

Safe and effective navigation of autonomous robots in hazardous environments

Derek Seward · Conrad Pace · Rahee Agate

Received: 22 May 2006 / Revised: 28 June 2006 / Accepted: 10 July 2006 / Published online: 8 September 2006
© Springer Science + Business Media, LLC 2006

Abstract The development of autonomous mobile machines to perform useful tasks in real work environments is currently being impeded by concerns over effectiveness, commercial viability and, above all, safety. This paper introduces a case study of a robotic excavator to explore a series of issues around system development, navigation in unstructured environments, autonomous decision making and changing the behaviour of autonomous machines to suit the prevailing demands of users. The adoption of the Real-Time Control Systems (RCS) architecture (Albus, 1991) is proposed as a universal framework for the development of intelligent systems. In addition it is explained how the use of Partially Observable Markov Decision Processes (POMDP) (Kaelbling et al., 1998) can form the basis of decision making in the face of uncertainty and how the technique can be effectively incorporated into the RCS architecture. Particular emphasis is placed on ensuring that the resulting behaviour is both task effective and adequately safe, and it is recognised that these two objectives may be in opposition and that the desired relative balance between them may change. The concept of an autonomous system having “values” is introduced through the use of utility theory. Limited simulation results of experiments are reported which demonstrate that these techniques can create intelligent systems capable of modifying their behaviour to exhibit either ‘safety conscious’ or ‘task achieving’ personalities.

Keywords Autonomous vehicles · Safety · Risk analysis · Unstructured environments · Task effective · Partially observable Markov decision processes · Real-time control system · Robot architecture

1 Background

The advent of fast, accurate and reliable differential GPS systems for the location of construction plant provides the opportunity for significant advances in the autonomous movement of such plant around construction sites. Such technology was demonstrated as long ago as 1996 by Caterpillar with its autonomous mining truck (AMT) (Geske, 2004), however this has still not become a commercial reality. The main reasons for this lack of success in the market are lingering doubts about both safety and economic effectiveness. This is despite the fact that this particular application was designed for use on relatively simple haul roads in large mines which are both isolated and largely free from other traffic.

One of the first applications of GPS for the real-time control of heavy plant involves the positioning of a large piling rig (Seward et al., 1996), however, largely because of safety concerns, no attempt was made to make the vehicle driverless during significant movements of the vehicle. Current work at Lancaster University is centred around LUCIE, the Lancaster University Computerised Intelligent Excavator – Fig. 1. This is a robot excavator that has demonstrated effective autonomous trenching in varying soil conditions, however, the focus of this paper concerns the work to achieve safe navigation of the excavator platform around an unstructured and dynamically changing construction site.

D. Seward (✉) · R. Agate
Lancaster University
e-mail: d.seward@lancaster.ac.uk

C. Pace
University of Malta



Fig. 1 Robot excavator under ‘training’

As robots evolve into systems with enhanced mobility and autonomy, their operational safety concerns increase and change in nature. Advanced mobile robots operating in dynamic and unstructured environments are unlikely to be able to rely on substantial robot isolation or motion constraints to ensure adequate safety, having to rely more on their adaptation to the environment to maintain safety. Thus behaviour generation will have to carry most of the ‘responsibility’ to maintain safe operation, owing to the undesirability of restricting availability through excessive environmental and operational constraints. Consequently, the onus on operational safety assurance will shift from an approach that constrains the system to an approach more fundamentally concerned with the robot’s behaviour. This brings about various new challenges in developing safe robotic systems.

One challenge is that the priorities of the robot may change depending upon its role and task. For example under normal commercial conditions ‘values’ such as *safety* and *cost efficiency* may dominate whereas for a military robot, under battlefield conditions, *task achievement* and *self-preservation* may be more important. Relative priorities may vary from day-to-day even for the same system.

Figure 15 shows an example of a simulated construction site with hazards. The object of this work is to develop a

strategy, and a means of implementation, that will enable an autonomous vehicle to move from the start point to a nominated target point with behaviour appropriate to its designated ‘values’.

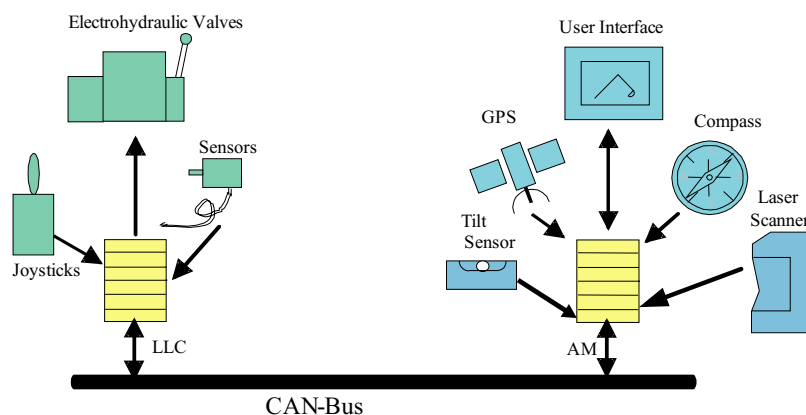
2 Excavator hardware architecture and sensors

The hardware platform used for this work was a standard JCB 801 tracked mini-excavator modified in the following ways:

- The standard manually operated hydraulic valves were replaced by a proportional electro-hydraulic system. For accuracy in control these are have ‘closed-centre’ spools. The problem of dissipating excess energy during periods of low activity was solved by replacing the existing fixed displacement hydraulic gear pumps with a single variable displacement swash-plate pump.
- Each joint in the excavator arm plus the slew and tracks were fitted with simple potentiometer sensors, interfaced to an analogue interface card, to provide closed-loop feedback control (Fig. 2).
- A two axis tilt-sensor was added to indicate the slope of the vehicle platform (pitch and roll)
- A Rotoscan scanning laser sensor was used to detect objects within the vicinity of the excavator (Seward et al., 2002)
- A Trimble differential satellite Global Positioning System (GPS) was added with an antenna on the roof of the vehicle. This can provide X-Y position to an accuracy of about 25 mm at an update rate of 5 Hz. Elevation data (Z) is less accurate at about 50 mm.
- A standard flux-gate compass was used to give orientation data. Such data can be obtained from the GPS when the vehicle is moving but not when the vehicle is turning-on-the-spot. (A more expensive alternative is to adopt two on-board GPS systems or multiple antennas).
- Two on-board compact PC104 computers, communicating via a CAN bus (Fig. 3), were added to perform the following functions:

Fig. 2 Hardware components



Fig. 3 Architecture for robot excavator

- The Activities Manager (AM) is responsible for tactical decision making during both autonomous digging and travelling. Behaviour is adapted in response to both sensors of the external world (exteroceptive) and internal performance monitoring (proprioceptive). This paper is largely concerned with the operation of this module.
- The Low-level Controller (LLC) controls the feedback loop formed by the joint/track sensors and the electrohydraulic valves to facilitate the desired movement of the arm or vehicle platform. It thus manages servo-control of the individual joints in addition to kinematic control of the entire arm or platform. Smooth and accurate control has been achieved using a technique known as Proportional Integral Plus (Ghu et al., 2004), but this is not the focus of this work.

3 Robot architectures

Architectural structure can be defined as the method by which a system is divided into subsystems, and how those subsystems interact with each other and with the external world to reach the set goal. A system ‘architecture’ primarily refers to the software and hardware framework for controlling the system. Architectural style refers to the computational concepts that underlie a given system. In the last decade, a number of system architectures have evolved (Arkin, 1998).

Architectural styles described in the technical literature can be classified into three categories: deliberative (characterised by top-down, hierarchical and model based control), reactive (characterised by simple sense-act behaviours), and hybrid. Hybrid style combines both reactive and deliberative control in a heterogeneous architecture. It facilitates the design of efficient low-level control with a connection to high-level reasoning. A problem is that there are about as many different intelligent robot architectures as there are robots, resulting in a lack of possibilities for reuse and collaboration. An attempt to produce a universal architecture is RCS. RCS (Real-time Control System) (Albus and Meystel, 2001) is a hybrid architecture in that it combines deliberative with reac-

tive components. The approach was developed at the N.I.S.T. (National Institute of Standards and Technology) laboratory in the USA and has been applied to a wide range of intelligent systems. The adoption of an existing framework such as RCS means that there is more chance of reusability of modules across different systems. In addition there have been moves to generate an RCS communications protocol and to create software design tools to facilitate system development.

3.1 Real-time control system-reference model architecture (RCS-RMA)

RCS partitions the control problem into four basic functional elements:

- behaviour generation (BG),
- world modelling (WM),
- sensory processing (SP),
- and value judgement (VJ),

along with sensors and actuators. RCS clusters these elements into computational nodes that have responsibility for specific subsystems and arrange these nodes in hierarchical layers such that each layer has characteristic functionality and timing. Each layer provides a rich and dynamic world model and a sensory processing hierarchy to keep the world model up to date. Each layer provides a mechanism for integration of deliberative (planning) and reactive (feedback) control. The RCS reference model architecture has a systematic regularity and recursive structure expressed in a canonical form that provides a basis for an engineering methodology (Albus and Meystel, 2001).

The Functional Elements perform the following functions (Albus and Meystel, 2001; Albus, 1991):

Value Judgment (VJ)

It is a process that

1. Computes cost, risk, and benefit of actions and plans.
2. Estimates the importance and value of objects, events, and situations.

3. Assesses the reliability of information.
4. Calculates the rewarding or punishing effects of perceived states and events.

World Modelling (WM)

Is a functional process that constructs, maintains, and uses a world model knowledge database (KD) in support of behaviour generation and sensory processing.

World modelling performs four principle functions:

1. It predicts (possibly with several hypotheses) sensory observations based on the estimated state of the world. Predicted signals can be used by sensory processing to configure filters, masks, windows, and schema for correlation, model matching, recursive estimation, and focusing attention.
2. It generates and maintains a best estimate of the state of the world that can be used for controlling current actions and planning future behaviour. This best estimate resides in a knowledge database describing the state and attributes of objects, events, classes, agents, situations, and relationships. This knowledge database has both iconic and symbolic structures and both short and long-term components.
3. It acts as a database server in response to queries for information stored in the knowledge database.
4. It simulates results of possible future plans based on the estimated state of the world and planned actions. Simulated results are evaluated by the value judgement system to select the best plan for execution.

Behaviour Generation (BG)

Performs the planning and control of actions intended to achieve or maintain behavioural goals.

Behavioural goal: a desired result that a behaviour is intended to achieve or maintain.

Desired result: a result that value judgment evaluates as desirable or beneficial.

Command: a name, a commanded action, and a command goal. Both commanded action and command goal may include parameters.

Sensory processing (SP)

Is a set of processes by which sensory data interact with prior knowledge to detect and recognize useful information about the world.

Figure 4 illustrates the relationships within a single node of the RCS architecture. Each node contains BG, WM, SP, and VJ processes, plus a knowledge database (KD). Any or all of the processes within a node may communicate with an operator interface. The interconnections between sensory processing, world modelling, and behaviour generation close a reactive feedback control loop between sensory measurements and commanded action. The interconnections between behaviour generation, world modelling, and value judgment enable deliberative planning and reasoning about future actions. The interconnections between sensory processing, world modelling, and value judgment enable knowledge acquisition, situation evaluation, and learning.

Within sensory processing, observed input from sensors and lower level nodes is compared with predictions generated by world modelling. Differences between observations and predictions is used by world modelling to update the knowledge database.

Each RCS node looks upward to a higher level node from which it takes commands, for which it provides sensory information, and to which it reports status. Each node also

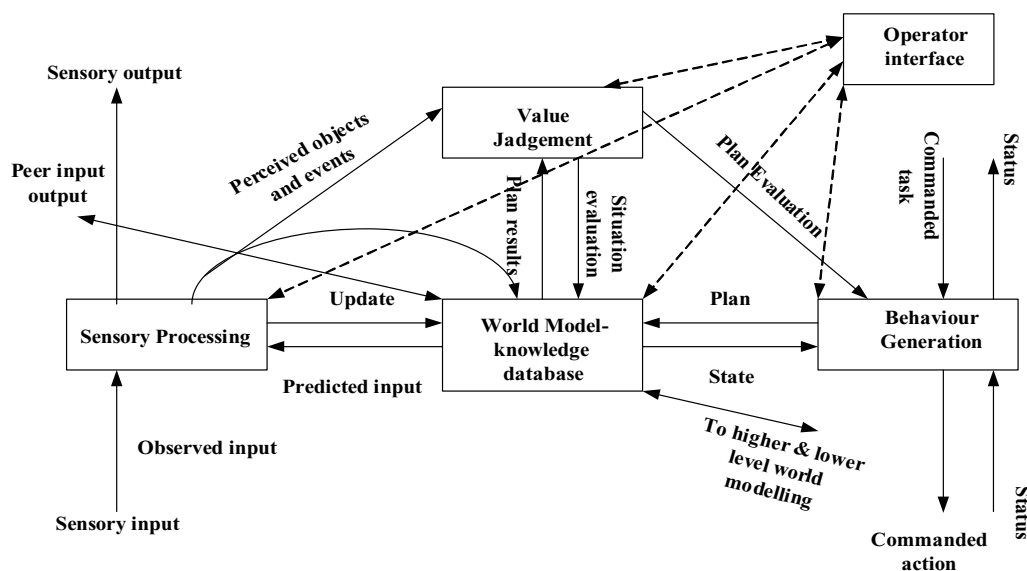


Fig. 4 An RCS node (Albus and Meystel, 2001)

looks downward to one or more lower level nodes to which it issues commands, and from which it accepts sensory information and status. Nodes may also communicate with peer nodes with which it exchanges information. A collection of RCS computational nodes such as illustrated in Fig. 4 can be used to construct a distributed hierarchical reference model architecture.

Each node acts as an operational unit in an intelligent system. Depending on where a particular node resides in the hierarchy, it might serve as a controller for one or more actuators, a subsystem, an individual machine, a group of machines comprising a manufacturing workstation, a group of workstations comprising a manufacturing cell, or a group of cells comprising a manufacturing shop. The functionality of each node can be implemented by a set of software processes or by a person or group of persons.

Figure 5 shows the organizational hierarchy in more detail; it illustrates both the hierarchical and horizontal relationships involved in the proposed architecture. Commands and status feedback flow hierarchically up and down a behaviour-generating chain of command. Sensory processing and world-modelling functions have hierarchical levels of temporal and spatial aggregation.

This is not a hierarchy in a classical sense of the term. The classical representation of a hierarchy refers to a tree-architecture, this is a multiscale organization in which vertical branches have horizontal connections.

The architecture is horizontal in that data are shared horizontally between heterogeneous modules at the same level. At each hierarchical level, the architecture is horizontally interconnected by wide-bandwidth communication between BG, WM, SP, and VJ modules in the same node, and between nodes at the same level, especially within the same command subtree. The horizontal flow of information is vo-

luminous within a single node but less between related nodes in the same command subtree. It has relatively low bandwidth between computing modules in separate command subtrees.

The volume of information flowing horizontally within a subtree may be orders of magnitude larger than the amount flowing vertically in the command chain. The volume of information flowing vertically in the sensory processing system can also be very high, especially in the vision system.

3.2 Implementing RCS-RMA in LUCIE

The first step is to perform a *task decomposition analysis* (Albus and Meystel, 2001) of the physical system to be controlled. In other words, the designer must identify the tasks or operations that this system performs and which task is performed by which actuator(s) or subsystem(s). The task knowledge is represented in the form of a hierarchical task decomposition where higher-level tasks are decomposed into lower-level subtasks (i.e., where the execution of a sequence of subtasks results in achieving the high-level task). Figure 6 shows the Organizational Hierarchy based on task decomposition analysis of an Excavator.

The next step is *defining the controller architecture*. The first and most important thing to consider is the layout of the physical subsystems and all the actuators and sensors of the system to be controlled. Each subsystem will have its own sensors and actuators (although these won't be unique to a specific subsystem). Then, based on the physical layout of the subsystems, the connections between them, the information flow, and the task decomposition analysis performed in the previous step, we define the controller architecture. This typically starts with assigning a control

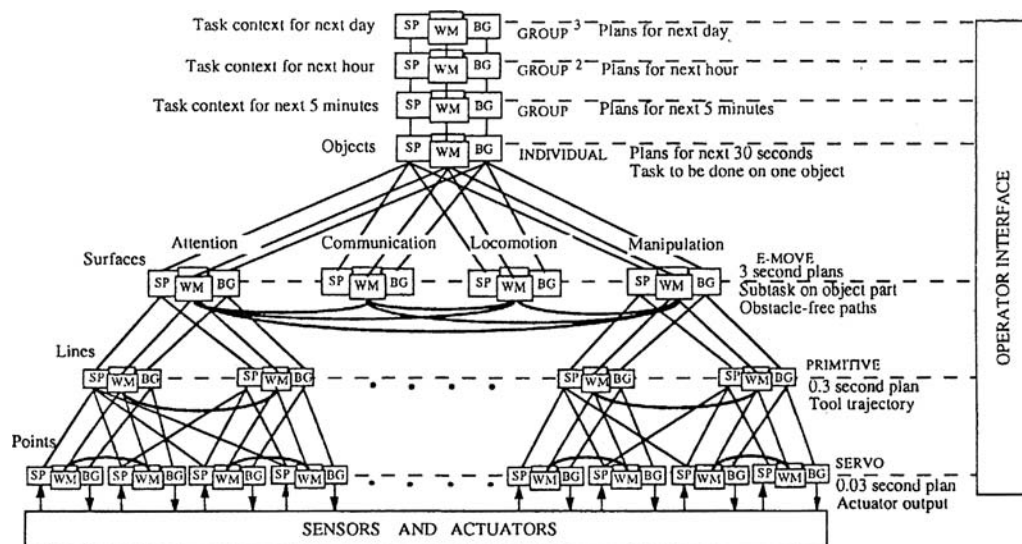


Fig. 5 A RCS reference model architecture for intelligent systems (1991)

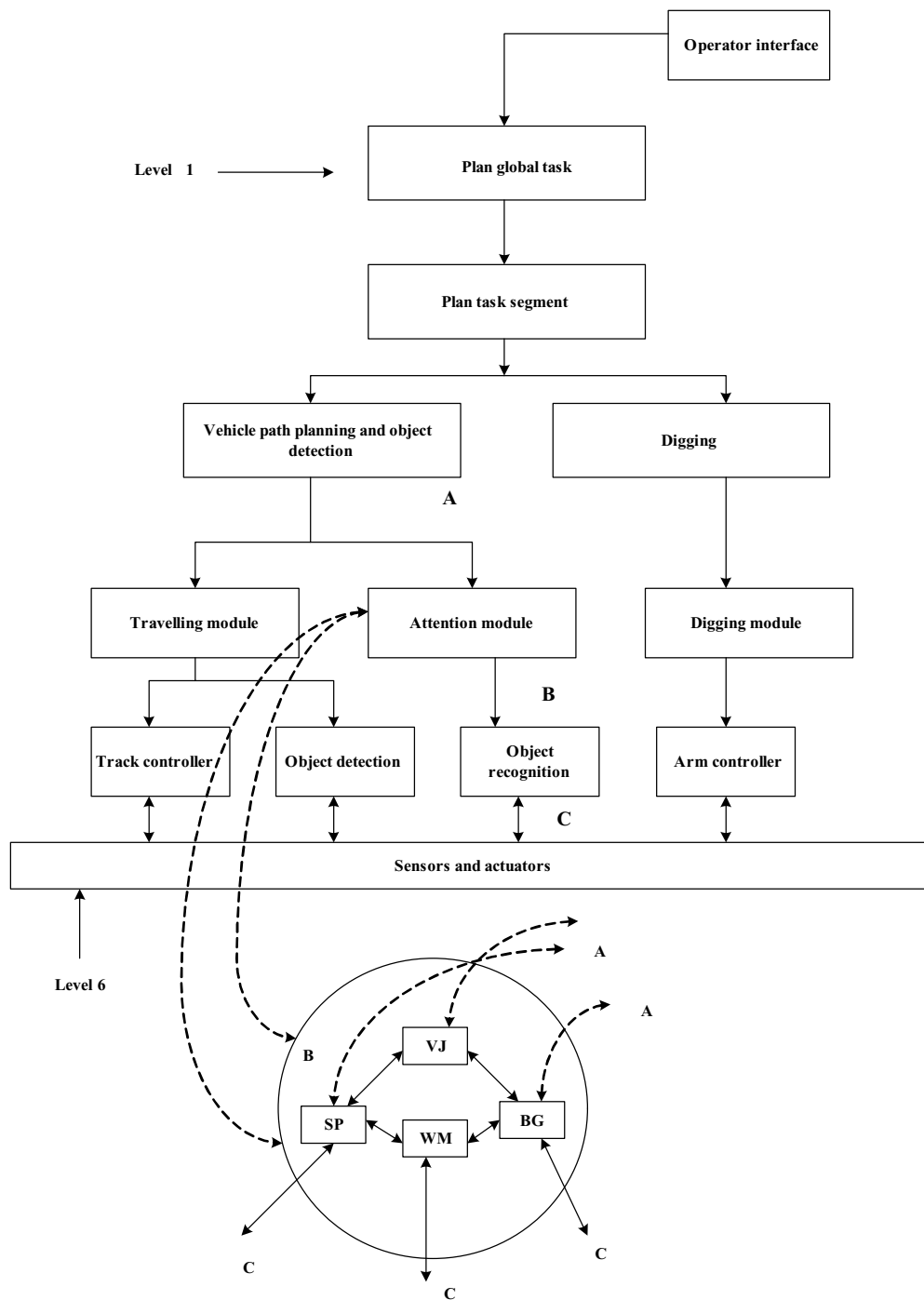


Fig. 6 Organizational hierarchy showing internal structure of one of the modules

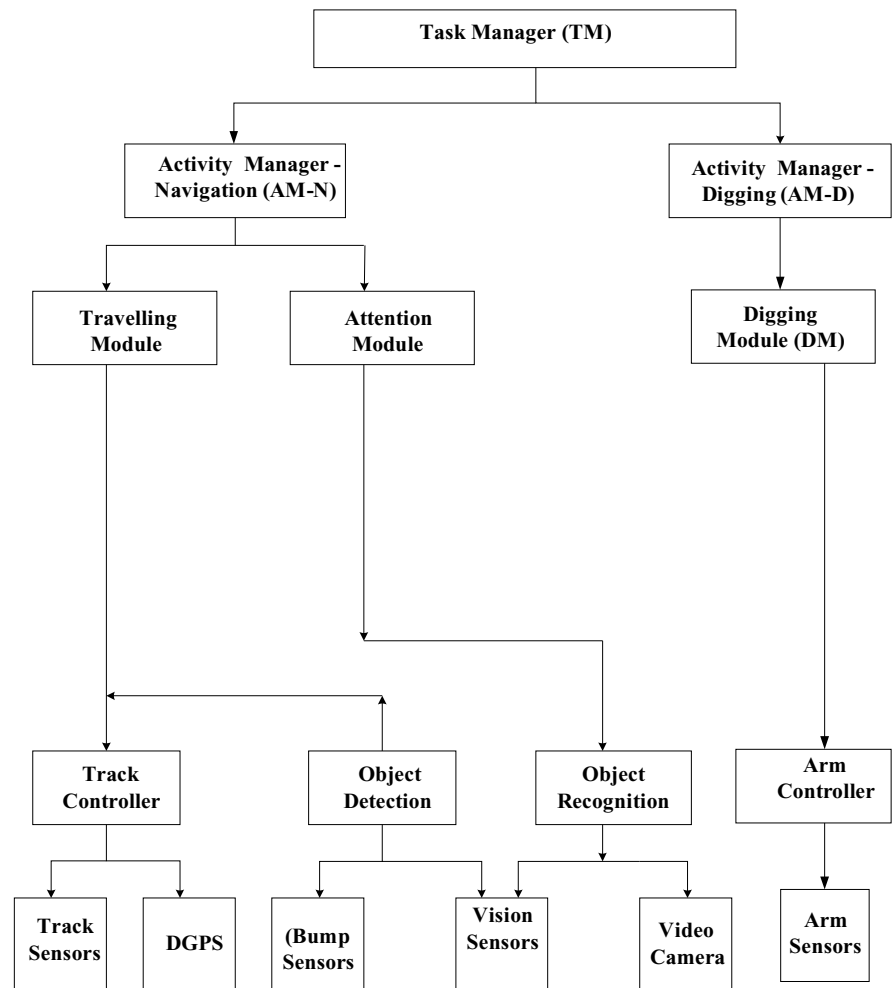
module to each actuator and sensor on the bottom of the hierarchy.

Figure 7 shows the 5-level hierarchy of the controller residing in mobile robot excavator. There is one or more number of modules at each level. The Task Manager Module (TMM) together with the Activity Manager-Navigation (AM-N) and Activity Manager-Digging (AM-D) represents

the first 3 levels of the organizational hierarchy shown in Fig. 6. The Attention Module (AM), Travelling Module (TM) and Digging Module (DM) correspond to the fourth level of the controller hierarchy.

As there are three independent operations of Attention, Travelling and Digging that are performed by three different subsystems at level 3 of the Controller Hierarchy, the cor-

Fig. 7 Controller architecture



responding Activity Manager is assigned as a coordinator to each of these subsystems at one level higher in hierarchy as shown.

The TM, AM and DM will play the role of carrying out decisions made by the respective Activity managers. But the actual task of controlling the vehicle motion is performed by Track Controller. Also Object detection takes place in Object Detection module OD. The Object recognition module OR is responsible for recognising the shape, size and type of obstacle. The excavator arm is controlled by the arm controller AC. This is actually level 2 of the controller hierarchy. This level along with the sensors can also be referred to as the low-level controller.

3.3 Considering safety in RCS

“Although RCS is reasonably generalised and multi-applicational system architecture, there are certain areas, which are in the development stage. It does not take into consideration safety in general.”

“There are several issues which need to be taken into account when analysing a system’s safety, particularly if the

system operates autonomously in an unpredictable and uncontrolled environment. The term safety is relative. There is no absolute safety. Due to this inability in assuring complete safety, it is necessary to secure a reliable degree of protection which fits the circumstances in which the application under development is being used (Pace, 2004)”.

The following questions in relation to embedding safety within RCS were considered:

1. Should safety be an additional value in the value judgement of the RCS? The value judgement parameter can take the form of a risk measurement (in the field of risk analysis, measurable risk values are considered so theoretically there are ways of quantifying the risk)
2. How would a safety value judgement be processed?—Potentially through the evaluation of possible accident risks for the proposed plans (i.e. the value judgement process is nothing more than a risk assessment process for the proposed plans with the resulting ‘value’ being the risk measure).
3. What are the implications of including safety in decision-making?—There will need to be some form of evaluation

of the dependability of the world model created as well as the sensory processing (i.e. how sure can one be that the world model and the system perceptions are a true reflection of reality).

4. How are risk measures integrated with decision-making? How can other values such as task achievement be considered by VJ and how does BG employ these values to generate actions?
5. How do the architectural layers influence safety management? Essentially this means operational risks can be considered on various time and spatial scales. The advantage is that the architecture layers act as a form of ‘safety firewall’ as decisions and actions are passed down the architecture, where each level has a role to play in eliminating hazards to which the robot is exposed. This is a very important issue and integral to the RCS as well as to the philosophy of considering safety throughout, planning and execution (which substantially departs from many approaches which only consider safety at the level of direct interaction with the environment).

The paper returns to these five questions later, however it is first necessary to gain a better understanding of the safety issues.

4 Implementation issues for ensuring safe behaviour

The distinctive features of mobile construction robotic systems when compared to conventional industrial systems, require an appropriate and systematic investigation and analysis of the system operation from a safety perspective. Since advanced robotic system behaviour is founded on the implementation of a computer based architecture, such systems fall within the definition of programmable electronic systems (PES) and thus potentially require a similar development strategy from a safety perspective (National Advanced Robotics Research Centre, 1992). Gaskill and Went (1996) highlights the complexity of robotic systems and the correspondence of such systems to PES’s, indicating once more the relevance of PES standards for managing the robotic system’s safety. The basis of such approaches is an adequate hazard analysis as the cornerstone to system safety.

4.1 Influential factors on robotic system safety—Environmental interaction and uncertainty

The focus here is on operational safety, and thus hazards originating from the interaction between robot and environment are of major interest. Hence a thorough definition of the interaction between robot and environment is necessary for defining any potential operational risks.

Defining the interaction between robot and environment is highly complex specifically owing to the unstructured nature of the environment and the inherent flexibility in the motion of the robot and its possible behaviour (Seward et al., 2000). The problem is further aggravated by the relatively high complexity of autonomous robotic systems in the way that they generate their own behaviour. It has been noted (ref 91) that ‘robots capable of a high degree of adaptability and performing a wide range of tasks, will also be capable of many modes of failure’ potentially requiring novel perspectives to the assurance of safety. Such complexity underlines the potential inability to carry out a full and complete analysis of the system-environment interaction and furthermore, in developing a system that can be verified to act safely in such interaction under all possible circumstances that may be encountered. Consequently, there is likely to be an innate uncertainty concerning the safe behaviour of such autonomous robots.

In the application of autonomous mobile robots uncertainty in the interaction of the robot with its surroundings potentially arises from various sources and at various stages in development. In [91] a number of implications regarding the development of such systems that highlight the problems related to uncertainty from a safety perspective are given, these being:

1. The need for flexible and dynamic interaction with the real world which requires real-time reactive abilities from the system. This however highlights the problem of predictability, considered to be a key attribute for safety.
2. The heavy dependence on sensory information to comprehend and interact with the world. Such information can rarely be guaranteed as certain and perceptual representation techniques generally require the need to allow certain degrees of uncertainty in the sensory interpretation.
3. Definitions and assumptions on the system and operating environment are required in order to bound the system development and related analysis. However, due to the potential nature of both system and environment, it may not be possible to specify in exact terms such definitions. For example, formally defining an unstructured environment may be impractical. This can potentially lead to problems at later verification and validation stages where the lack of specificity may give rise to an inability in confirming the achievement of the required system characteristics.
4. The use of computing technology, which is essential in achieving the required functionality of such autonomous robotic systems, and the underlying knowledge representation and management, can give rise to safety concerns with regards to the integrity and reliability of the software.

4.2 A hazard analysis

An initial hazard identification was performed on the autonomous excavator case study. The principle hazard classes were defined as:

1. Collision with an object on the surface
2. Collision with an underground object during digging
3. Toppling of the excavator

A back-ward and top-down approach was then adopted using Fault Tree Analysis (FTA) in order to define hazard causal factors. The FTA provides a logical breakdown of a system's failure starting from a top event defining the accident or hazardous event, and proceeding to the causal factors in the form of a Boolean logic tree. Fault trees were constructed to provide a better insight into the hazard dynamics, by identifying;

1. at which point during an operation there is danger of a hazard occurrence and
2. how the hazard might best be avoided from a system dynamics point of view, or, failing that, detected and an accident avoided.

A typical fault tree originating from this analysis can be viewed in Fig. 8. What is immediately apparent from the fault trees is the need to identify the environmental features contributing to the causal branches leading to the accident/

hazard occurrence. In other words, most causal branches are linked to specific environmental features and the limitations in the system's ability to perceive such environmental features. This differs from conventional fault trees where the events are mainly component failures. A typical example in this case is the inability to correctly foresee the effect of terrain conditions, such as traction, on the robot motion. This makes the robot expect a certain path to be followed, when in reality variations from the expected path are likely to occur given the traction effects on system motion. Other examples include the various effects on the sensory suite resulting from operation over rough terrain, potentially inducing interference as well as anomalous data patterns, eventually leading to misinterpretations of the sensory data. All such features are directly linked to specific characteristics in the environment, and thus, have to be represented in a manner that clearly indicates such dependency.

Outcome of the hazard analysis

The principle conclusion from the hazard analysis is that autonomous systems which offer some form of safety assurance need to exhibit safety control activities that mitigate hazards. Such activities are based on the system's operational capabilities under the various possible states and conditions that may be encountered during task execution. Knowledge of the system's *operational state* thus becomes an initial objective for achieving safe behaviour, where such knowledge

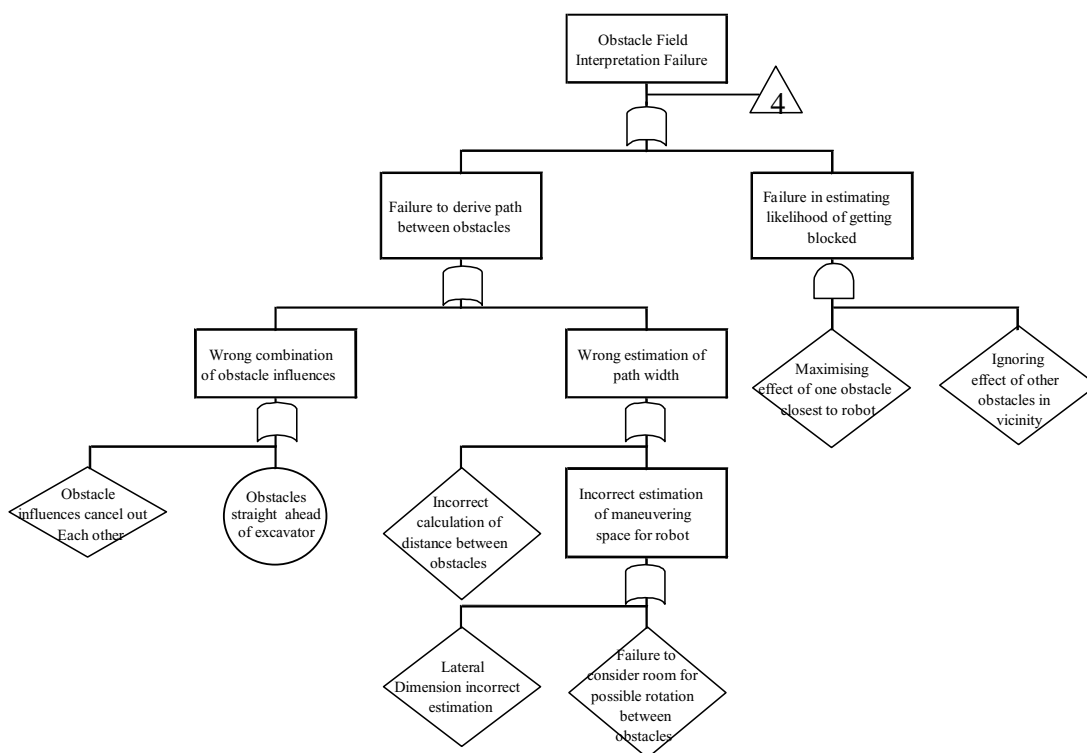


Fig. 8 A Typical Fault Tree applied for analyzing robot system safe behaviour

is substantially dependent on the ability to identify faults and failures within the system as well as anomalous system operational states.

A system should additionally be capable of defining the effect that its operational state has on the capacity to achieve the required task objectives whilst maintaining system operational safety. Again, the safety analysis provides valuable insights on how such a system ability is developed. An effective system must have:

1. The ability to detect both the system and environmental states that generate hazards. This will require the interpretation of sensor data and it is important to understand the degree of confidence in this interpretation as a result of underlying perceptual and cognitive uncertainties.
2. The ability to comprehend the safety related consequences of specific system operations and actions, prior to action execution. Knowledge of the safety consequences provides the basis for safety decision-making. Once more, awareness of the degree of confidence in the interpretation of action consequences is critical in providing an appropriate representation of operational risks.
3. The ability to directly influence action decision-making, so as to constrain or adapt actions deemed as unsafe or dangerous. In this manner, the prior safety evaluation would provide an approach to safety that reflects the knowledge about the system operation and its limitations.

Overall the autonomous system must self-manage its behaviour so as to minimise risk, where risk is defined as the combination of the likelihood of an accident occurring and its severity. The system needs to select an action or plan for execution, which, in conjunction to the achievement of the operational task or goal, also aims to reduce risk, to an extent that the operational ‘risk’ is justified given the expected benefits from executing such an action. In this perspective, the robotic system’s decision-making process from a safety point of view, becomes nothing more than a self-risk evaluation and reduction exercise. Figure 9 below provides a breakdown of the principle steps in such a risk assessment process.

5 Implementing a safety management approach using partially observable Markov decision processes

A clear approach is required for implementing the activities shown in Fig. 9. The lack of certainty with regard to knowing both the operational state and the consequences of actions leads to the adoption of a probabilistic model. From a semantic viewpoint, probabilities give a representation of uncertainty in a coherent and comprehensible manner. Furthermore, and as importantly, probability lends itself to a strong formal, mathematical representation which, from a safety management perspective, allows visibility in the assessment of the system’s level of safety and consequent decision-making. A suitable implementation technique is the adoption of Partially Observable Markov Decision Processes (POMDP).

5.1 Partially observable Markov decision processes

A Markov decision process (MDP) is a technique for making decisions about actions or behaviour in the face of uncertainty. It starts from the assumption that the world (the system and its relevant surroundings) exists in the form of a finite number of discrete states (S). Furthermore there is a set of actions (A) each of which can transform the world from one state (s_i) into another. However a particular new state is not certain. Knowing the current state of the world, the process evaluates probabilities for certain states resulting from certain actions. The process always looks forward and does not take account of states before the current one. Partially observable Markov decision processes (POMDP) additionally assume that knowledge about which state the world is in at any time is not certain, but also subject to probabilities. More formally POMDP models a discrete state system operating in discrete time. State identification uncertainty is based on a set of observations and a related probabilistic observation function, and action consequence uncertainty is defined through a probabilistic transition function that computes the transition probability to future system states for each possible action.

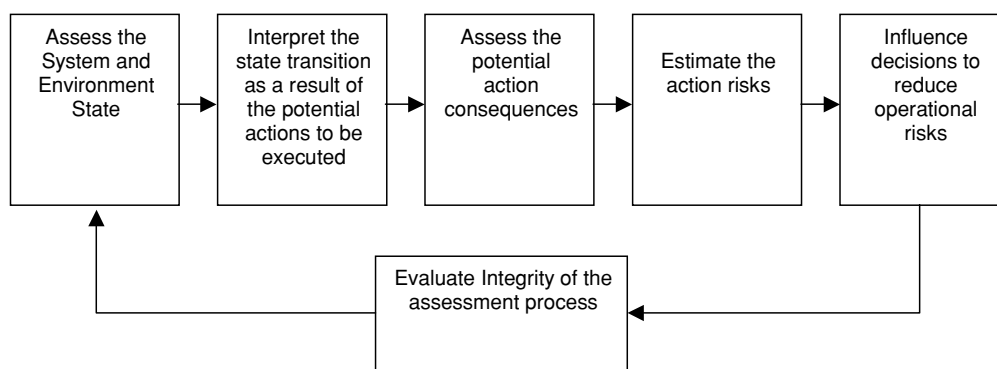


Fig. 9 The principle steps in a self-risk assessment and management process

In mathematical terms the POMDP can be represented (Kaelbling et al., 1998) as a tuple $\langle S, A, T, R, \Omega, O \rangle$ where

- S is a finite set of states s_i of the world,
- A is a finite set of actions a_i which can be taken,
- $T: S \times A_d \rightarrow \Pi(S)$ is the state-transition function, where for each state s_i and action a at time t , a probability distribution over the states is given, where $T(s_i, a, s_j)$ represents the probability of ending in state s_j at time $t + 1$.

$$T(s_i, a, s_j) = \Pr(S_{t+1} = s_j | S_t = s_i, A_t = a) \tag{1}$$

- $R: S \times A_d \rightarrow \mathfrak{R}$ is the reward function, giving the expected immediate reward gained by the system for taking each action a in each state $s, R(s, a)$.
- Ω is a finite set of observations that the system can make of its world, and
- $O: S \times A_d \rightarrow \Pi(\Omega)$ is an observation function, giving, for each action and resulting state, a probability distribution over possible observations. Here $O(s_j, a, o)$ signifies the probability of making an observation o at time $t + 1$, given that action a is taken at time t and with the system landing in state s_j at time $t + 1$;

$$O(s_j, a, o) = \Pr(\Omega_{t+1} = o | A_t = a, S_{t+1} = s_j) \tag{2}$$

The following sections describe the implementation of POMDP’s within a safety reasoning and management framework.

5.2 State definition

From an operational safety perspective, the scope of the POMDP is that of modeling the system’s state of safety and of serving as a basis for managing and ensuring adequate operational safety. Consequently, within such a model, the system states need to synthesize the system’s likelihood of an accident (or accidents) and the accident outcome or severity. In this respect, system states of interest can be defined from the identification of the robot’s hazardous states that arise out of the hazard analysis. A system state as defined within the POMDP model would thus be a reflection of the presence or absence of an operational hazardous state and consequent accident event.

5.3 Mapping safety states into hazard state likelihoods

To facilitate the definition of the state, a set of state variables is introduced, where these variables and their values embody the causal factors and conditions which give rise to the hazardous state. For example, state variables can include the state of specific sensors or actuators, their values reflecting the level of integrity of the devices. By calculating the

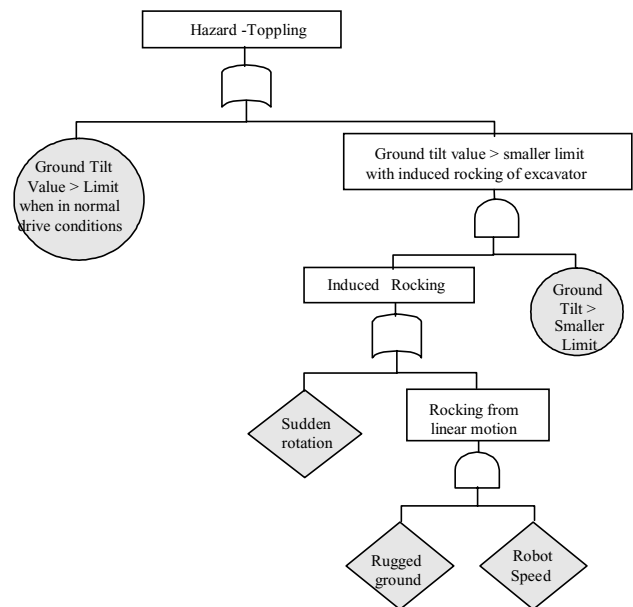


Fig. 10 A Fault Tree used for mapping state variable values into a hazardous event

likely values (and their probabilities) of these state variables, from the sensory observations as well as from knowledge of the intended actions, it is possible to extract the likelihood of the system state, which in turn allows an estimate of the likelihood of a hazardous state. A mapping between state variables (and their values) and the occurrence of a hazardous state is thus necessary to allow an appropriate safety interpretation and to be able to evaluate the risks incurred in system operation.

An effective mapping can and should generally be directly ‘extracted’ from the hazard analysis. The hazard analysis represents the link between the possible system hazardous states and their causal factors, that is the system state variables. For this purpose a fault tree model can be applied, as illustrated in Fig. 10, where the fault tree logic provides the appropriate function for converting the state variable value probabilities (the basic events) into the hazard state probability (the top event).

It is important to note that the fault tree mapping provides a means of directly linking the results of the hazard analysis to the system’s reasoning framework for detecting the presence of hazardous states.

5.4 Observing system states and defining state uncertainty

As was indicated earlier on, it is necessary to provide a state definition process which allows for the consideration of the constraints in the perceptual and cognitive processes used for identifying the system state. For this purpose, state variable values can only be identified within a probabilistic framework, through a set of observations, which in turn

embody the sensory information and knowledge generated from other sources (such as prior maps of the environment).

Defining such an observation function to evaluate state probabilities from a set of observation variable values requires the application of appropriate tools to handle the underlying probabilistic computations. For this purpose, Bayesian networks have been used to derive the state variable value probabilities.

5.5 Transitioning to future states

Defining the action transition function in order to identify the future probable states at future discrete time-steps, is problematic owing to the need to estimate the individual variable value probabilities for each possible system action. Again, Bayesian network based models have been applied for providing the transition probabilities computation.

Future actions generally depend on the operational abilities of the system and are mostly task achievement related. However system actions may not only represent physical motion requirements but may also allow other forms of behaviour. Of particular interest here, and found in POMDP applications, is the definition of explorative type actions that consist of additional observations to reduce operational uncertainties, rather than goal-directed physical actions. Such actions can be of particular use when high risk consequences of physical actions result from high uncertainties on the current and future system states.

5.6 Decision-making within the POMDP

Through the probability distribution of future state variable values as well as through the system state—hazardous state mapping, the likelihood of future hazardous states for each possible action is identified. Knowledge of the consequences of such states helps define a ‘reward’ value function which provides a quantification of such consequences. Such rewards as assigned to the individual states can be used as the basis of decision-making and action selection.

Reward assignment, in practice can be allocated to the individual hazardous states reflecting each hazard state consequence severity. In this sense, system states can be classified as safe or as states leading to accidents (hazardous states). Based on this approach it is logical to expect that a safe state is allocated a higher reward than an unsafe state. In the case of safe states there will be nothing to distinguish their level of preference from a safety perspective, and therefore a common reward assignment $R(s_{\text{safe}}) = 0$ can be used. For unsafe states, it is quite natural to assume that the least preferred system state is the one that leads to the worst hazard consequences. Penalties can thus be attributed to hazard states based on their consequence and allocated to the states in which such hazard states will occur.

Thus, if a state s_{Acc} represents a set of state variable values that indicate the occurrence of ‘ k ’ individual hazard states, h_1, h_2, \dots, h_k , then the corresponding ‘reward’ assigned to such a state will be given by (4) where $Sev()$ is the quantified penalty that reflects the severity of the hazard state consequence.

$$R(s_{\text{Acc}}) = \sum_{i=1 \dots k} Sev(h_i) < 0 \quad (4)$$

Consequently, from the rewards values allocated to the individual possible system states, defining the value attributable to an action would simply require a weighted sum of the possible future states arising from an action execution, the weighting factor being the probability of occurrence of the individual states.

Thus with a current state probability distribution $\Pr(S_t)$, the expected reward to be gained by taking the action a in the next time step, i.e. the value assigned to the action, $V(a)$ is given by;

$$V(a) = \sum_{s_i \in S} \sum_{s_j \in S} \Pr(S_t = s_i) T(s_i, a, s_j) R(s_j) \quad (5)$$

Given that the transition function in Eq. (5) provides an evaluation of the individual state variable value probabilities, to facilitate the derivation of the value attributed to an action, it is possible to bypass the identification of the individual future system states and their probabilities, and instead derive the probabilities of the individual hazardous states directly from the probabilities of the individual state variable values (through the fault tree mapping functions) Thus the rewards (severity values) attributed directly to the individual hazardous states can be applied as shown in (6). Here the individual hazardous state probabilities $\Pr(h_i)$ are derived from the future state variable value probabilities through the application of the fault tree mapping tool, with the summation being over all possible hazard states.

$$V(a) = \sum_{i=1, \dots, k} \Pr(h_i) Sev(h_i) \quad (6)$$

5.7 Risk assessment and integration with task objectives

It is interesting to note that (6) represents in explicit terms the risk taken in executing action a , since safety related risk is defined as the combination of the accident probability and accident consequence or severity (Arkin, 1993). The resulting outcome is that the process of determining the action consequence assessment reflects nothing more than an action risk assessment process.

With a risk assessment on the individual possible actions, based on the action consequences, it is possible to weight

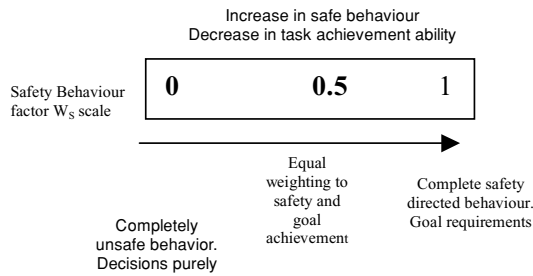


Fig. 11 A safety behaviour weighting factor (W_S) scale

such risks with the expected estimated gains of the system performing such an action, primarily through task achievement. Given that actions will be positive or negative from the perspective of task achievement, based on whether each action brings the system closer to completing its task, the balance between risk and task achievement can be achieved through a simple weighting function of both the safety and task achievement action values. The weighting function will potentially take the following form;

$$V_{Tot}(a) = W_S V_{Safe}(a) + (1 - W_S) V_{Task}(a) \tag{7}$$

where V_{Tot} , V_{Safe} and V_{Task} represent the total, safety related and task related values attributed to the action a . In the above equation, W_S is simply a weighting factor ranging between 0 and 1. The weighting factor reflects the relative importance attributed to safety maintenance when compared to task achievement, with $W_S = 1$, indicating that task achievement objectives are ignored when compared to safety maintenance, and vice-versa for $W_S = 0$. Figure 11 illustrates the effect of varying the weighting factor on a simple scale.

It can be seen that the application of the weighting factor allows the system to switch behavioural patterns from one which is purely safety oriented to one which is purely operational task oriented. The use of fault trees from the safety analysis provides a convenient measure of the risk associated with an action, but further work is required to derive a similar measure for task effectiveness. For the case of “travelling”, a simple measure is to consider the degree to which a particular action takes the vehicle towards the target location.

5.8 A graphical representation of the process model

Figure 12 illustrates a computational process model that depicts the process of safety risk assessment and management for the next time step, as described above. The process starts with the estimation of the current state belief or probability distribution through the observation function, given the observation information, the prior state and action knowledge. Given the probability distribution of the current state $Pr(S_t)$

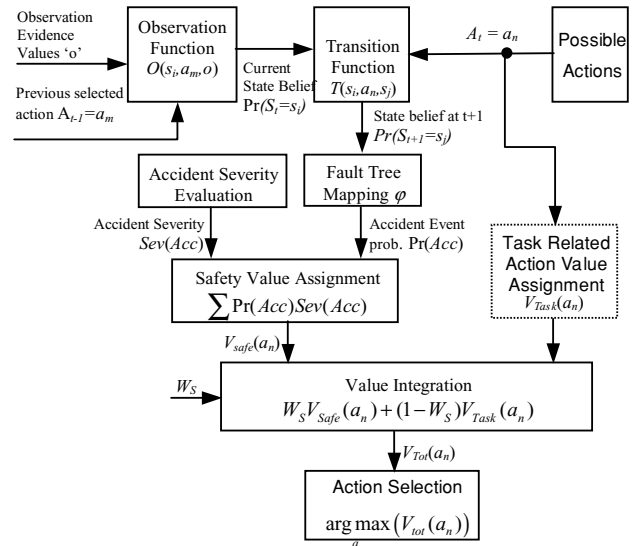


Fig. 12 A POMDP computational process model for safety risk assessment and management

described by the state variable value probabilities, the likely future state probability $Pr(S_t + 1)$ (and the corresponding state variable value probabilities) can be evaluated through the transition function for each possible action decision. With the attainment of such a future state probability distribution, the accident occurrence likelihood and the relevant accident severity can be evaluated, in the former case by using a fault tree based mapping between state variable values and the top accident event. With both accident likelihood and severity ‘reward’ evaluated, the risk in taking each action can be derived, giving the safety related value attributed to the action. This is then integrated with the task achievement related value with the weighting factor indicating the relative importance of the two operational objectives. Finally action selection becomes just a process of selecting the action with the maximum total value.

5.9 An example of a simple safety decision

To illustrate the above concepts an example of the vehicle possibly moving over rough terrain is illustrated. Figure 13 shows an example of a much simplified fragment of a decision tree to illustrate the working of POMDP. The system is 95% confident that the current state is S_1 , which is defined by state parameter values of traveling at ‘medium speed’ over ‘rough terrain’. It is faced with a choice between two possible actions, A_1 and A_2 . If A_1 is chosen, which consists of a gentle turn to the right, there is an 80% probability of moving to state S_2 , or a 20% Chance of moving to state S_3 . The former state is characterised by smooth terrain and the latter by rough. As they are the only realistic results of action A_1 , they sum to 100%. Action A_2 , also has two possible state

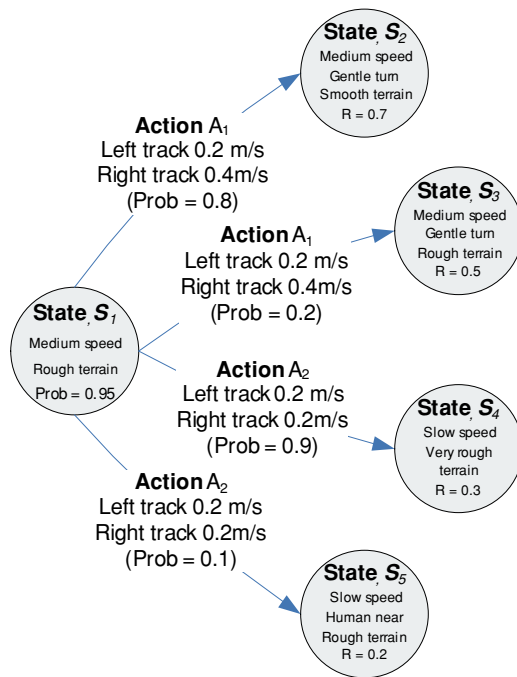


Fig. 13 An example of decision making

outcomes, one of which, S_5 , takes the vehicle near a human. Although there is only a 10% chance of this happening.

Each of the four possible new states are now subjected to a safety assessment by interrogating a fault tree with the future state parameters as the basic events. A ‘reward’ R reflecting the severity consequences (or lack of) for each state or its undesirability is then determined. By multiplying the reward with the probability of occurrence of the state, and summing up the values for each possible state outcome for an action, the action value in terms of safety is obtained (indicating how risky each action is). This process has to be repeated for all possible current states, since this will influence the interpretation of the transition to future states. Finally, the risk related values for each action under different current states are summed up and weighted according to the probability of the current state, providing an action risk value which incorporates both the current state uncertainty as well as the future transition uncertainty. Actions will also be assessed for task achievement (e.g., which action takes the robot closest to the target) and the two assessments are combined using the weighting factor to produce the action value V for each possible action, on which action selection will be based (i.e. selecting the action with the highest value V)

6 Integrating POMDP into RCS

We have already dealt with the choice of RCS-RMA (real-time control system—reference model architecture) for developing mobile robot architectures. Although RCS is a rea-

sonably generalised and multi-applicational system architecture, there is still room for increasing the robustness of the architecture:

1. RCS does not explicitly take safety into consideration, which, as demonstrated above, needs to be taken into account, particularly if the system operates autonomously in an unpredictable and uncontrolled environment.
2. RCS in its present form does not deal with uncertainty. An interpretation of the world state based on partial information from an economic suite of sensors invariably contains a degree of uncertainty. This must affect behavioural decision making and hence needs to be incorporated into the framework.

As discussed above, partially observable Markov decision processes (POMDP) provide an effective way of dealing with uncertainty through probabilities. The assumption behind the development of RCS was to develop an effective and realistic method to map elements of natural intelligence onto artificial systems. Utility theory (Joyce, 1999) argues that humans make decisions in order to maximise benefits (pleasure) and minimise costs (pain), or as stated by the eighteenth century Utilitarian philosopher Jeremy Bentham, the purpose of activity should be to “maximise the greatest happiness of the greatest number” (Bentham, 1789). Inherent within this decision process is an, often subconscious, consideration of the probability of reaching the desired outcome. Also different humans can reach different conclusions about the most appropriate action to choose depending upon their current “values” or personality. Thus a cautious and safety conscious individual would make different decisions to one who is bold and task focussed. Similarly, the POMDP approach to decision making utilises probabilities and rewards actions that maximise benefits as perceived by the “values” of the system. Hence the integration of POMDP within RCS enhances its scope for dealing with decision making in the face of uncertainty. Also, as shown below, POMDP model maps very well onto RCS-RMA and can be modelled robustly and precisely using mathematical models.

6.1 Development of POMDP process in RCS-RMA

Owing to the above, RCS-RMA is used as a framework for developing complex robotic systems and POMDP is proposed for managing probabilistic decision making at appropriate hierarchical levels of RCS. So in essence, a POMDP modification to RCS-RMA is proposed which can deal with uncertainties. This hence provides a strong basis for safety management within the action selection/decision making process of an RCS framework. Furthermore, the fact that this POMDP decision making can be incorporated into several hierarchical levels means that the system provides “defence in depth” against accidents. I.e. an upper RCS level can take

account of safety at the strategic planning level whereas a lower level makes safe tactical decisions. Thus for the robot to perform a safe and effective task, there is an increased necessity that safe behaviour should be intrinsic to the system behaviour generation (Pace, 2004).

For the development of POMDP processes in RCS it is important to be clear about definitions related to the model development.

State: The state of the system comprises not only the vehicle state but also the surrounding environment of the vehicle. The action the robot decides to select is highly affected by its knowledge of the surroundings. If a given state only represents the vehicle state without representing environment then the expected state (from WM) and evaluated state (to WM from SP, which consists of sensory data regarding vehicle and surroundings) will be conflicting.

The vehicle state consists of vehicle parameters, such as position, speed, power requirements, angle of tilt etc. It also consists of possible failure or fault states of components.

The Environment state consists of the position of obstacles, condition of the ground, weather conditions, map of the area, planned paths etc.

Current system state: This is the information/data that the system formulates regarding the vehicle state and environment at time t (current time step).

Updated state: This is update on the current state S_t depending on current observations. This is not just the vehicle state, but also the environmental state. Note that this is in the same time step t .

Future state: Future state is the estimated state in next time step, i.e. $t + 1$ using the transition function. It is more specifically the state S_{t+1} the system estimates, depending on the current state S_t and the observations made in time step t .

Future Observed State: This is the future state calculated using the Observation Function. This is the state dependent on which the system chooses the action to take in time step $t + 1$ so as to reach to time step $t + 2$. This is not required to be calculated if only one time step is considered.

Plan: Plan is the action chosen or the command executed in current time step. This also contains the list of actions available to take in next time step.

Plan evaluation: Plan evaluation is the pruned list of actions available with their respective values (rewards)

Figure 14 below represents the rough mapping of POMDP and RCS.

As already explained, each level of RCS.RMA has one or more nodes and each module has one node (Albus and Meystel, 2001). A node on the higher level communicates

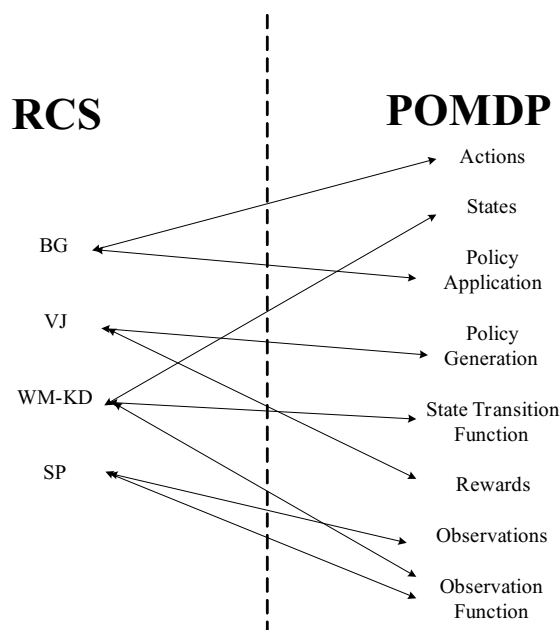


Fig. 14 Mapping RCS and POMDP

with nodes on the level below. One of the important and special properties of RCS-RMA is reusability. This property is very desirable as it makes the architecture cost effective and also simpler to manage. It also reduces the designer's efforts, as the designer has to design the structure/software only once and then repeat it in each module of RCS based architectural hierarchy. By developing a generalised model of RCS management using POMDP, the basic functional property of RCS, i.e. reusability remains intact.

In the following sections a generalised POMDP process model is developed in RCS (Fig. 15). Future work will concentrate on applying it to a 'n' level hierarchical RCS structure to demonstrate it.

The role of each element is as follows:

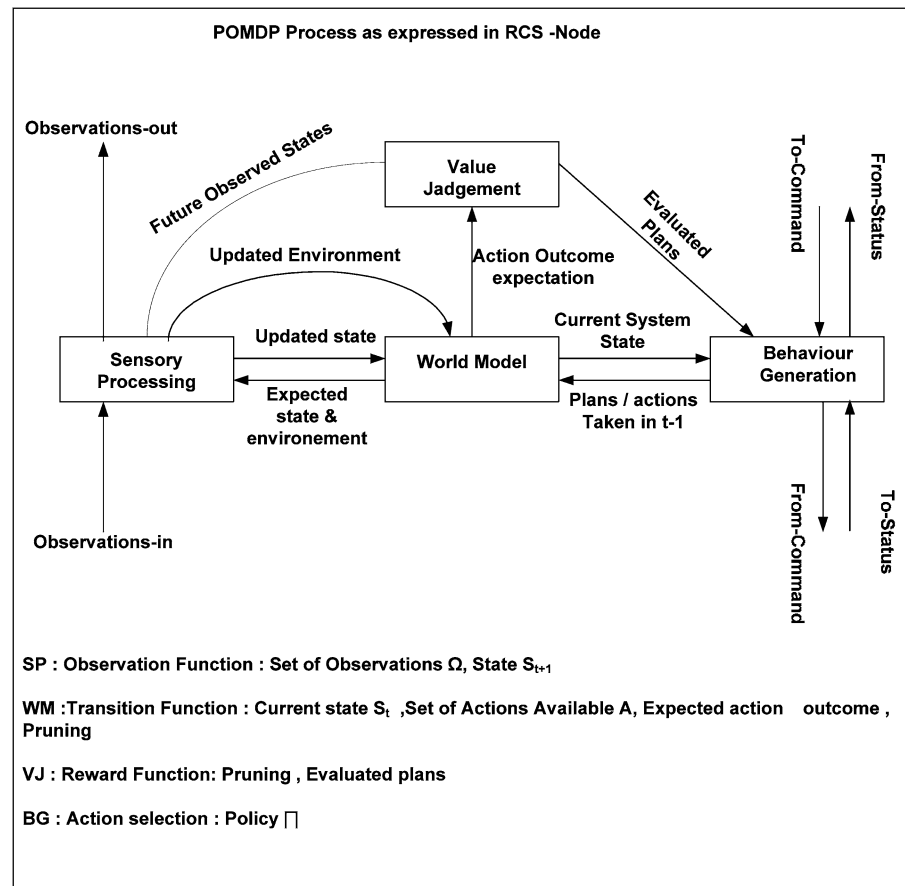
Sensory Processing: determines the current state and its probability (vehicle and environment) based on expectations (from WM) and on actual observations (from Sensors). If the expected state and observed state agree then SP becomes more certain about its current state. If they are in conflict the probability reduces, and the system becomes less certain about its current state.

World Model: stores the current belief state and computes the transition probabilities to new states given the available actions (from BG) by operating transition functions. Hence identifies future expected states and environment changes.

Value Judgement: determines the reward values to be assigned to the individual actions

Behaviour Generation: selects actions based on maximising rewards and executes them.

Fig. 15 POMDP process development in an RCS computational node



7 Experimentation with a case study and results

A software-based simulator was constructed for testing the effectiveness of different control strategies. The simulator consisted of three components:

1. An environment simulator that facilitates the creation of a virtual world complete with contoured sloping ground, various hazards and start and target points—see Fig. 15
2. A simulated robot vehicle that can be driven in the virtual world and is equipped with a range of sensors for the detection of features within the world. Realistic user interfaces to the sensors were developed so that the user can observe what the robot “sees”
3. An independent controller that can receive sensor data, make decisions and command the vehicle. This controller can be used in the simulated virtual world but could then be transferred to a real physical robot with real sensors and actuators.

The POMDP based reasoning model was adopted for a case study analysis based on the excavator simulation. The case study involved the consideration of the simulated excavator traveling over terrain where a trench is present along the predefined path of the excavator, and hence resulting

in the excavator being exposed to a potential toppling accident.

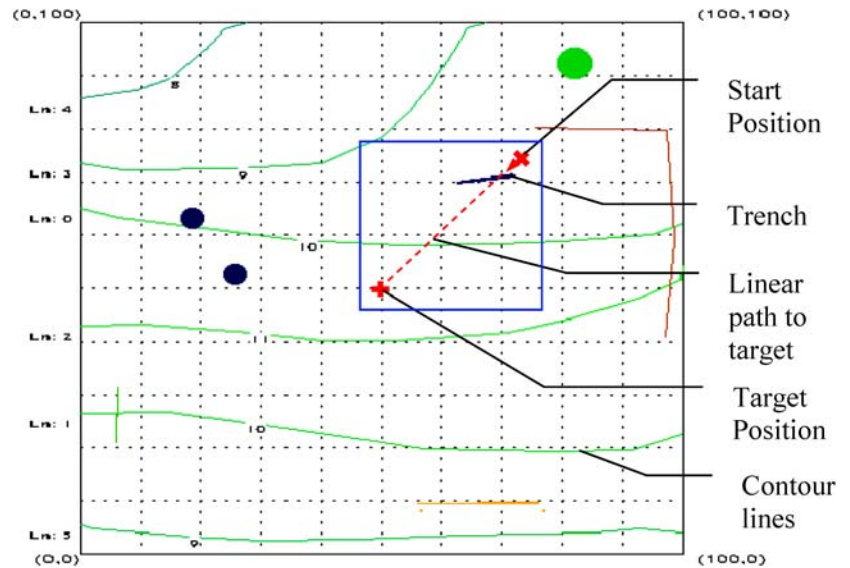
7.1 The experiment

Figure 16 illustrates the simulated environment for the experiment and consists of the virtual environment test layout, indicating the starting configuration of the robot and the target position. This test layout has been used for a number of comparative experiments to illustrate the effects of the parameters of interest. The environment model has also been used with different start and target positions, in order to monitor the behaviour under different environmental circumstances.

The scope of the experiment was to evaluate the robot behaviour in trying to avoid the trench under diverse operating conditions. Avoiding the trench, and hence avoiding the risk of a toppling accident is dependent on the knowledge on the terrain conditions available to the robot. Based on such knowledge, the robot is able to take evasive actions in order to select a safe path towards the target.

Knowledge of the presence of the trench can be obtained either by predefined maps or alternatively via actively searching for terrain conditions through the use of a laser range

Fig. 16 Simulated virtual environment



sensor. These define diverse actions which the robot can execute, depending on the values assigned to the individual actions. Here value assignment for future actions is dependent on *safety* and *task achievement*. The task achievement reward is simply allocated on the basis of the extent to which actions tend to bring the robot closer and more readily to its target position. No consideration is given by the task value assignment to the terrain conditions, and therefore avoidance of the trench depends purely on the action safety value assignment, based on the application of the POMDP safety reasoning and management framework.

7.2 Representing environmental information uncertainty

Multiple experiments were carried out with different levels of uncertainty in the robot's knowledge of the environment prior to execution of the task (i.e. the world map available to the robot). Uncertainty in the knowledge of the environment was based on

1. an uncertainty in the position of terrain features that influence the robot's tilt, such as the trench position. Thus, the robot is knowledgeable about the presence of environmental features that induce toppling (i.e. the trench), but is uncertain about the relative position of such features with respect to the robot itself.
2. an uncertainty in the terrain conditions. Here the uncertainty results in a lack of knowledge of the terrain tilt features, and is not related to any other position or orientation uncertainty.

Specific uncertainty levels were adopted during experimentation, each identifying specific environment position and tilt (terrain condition) related uncertainty parameters (i.e. probability distributions). Thus, for each set of uncertainty parameters, a different a-priori world representation would

have been available to the robot, influencing the risk assessment ability of the robot in determining whether action consequences would result in toppling. Furthermore, experiments were carried out with different weighting factor W_s values, in order to evaluate the influence of shifting robot's sensitivity to safety.

Within the simulation runs, both task and safety related decision making occurs in discrete time steps of 2.5 s. Actions selected for execution are then fed to a lower level control which simply provides appropriate steering and drive signals to the simulated robot.

7.3 Simulation test results

The motion patterns depicted in Fig. 17 illustrate the robot behaviour under different uncertainty levels where (a) illustrates a positional uncertainty standard deviation of 0.25 m and (b) illustrates a positional uncertainty standard deviation of 0.5 m. The two graphs also illustrate the path followed for different weighting factor (W_s) values, reflecting the varying relative importance given to safety assurance with respect to task achievement. From the diagrams it is noticeable that the increase in uncertainty with regards to the position of the trench causes the robot to steer away from the trench at an earlier point along the path. Furthermore, in the case of the system operation using a weighting factor value of 0.75, there is an increased 'repelling' action arising from the trench (again the robot steers away at an earlier point along the path).

At this stage it should be pointed out that the behaviour of the system is dependent on a limited 2-step look-ahead search in terms of safety assessment and therefore results in a relative myopic view of the safety consequences of actions. The effect is that the safety values assigned to actions only reflect a limited look ahead and consequently, the robot

Fig. 17 Effect of positional uncertainty of environmental features for a two step look-ahead strategy for a positional uncertainty of 0.25 m standard deviation (dashed line) and 0.5 m standard deviation (solid line) with a weighting factor (a) $W_s = 0.5$, (b) $W_s = 0.75$

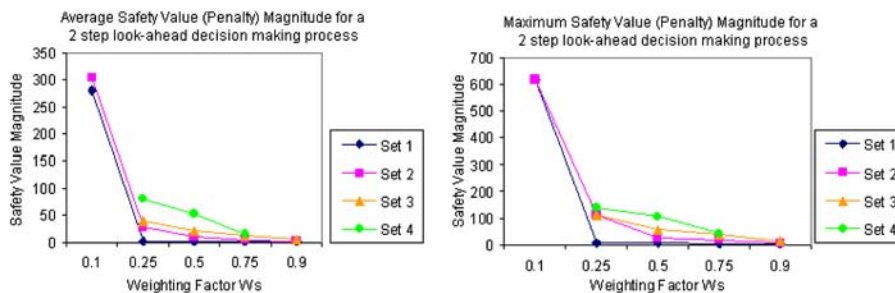
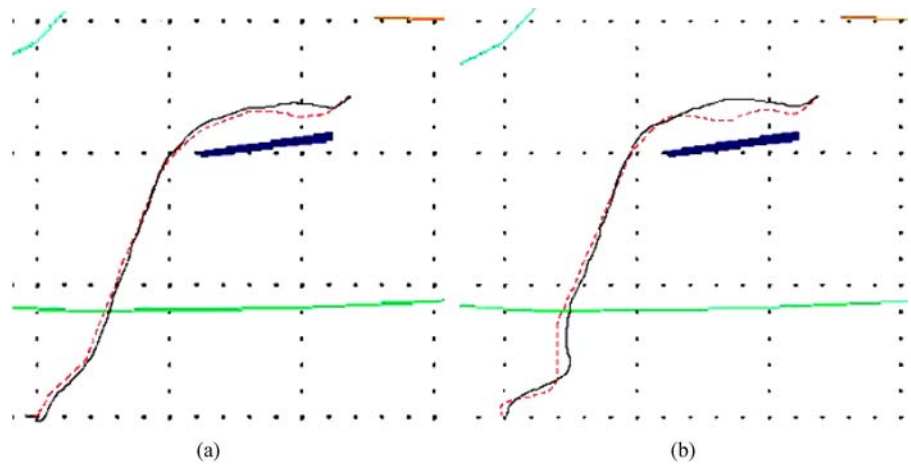


Fig. 18 Graphs Indicating the average and maximum action safety values (or penalties) during the simulation test runs for different weighting factors and levels of uncertainty (Set 1—No uncertainty in terrain conditions, Set 2—a tilt standard deviation of 3°, Set 3—a tilt standard deviation of 5°, Set 4—a tilt standard deviation of 8°). It can be noted

that for $W_s = 0.1$, the effect of the action safety values is insignificant and therefore the robot does not succeed in avoiding the trench obstacle, resulting in a high safety value (penalty) corresponding to the toppling event

may be allowed to steer towards the trench. An additional factor which can be noticed is the robot action just before reaching the target position, particularly in the case of the higher weighting factor. Here the robot aligns itself with the slope to reduce the likelihood of lateral toppling, given that it ‘knows’ it is more stable along the longitudinal axis. However, once more due to the relative myopic view towards safety, consideration is not given to the need to steer towards the target position at a later stage.

Table 1 illustrates the effects of another scenario with a high level of uncertainty in terrain conditions (here a standard deviation of 8 degrees in the tilt position is included in the

system’s prior knowledge on terrain tilt). The table indicates the number of total action decision steps during a simulation run and the corresponding number of action decisions which are purely observation related. It can be noted that due to the high level of uncertainty, the robot starts acting very tentatively and repeatedly selecting actions based purely on stopping and observing the terrain conditions in order to reduce such terrain knowledge uncertainty. This is particularly so when the weighting factor increases, ultimately resulting in the robot having to stop almost after every traveling action to observe the terrain ahead.

The graphs in Fig. 18 illustrate the average and maximum safety related values (or penalties) during execution of the simulated test runs. These values represent the level of safety related risk that the system estimates itself to be in during execution of the simulated test runs. The magnitude of the average safety and maximum safety value (or penalty) are plotted against the safety behaviour weighting factor W_s for each simulated experiment. It should be reiterated here that the action safety value assignment is independent of the W_s , and simply reflects the level of safety that the system estimates itself to be in, independent of the risk-averse nature that the decision maker exhibits.

Table 1 Action Decision Steps and observation related actions for various weighting factors, given a prior ground tilt knowledge uncertainty with an 8° tilt standard deviation

Weighting factor	No. of action decision steps	No. of observation related action decisions taken
0.25	21	0
0.5	22	1
0.75	31	9
0.9	49	23

What is apparent from these graphs, is that both the average and the maximum action safety values drop in magnitude with respect to an increase in W_s . This indicates that as the system becomes more risk-averse (increasing W_s), it tends to expose itself to lower levels of risk. Additionally, the system tends to experience only a minimal increase in risk for the same value of W_s as the uncertainty in the environment prior world knowledge increases. What this signifies is that a specific weighting factor value tends to result in a fairly constant maximum estimated risk that is experienced by the robot, no matter what the level of uncertainty is.

8 Conclusions and further work

1. The problem of managing autonomous robots in unstructured environments is a complex one, particularly if they are to be both safe and effective in performing worthwhile tasks. However such challenges must be faced if autonomous mobile robots are to become a commercial reality.
2. The adoption of a common framework or architecture for intelligent systems encourages the use of reusable modules across different systems. RCS-RMA is a flexible framework that appears to be a suitable candidate for adoption for a wide range of systems.
3. The design of mobile autonomous systems requires a detailed and comprehensive safety analysis at the early stages of the design lifecycle, so that self-safety management can become an inherent part of system behaviour. Fault trees appear to be very useful tools for real-time decision making as they provide an explicit representation of operational risk.
4. The partially observable Markov decision process (POMDP) is a technique that can be used within autonomous systems for probabilistic decision making with regard to action selection. It has been shown to be computationally tractable for real-time applications although if there is a need to look more than one step ahead, the processing time increases steeply. There is also a need to prune less probable action selections to speed-up solution times.
5. The use of a reward function, combined with a weighting factor, within POMDP enables the behaviour (or personality) of a system to be adapted easily to suit operational requirements. For example it has been shown how focus can be changed from safety consciousness to task achievement. Future work will concentrate on extending this concept to multiple “values”. For example a system may need to demonstrate the ability to take varying account of cost, quality, time, self-preservation etc. as well as safety and task achievement. Current intentions are to follow the utilitarian lead and combine the rewards for multiple values to create overarching “emotions” such as “goodness”.
6. It has been shown that POMDP can be integrated within RCS to generate probabilistic decision making levels where appropriate. Future work involves the construction and testing of a complete RCS system architecture. Also the creation of a suitable software tool to enable autonomous robot designers to build such systems, in a straightforward manner, without having to construct the underlying algorithms.
7. Finally a computer based simulation has demonstrated that the POMDP technique can be successfully integrated within a mobile robot controller and that the resulting autonomous behaviour is sensitive to variations in both the safety weighting factor and the degree of uncertainty in sensor data. Where uncertainty is high, the value of performing additional observation actions to increase confidence in sensor data can be seen.

References

- Albus, J.S. 1991. Outline for a theory of intelligence. *IEEE Transactions on Systems, Man and Cybernetics*, 21(3): 473–509.
- Albus, J.S. and Meystel, A.M. 2001. *Engineering of Mind: An Introduction to the Science of Intelligent Systems*. Wiley, Chichester.
- Arkin, R.C. 1993. Survivable robotic systems: Reactive and homeostatic control. In: M. Jamshidi and P.J. Eicket (Eds.), *Robotics and Remote Systems for Hazardous Environments*. Prentice Hall, pp. 135–154.
- Arkin, R.C. 1998. *Behaviour-Based Robotics*. MIT Press, Cambridge, MA.
- Bentham, J. 1789. *An Introduction to the Principles of Morals and Legislation* from Utilitarianism, Fontana, 1973.
- Gaskill, S.P. and Went, S.R.G. 1996. Safety Issues in Modern Applications of Robots reliability. *Engineering and System Safety*, 53:301–307.
- Geske, D.M. 2004. *The Future is Still in the Future for Autonomous Haul Trucks*. Diesel & Gas Turbine Publications, Gale Group.
- Ghu, J., Seward, D.W., and Taylor, C.J. 2004. The automation of bucket position for the intelligent excavator LUCIE using the Proportional-Integral-Plus (PIP) control strategy. *Journal of Computer-Aided Civil and Infrastructure Engineering*, 19:16–27.
- Joyce, M.J. 1999. *The Foundations of Causal Decision Theory*. Cambridge studies in probability, Induction, and Decision Theory. ISBN 0-521-64164-0.
- Kaelbling, L.P., Littman, M.L., and Cassandra, A.R. 1998. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134.
- National Advanced Robotics Research Centre, 1992. Safety and Standards for Advanced Robots—A First Exposition.
- Pace, C. 2004. Autonomous safety management for mobile robots. Lancaster University, PhD Thesis.
- Seward, D.W., Ward, J., Findlay, J., and Kinniburgh, H. 1996. The automation of piling rig positioning using satellite GPS. *13th International Symposium on Robotics in Construction*, Tokyo pp. 223–230.
- Seward, D.W., Pace, C., Morrey, R., and Sommerville, I. 2000. Safety analysis of autonomous excavator functionality. *Reliability Engineering and Systems Safety*, 70:29–39.

Seward, D.W., Quayle, S.D., Zied, K., and Pace, C. 2002. Data interpretation from leuze rotoscan sensor for robot localisation and environment mapping. *19th International Symposium on Robotics in Construction*, Washington, USA, pp 343–348.



Derek Seward is Professor of Engineering Design at Lancaster University and currently holds the post of Director of Regional Outreach for the University. He has spent over fifteen years researching in the field of construction automation and robotics including the use of satellite GPS guidance, artificial intelligence and system safety.



Conrad Pace received his M.Sc. Degree in Mechatronics and his Ph.D. in Engineering from Lancaster University, U.K., in 1997 and 2004 respectively. He currently holds the post of lecturer at the University of Malta, Malta, where he lectures on Mechatronic systems and Safety Engineering. His main research interests include mobile robot navigation and system safety.



Rahee Agate is PhD candidate at Lancaster University, UK, in the Intelligent Control Group. Research interests include mobile robotics, construction robotics, emotions modelling in artificial systems and safety integration within mobile robot architectures.