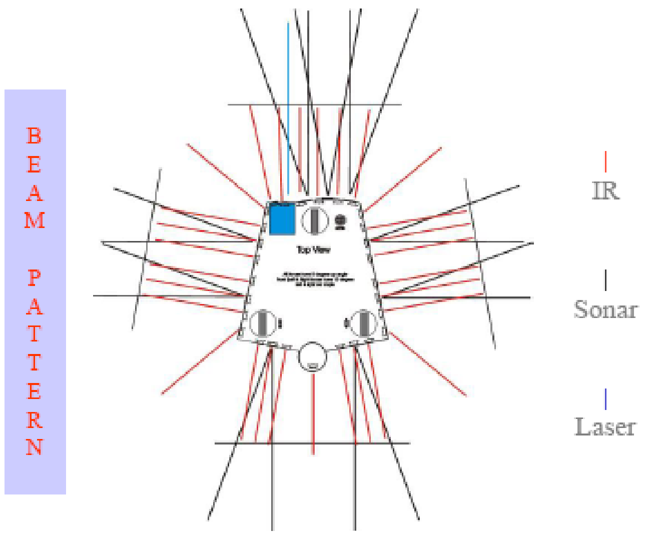


Fig. 20. Sensor influence map for ODIS.

- Localization sensors such as a fiber optic gyro (FOG, similar to a compass) and GPS.
- Sensors to aid in localization, including wheel encoders for odometry or dead-reckoning computations (relative encoders for wheel speed and absolute encoders for steering angle).
- Vehicle health sensors, such as current sensors on each motor, temperature sensors for the motors and inside the chassis, battery voltage sensors, and accelerometers to measure stresses.

Sensors are important for UxVs not only for self-sensing and localization, but also for safety. Taking as an example an autonomous tractor, safety scenarios include stopping the vehicle if:

- Tractor leaves field boundary or deviates from path.
- Unavoidable obstacle within given threshold.
- Communication disrupted or lost.
- GPS dropout corrupts position information.
- Computer failures occur.

Some safeguards to ensure that such vehicle halt actions occur include the use of a sensor suite for detecting vehicle path obstructions, a redundant radio link to protect against wireless communication dropout or corruption, and the use of odometry to complement/supplement GPS. Fig. 21 shows some of the awarenenss issues that must be addressed by the sensor suite.

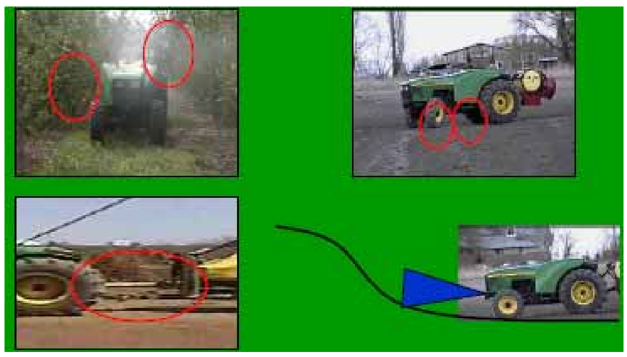


Fig. 21. Safety awareness issues.

In the case of the autonomous tractor, these issues were addressed through the use of a three-tiered proximity detection system, whose sensor influence map is shown in Fig. 22.

We conclude by commenting on localization. Localization (where am I?) is arguably the most fundamental problem autonomous systems. It is defined as the technique through which a robot is able to update its position (and also orientation) in the world. In outdoor unmanned systems, one might suppose that GPS solves the problem. However, it is often the case that GPS signals are lost or denied. Thus, more sophisticated techniques using cameras, maps, and statistical inference strategies, such as SLAM (simultaneous localization and mapping), are active areas of research.

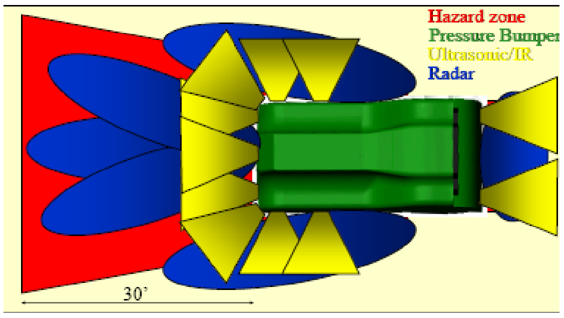


Fig. 22. Proximity detection system.

4.3 Scope Creep and Design Evolution

Inevitably, all projects experience either *scope creep*, in which the customer keeps changing things and/or adding to the requirements, or *product evolution*, in which the system is enhanced or changed in fundamental ways, or both! In the case of the ODIS project, after the autonomous unmanned vehicle was developed, the customer requested a tele-operated-only version (subsequently named the ODIS-T). Later, a different customer requested another full-up autonomous version to be used for semi-autonomous operation (ODIS-S). In the case of the ODIS-T, the desire was to have an unmanned (not autonomous) vehicle that could provide an operator with the benefit of stand-off surveillance without the complexity and reduced capability of autonomy.[3]

4.4 Mission Payloads

It is common to see an unmanned vehicle or autonomous system effort focused on making the vehicle "move by itself" or by remote control. But, in practice, the system is expected to deploy a payload of some type. Such payloads are typically sensors. In the case of the ODIS-T robot, the sensor suites deployed included those used in many unmanned systems:

- Visual – pan/tilt imaging camera
- Passive & active thermal imaging
- Chemical sniffers – i.e. nitrates, toxic industrial chemicals
- Night vision sensors
- Acoustic sensors
- Radiation detectors – i.e. dirty bombs
- Biological agents detection
- MEMS technology – multiple threats
- License plate recognition
- Infrared thermal imager

[3] It is ironic to note that in practice, for UGVs, autonomy does not necessarily imply improved capability, because, although we can design a high degree of inherent capability in the electro-mechanical system, the associated algorithms for perception and decision-making are not yet equally mature.

Fig. 23 shows an infrared thermal image obtained by an ODIS robot. Such a sensor is useful in law enforcement applications.

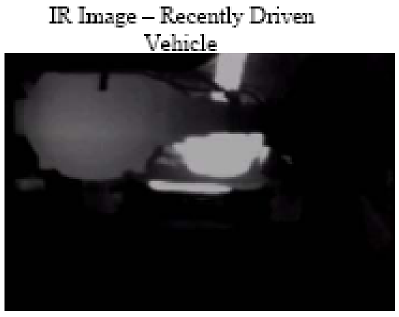


Fig. 23. ODIS IR sensor feedback

Mission payloads can be actuators as well as sensors, such as a fire hose, a robot arm, a cutting tool, etc. Different CONOPS will produce different mission payload requirements. Fig. 24 shows a mobile manipulator based on the ODIS robot.

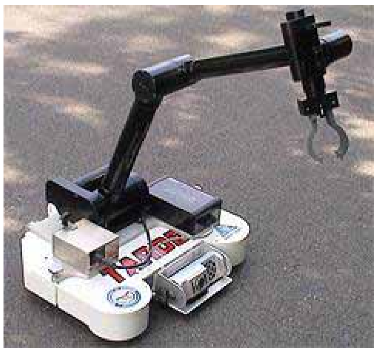


Fig. 24. ODIS mobile manipulator.

4.5 Other UxV Inherent Capability Concepts

There are a number of other concepts related to the inherent capability of UxVs. One concept is the idea of cooperative or coordinated or *swarm* behaviours, such as shown in Fig. 3 above. Another interesting concept is the so-called *marsupial* behaviours. Fig. 25 shows a large robot built to carry the ODIS robot. Notionally, the large robot would patrol a parking area (or be deployed by security guards from a distance), performing coarse-grained inspection, and the ODIS robot would be released from the larger robot to perform fine-grained inspection.

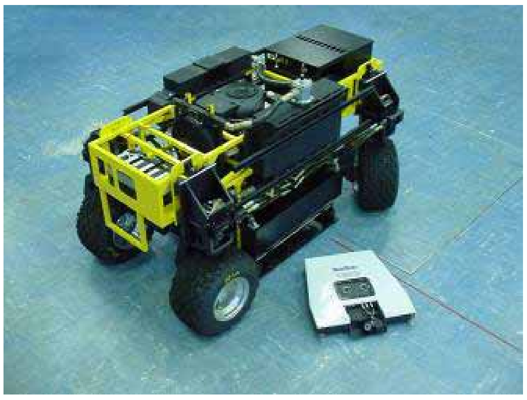


Fig. 25. Marsupial parking lot surveillance system.

Finally, we mention that many times an autonomous system is developed by *retrofitting* existing vehicles, with actuation of steering, braking, drive, etc. This is especially common when dealing with larger automated vehicles (e.g., tractors or construction equipment) to be used by security and law enforcement personnel for fire-fighting, road-block and debris clearing, building breaching, crowd control, explosive ordinance disposal, etc. Typically such vehicles are retrofitted using an "automation kit." This requires adding actuation as well as sensing. Fig. 26 shows how an existing farm tractor was retrofitted with actuators for remote control and autonomy. Fig. 27 shows the sensors.

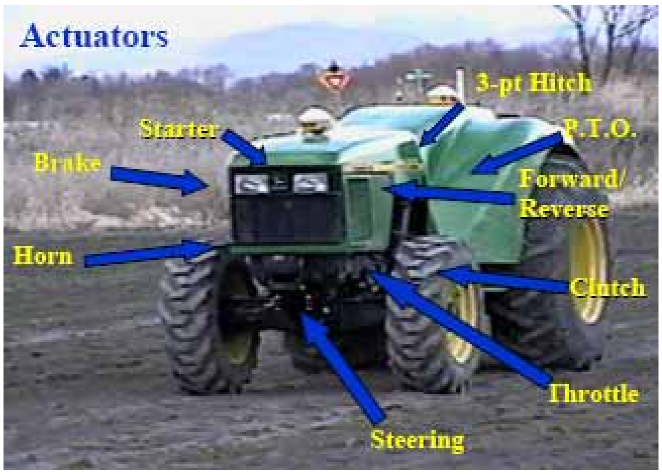


Fig. 26. Actuators used to retrofit a tractor.

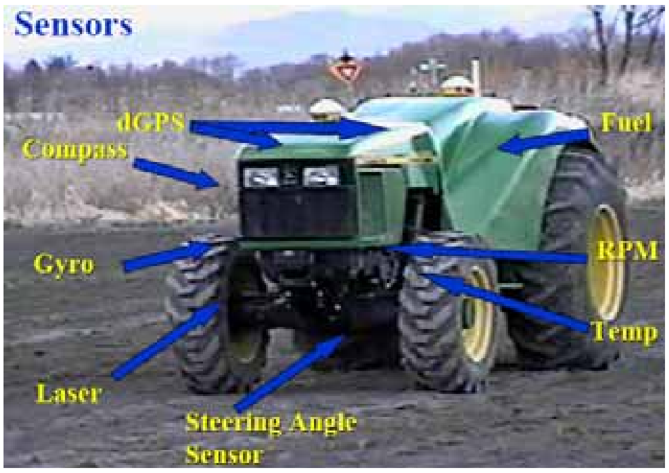


Fig. 27. Sensors used to retrofit a tractor.

5. CONTROL OF UNMANNED SYSTEMS

Once one has in their hands a vehicle with some level of inherent mobility capability and the ability to "fly-by-wire," it is necessary to develop algorithms exploiting that inherent capability. Control and decision algorithms are what transform a collection of wheels or tracks, chassis, electronics, and sensors into a smart, mobile unmanned autonomous vehicle. We consider two aspects of control for autonomy: servo and path-tracking control and intelligent behaviour generation for mission planning and execution, which "wrap around" the inherent capability components as depicted in Fig. 28.

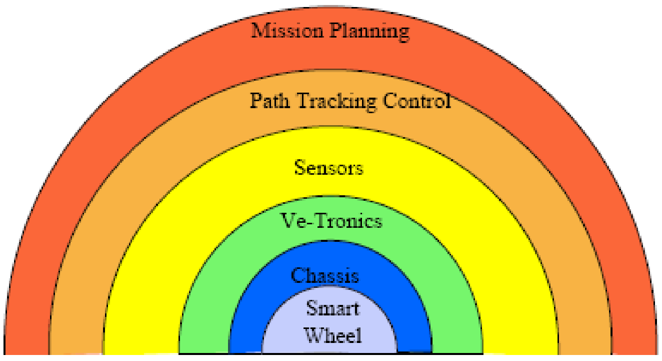


Fig. 28. Control and intelligent behavior generation.

5.1 Architectures

Algorithms for intelligent behaviour generation and control must be organized in a suitable architecture. A large number of architectures and strategies have been proposed for intelligent behaviour generation for UxVs. These include:

- Subsumption (Brooks) – The subsumption architecture uses input-output relations to solve small problems with a prioritization of actuation. For example, a simple autonomous system may have two behaviours: forage or flee. When there is no danger, forage has priority. When danger is present, fleeing has priority. This approach can be called a reactive approach.

- Behavior-based, hierarchical (Arkin) – This architecture uses specific motor schemas for specific behaviours, with higher-level behaviours formed form lower-level behaviors. This is often called a deliberative approach.

- Behavior-based reinforcement learning (Barto/Sutton/ Anderson) – In reinforcement paradigms, the output of a difference engine measuring state-goal mismatch dictates actions to minimize that mismatch. These actions are typically probabilistic.

Of course, one can consider combinations of these architectures. Below we present an architecture that is both deliberative and reactive, where an AI-based planner devises deliberative actions, but reactive actions are possible.

We also note that a number of formal codes have been developed to implement architectures for autonomous systems. These include the 4D/RCD system from the U.S. National Institute of Standards and Technology, the JAUS (Joint Architecture fro Unmanned Systems) from the U.S. DoD (now a developing SAE standard), and a number of industry-specific standards currently under development. (e.g., the US STANAG 4586 for UAVs).

In the remainder of this section we will give an overview of a multi-resolution architecture to implement a "task decomposition" approach to control. As shown in Fig. 29, at the lowest level actuators run the robot. At the middle level the path tracking controllers generate set-points (steering angles and drive velocities) and pass them to the low level (actuator) controllers. At the highest level the mission planner decomposes a mission into atomic tasks and passes them to the path tracking controllers as command-units.

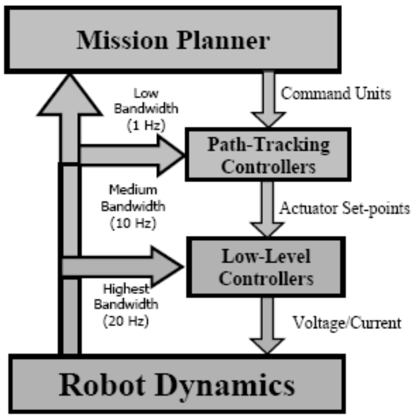


Fig. 29. Multi-resolution architecture.

5.2 Low-Level and Path-Tracking Control

At the lowest level, servo-control and basic kinematics considerations can be used to devise control algorithms to both force the system's actuators to follow desired setpoints and to force the vehicle's body-centered motion to follow a desired path. Additionally, when odometry or GPS and inertial data are available, coordinate transformations can be used to force the vehicle to have a desired open-loop velocity vector in inertial space. To actually track a desired path in inertial space, path tracking strategies can be broadly classified into two groups:

1. Time trajectory-based (temporal), whereby the desired path is parameterized into time-varying set-points. In this case the locus of these set-points follow (in time) the desired trajectory (in space).
2. Spatial-based, whereby the desired path is parameterized as a function of space and all control actions are based on the distance of the vehicle from the desired path. Velocity as a function of path can be included in such algorithms.

We have implemented a variety of each type of controller on our robots. In general, spatial-based strategies are safer, but time-based strategies may be more accurate. For more details please refer to the references.

5.3 Intelligent Behaviour Generation

We classify behaviour generation strategies the evolution of their development. "First Generation" strategies used waypoint navigation, with splines used to define paths between waypoints. This can be viewed as a user-based path generation strategy.

"Second Generation" strategies decompose the path into primitives with fixed input parameters. For example, a command sequence might be:

1. Translate to (20,20) with a velocity of 1.414.

2. Translate to (20,40) with a velocity of 1.

Such a sequence might be implemented via a set of primitives such as those shown in Fig. 30. We refer to such a strategy as open-loop path generation.

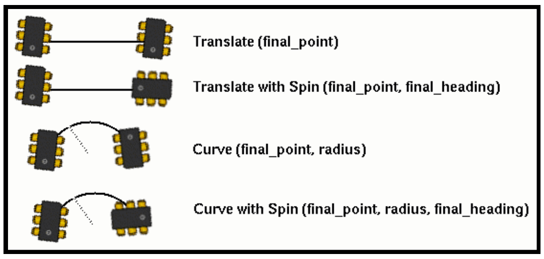

Translate (final_point)
Translate with Spin (final_point, final_heading)
Curve (final_point, radius)
Curve with Spin (final_point, radius, final_heading)

Fig. 30. Motion primitives.

The "Third Generation" of intelligent behaviour generation strategies that we have studied also composes paths into primitives. However, in this case the variable input parameters depend on sensor data. We call this sensor-driven path generation strategy a delayed commitment approach. To see the difference from the second generation strategy, notice that the second generation approach depends upon how accurate your original plan might be. The problem with instantiating a primitive such as that shown in Fig. 30 with actual numbers after a planning activity is that it is not adaptable; it pre-determines the commands based on the information available at planning time. A more adaptable approach is to leave the determination of the input variables until execution time, based on the perceived environment.

To illustrate, consider the problem of controlling ODIS to approach a vehicle in a parking lot from the back and to align itself between the two back wheels. Since the exact location of the car is not known at planning time, a traditional script command specifying exact coordinates will not be possible. However, in the third generation intelligent behaviour strategy, the path for ODIS to take can be specified as

(ALIGN-ALONG (LINE-BISECT-FACE CAR_001) distance)

where "line-bisect-face" is a function that returns a line that bisects the closest face of the object CAR_001 (see Fig. 31). This line will be undefined initially, but once the object CAR_001 has been detected and a spatial model built, the line will be defined based on the perceived position of CAR_001. Here the determination of the path to be followed is delayed until execution time, based on the result of sensing the environment. In this way the paths followed naturally adapt to the local environment encountered by ODIS.

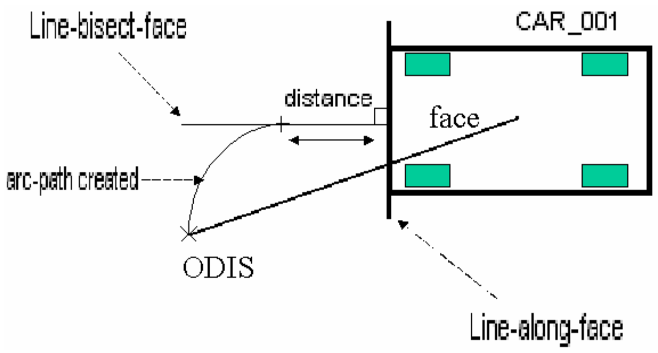


Fig. 31.Delayed-commitment approach.

5.4 Software Implementation

Command Grammar: A collection of commands such as illustrated in Fig. 31 can be viewed as a grammar. In our research we developed a command environment called MoRSE to implement our delayed commitment approach (Mobile Robots in Structured Environments). The language is characterized by

- A high degree of orthogonality with a number of small orthogonal constructs that effectively span the action space of the robot and can be mixed and matched to provide almost any behaviour.
- Variables that include standard integer, floating point, and geometric data types, such as points, lines, arcs, corners, pointsets, etc.
- Data constructs for objects in the environment, to be fit and matched to data.
- Geometric computation functions for building arcs and lines from points, returning points on objects, extracting geometry from environment objects, generating unit vectors based on geometry, fitting functions to turn raw data into complex objects, and vector math

A key feature of MoRSE is the *command unit*. These are a set of individual *commands actions*, which define various vehicle actions that will be executed in parallel, including:

- Commands for XY movement:
  - moveAlongLine(Line path, Float vmax, Float vtrans)
  - moveAlongArc(Arc path, Float vmax, Float vtrans)
- Commands for Yaw movement:
  - yawToAngle(Float angle_I, Float rate)
  - yawThroughAngle(Float delta, Float rate)
- Commands for sensing:
  - SenseSonar
  - SenseIR
  - SenseLaser
  - Camera commands
- A set of rules defining how these commands may be combined:
  - At most one command for XY movement
  - At most one command for yaw movement
  - Only one Rapid-stop command
  - At most 1 of each sense command (laser, sonar, IR)
  - At most 1 command for camera action
  - No XY, yaw movement, and senseLaser commands allowed with Rapid-stop command
  - No yaw movement command when a senseLaser command is used

System Architecture: These command actions and units are then implemented in an architecture designed to allow for the various layers of control and decision-making needed to produce autonomous behaviours. This architecture is shown in Fig. 32. For planning and intelligent behaviour generation, higher-level tasks are defined as compositions of lower-level tasks. In our hierarchy we define

- Mission-level tasks – These are user-defined tasks to describe what the system should do, such as {Move to point}, {Characterize a stall}, or {Inspect a stall}. In our ODIS implementation we had 12 such high-level tasks.
- Tasks and Subtasks – These are variable and planned actions. For instance, {Characterize a stall} might be decomposed into tasks such as "move to the end of the stall" then "move to the desired stall" then identify the bumper", .etc. Subtasks would be a series of actions needed to achieve a task. For example, "identify the bumper" might be decomposed into "rotate through 45 degrees while sensing with sonar" then "fit line to data" then "move to the mid-point of the line at a distance of 50 centimetres."
- Atomic Tasks (scripts), Command Units, and Command Actions – These are the final decomposition, whereby subtasks are broken up into a series of pre-defined actions, such as the "moveAlongLine" command given above.

Deliberative and Reactive Planning: The process by which user-defined mission-level tasks are decomposed into command units and command actions uses AI-based planning techniques which involve defining an appropriate graph structure for the problem and doing an A*-search over the graph. It is beyond the scope of this paper to describe this process. The interested reader is referred to the references for more information. This process, however, is a deliberative planning approach. That is, given a mission-level task, and current information about the world, we define a plan to achieve the task.

Sometimes, however, it is necessary to deviate from the plan by invoking reactive behaviours. In our system, reactive behaviours are induced via several different approaches. These include (see Fig. 33):

- Localization thread: The localization thread compares expected positions to actual sensor data and makes corrections to GPS and odometry as needed. This is particularly important, as ODIS's missions require lots of motion, including rotation. Problems occur in "knowing where we are" can occur due to GPS dropout and drift in fiber-optic gyro data. The two solutions we have used include landmark-based localization using ODIS's laser and wireless visual-servoing using ODIS's camera.

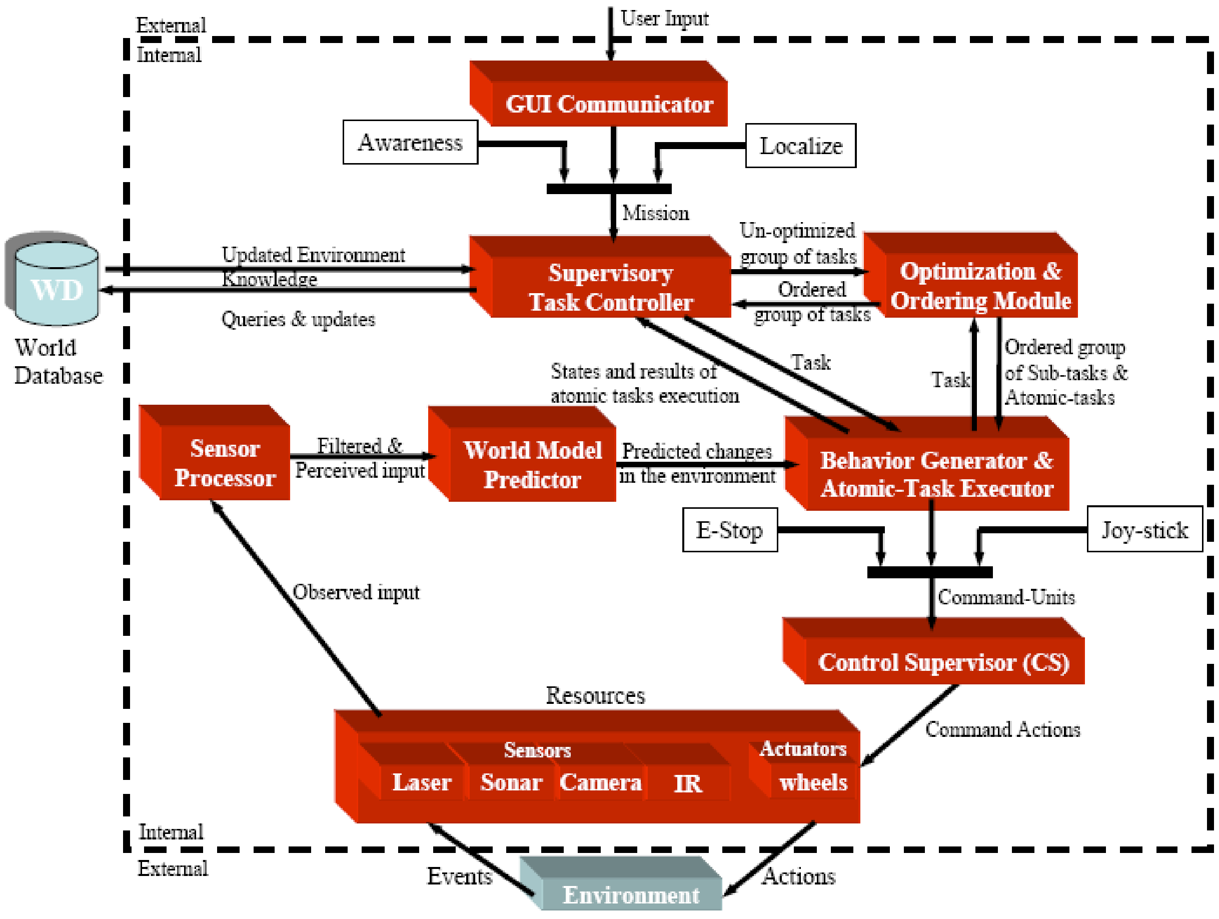


Fig. 32. Complete architecture.

- Awareness thread: The awareness thread interacts with the execution thread based on safety assessments of the environment

- Logic within the execution thread and scripted adaptive behaviours: In our system, exit conditions at each level of the hierarchy determine branching to pre-defined or adaptive actions or to invoke a re-plan event, as shown in See Fig. 34.

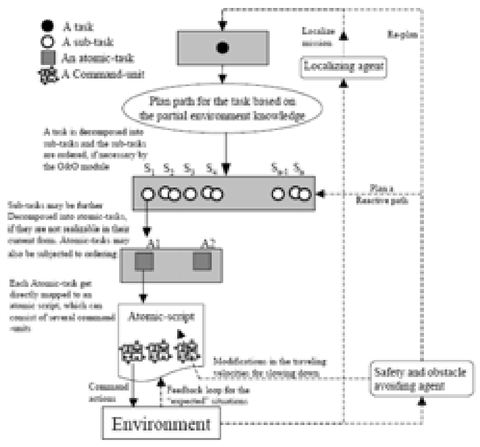

A task
A sub-task
An atomic-task
A Command-unit
Localize mission
Re-plan
Localizing agent
Plan path for the task based on the partial environment knowledge
A task is decomposed into sub-tasks and the sub-tasks are ordered, if necessary by the O&O module
S_1 S_2 S_3 S_4 S_{n-1} S_n
Plan a Reactive path
Sub-tasks may be further Decomposed into atomic-tasks, if they are not realizable in their current form. Atomic-tasks may also be subjected to ordering
A1
A2
Each Atomic-task get directly mapped to an atomic script, which can consist of several command units
Atomic-script
Modifications in the traveling velocities for slowing down
Safety and obstacle avoiding agent
Command actions
Feedback loop for the "expected" situations
Environment

Fig. 33.Reactive behaviors.

6. CONCLUDING COMMENTS

We conclude with a brief observation about future research. We believe that advances in autonomy will require new ways of thinking in the areas of perception, decision-making, and architecture (but not traditional control). We have not addressed perception in this paper and leave this topic for another time. However, regarding architectures, it is our view that new paradigms for architectures are needed. In particular we believe that the discrete-event dynamic system perspective

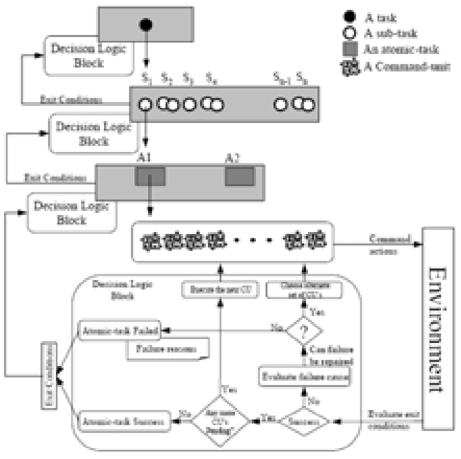


Fig. 34.Feedback during the execution thread.

for developing a formal approach to grammar-based strategiescan be useful. From this point of view, the mobile robot behaviour generator can be interpreted as a discrete-event dynamic system (DEDS) as depicted in Fig. 35. In this interpretation commands and events are symbols in an alphabet associated with a (regular) language. In this formalism our goal is to develop methods for the automatic generation of scripts. Suggested approaches for synthesis include Petri nets and recent results on controller design for finite state machine model matching. This approach can also allow for feedback as a natural aspect of deliberative behaviour, such as the our use of feedback during the execution thread in our MoRSE system depicted in Fig. 34. Finally we note that the role of memory in intelligence needs to be understood better, as does the idea of adaptable architectures that can allow "growth" of knowledge (e.g., evolution of language), as well as other biologically-inspired modelling and architecture ideas such as swarms (e.g., bacterial foraging, ants/bees, etc. – more later), multi-agent systems, self-organization, complex adaptive system (e.g., membrane formation), and cybernetic viewpoints.

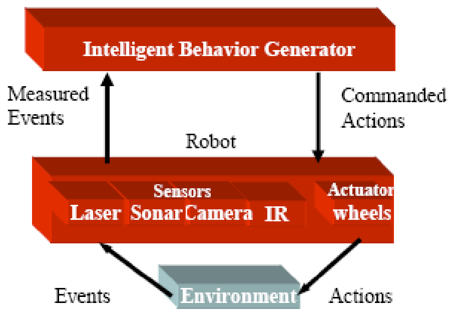


Fig. 34.DEDS view of an autonomous mobile robot.

7. CITATIONS

Our approach to citation in this paper is unconventional. The information presented has been based on the author's knowledge accumulated over a ten-year period on a number of projects. We have chosen not to give individual references. Instead, we indicate here several representative publications. A complete list of publications related to the projects and topics described in the paper, including many downloadable links, can be found at http://www.csois.usu.edu/.

A basic overview of the ODIS concept can be found in:

Flann, N.S., Moore, K.L., and Ma, L. (2002). A small mobile robot for security and inspection applications in *Control Engineering,* Vol. 10, No. 11, Nov. 2002, pp. 1265-1270.

An overview of the ODIS vehicle design and systems engineering can be found in:

Moore, K.L, Flann, N.S., Rich, S., Frandsen, M, Chung, Y-C., Martin, J, Davidson, M., Maxfield, R, Wood, C. (2001). Implementation of an Omnidirectional Robotic Inspection System (ODIS) in *Proceedings of SPIE Aerosense 2001,Conference on Unmanned Robotic Vehicles*, Orlando, FL, April 2001.

A general overview of our overall ideas on architecture as well as system engineering can be found in:

Moore, K.L, Flann, N.S. (2000). A Six-Wheeled Omni-directional Autonomous Mobile Robot in *IEEE Control Systems Magazine,* Special Issue on Mobile Robotics, vol. 20, no. 6, pp. 53-66, December 2000.

Our ideas on intelligent behaviour generations, including the delayed commitment approach have appeared in many papers. The following are representative:

Bhal, V., Moore, K.L. (2003). Multi-Robot Autonomous Parking Security System in *Proceedings of the 18th IEEE International Symposium on Intelligent Control,* IEEE ISIC'03,Westin Galleria Houston, Texas, October 5-8, 2003.

Flann, N.S., Davidson, M., Martin J., Moore, K.L., (2001). Intelligent Behavior Generation Strategy for Autonomous Vehicles Using a Grammar-Based Approach in *Proceedings of 3rd International Conference on Field and Service Robotics FSR2001*, Helsinki University of Technology, Otaniemi, Espoo, Finland June 11 -13, 2001.

Moore, K.L., Flann, N.S., (1999). Hierarchical Task Decomposition Approach to Path Planning and Control for an Omni-Directional Mobile Robot in *Proceedings of 1999 IEEE International Symposium on Control, Intelligent Systems and Semiotics,* pp. 302-307, Cambridge, MA, Sept. 1999.

Control strategies developed in our research have been detailed in the following:

Davidson, M., Bahl, V. (2001). The Scalar E-Controller: A Spatial Path Tracking Approach for ODV, Ackerman, and Differentially-Steered Autonomous Wheeled Mobile Robots in *IEEE International Conference on Robotics and Automation*, Seoul, Korea, May, 2001.

Moore, K.L., Davidson, M., Bahl,, V., Rich, S., Jirgal, S. (2000) Modelling and Control of a Six-Wheeled Autonomous Robot in *Proceedings of 2000 American Control Conference*, pp. 1483-1490, Chicago, Illinois, June 2000.

ACKNOWLEDGEMENTS

The author would like to thank all his various colleagues and co-authors from the past ten years who have worked with him in the area of autonomous systems, especially Robert Gunderson, Nicholas Flann, YangQuan Chen, Carl Wood, and Don Cripps of Utah State University, Mel Torrie and Matt Berkemeier of Autonomous Solutinos, Inc., Gordon Olsen of VPI and Monte Frandsen and Morgan Davidson of USU's Space Dynamics Lab, and Vikas, Bahl, Lili Ma, Hitesh Shah, and Zhen Song.