Partially Observable Markov Decision Processes for Fault Management in Autonomous Underwater Vehicles

by

Kathleen A. Svendsen

Submitted in partial fulfilment of the requirements
for the degree of Master of Applied Science

at

Dalhousie University
Halifax, Nova Scotia
July 2019

Table of Contents

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Proposed is a partially observable Markov decision process (POMDP) model-based schema as the basis of a fault manager system for use by autonomous underwater vehicles (AUV) undergoing long endurance missions with the operator far from the AUV. The thesis explains the reasoning behind using POMDP over traditional static look-up tables to achieve a more autonomous system. The objective was to develop POMDP models for two illustrative AUV sub-systems – depth and power-management. These models were used as fault managers for a series of simulations for each sub-system, individually, and then when there are interactions. This novel solution demonstrated the validity of POMDP as the basis for a fault manager in accounting for the inherent partial observability of AUV states and their environments. Future work aims to expand this with more AUV sub-systems and test on hardware-in-the-loop simulators.

GLOSSARY

- **Action**: the fact or process of doing something. Examples are turning a sensor off, sending out a communication message, adjusting the sampling rate of acoustic waves, alternating mission plan, etc.
- **Algorithm**: a process or set of rules to be followed in calculations or other problem-solving operations, especially by a computer.
- **Artefact model**: Estimated model of system built on observations and predictions made by the control system by sensing the vehicle and its environment.
- **Autonomy**: This refers to the intelligence of the machine to act independently of a human operator. It should also be noted that it incorporates the ability to resolve scenarios rather than run on a pre-set script.
- **Autonomous Vehicle:** autonomous robotic mobile system that operates independently of an operator.
- **Autonomous Marine Vehicles**: Autonomous vehicles that operate in marine settings. Includes aerial, surface and underwater vehicles.
- **Autonomous Underwater Vehicle**: Autonomous vehicles that operate primarily underwater.
- **Backup step**: This refers to a POMDP policy solver value iteration's method of how updates propagates information in reversal temporal order through the value function.
- **Belief State**: This refers to the probability distribution of the state of the system. It implies that the exact state is not known but can be estimated given observations and previous estimations.
- **Boyen-Koller**: An algorithm for factorization of belief states for approximation in DBN interference.

- **Bottom-lock**: This is when the vehicle is positioned such that it can sense the seabed. I.e. the altitude, pitch, and roll are within range for the sensors to be pointed towards the seabed (same as DVL-lock).
- **Component:** This is a physical piece of the system. For example, a battery or its wires are a component.
- **Complement Probability:** Given P(A) the complement is P(A') = 1 - P(A).
- **Conditional Probability:** of an event B is the probability that the event will occur given the knowledge that an event A has already occurred. This probability is written P(B|A), notation for the probability of B given A. Note if B is independent of A then P(B|A) = P(B).
- **Continuous Models:** A model based on continuous time.
- **Discrete Models:** A model based on discrete timesteps.
- **Doppler velocity Log**: is a sonar system that measures motion under water. It does so by measuring the velocity in relationship to the ground. DVL-Lock is when the DVL is able to sense the ground and can generate accurate velocity measurements (same as bottom-lock).
- **Fault:** This is a change (usually degrading) in the system from the normal conditions.
- **Factorization**: A method of simplifying probabilistic graphical models.
- **Fault Management System**: This refers the entirety of the intelligence for mitigation of possible faults and failures of the system.
- **Global Minimum:** In statistical analysis this is when the best-fit set of parameters is given the entirety of the model space and represents that most desired solution to a given problem. Solving these can be costly and often local minimums are used instead for simplicity and time/processing saving.

- **Group:** A specific group of states, observations, or actions that are mutually exclusive, for example, a group of depth states ‘shallow’ or ‘deep’ in reference to the position within the water column.
- **Intelligence:** This refers to the computational algorithms and methods used to perform the mission. The more intelligent the system the more it is able to handle complex tasks with human intervention.
- **Inference**: A conclusion reached on the basis of evidence and reasoning
- **Intractable:** A intractable model is one that can be solved in theory (e.g. given enough time and resources), but for which in practice any solution will take too long and be too computationally heavy to be feasible.
- **Joint-Actions:** These are the combined groups of actions that form all possible actions for the vehicle. For example, if actions groups are A, B, and C then the joint actions are: ϵ $A_1\_B_1\_C_1$ to $A_i\_B_j\_C_k$.
- **Knowledge:** Is inferred information about the vehicle and the environment that informs not only specific measurements but the relationships and context in which these measurements are acquired.
- **Local Minimum:** In statistical analysis this is when the best-fit set of parameters is given for a specific subset of a model. Determining best fit algorithms can sometimes result in these local best-fit rather than the overall global best fit of the entire model space.
- **Look-Up Table:** These are static pre-compiled lists of actions that a system can use when triggered by a specific situation.
- **Mission**: The mission is an assignment that the system is being tasked to accomplish. This may include lesser value optional assignments.
- **Monte Carlo:** are algorithms that use repeated random sampling to obtain numerical results.

- **Non-observable variables**: cannot be directly measured. These are instead inferred from other variables.
- **Observation:** Observation for our purposes is a discrete value for a variable that is used by a POMDP model to predict its state. Observations can also mean sensor data, inferred knowledge from sensor data, or variables that can be measured directly or non-directly through combining data.
- **Partial Observant:** Real-world applications such as autonomous robotics are inherently partially observant in that they can only inquire data from a limited set of sensors in order to determine their state and environment. This means they will only have a partial understanding of their state and environment.
- **Probabilistic Graphical Model:** A framework for representing causal relationships between variables that have probability distributions.
- **Q-MDP:** Modified MDP solver that uses a Q vector to account for belief of a POMDP.
- **Robot Operating System (ROS):** is a service-oriented middleware that is open source and commonly used for robotics [1].
- **Sub-system:** A sub-system is a system that is part of the larger system.
- **System:** A group of interacting elements or technologies having a functional relationship that when grouped or integrated provide some processes or services [2].
- **Space:** A non-null set which represents all possible outcomes and probabilities.
- **Stochastic:** having a random probability distribution that can be statistically modelled but may not be predicted precisely.

ACKNOWLEDGEMENTS

This thesis has been years in the making and would not have been possible without the support and guidance of the many people in my life. I would like to first thank my advisor Dr. Mae Seto who has acted not only as a mentor, but as a peer and friend as well. She has been a great source of support, insightfulness, and immense knowledge that has helped me grow both as an engineer and as a person. I would also like to thank the support of my colleagues at Lloyds Register's Applied Technology Group who have all been supportive throughout my endeavor. I would like to express my gratitude to my parents and brother for their unwavering encouragement in pursuing science and technology throughout my life. Finally, I would like to thank my significant other who has been an invaluable source of advice, steadiness and support in this undertaking.

INTRODUCTIONS

Autonomous underwater vehicles (AUVs) are effective platforms applied across marine sectors by military, commercial and scientific research. These vehicles function independently of operators to perform tasks such as route survey, mine counter-measures, search for downed aircraft, and inspecting pipelines to name a few examples.

1.1 Motivation

AUVs can operate submerged for hours, in some cases days, without fatigue, in contrast to human divers performing similar tasks [3]. Unlike tethered remotely operated vehicles (ROVs) [3], AUVs can transit great distances (in some cases, up to 1000 km [4]) from their human operators in, around, and underneath submerged structures.

The marine sector comprises 3.2% of the global economy and is expected to more than double and reach $3-trillion (US) by 2030 [5]. Given this expected level of underwater activity, AUVs are predicted to become widespread in the maritime industry as necessary tools [2].

AUVs are limited by critical sub-systems such as power distribution and navigation. Underwater communications are limited primarily to acoustics due to the rapid attenuation of electromagnetic (e.g. radio and light) signals underwater [6]. Power capacity is limited underwater, traditional air-breathing diesel generators and other combustion-based power sources cannot be used. Consequently, the vehicle must carry its entire energy needs in the form of batteries. Navigation is also difficult in underwater settings, from no access to resources like Global Positioning Satellites (GPS) without surfacing. While submerged the vehicle must approximate its location using dead-reckoning with, for example, an inertial navigation system (INS) which measures the vehicle acceleration vector to approximate its location. At the moment, underwater navigation is achieved with an INS, Doppler velocity log (DVL), compass and acoustic sensor working together. Additionally, underwater environments are unstructured and

dynamic, and the vehicle must navigate areas that range from cluttered debris, to featureless landscape without prior knowledge of the environment. With purchase costs upwards of hundreds of thousands to millions of dollars, vehicle failure or loss can result in considerable loss for owners and operators.

As AUVs are increasingly applied to complex underwater missions in unstructured environments for extended durations [7], more robust intelligent control schemas and fault management systems are required.

1.2 Thesis contributions

The main contribution of this thesis research contribution is development of a high-level proof-of-concept fault management system using a partially observable Markov decision process (POMDP) deliberative model-based implementation for autonomous underwater vehicles. This proof-of-concept is developed by building an AUV simulator framework to validate, a POMDP modeller and implementing a solver to generate policies for the fault manager.

1.3 Thesis organization

This thesis is organized into the following sections: Chapter 2 is a literature review that covers autonomous underwater vehicles, their challenges, fault-management, intelligence schemas, probabilistic graphical models and some implementations. Chapter 3 proposes the design of a new fault management tool using topics that were covered in the literature review such as model-based architectures and partially observable Markov decision processes. Chapter 4 describes the implementation based on this design. Chapter 5 presents simulations and their results using the implemented system. Chapter 6 discusses and draws conclusions from the simulations, and Chapter 7 offers suggestions for future development of the system.

2 LITERATURE REVIEW

This chapter begins by describing underwater operations and the limitations imposed by the environment. Then it covers autonomous underwater vehicles and their main sub-systems. Section 2.3 covers fault management. Section 2.4 and 2.5 present different intelligence schemas. Sections 2.6, 2.7 and 2.8 address the in-situ computation challenges, Bayesian statistics, and the Markov assumption. Section 2.9 reviews probabilistic graphical models that draw on the stochastic tools from the previous sections. Sections 2.10 and 2.11 review Markov decision processes and partially observable Markov decision processes, respectively. These sections also include methodology for generating policies and example implementations relevant to the thesis topic.

Autonomous underwater vehicles operate in dynamic and uncertain environments. The environment imposes operational restrictions like limited communications [8], energy capacity [7], lack of universal references for localization and navigation, and spatial-temporal fluctuations that vary daily, weekly, monthly and seasonally.

State-of-the-art AUVs have an important role as intelligent sensor platforms in this industry since they can operate over long and often dangerous missions, autonomously. However, to facilitate longer and increasingly more complex missions, a vehicle's autonomy (on-board intelligence) must be enabled to work in this dynamic environment which can create unexpected issues. AUV fault management systems must be engineered to be robust in the face of uncertainty and the unexpected.

Underwater operations are difficult due to complexities in their environment.

2.1 Underwater operations

The underwater environment is among the harshest to operate in from a technical and safety standpoint. Human operations over water carry the inherent risk of drowning. Autonomous underwater vehicles are important to the growth of this ocean industry given their ability to function without putting humans in harm's way. The primary challenges AUVs face are: limited communications with the operator,

navigation and localization within the environment, and sufficient on-board energy capacity to carry out missions.

AUVs use acoustic communications in their work where possible due to the rapid attenuation of electromagnetic (e.g. radio and light) signals underwater [6]. Nonetheless, acoustic communication channels are noisy and limited in bandwidth and range. Latencies, reflections, absorption, and scattering of acoustic signals make underwater acoustic propagation problematic at best [6]. As acoustic signals are the primary means of AUVs communicating with their operators, this means AUVs operate with little human support or intervention at longer ranges (tens of km) [7]. Therefore, AUVs must work mostly autonomously and can only communicate with operators at low bandwidths.

Operations at sea are resource-intensive which means infrequent support to vehicles deployed for extended durations compared to land-based ones. Navigation is difficult with GPS only available while surfaced, limited navigational landmarks, and influences from the environment (currents, sea states, etc.) on vehicle positioning. Additionally, since most communications underwater must be conducted via acoustics, the inherent range (~ kilometers) and bandwidth (<10 kHz) are quite limited [9]. Typical energy conversion methods employed by other vehicles like internal combustion engines and solar cells are unavailable at depth to AUVs, due to the lack of oxygen and sun, respectively.

These various environmental restrictions result in a system that must operate with minimal or no support. If a fault occurs, it is unlikely that it can be communicated to operators in a timely fashion or that operators are readily able to intervene to provide fault recovery. The system must also be intelligent and self-sufficient (i.e. truly autonomous) enough to operate independently on missions and to detect, identify and resolve anomalous (fault and failure) events. A fault is an abnormal condition or defect

at the component, equipment, or sub-system level which may lead to a failure. A failure is the state or condition of not meeting a desirable or intended objective and may be viewed as the opposite of success [10].

Another environment that shares many of these limitations is space.

2.1.1 Space operations

Space vehicles have similar challenges to underwater ones. They operate with limited communications and access after deployment is in harsh and unstructured environments – difficult for an operator to access. Similarly, their missions are complex and require robust fault management [11]. Due to these commonalities between ocean and space, and the relatively more mature fault management systems applied to space vehicles, space vehicles provide a relevant reference for development of autonomous underwater vehicle fault management systems.

2.2 Autonomous Underwater Vehicles

AUVs are a class of autonomous vehicles with a wide range of applications in marine settings and operate primarily underwater.

A system can be defined as “a group of interacting elements or technologies having a functional relationship that when grouped or integrated provide some processes or services” [2]. An AUV’s system is a set of components and/or algorithms working together towards a common goal. For example, physical components include batteries, sensors, actuators and logic (algorithms) like those used for power distribution, data acquisition, thrusters, and processing. The external environment could also be considered a system that acts, and is acted on, by the AUV.

Growth in the marine industry means there is greater demand for complex and extended duration missions. AUV missions have become longer, riskier, and by extension, more complex with decreased human interaction [12]. An example of complex missions are naval mine counter-measures, where an AUV must survey large

areas (e.g. 10 nm x 10 nm) and determine if, and where, mine-like objects may be located [13]. Long endurance missions like under-ice surveys [14] are susceptible to vehicle failures due to the shear long mission duration. Such AUVs could benefit from robust on-board autonomy to determine recovery or mitigation strategies. For example, robust AUV autonomy must handle complex failures like unanticipated low power [8] or reduced hydroplane functionality [7]. With true AUV autonomy the vehicle can autonomously make decisions, enact these decisions through applied actions, evaluate the results of these actions and adapt. This is paramount for more complex and longer missions in dynamic underwater environments [8].

AUVs can operate in different modes depending on their mission. For example, the vehicle may be set to a depth-keeping mode where it attempts to maintain a constant depth (relative to the water surface) within the water column. An application that may use this would be for collecting water samples [15]. Alternatively, another mode is altitude-keeping where the vehicle attempts to maintain a constant stand-off (distance) from the seabed. This is useful for missions like route surveys, where a map of the seabed is generated, or searches for objects like mines or wrecks, where it is desired that the sensor on-board the AUV be at a constant distance from the seabed to create a map.

To enable AUVs to maneuver underwater, they are equipped with maneuvering and propulsion systems.

2.2.1 Actuators

AUVs often employ propeller-based propulsion systems. Some AUVs make use of gliding [16] or more bio-inspired propulsion such as fish-styled swimming [17]. AUVs that are torpedo-shaped often propel, like submarines, with a propeller pushing the vehicle forward. Their speeds usually range between 2 to 5 knots [7]. Fins (or hydroplanes) are used to maintain or alter the vehicle's attitude (yaw, pitch, and roll, see Figure 2-1) in a three-dimensional space by changing their deflection angles to

generate more or less lift as required. Vehicles can change their depth in two modes. The first, commonly referred to as the 'flying' mode, uses changes in the vehicle pitch to change (fly to) depth, like an aircraft. Some AUVs are equipped with vertical thrusters, or hydroplanes, located at the center of hydrodynamic forces to change the vehicle's depth without changing its pitch. The choice of one or the other depends on the mission and on-board payload sensor.

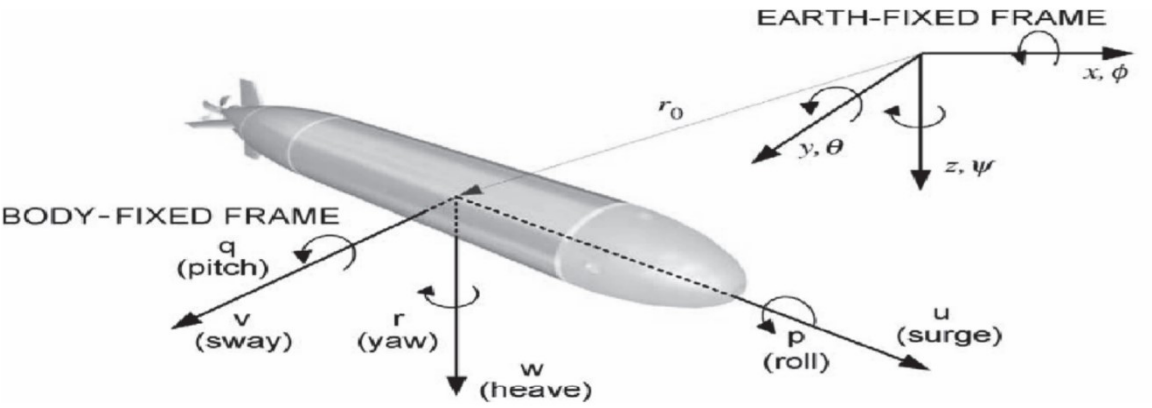

Figure 2-1 Coordinate system for a six degrees-of-freedom AUV [90]

AUVs' have two types of buoyancy control systems. The first is a static buoyancy system usually in the form of syntactic foam (glass air bubbles embedded in a matrix) designed for underwater applications (e.g. at the top of a remotely operated vehicle). The second, is a variable ballast control system [18] which changes the mix of water or air in the on-board ballast tanks to achieve the desired buoyancy. This is used in submarines and large AUVs [19]. Often AUVs are slightly positively buoyant as a fail-safe to ensure the AUV surfaces in the event of a power loss. There are buoyancy systems that change over the course of the mission to help the AUV adaptively maintain trim as it transits into water with a different salinity or temperature (since these effect water density) [20]. AUVs can also make use of weights which are dropped for an emergency rise or large changes in weight in water due to payloads being deployed.

AUVs can acquire measurements on their environments, and themselves, through sensors.

2.2.2 Sensors

Sensors are instruments that make measurements on environmental and vehicle variables (i.e. salinity, pressure, accelerations, etc.). Satellite communication, radio, optical, and other electromagnetic signals are only possible when surfaced; due to the effects of water attenuation which can only propagate short distances. Consequently, AUVs primarily rely on acoustic energies and waves. Acoustic waves can propagate much further than electromagnetic waves underwater but are limited in bandwidth as their carrier frequencies are only 10's of kHz (compared to MHz and GHz above-water). Additional complexities with acoustic waves include: reflections and refractions, and unique to acoustic are low sound speed, multi-path, internal waves, high ambient noise [6] [21], and convergence and shadow zones [22].

Given the general lack of underwater positioning references, underwater navigation is often achieved through dead reckoning. This is improved with inertial navigation systems (INS) that sense the vehicle acceleration vector and double-integrate that to determine displacement. Often, these INS are coupled with a speed-over-ground velocity measurement through a Doppler velocity log (DVL) sonar to slow the overall error growth. DVL-lock is a state where the DVL is within range to sense the seabed (or surface if pointed upwards) and thus contribute a speed-over-ground measurement to bound the error growth from a pure inertial measurement. DVL-lock may not be possible in deep water or if the AUV is operating at high altitudes with respect to the DVL range. Bottom-lock is when the AUV is within range to the seabed and appropriately angled for all desired sensors to measure the seabed, in lieu of other sensors DVL-lock can be assumed to be bottom-lock. Long duration missions could have the AUV surface for GPS calibrations to zero the accumulated error from dead reckoning. This creates overhead and consumes energy if the AUV is operating in deep water. As well, this is not an option if the AUV is operating under ice.

Deployed transponders can be used to determine the vehicle's position through acoustic positioning similarly to GPS [6]. These can be fixed to underwater nodes, marine structures and surface vehicles. However, they require the external systems to be in place and have limited accuracy depending on range and setup.

LiDAR (light detection and ranging) is a remote-sensing method using coherent light to measure range. This is limited due to the high attenuation of electromagnetic energy underwater. However, there are underwater LiDAR in the blue-green wavelength range that could achieve 50 meters, or more, range. Generally, LiDAR is used over short ranges for tasks like inspecting structures [23]. 3D lasers can be used for triangulation and achieving high resolution 3D images of underwater objects [24]. Optical short-range transmitter/receivers are a current research area to increase communication and data transfer in underwater systems [25]. While electromagnetic waves do not travel far underwater, they have higher bandwidths (by orders of magnitude) which allow for faster and larger data uplinks.

Sensors have limits in their dynamic range and can be noisy. One way to address this is to use state-estimation methods that fuse the sensor measurements with other sensing modalities (e.g. INS fused with an acoustic sensor like a DVL) and to acknowledge their noisier character. For this, a sensor model is needed which also captures the sensor's noisy measurements through a probabilistic model [26]. Sensors, and their subsequent measurement processing, require energy. AUV sub-systems for communications, actuation, ballasting, navigation (INS, compass and DVL), sensor processing, power management, etc. all require energy. Therefore, a critical AUV sub-system is the power management system which administers how the energy (and subsequently, power) is distributed across the sub-systems.

2.2.3 On-board power management systems

Underwater vehicles are limited in energy conversion options due to their environment. For example, solar panels can only be used when surfaced; however, the energy source from this method is low efficiency and is only enough to charge secondary batteries [16]

Internal combustion engines cannot be used due to lack of oxygen, although closed-cycle diesel engines and Stirling engines can be used for short durations, they are not typically implemented on AUVs [27]. Nuclear reactors are used on submarines but are generally less feasible on AUVs which are much smaller and do not have on-board operators. Another option are fuel cells [27].

However, energy sources specific to AUVs are generally batteries [27] [18].Sensor data acquisition, actuation, and processing energy requirements must be balanced with vehicle support sub-systems and mission requirements. Typically, the energy required for a mission is defined as follows (equation 2-1 and 2-2):

$$energy = \frac{(power_{propulsion} + power_{payload\ sensor} + power_{vehicle\ equipment}) \times distance}{velocity} \quad (2\text{-}1)$$

or:

$$energy = (power_{propulsion} + power_{payload\ sensor} + power_{vehicle\ equipment}) \times mission\ time \quad (2\text{-}2)$$

such that:

Table 2-1 Energy requirements

power type	*propulsion / actuation*	*payload*	*vehicle equipment*
sub-system	propellers	sensor	embedded processor
	fins	electronics	navigation (INS, DVL, etc)
	rudders	data logger	communications
	hullform drag		buoyancy control
			energy storage
			miscellaneous

The total power for each of the three power types are usually known for a given vehicle state. Long-endurance missions require that more batteries must be onboard

which usually translates to a larger AUV hullform resulting in greater drag. There is an optimal tipping point between energy carried, endurance (includes payload sensor needs) and hullform [18].

The ability for successful, long-duration, and complex AUV operations, and to a lesser extent, vehicle survivability could be enhanced through on-board vehicle autonomy.

2.2.4 Autonomy

Autonomy refers to the on-board intelligence of the vehicle and its ability to operate without direct human supervision. This can be broken into four levels that distinguish the vehicle's independence from its operators (see Figure 2-2). A system that is remotely controlled by a human operator has no autonomy, such as an ROV. A human-delegated system (Figure 2-2) is one which follows mostly scripted instructions. Factory conveyor line robots often fall under this. Human-supervised (Figure 2-2) refers to systems that can make decisions but communicate with operators while performing tasks. For instance, autonomous spacecraft can operate independently but rely on operators for their prescribed recovery actions from complex faults [29]. A fully autonomous vehicle (Figure 2-2), however, may have limited or no operator to rely on and must make all decisions, implement its decisions through actions and evaluate and adapt as required. Due to the range and bandwidth limitations of underwater communications, an AUV must be as fully autonomous as possible [30].

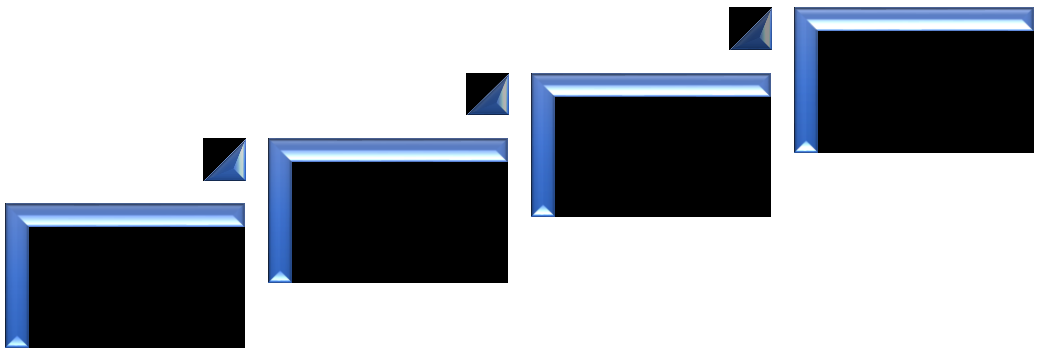

Figure 2-2 Levels of autonomy increase from human operated to fully autonomous [31].

Given the limited communications with operators, a fully autonomous vehicle requires a robust fault manager as part of its autonomy to address unanticipated issues that may lead to mission failure or loss of vehicle. This is how AUVs can increase their endurance long missions.

2.3 Fault management

"A fault is an unintended or unanticipated change, i.e. an anomaly, or defect in the AUV's system function or state which interferes with its nominal operation by causing performance deterioration. The fault can be by an incorrect step, process or damage to the AUV" [32]. A fault may lead to a failure. A failure is the state or condition of not meeting a desirable or intended objective and may be viewed as the opposite of success [10].

A fault-management or fault-detection, identification, and recovery (FDIR) system is a methodology to determine if a fault or failure has occurred (detection), what the fault/failure is and how it affects the vehicle sub-systems (identify, diagnose, localize), and finally, devise the most appropriate action to mitigate its impact (recovery) on the vehicle and/or mission.

Fault management systems are common in robotic and computerized systems. However, currently most fault management systems for AUVs operate with static pre-set actions for faults [7] and thus cannot adapt to complex or unanticipated faults/failures. Faults and failures can be probabilistic [3] (i.e. their occurrence is not deterministic) and AUVs operate in a dynamic environment which can have uncertain effects on the vehicle [33], among them, trigger faults/failures. Fault management for AUVs would need to address this.

A fault-management system can be distributed, decentralized, or coordinated [34].

- A distributed fault-management system is a single entity that may be spread (distributed) over multiple sub-systems to off-load computation effort, but it is essentially a single overarching system [34].
- A decentralized system has each sub-system maintain its own fault management [34].
- With a coordinated system, each sub-system maintains a fault management system with an overarching coordinator that provides oversight to ensure the sub-system recovery measures are consistent with the overall system and doesn't have undesirable interactions with other sub-systems [34].

The first step in fault management is to detect that a fault or failure has occurred.

2.3.1 Detection: direct and indirect observations

To sense the vehicle state, the fault management (or other) system makes internal and external measurements of the vehicle. Variables which can be measured or sensed directly by sensors are observable variables [35]. An example is a depth (pressure) sensor to determine the vehicle's depth in the water column. The data that the sensor collects is a direct measurement that is part of the vehicle's state.

Non-observable variables are those that must be extrapolated or inferred, indirectly, by combining or interpreting data from other sources [35]. The INS measure the AUV acceleration vector, and with information from other sensors, can provide a hypothesis on the vehicle pose.

Once it is established that a failure has occurred, the exact failure and its causes must be identified (diagnosed).

2.3.2 Identification and diagnosis

The fault management-system desired here must diagnose the cause of the fault. Inference, meaning a conclusion reached based on evidence and reasoning, is used to combine data and information from observable and partially-observable (in

Bayesian networks these are also referred to as non-observable) variables to diagnose the fault/failure.

Inferring failure states can be difficult. Multiple known faults could manifest similar symptoms and unknown faults may also trigger the same indicators [36]. Sensor measurements are limited to what is available on the vehicle and may be affected by the fault. Failures are often inferred from multiple data sources [7]. For example, the DVL may fail to achieve DVL-lock (i.e. sense the seafloor) because the vehicle is at too high an altitude or the vehicle pitch or roll is so high that the seabed it is pointing at is not within detection range.

Knowledge (Figure 2-3) is inferred information about the vehicle and the environment that informs not only specific measurements but the relationships and context in which these measurements are acquired. The fault manager can better determine failure states and fault causes by combining knowledge of the system to distinguish between faults.

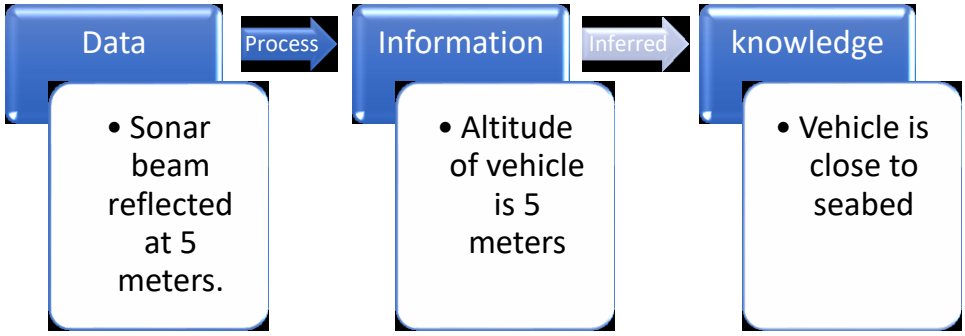

Figure 2-3 Sensor data to knowledge of vehicle state. Raw data is processed and combined into information that represents information on observable and non-observable variables that are used to infer knowledge of the state [31].

A cascade fault is when one failure triggers subsequent faults which may obscure the cause of the original fault [12].

Fault detection and identification are often studied in tandem since detecting the existence of a problem is not particularly useful unless the nature or cause of the fault can be identified. Once the fault-management system has identified the root cause of the failure, it needs to devise an action to recover or mitigate its impact on the vehicle and mission.

2.3.3 Recoverability

The desired fault management system must make decisions and determine the best recovery or course of action given a failure and then implement the action. Depending on the fault, there may not be a best recovery and the system may be left in a long-term degraded state until operators can intervene. One method of recovery is control reconfiguration [30].

With control reconfiguration the fault manager reconfigures a system to mitigate repercussions from the fault. This may involve the selection of a new configuration or relying on alternative sensor inputs and actuator outputs to continue the mission [30]. For example, the control system could restart a degraded sub-system to reset to a default initial state. Alternatively, a vehicle may reconfigure to accommodate the fault. For example, it could redistribute the actuator control to account for a jammed fin on an autonomous underwater vehicle [30]. Another method would be to turn off unnecessary processes and sensors to limit power consumption if the fault was an unanticipated low-energy state.

Fault management is necessary for AUVs to overcome failures and successfully (as much as possible) complete missions. To that end, different schemas have been developed to incorporate intelligence modelling for a vehicle's autonomy.

2.4 Intelligence modelling

The fault management system determines how the system senses and interprets the vehicle and surrounding environment. The two common schemas are model-free and

model-based. The most common, and often simpler, implementation is the model-free one.

2.4.1 Model-free based schema

A model-free system does not have a unifying overview of itself and the environment. Often these systems are divided into behaviours that perform specific actions such as diving to a specified depth or transiting to a waypoint. Each behaviour represents a discrete observable action that can be performed [37] in response to a set of conditions.

The model-free based systems can use rule-based tables for fault mitigation. Rule-based (or look-up) tables statically map a set of conditions, or rules, to a response action [9]. These rules are often pre-set by experts before a mission [32]. While these methods can be effective, they can be limited by the inherent partially observable (incomplete system representation from limited sensors) nature of the system and cannot predict and mitigate unknown (new) or future faults and failures [38]. Scalability of the look-up tables schemas are difficult when applied to more complex systems [39]. An alternative to model-free are model-based schemas.

2.4.2 Model-based schema

Model-based schemas generate models of the vehicle and its environment. These models are internal representations used to aid in the vehicle control and decision making. These models are generated by combining data from sensors, prior available hardware characteristics (sensor failure rates), a priori available measurements (e.g. water density from salinity, temperature and pressure), expert estimations (likelihood of system failures or issues), and inference from combining observations. They can also involve either models of the vehicle, or its sub-systems, that capture the evolution over a mission. Ideal models (built on how the system *should* operate) are sometimes referred to as the plant [40] or nominal models [41].

There are three types of model-based analyses: analytical, knowledge-based, and stochastic [12].

- Analytical models are for systems that operate on first principles such as feedback controllers. These are aimed at low-level hardware and component level operations [12].

- A knowledge-based approach uses a model built on in-depth knowledge and insight of the structure, components, and interaction of the components in the system. The model captures the ideal system [41] and compares it against the observed model (also known as the artefact [42]) to find inconsistencies (sometimes referred to as consistency-diagnosis [41]). If inconsistencies are found these are determined to be faults [12] [40].

- Stochastic models use probability distribution functions in their world model to account for the only partial observability and uncertainty in a dynamic ocean environment. This is achieved using statistical filters to update the probability distributions of the system's model as measurements become available [12].

Model and behaviour-based control schemas can be hybridized to create more complex and flexible control paradigms [9].

2.4.3 Model-behaviour hybrids

Model-behaviour hybrid designs are common in modern robotics for modular design and greater flexibility [9]. Behaviours allow the system to encapsulate its controls and react quickly to unanticipated changes. This combined with a world model that is incrementally updated, allows the vehicle to logically choose behaviours to optimize mission execution.

An example is an architecture that uses a controller, called a hybrid automata machine, which chooses actions to accomplish the mission goals. These finite-state

machines encapsulate states that are continuous models of complex behaviours [43]. Each model represents the behaviour's actions and maps how the vehicle's state will change over time. The control system determines how to transition the behaviours based upon pre-set state limits.

Behaviours can also be blended for precise switching between states which grants greater control and smoother transitions between actions. The behaviour values are added based upon a sliding control which determines to what degree any given behaviour affects the robot's final actions. An example of this blended state is a behaviour to 'go-to-goal' and 'avoid obstacles'. While approaching an obstacle, the avoidance behaviour might assign higher priority for decision-making, but as the vehicle gains distance from the obstacle it would revert to its 'go-to-goal' behaviour.
Control systems, including fault management ones, operate in two types of measurement and time modelling: discrete and continuous.

2.4.4 Discrete and continuous systems

Discrete systems model the world in discrete timesteps while continuous systems model the world as continuous functions. Discreet time is commonly used in robotics due to the finite sample rate of sensors. If a sample rate is sufficiently high to capture the highest frequency phenomena (i.e. the Nyquist frequency [44]) it is indistinguishable from a continuous model. This is a driver to determine the appropriate sampling frequency to use.

The nature of computations requires systems to operate in discrete time. The system clock of the central processing unit (CPU) dictates the unit speed at which digital information can be processed. Digital-to-Analog Converters (DAC) and Analog-to-Digital Converters (ADC) allow for sufficiently sampled signals to be converted back and forth. Processing data in continuous time is generally more computationally intensive due to the high sample rate although to do otherwise may lead to not capturing vital

information between samples (i.e. the Nyquist rate was not achieved). Many systems process information in discrete time to minimize computation.

Intelligence modelling can be built using different architectures for the system design.

2.5 Architectures

Each type of fault management schema can be further broken into reactive, deliberative, [32] or hybrid architectures. These architectures encapsulate how the intelligence of the vehicle is formulated and implemented.

Software architectures are an implementation of the fault management to run the autonomous system. “An architecture provides a principled way of organizing a control system. However, in addition to providing structure, it imposes constraints on the way the control problem can be solved” [37]. There are different architectures ranging from simple reactive-based control schemas to complex learning algorithms. The type of architecture required depends on the existing capabilities of the autonomous vehicle, its desired capabilities, the mission, and the environment it interacts with.

To start, reactive architectures are common due to their design simplicity.

2.5.1 Reactive architecture

Reactive architectures are generally fairly simple, they operate using a sense-plan-act schema [45] that reacts to events with mostly pre-set responses. These architectures are useful for fast decision-making (e.g. obstacle avoidance) in dynamic environments. They are generally used with model-free or behaviour-based systems.

Another architecture type is a deliberative architecture.

2.5.2 Deliberative architecture

Deliberative architectures are designed primarily for model-based systems where the system can plan its mission against models of the vehicle and its environment

[45]. These models facilitate deliberative mission-planning and predictions on mission execution.

Some designs feature a hierarchical structure where the model is broken into layers, with each layer providing sub-goals and instructions that are propagated to subsequent layer(s) [37]. Sensors update the vehicle's model and actions are filtered through the layers to arrive at the best course of action. The top-most layers focus on the end-goals of the mission. These long-term goals are processed into short-term goals for immediate actions. The bottom layers comprise of direct control actions. These architectures are suited for complex tasks but usually require complete world models and thus may be slower to execute, even for simple actions [37]. It can be difficult to develop or have complete world models to work with.

An example of this is a deliberative layered architecture that is proposed for discrete and continuous-time systems based on consistency-based reasoning [46]. Consistency-based reasoning compares the ideal model of the system to the sensed one and searches for inconsistencies that could resulted from failures in the vehicle. This consists of five (5) main layers: data validation, propagation and prediction, hypothesis generation, hypothesis testing, and hypothesis discrimination. The steps are designed to parallel the processing steps for consistency-based reasoning [46].

The implementation of intelligent systems requires algorithms for processing information from data. They can vary in computational complexity. The deliberative architecture is of interest in this thesis.

Algorithms and decision-making all require time and processing resources to execute. It is useful to compare different methods based on the computational difficulty to ensure the balance between accuracy and computational time (i.e. time it takes to solve the problem).

2.6 Computational Difficulty

AUVs possess limited computational resources and as such can be limited in computing power. Algorithms are parsed into different groups based on the complexity of their solutions (see Figure 2-4).

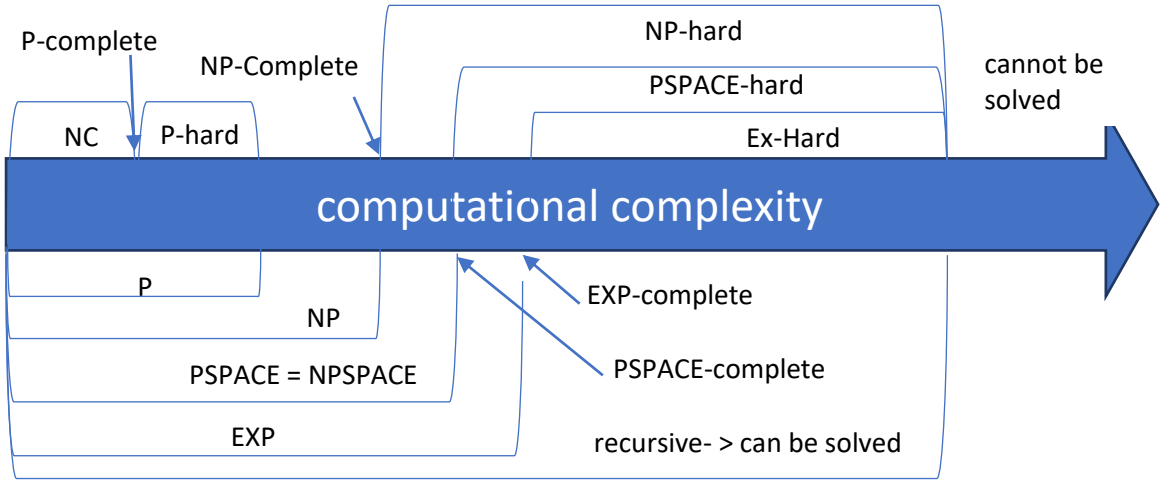

Figure 2-4 Computational complexity terms comparison [48] [47]. exp stands for exponential and P stands for polynomial. NP stands for nondeterministic polynomial.

The various computational difficulty classes modelled in Figure 2-4 are as follows:

- NC (Nick's class): problems can be solved in poly-logarithmic time when parallelized with a polynomial number of processors [47].
- P-hard: problems cannot be solved in poly-logarithmic time when parallelized with a polynomial number of processors [47].
- P-complete: refers to problems both NC and P-hard. Assuming NC ≠ P [47].
- P-time: refers to algorithms that can be solved in polynomial time.
- NP-time: refers to algorithms whose solutions can be checked in polynomial time [48].
- NP-hard: refers to algorithms that are at least as difficult to compute as the hardest NP-time solution [48].
- NP-complete: are algorithms that are both NP-time and NP-hard [48].

- EXP time: refers to algorithms that can be solved in exponential time [48].
- EXP-hard; refers to algorithms that are at least as difficult to compute as the hardest exponential-time solution [48].
- EXP-complete: are algorithms that are both EXP -time and EXP -hard [48].
- PSPACE: are algorithms that can be solved in polynomial space [48].
- PSPACE: refers to algorithms that are at least as difficult to compute as the hardest PSPACE solution [48].
- PSPACE: are algorithms that are both PSPACE and PSPACE-hard [48].
- recursive are algorithms that can be solved [48].

Algorithms that fall in polynomial time are generally seen as ideal with NC algorithms useful since they can be processed on parallel processors to further reduce their computational time. It is important to account for the computational complexity of the models in the autonomy. This will be further discussed in sections 2.10 and 2.11.
An important statistical field for this thesis' purposes are Bayesian statistics.

2.7 Bayesian statistics

Non-deterministic quantities that have inherent uncertainty can be modelled through random variables as probability distributions. With Bayesian statistics, inferences or hypotheses on such variables are updated as evidence (measurements) accumulate. The 'belief' in an event can be based on its prior state updated with measurements. Bayesian-based methods are powerful decision-making and inference tools for stochastic systems [35]. Bayesian statistics provide a framework to represent the internal 'belief' of a robot's state, even in the presence of partial observability, which is inherent in systems where it is not possible to measure all variables that define its state (e.g. AUV underwater localization).

Bayes' theorem (equation 2-1) relates the probability of an event, conditioned on another related event, based on prior information and observations. It states that the probability of event ***A***, given event ***B*** occurred, is equivalent to the conditional

probability of event ***B*** given ***A*** multiplied by the marginal probability of event ***A*** (the unconditioned prior) normalized by the marginal probability of event ***B*** (the evidence) (equation 2-3).

$$\boldsymbol{P(A|B)} = \frac{\boldsymbol{P(B|A)P(A)}}{\boldsymbol{P(B)}} \tag{2-3}$$

In addition to Bayes' Theorem, another algorithm that is often used in stochastic modeling is the Markov assumption.

2.8 The Markov assumption

Markov's assumption is that the future is conditionally independent (equation 2-4) of the past given the present [35].

$$(X^{t+1} \perp X^{0:t-1} \perp X^{T}) \tag{2-4}$$

A process is said to be Markovian if it adheres to this assumption. Many dynamic probabilistic models use Markov's assumption to reduce complexity over temporal models. Equation 2-5 shows that at each timestep the future timestep's state probability distribution only depends on the current state's probability distribution given that at each timestep, the current state's probability distribution is updated [35].

$$P(X^0, X^1, \dots, X^T) = P(X^0) \prod_{t=0}^{T-1} P(X^{t+1}|X^t) \tag{2-5}$$

Probabilistic graphical models, which are often used to represent stochastic models, can be Markovian for temporal variances [35].

2.9 Probabilistic graphical models

Probabilistic graphical models (PGMs) are graph-based representations to compactly encode complex distributions over high-dimensional spaces. Each node of a variable is represented as a probability distribution (or mass) function [35]. Connections between nodes represent the causal relationships between them. A section of PGMs are described in the next few sections. The PGMs selected are variations of Bayesian Networks and in some cases are components of POMDPs which are of interest to this thesis.

2.9.1 Bayesian networks

A Bayesian network is a type of probabilistic graphical model that uses Bayesian statistics to infer information about the system. Bayesian network representations are directed acyclic graphs (DAG) which capture probabilistic causal relationships between observable and non-observable variables through 'edges' or directed connections between the nodes (see Figure 2-5) [35].

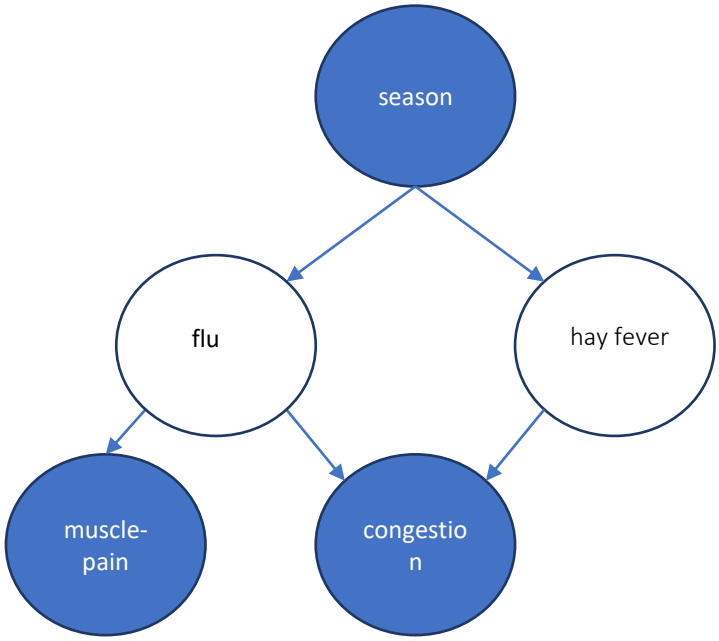

Figure 2-5 Example Bayesian network for diagnosing flu verses hay-fever given symptoms and season [35]

In the example Figure 2-5 the observable nodes are the season, muscle-pain and congestion. The non-observable intermediate nodes are for flu, and hay-fever. The cause of the illness must be inferred from the symptoms and the time of year.

Partially-observable (also known as non-observable) variables are of two types: targets and intermediate. Targets are information that is desired by the system while intermediate variables allow for help in managing the size of the conditional probability table, transparency to relationships and highlighted interactions between variables [49].

A Bayesian network can be used to model temporal systems using a dynamic Bayesian network.

2.9.2 Dynamic Bayesian networks

Dynamic Bayesian networks (DBN) permit time evolution in the system [50]. DBNs model this by allowing new observations to be made, and prior belief distributions to be updated, via Bayes theorem and the Markov assumption. The structure of the DBN itself

does not change but causal relationships between variables and the next timestep are represented [35]. For example, the vehicle's position is affected by the previous timestep's position and velocity if one uses dead reckoning.

Dynamic Bayesian networks are used across a wide range of applications with stochastic temporal systems. One such example implementation was a tool for modelling AUV mission risk.

2.9.2.1 AUTOSUB AUV MISSION-BASED RISK ANALYSIS TOOL

An example of a DBN used in autonomous vehicles is a mission-based risk analysis tool that gives the likelihood of vehicle loss given the mission and environment parameters [49]. While not a direct control system it demonstrates the stochastic nature of AUV missions and how DBNs could be applied. The DBN infers the risk of vehicle loss to determine if a situation puts the vehicle at risk.

This was applied to the AutoSub AUV developed by the National Oceanography Centre (NOC) (Southampton, United Kingdom) for ocean research. Their implementation used a Bayesian belief network model combined with Monte-Carlo simulations to generate 'risk-envelopes' that create Kaplan-Meier survival plots [51] [52]. The Bayesian belief network drew on an expert panel to assign the required probabilities [53] [54]. Later, these values were updated using Bayesian networks that compare prediction versus the actual occurrence to better model the probability of vehicle loss [55].

This example demonstrates the applicability of stochastic temporal modelling to AUV missions and underwater environments. Another useful example of DBNs is for modelling of cascade failures.

2.9.2.2 CASCADING FAILURE PROPAGATION

DBNs were used to model cascade failures for their trade-off between analytical tractability and representation of propagating cascade failure events. The stochastic

model provides a representation of the physical model for the evolving cascading phenomena [56].

This example demonstrates the ability of DBNs to handle inference and determine the root causes of faults and failures which is for fault management systems. A DBN that applies the Markov assumption to determine a system's state is a Markov chain.

2.9.3 Markov Chain Monte Carlo

A Markov chain Monte Carlo is a sequence of possible states that use the Markov assumption to make the conjecture that the current state only depends on the previous state. Monte Carlo meaning algorithms that use repeated random sampling to obtain numerical results. A transition function (equation 2-6) is applied so the probability of the next state [s'] is conditional on the current state [s] [35] (see Figure 2-6 A Markov chain example). This probabilistically models a system's evolution to account for changes between states.

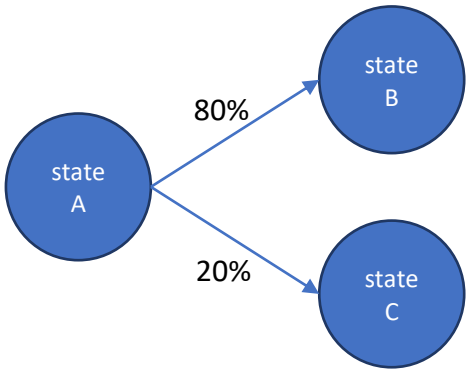

Figure 2-6 A Markov chain example

$$T(s'_i | s_i) \tag{2-6}$$

Markov chains can be applied to many problems since they can handle changes in states. A relevant example is the application of Markov chains to AUV risk assessment during critical mission phases.

2.9.3.1 MARKOV CHAIN FOR CRITICAL PHASES IN AUVS

A Markov model was applied to estimate risk to the vehicle during different phases in an AUV deployment, operation, and recovery. Faults were differentiated based on the mission phase [57]. This example demonstrates the strength of the Markov assumption to determine a vehicle's state and risks in transitions for a fault management system.

Fault management systems must also act to mitigate fault and failure impacts on the mission. Decision networks are Bayesian networks that allow decisions or *actions*.

2.9.4 Decision networks (DN)

DNs are modified Bayesian networks that include 'decision nodes' and 'value nodes' [58]. Decision nodes are possible actions that affect variables with the value nodes representing the reward or cost of associated with the variable nodes and decision nodes. A policy is the analysis of the value nodes to map the best actions that have the highest reward or lowest cost to each possible state. Decision networks are useful for autonomous vehicles since they incorporate actions in the model and make comparisons to determine the optimal outcome.

For example, a DN used to determine the best action to repair a faulted system is briefly described next.

2.9.4.1 FAILURE COUNTER-MEASURES

A decision network was applied to a case of a repairable hydraulic/ thermal system. The DN computed the system reliability, expected cost of the value nodes enacted, and the importance measure of components (meaning the criticality of the component failing). It then selected the 'best' countermeasure / repair for the system [58].

This example demonstrates that a DN can be used to choose actions to mitigate faults. Another interesting example is the application of a DN as a countermeasure for external attacks.

2.9.4.2 ATTACK/ DEFENSE COUNTERMEASURES

A decision network was applied as a countermeasure to an attack / defense scenario of the Supervisory Control and Data Acquisition (SCADA) system. This application demonstrated that the DN can have complex modeling and a fault management system which was able to apply counter-measures to possible failures from external attacks [59]. This is a useful example since the faults / failures that a vehicle fault management

system may have to consider responses to external forces (e.g. interference from wildlife).

Decision networks that model temporal variances and multiple actions over time are dynamic decision networks.

2.9.5 Dynamic decision networks (DDN)

DDNs, like their Bayesian network counterparts, are extensions of the decision process that account for temporal changes. They continually update their beliefs as each timestep provides new observations which can result in new actions to perform.

DDNs are relevant for fault management, here, since they account for stochastic relationships between variables, fault mitigation actions, and the temporal evolution of the vehicle. An example is the Anomaly Resolution and Prognostic Health Management for Autonomy (ARPHA).

2.9.5.1 ANOMALY RESOLUTION AND PROGNOSTIC HEALTH MANAGEMENT FOR AUTONOMY

The ARPHA system is part of the VERIFIM project [29]. Its goal was to investigate methodologies for autonomous on-board FDIR by the European Space Agency and Thales/Alenia Italy [29]. This fault management system was developed for autonomous spacecraft and was demonstrated in a case study for the ExoMars Rover power system. This system uses dynamic decision networks to develop policies to determine the best response to a fault/failure incident [29].

These DDNs were developed with an extended dynamic fault tree (EDFT) language [60]. ARPHA uses a junction tree to propagate observations and actions in the model and then computes the expected utility and future belief. A Boyen-Koller algorithm [61] provides the posterior probabilities over variables of interest to determine the system's state and policy evaluation for actions to take [38].

Dynamic decision networks that apply the Markov assumption are Markov decision processes.

2.10 Markov decision processes

Markov decision processes (MDP) are temporal dynamic decision networks with decision nodes, transition functions, and reward functions. They are Markovian processes since they assume the previous state is the best predictor of the current state (see Table 2-2). MDPs are used to model decision processes given a probabilistic system state [s] that transitioned from the system's initial state (prior) by applying an action, control or command.

Table 2-2 MDP tuple (s, a, T, R, h, γ)

s:	finite number of AUV & environment states
a:	finite set of actions the AUV can perform
T:	state transition probability function $T(s' \mid s, a)$
R:	reward function $R(s, a)$
h:	horizon
γ:	discount

Markov decision processes are an augmentation of Markov chains; the difference is the addition of the choice of actions [a] and rewards for choosing those actions (see Figure 2-7). This process is captured in the state transition function (see equation 2-7). Additionally, the Markov assumption uses only the system state from the previous timestep to calculate the current probabilistic system state [62]. For example, given state A and action k, there is an 80% probability the new state is B, but there is also a 20% probability that the new state may be C if states B and C constitute the entire set of possible states resulting from applying action, k (Figure 2-7).

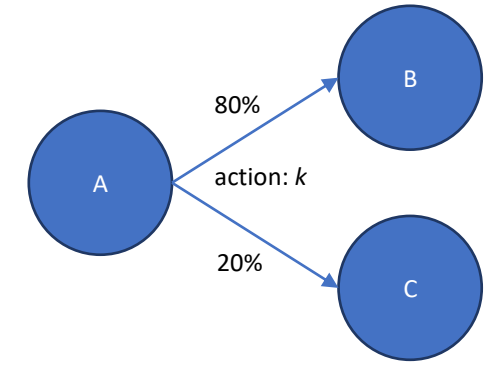

Figure 2-7 Example of $T(s' \mid s, a)$ in which possible s' are states B and C, s is state A and action is k

$$T(s'_i \mid s_i, a_j) \tag{2-7}$$

$$R(s_i, a_j) \tag{2-8}$$

The reward function (see equation 2-8) gives a cost/benefit for performing an action given the state. For example, a positive reward may be given for the vehicle rising when its state is too deep in the water column and a negative reward when it descends. The horizon is a count of how many future steps to plan beyond the immediate one. For example, a horizon of 3 may have 3 sequential actions that are performed to result in the desired state. The discount is applied to the reward so future rewards are less than immediate rewards. This allows the control system to focus on immediately beneficial actions over future ones for a more responsive system. A policy can be created that maps all possible states given their prior state and the action applied.

The utility function combines the reward and likelihood of an action transitioning the state to the rewarded one – the 'Bellman Equation' (equation 2-9). The optimal equation is the maximum utility action given the state (equation 2-10) [63].

$$V(s_i) = R(s_i) + \Upsilon \sum_{j=1}^{N} V(s_i) T(s'_i | s_i, a_j) \tag{2-9}$$

$$Policy\ \pi(s_i) = \underset{a}{argmax} \sum_{j=1}^{M} V(s_i) T(s'_i | s_i, a_j) \tag{2-10}$$

Each timestep's state is affected by the previous state and action. The reward received is affected by the state, the new state, and the action performed (Figure 2-8).

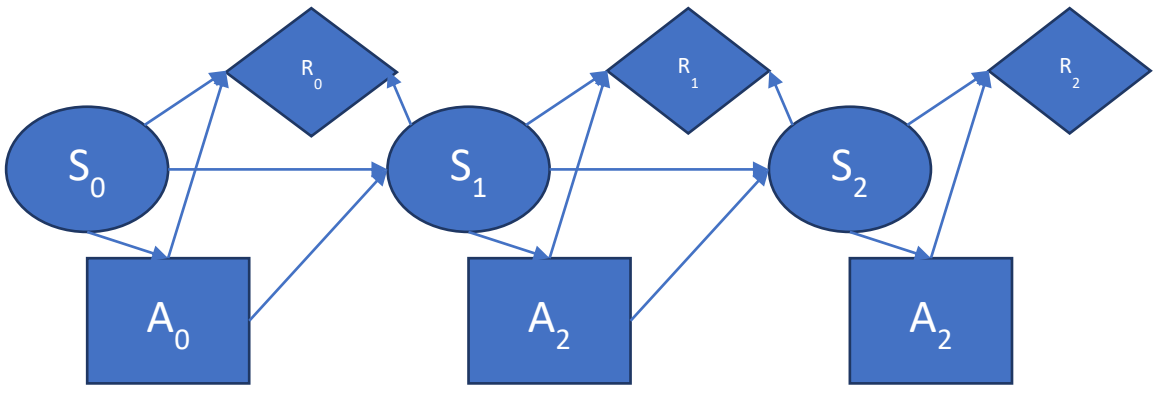

Figure 2-8 Markov decision network over 3 timesteps [91].

A *policy* generated for a system maps the states to actions that have the best utility function. A common way to generate the policy for an MDP is through a value iteration solver.

2.10.1 Value iteration solver

Value iteration calculates the utility of all actions for a given state. It then selects the action with the highest utility for the given state. The policy is generated by iterating through all possible actions a state might execute and map each state to an action and probable new state [64]. The MDP value iteration can be solved in polynomial time and is P-complete [65] [66].

Markov decision networks have been applied to several applications which are discussed next.

2.10.2 Example control systems

MDPs are increasingly applied to real-world systems since they model stochastic systems and provide actions to those systems as part of the vehicle's autonomy. One example is an MDP applied to a faulted AUV.

2.10.2.1 REAL-TIME OBSTACLE AVOIDANCE FOR UNDER ACTUATED AUV

A Markov decision process was used to plan the kinematic motion of an underactuated AUV for real-time obstacle avoidance. This was performed in an unknown environment with no prior knowledge of obstacles and an unknown sea flow vortex was applied. The system used a combination of geometrical rough path-planning and MDP-based target path-tracking which was supported by reactive collision avoidance [67]. This example demonstrates the applicability of MDP for AUV fault management.

Another example is MDPs applied to AUVs for tracking of targets.

2.10.2.2 TRACKING A ROW OF DISCRETE TARGETS WITH AUVS USING MDP

A Markov decision process was applied to a vehicle control system to determine search areas for naval mine countermeasures [68]. A grid is overlaid onto the area the AUV searches for targets in. Each individual target does not need to be found since mine

disposal (typically with underwater charges) affects an entire area and nearby undetected mines would also be neutralized. The search area was divided into cells, based on the overlaid grid, where each cell's probability distribution relates the likelihood it contained targets. The MDP determined which cells to search for new targets and minimizes mission time given the likelihood of a cell containing a target and the distance from cells confirmed to contain a target [68].

This complex modelling is a useful example for AUV fault management as it demonstrates MDPs can be used to predict possible states, given current information, and devise actions accordingly. Collision avoidance for UAVs is another example of predicting faults and taking actions to mitigate them before they because issues.

2.10.2.3 AUTONOMOUS COLLISION AVOIDANCE WITH DELAYED PILOT COMMANDS

An MDP was used to compute the optimal wait strategy for an unmanned aerial vehicle's (UAV) pilot control input. It determines whether an expected communication latency is larger than the optimal wait time, and if so, to autonomously maneuver the UAV to avoid the obstacle. This methodology must determine the likelihood of a delay and autonomously act, as opposed to wait for the pilot input [69].

This example also demonstrates an MDP that chooses between actions set by the pilot (UAV would have pre-set missions) and those that benefit the vehicle health.

Pre-planning collision avoidance is another relevant example for a fault management system.

2.10.2.4 COLLISION-FREE TRAJECTORY GENERATION FOR UAVS

An MDP-based algorithm combined with the backtracking method, was applied to optimally re-route a UAV to avoid both static and moving obstacles, track moving targets and solve for the best route to avoid them. A flatness-based trajectory planning method was applied to integrate the UAV physical constraints [70].

This example demonstrates planning long-term actions to avoid obstacles. This is of interest to fault management since obstacle avoidance is local vehicle navigation.

AUVs are uncertain about their state due to sensor limitations resulting in partial observability of themselves and their environment. Markov decision processes are limited as they *assume the state is always known*. Partially observable Markov decision processes address this gap.

2.11 Partially observable Markov decision processes

Partially observable Markov decision processes (POMDP) incorporate the MDP and the assertion that the system's state is only partially observable and could be approximated with a belief distribution over all possible system states (see Table 2-3). A belief state is a probability distribution across the possible state space representative of the vehicle's current state, given its past history of actions and observations [71].

Table 2-3 POMDP tuple (*s, a, o, T, R, O, b*)

s:	finite number of AUV and environment states
a:	finite set of actions that the AUV can perform
o:	finite set of observations the AUV can make
T:	transition probability function *T(s'| s, a)*
R:	reward function *R(s, a)*
O:	probability of observation function *O(o| s, a)*
b:	belief distribution of across all possible states for which state the vehicle is in.
h:	horizon
ϒ:	discount

An observation is a discrete value for a variable that is used by a POMDP model to predict its state. The probability of the making-an-observation function (see equation 2-11) maps the conditional probability of making that observation given the state and action that was executed.

$$O(o|s', a) \tag{2-11}$$

This function along with the state transition function in the MDP model captures the uncertainty of the system's state estimations [72]. This allows the MDP to model the uncertainty of the state as well as the transition in state from actions.

For each timestep, given a series of observations, the belief distribution of the state is updated using equation 2-12 with the normalizing constant (η) for the belief update defined in equation 2-13 [63].

$$b'(s') = \eta\, O(o|s', a) \sum_{s \epsilon S} T(s'|s, a) b(s) \tag{2-12}$$

$$\eta = \frac{1}{\sum_{s' \epsilon S} O(o|s', a) \sum_{s \epsilon S} T(s'|s, a) b(s)} \tag{2-13}$$

Like the MDP, each timestep of the state is affected by the previous state and action, and the reward is affected by the state, the next state and the action chosen. The action chosen is affected by the observations made, which are used to build the state belief since the state cannot be observed directly (Figure 2-9).

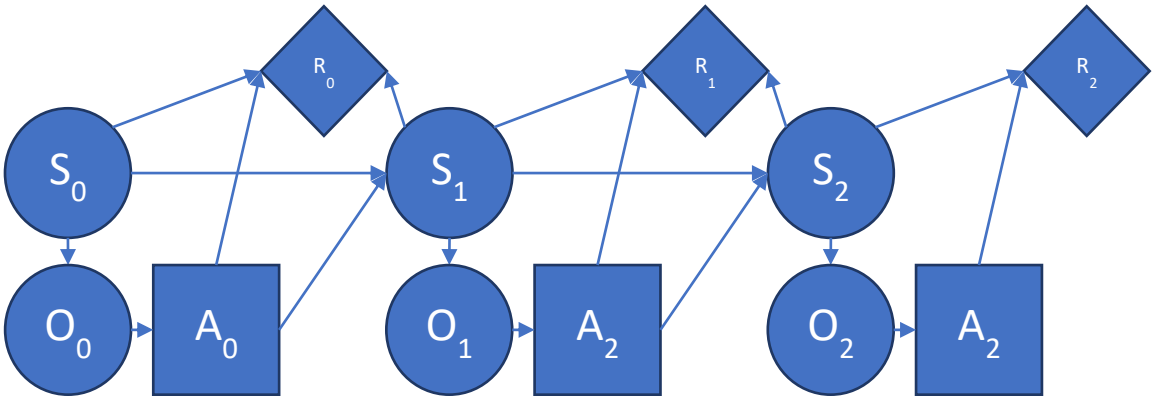

Figure 2-9 Partially observable Markov decision network over 3 timesteps [91] (O=observation, S = state and A = action).

A major draw-back of the POMDPs is their high-dimensional space. Additionally, when planning for a horizon of more than one timestep, the complexity is exponential since each state is probabilistic and the next timestep must account for the previous state being known. The conditional probabilities of the observation function can be generated from the vehicle's sensor models [73].

The process to generate a policy (maps belief states to best actions for maximum reward) is made more complex by the POMDPs lack of known state. This could be

addressed by solvers that attempt to minimize the solving complexity while still generating optimal policies for large state spaces.

2.11.1 Solvers

There are POMDP solvers, however, many only approximate the POMDP due to its high-dimensionality. The solvers vary due to the models they are applied to, their execution speed, assumptions of the environment, etc. Some solvers only seek a local optimal and others are combined with state estimators like particle filters. Solving POMDPs is highly intractable generally, partly due to the optimal policy potentially being infinitely large [74]. POMDPs for finite horizon cases are PSPACE-complete [65] [75]. This makes their solving time grow exponentially [76] for larger state, observation, and action spaces. This could be an issue when applied to AUVs and as such alternative solving methods, such as approximate solutions, may be preferable. The following solvers were selected for review based on their precursor to other solvers, and their appropriateness to autonomous vehicle applications.

Value iteration, with a known reward function, forms the basis for POMDP solvers and is discussed next.

2.11.1.1 VALUE ITERATION

Similar, to MDP value iteration, a POMDP value iteration calculates the utility relative to the payoff (reward) function [64]. However, POMDPs cannot directly determine the state. Instead, a piece-wise linear convex (PWLC) function must be developed. The PWLC maps the state belief and actions to the states they will end in [77]. Figure 2-10 shows a system with two states (0 and 1) and two actions (a1, a2). Initially, when b is within the first state, a1 (blue line) has the highest utility in reward. In the second timestep a2 (green line) would have the highest reward.

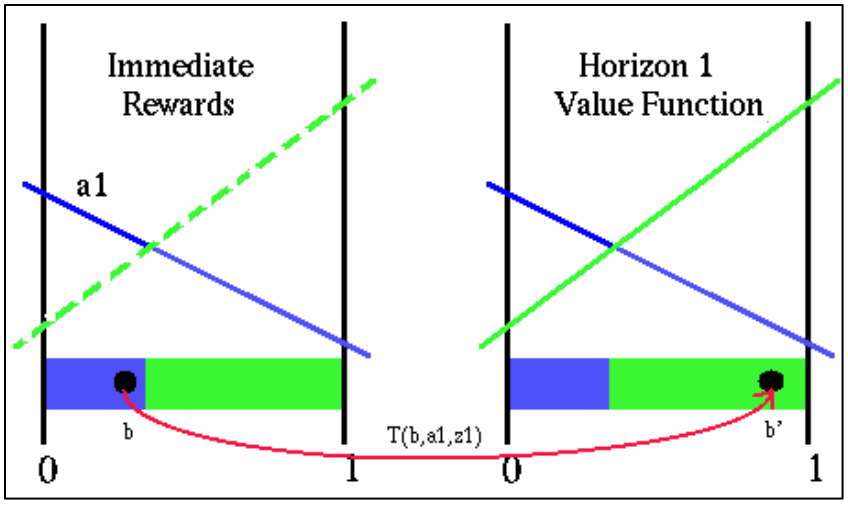

Figure 2-10 Value of a fixed action and observation [77]. The two possible states are 0 and 1. The probability of being in either is given by the blue and green bar between 0 and 1. Each state has a utility function (blue and green lines). In the first timestep (left) given the belief b the highest utility value is from the action a1 (blue line). This results in the belief changing states from 0 to 1 in the next timestep (right).

The value iteration algorithm defines the basics in many of the alternative solvers. However; it is time-consuming, intractable and consequently, inefficient. Additionally, the value iteration computes value functions for ALL belief states not just the relevant ones, some states will be unreachable from the current vehicle state. Other solvers search the subset of the state space with states, and actions that are attainable, desirable, and probable [64].

An alternative similar method to value iteration is policy iteration.

2.11.1.2 SONDIK AND HANSEN'S POLICY ITERATION

Policy iteration (proposed by Sondik [78]) iterates through possible policies using the utility function and attempts to improve the policy from the previous one. The policy iteration algorithm can converge to a policy arbitrarily close to the optimal policy [78]. The policy evaluation step converts a policy to an equivalent, or approximately equivalent, finite-state controller which allows the value function of a finite-state controller to be computed in a straightforward way. However; the conversion between these representations is complicated and difficult to implement, thus its limited use [79].

Hansen proposed an improved policy-iteration algorithm and a new heuristic search algorithm that solves infinite-horizon POMDPs through searching for a policy space of finite-state controllers [79].

This solver, along with the previous solver, forms the basis of modern solvers however, they are unable to generally solve large state spaces. A solver that was developed to improve solutions for POMDPs was the witness algorithm.

2.11.1.3 WITNESS ALGORITHM

The witness algorithm explores a finite number of regions that are a partition of the state space imposed by the PWLC property of the value function [71] [80] [63]. It simplifies the solution of POMDPs to evaluating each action separately (rather than together like previous solvers) and then combines the value functions at the end. This solver was more efficient than previous versions however; it was still only efficient for smaller state spaces. Its improvements are often featured in other solvers. A larger improvement for solvers for larger state spaces was the point-based value iteration.

2.11.1.4 POINT-BASED VALUE ITERATION

Point-based value iteration (PBVI) selects a small set of representative belief points and iteratively applies value updates to those points. It applies explorative stochastic trajectories to select belief points, thus reducing the number of beliefs points necessary to find a good solution. PBVI focuses planning on reachable beliefs and due to its use of fixed belief points set, is able to perform fast value backups [81]. A further development of the PBVI is the implementation of Perseus.

2.11.1.5 PERSEUS

Perseus uses the randomized point-based value-iteration algorithm to operate on large belief sets that are sampled by simulating random trajectories through belief space. Approximate value iteration is performed on the belief set by applying several 'backup' stages, which ensures that each backup incrementally improves the value of

every point in the belief set. A backup state refers to the value iteration's method of updates to propagate information in reverse temporal order through the value function. Additionally, a single backup may improve the value of many belief points. Perseus backs up randomly selected subsets of points in the belief set, sufficient to improve the value of each belief point in the set. This reduces the computation burden on the back-up stage which can be intensive [82].

Similar to PBVI augmentation of value iteration, an alternative to policy iteration is the method of point-based policy iteration.

2.11.1.6 POINT-BASED POLICY ITERATION

A point-based policy iteration (PBPI) algorithm for infinite-horizon POMDPs was implemented by combining policy iteration (Section 2.11.1.2) with point-based value iteration (Section 2.11.1.4). PBPI allowed for the faster convergence of policy iteration and the high efficiency of PBVI for policy improvement. PBPI is generally more robust and requires fewer points for solving than PBVI which leads to increased scalability and robustness compared to its predecessor [83].

Alternatively, to variations on value and policy iteration are approximations of POMDP as MDP to solve for policies. One such solver is the augmented Markov decision process.

2.11.1.7 AUGMENTED MARKOV DECISION PROCESS

The augmented Markov decision process (AMDP) approximates the POMDP as a reduced MDP model to solve [64]. The AMDP compresses its space of beliefs to a mode incorporating the uncertainty of the distribution (i.e. distribution and entropy). It assumes that the "belief space can be summarized by a lower dimensional 'sufficient' statistic f, which maps belief distribution into a lower space" [64].The probability functions and reward functions for the augmented space are than learned. Once these new augmented functions are generated value iteration is used to generate the policy [64]. It then solves the policy of the approximated belief space as an MDP. This

however; is disadvantageous due to the compression of the model being computationally demanding [84].

Another POMDP solver that uses an approximation of the POMDP as an MDP is the Q-MDP.

2.11.1.8 Q-MDP

A Q-MDP is a type of solver that approximates the solution of a POMDP by first solving the MDP and then applying the state belief distribution to the MDP policy to determine the POMDP policy [64]. The Q-MDP algorithm is the same complexity as MDP value iteration [75] [85] which simplifies its computational complexity to P-complete. The Q-MDP name is derived from the fact that single action value functions *V(s, a)* were historically referred to as Q-functions [80].

The Q-MDP makes the (usually false) assumption that after one step of control, the state becomes fully observable. This can be an issue since the algorithm assumes that any ambiguity will fall away in the next step, causing it to lean towards neutral actions and thus has difficulty in choosing information gathering actions [80]. However, while using direct control strategies in real world problems, Q-MDP policies perform very well [85].

Equation 2-14 is the Q-MDP value update of the MDP value function and Equation 2-15 is the Q-MDP policy update.

$$Q(s_i, a_j) = r(s_i, a_j) + \sum_{k=1}^{N} V(s_i) \times T(s'_i | s_i, a_j) \quad (2\text{-}14)$$

$$\textit{Policy } \pi = \underset{a}{argmax} \sum_{j=1}^{M} b(s_i') Q(s_i, a_j) \quad (2\text{-}15)$$

The Q-MDP solver is promising for the thesis application due to its ability to handle large state spaces with minimal computational complexity. PBVI, PBPI and Perseus are also plausible for this thesis' application. Value iteration, policy iteration, and witness algorithm, while relevant to discuss due to them being much of the underlying processes of more modern solver implementation, are not suitable to be directly applied for the purposes here. The AMDP solver requires intense computations

for compression which an embedded AUV processor may not be able to handle and is thus less relevant for purposes here.

Modern solvers often combine solvers with other methods (such as particle filters) or make assumptions to reduce complexity and solution state space. This with additional increases in computing power allows for more feasible application to real-world problems.

2.11.2 Example implementations

While still primarily research tools due to their complexities and computation requirements, POMDP models are increasingly applied to real-world systems. This is due to their ability to account for decision-making, stochastic models, conditional relationships, probabilistic state transitions and system and environment partial observability.

An example of this is a POMDP applied as a survivability agent (similar to a fault management system).

2.11.2.1 AGENT SURVIVABILITY SYSTEM

A POMDP model was applied to a survivable agent system for environmental threats, sensor observations, and the composite state of its agents. This system models large scale agent population and is assumed to exist in an unstable environment that is subjected to inadvertent and deliberately induced failures [86].

This application demonstrated the use of POMDPs for survivability which is similar to fault management in that it determines actions that best promote the health or survivability of the agent. Another implementation of POMDPs is for naval mine classification.

2.11.2.2 MULTI-VIEW TARGET CLASSIFICATION WITH AUV

A POMDP model was applied to the multi-view underwater mine classification problem. The POMDP policy adapted an AUV route to determine the number and

viewed aspects to confidently classify a mine-like object. The POMDP's states correspond to target-aspect pairs for different object types. Actions incorporate AUV paths and classifying the object that is being interrogated. The reward function is defined to treat misclassifications equally, rather than heavily penalizing certain types [13]. This was evaluated using a set of synthetic aperture sonar data collected during NATO Undersea Research Centre experiments. The POMDP reactive method outperformed other deliberative methods of classifying targets [87].

This example demonstrates how POMDP models can be used to distinguish between similar objects. Fault management similarly differentiates between normal and faulted states that may manifest similar observations.

Another example of an applied POMDP model is for UAV path-planning.

2.11.2.3 PATH-PLANNING FOR UAV

A POMDP control system was applied to passive detection in UAV path-planning. Both the state and action spaces were modelled in continuous time. The posterior probability distribution of the target state was estimated with an Unscented Kalman Filter. The control system was simulated in MATLAB [88].

Similarly to MDPs (Sections 2.10.2.3 and 2.10.2.4), POMDP models are useful to determine actions over time that mitigate failures such as collisions.

These examples demonstrate the viability of POMDPs for real-world applications and autonomous vehicles in particular.

In summary, from the literature review, underwater environments are complex, dynamic and only partially observable. AUVs are relevant and useful tools for underwater applications since they can act independently of operators. AUVs however, must be enabled with intelligent autonomy to facilitate complex and long-duration missions. To this end, fault management is important as it facilitates a vehicle's ability to mitigate current and potential failures. The implementation of a fault management

system can vary in its: representation of time; model; design of the underlying architecture, and the way it makes decisions.

Of interest are deliberative stochastic model-based systems that operate in discrete time. Fault management systems can implement these using partially observable Markov decision processes.

POMDPs are part of a larger group of models called probabilistically graphical models which capture relationships between observable and non-observable stochastic variables. Non-observable variables can be inferred through their conditional probabilities with observable variables. PGMs are used in a wide range of applications including fault management and related problems like collision avoidance and classification of observed targets.

POMDPs are directly used in some autonomous vehicle systems to path-plan, virtual survivability modelling, and mine classification. These examples demonstrate the robustness and capabilities of POMDPs over a wide array of tasks in dynamic environments. The POMDP model's ability to capture stochastic systems and causal relationships, as well as determine the 'best' actions to achieve desired vehicle states, makes it a powerful tool for fault management.

At the time of this thesis, there is no public literature on implementation of POMDPs for AUV fault-management. Therefore, this thesis proposes to develop a proof-of-concept fault-management system that applies a partially observable Markov decision process fault manager for an AUV. The fault manager implementation uses a discrete-time deliberative stochastic model-based system.

The rest of this thesis is laid out as follows. Chapter 3 contains the methodology which covers the reasoning for the design and tools selected. Chapter 4 describes how

the system is implemented. Chapter 5 presents results from the revealing simulation test cases for the fault management system. Lastly, Chapter 6 presents conclusions and Chapter 7, recommendations for future work.

3 METHODOLOGY

This thesis develops a stochastic, deliberative, model-based fault management system with partially observable Markov decision processes for autonomous underwater vehicles. A proof-of-concept implementation was tested by simulating two AUV sub-systems and developing POMDP models for the fault manager. These POMDP models were processed with a Q-MDP solver to create policies to select the best action at a given timestep and belief state. The AUV sub-systems chosen were fairly simple in terms of complexity but this was desired to focus on proving the application of applying a POMDP. While these specific sub-systems could be controlled by more simple control systems it is proposed that the model be eventually expanded to more complex sub-systems such as navigation.

To be consistent in the terminology (due to words having multiple meanings depending on context) from here on:

- 'observations' refer to judgements on measurable discrete system variables that determines the state. For example, a depth sensor reading may read 50 meters and the observation the AUV might make on this is that the vehicle depth is 'deeper than desired'
- 'actions' refer to commands given by the POMDP fault manager to the AUV to enact. For example, an action may be to 'abort the AUV mission and surface'.
- 'states' refer to non-observable discrete variables that represent the vehicle's health and pose in the environment.
- 'belief states' refer to the probability distribution the POMDP fault manager has over all possible states.
- 'confidence of a state' refers to the likelihood/probability of the vehicle state.
- 'measurements' are simulated AUV sensor measurements to infer the observations.
- 'groups' refers to the mutually exclusive values of states, observations, or actions variables, for example a group of energy capacities could include high energy

capacity, and low energy capacity. Joint-actions, joint-states, and joint-observations are various combinations of the different groups (variable values).

- bottom-lock and DVL-lock are equivalent for our purposes (normally bottom-lock would refer to all sensors but since DVL is the only one accounted for in this model it is used as the reference for bottom-lock).

The fault management system models in discrete time with each timestep accessing new simulated sensor measurements. *These measurements are used to generate observations for the vehicle*. The fault manager uses these observations to update the belief state distribution of the vehicle. Once the belief state for the AUV is known, the policy would be used to select actions for the AUV to perform. The next timestep would update with new observations (i.e. actions are enacted almost immediately).

This system was implemented in the open-source middleware Robot Operating System (ROS) [1]. ROS was selected because it is relatively easy to use and has available third-party robotics tools.

The fault management system was developed around four main ROS nodes (see Figure 3-1). The mission-control node administers and runs each simulation in the mission-control input file (see section 4.1). The logger node generates log files from the simulations (see section 4.2). The POMDP node is the fault manager that generates actions to perform (see sections 3.2 and 4.3). Finally, the AUV simulator models a vehicle (Sections 3.1 and 4.4), makes observations from simulated sensor inputs, and enacts actions from the fault manager (POMDP).

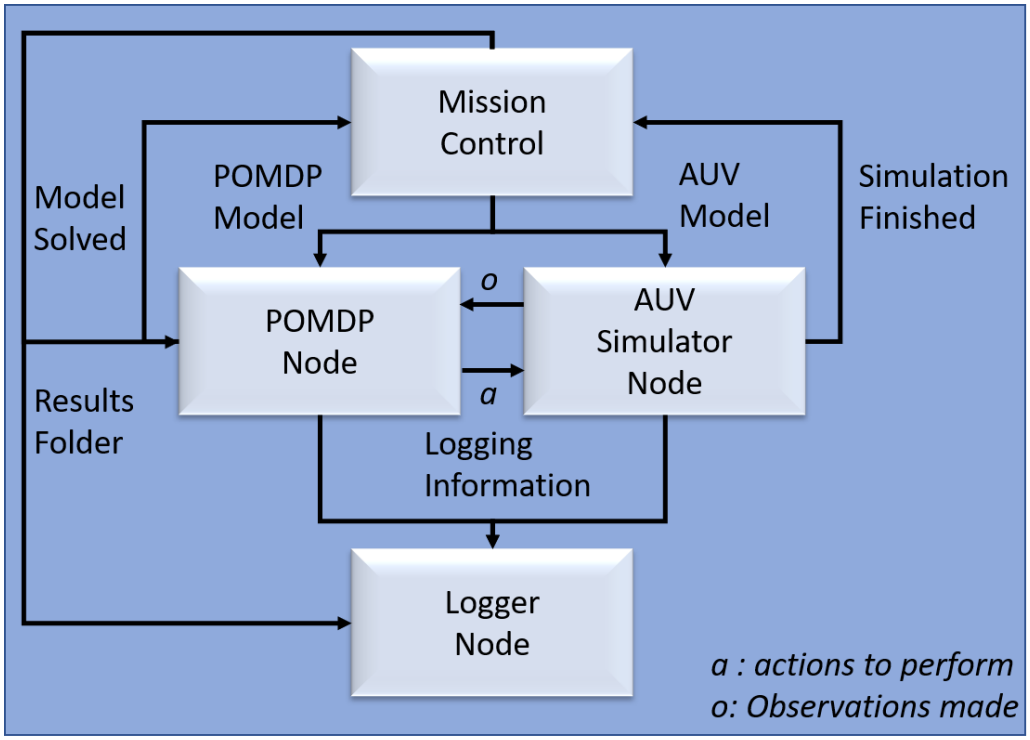

Figure 3-1 Implementation of AUV fault management system using ROS

3.1 AUV Simulator

The AUV simulator consists of two sub-systems: depth and power management. They can operate independently of one another or have dependencies between them.

The depth sub-system addresses the AUV depth and pitch where depth is a component of the AUV pose (state), has observations that are easy to interpret, and can easily integrate real-world data for test purposes.

The power management sub-system is a critical AUV component since energy is a limiting factor in mission success. The energy consumed each timestep is generated by input values from a pre-set log file. A logical progression in the future would be to integrate the energy consumption over all sensors, motors, actuators, and processors. The power-management sub-system calculates the estimated time to mission completion by comparing against a pre-set total mission time and elapsed time since

mission start. This sub-system's actions can limit energy use and abort the vehicle mission.

The AUV simulator starts when the mission control sends it an initialization file (Figure 3-3a). This initialization file contains AUV parameters like maximum depth and initial energy capacity. Next, the simulator initializes the two sub-systems' variables as defined in the initialization file. If the variables are not set for a sub-system, the vehicle disables that sub-system by default (facilitates independent sub-system testing, see Appendix C – Table). Finally, initial vehicle observations (judgements on measurements) are made and sent to the fault manager (POMDP node).

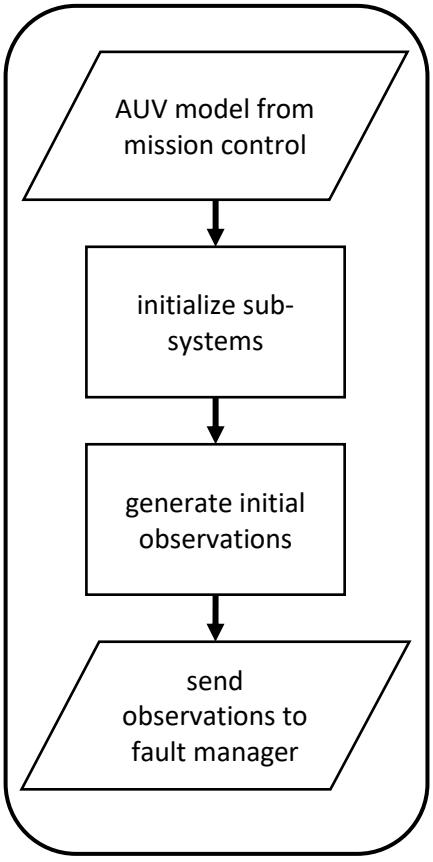

Figure 3-3a AUV simulator initialization. Receives AUV initialization file from the mission control.

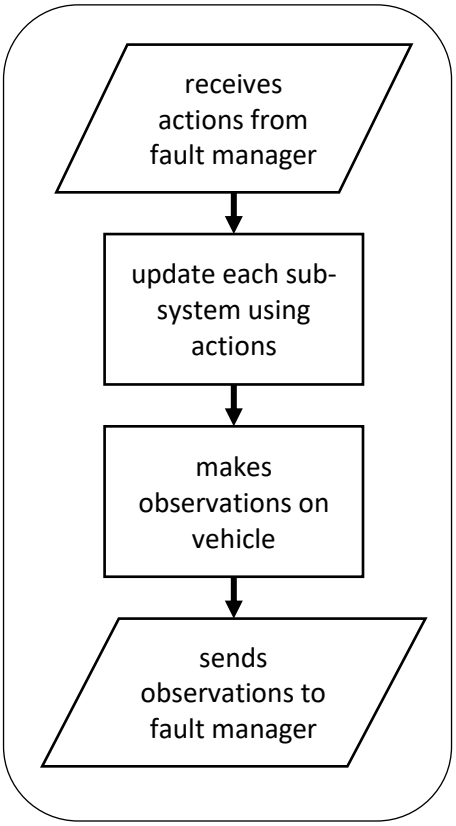

Figure 3-3b AUV simulator update for each timestep.

At each timestep (see Figure 3-3b) the simulator receives new actions from the fault manager. Then, the sub-system response to these actions are simulated and the AUV simulation model is updated. The simulator then acquires sensor measurements and processes these into observations. These observations are then sent back to the fault manager for inference (deliberation). The sensor measurements are taken from preset log files either generated or real-world bathymetry.

The vehicle simulator passes the observations to the POMDP-based fault manager which then returns actions for the vehicle to enact (execute).

3.2 POMDP

The partially observable Markov decision process (POMDP) is the fault manager's deliberative stochastic model-based schema. It models the vehicle's state, possible actions, and observations. A POMDP system was chosen for a number of reasons.

1. It captures the partial observability of the vehicle and its environment. Due to sensor limitations the vehicle does not know its state well. Additional information can be inferred from fusing and processing multiple sensors, but these have an associated error (noise). The POMDP maintains a probability distribution, across all possible states, to determine what the most likely one(s) may be.

2. With Bayesian statistics inherent in the POMDP, it is possible to update the state distribution from the conditional probabilities of the observation (in the sense of measurements, not judgements) function (sensor model) and prior actions.

3. Markov's assumption asserts that the future state is independent of past ones and only dependent on the present one (see section 2.8). This simplifies the historical state information to retain.

4. The state transition is modelled as a probabilistic function to capture the uncertain nature of the transition given an action (see Section 2.10).

5. The POMDP models both independent and dependent sub-systems and determines actions for the vehicle.

The fault manager implements its stochastic model by building and solving a POMDP model (see Figure 3-4). The POMDP model is generated from an input model file with information on *actions, states, observations* and the probability of *making-an-observation* (given state and action), *transitioning-between-states* (given prior state, actions) and reward functions.

The probabilities and rewards were set based on developer knowledge. A more accurate representation of an AUV would require detail knowledge of the sensors (to create higher-fidelity models) and environment. For more information on how the POMDP model was built see Section 4.3.2.

Once the model was generated, a solver was implemented to develop the policy. The Q-MDP [64] solver was implemented here. Section 2.11.1.8 covers Q-MDP in more detail. The Q-MDP operates by first solving an MDP and then updating the policy to account for

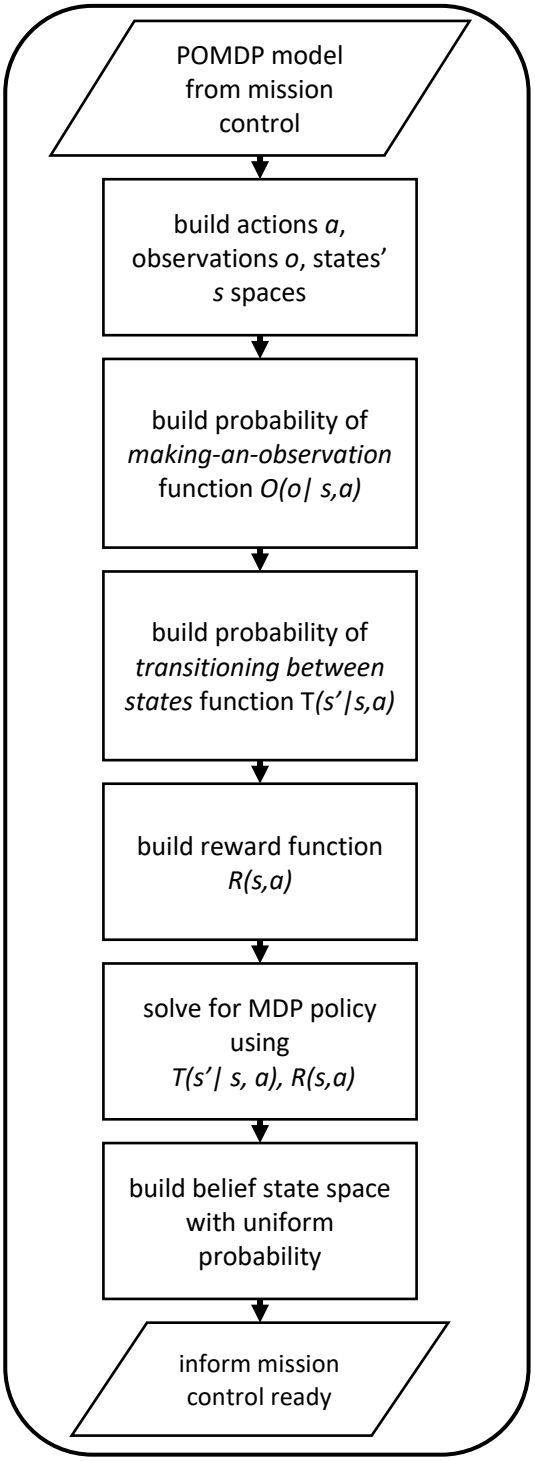

Figure 3-4 Fault manager initialization. Reads in POMDP model file given by mission control.

the partial observability given the state belief for each timestep. The POMDP would then generate an initial belief state (uniform probability over all possible states) and then informs the mission control to begin the simulation.

Over the course of the simulation (see Figure 3-5), at each timestep the simulated AUV would generate observations and pass them to the fault manager. These observations updated the belief state of the vehicle model, which using the policy, would determine an action. Then, the action would be performed by the simulated AUV and the process would repeat until the simulation ends.

The simulation ends when there is no new simulated sensor data passed to the AUV. Additionally, if an abort is determined by the POMDP the vehicle will surface with the remaining power. If no depth sub-system was initialized, the abort triggers the end of the simulation.

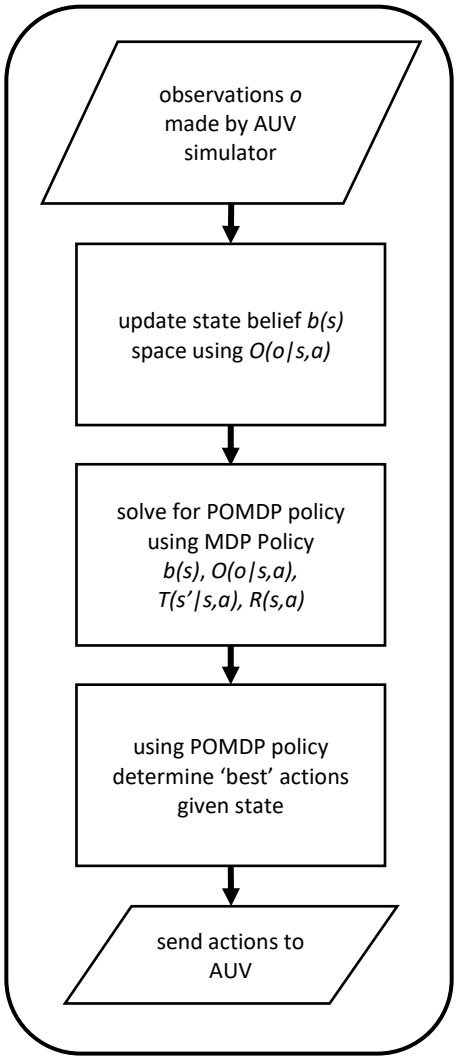

Figure 3-5 Fault manager update.

As previously mentioned, the solver implemented was the Q-MDP method. In the next section the use of this will be explored further.

3.2.1 Q-MDP

A Q-MDP solver [64] [80] [89] was chosen to generate the policy for several reasons.

1. Value iteration of the Q-MDP is of the same complexity as MDPs [85]. This can greatly reduce solving complexity compared to solving a POMDP directly (polynomial, versus exponential time to solve).

2. The solver handles large state and observation spaces which are necessary for the model.
3. The model designed for the proof-of-concept did not include actions that the Q-MDP would discriminate against due to its solving method.
4. Q-MDPs are well equipped to deal with real-world applications, like an AUV.

Value iteration [64], policy iteration [78] [79], and witness algorithms [71] [80] [63] were not chosen due to their inability to solve large state spaces. The AMDP [64] [84] requires a computation intensive conversion between the POMDP and MDP spaces. The point-based value/policy iteration [81] [83] and Perseus [82] algorithms were not implemented as the Q-MDP was found sufficient and these offered no immediate foreseen benefit. (see Section 2.11.1). POMDPs have many other solvers and future work should compare solvers to determine the optimal ones for AUVs. However, for this thesis' purpose the Q-MDP was deemed sufficient.

The Q-MDP solver [64] [80] [89] [85] solves the MDP first. Then, at each new timestep, the Q-MDP policy was solved by combining the vehicle belief state with the MDP policy and value function. Then, the Q-MDP policy determines the vehicle actions.

This POMDP fault management and AUV simulator were implemented as ROS nodes in C++.

4 IMPLEMENTATION

The proof-of -concept fault manager and AUV simulator framework was implemented as four ROS nodes: Mission Control, Logger, AUV Simulator, and the POMDP (fault management system). These four stand-alone nodes communicate through publishing and subscribing to messages. Their interconnections are shown in the ROS rqt graph (see Figure 4-1) which shows the nodes' messages that are published and subscribed to.

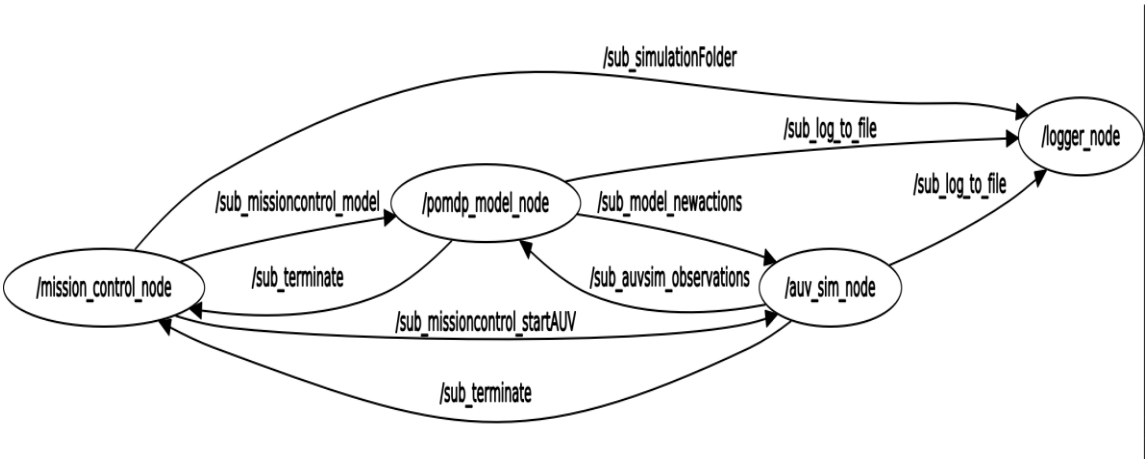

Figure 4-1 ROS rqt graph of the AUV fault management system developed.

The startup node is the mission control which sets up the other nodes.

4.1 Mission Control

The mission control node manages the execution of multiple distinct simulations run in sequential order (see Figure 4-2). The mission control file (see Appendix B - Mission Control File) contains the file with the POMDP model (see Appendix A – POMDP Model File) to be built, the AUV simulation initialization file (see Appendix C – Table) and the path where the simulation results are written.

Once the mission file has been read and parsed, the mission control sends the results folder path to both the logger and the POMDP nodes. Then, after a short wait, to ensure the message was received by the logger and that it has generated the files, the input model file is broadcast to the fault manager (POMDP node) to build and solve the POMDP model. Then, the POMDP node publishes a message, which the mission control node subscribes to, that a policy has been generated (if it failed to generate a policy the simulation will terminate). The mission control then sends the AUV simulation initialization file path to the AUV simulator node. Once the simulation is

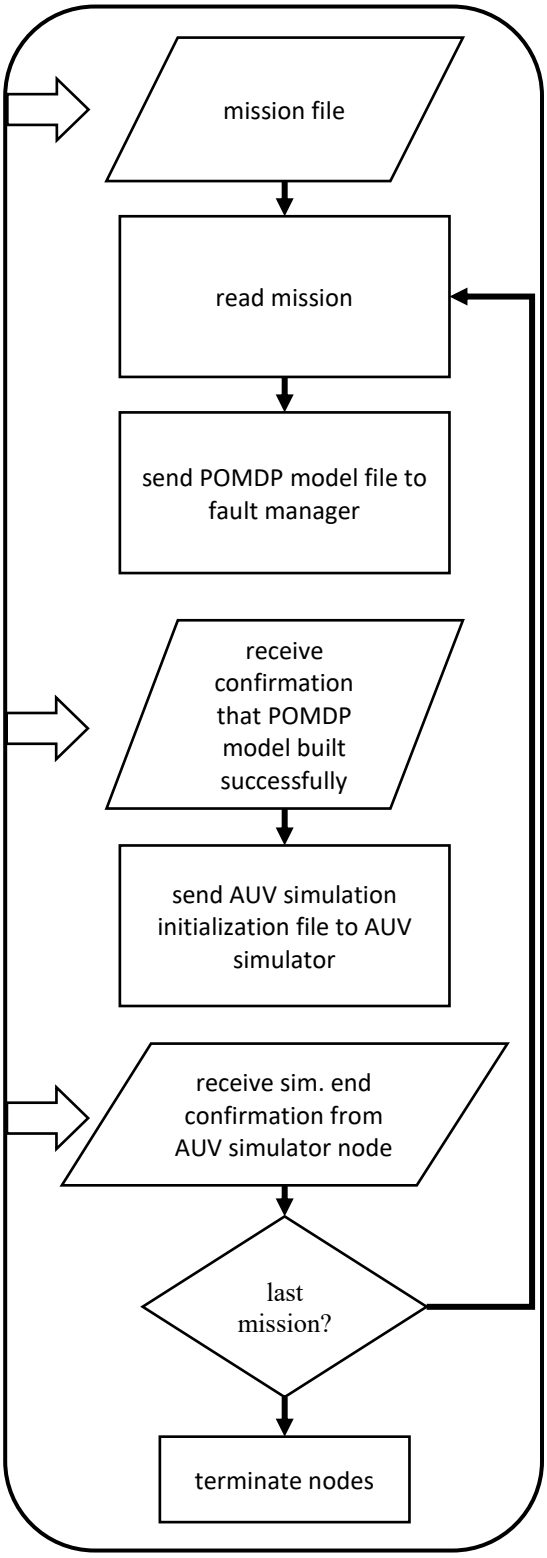

Figure 4-2 Mission control execution steps

complete the AUV simulator node publishes a *finished* message, subscribed to by the mission control node, which then reads the next mission file in the queue. If there are no more in the queue, the mission control node shuts itself down and terminates the other three nodes.

4.2 Logger

The logger generates and updates results files from the AUV simulation. This node starts when it receives the path to the simulation folder (location where the result files of the current simulation are stored) and generates the results files. Note, existing results files in the folder will be overwritten. Each new simulation should have its own results folder.

Over the course of the mission, the logger receives messages from other nodes and appends a message as a new line in the selected file. The logger updates these files until either a new simulation folder is received, or it is terminated by the mission control.

4.3 POMDP

The POMDP is the fault management system and operates in three modes: the first builds the POMDP model from the input model file, the second solves for an initial policy of the MDP model, and finally, it updates its belief of its joint-state distribution and solves for the Q-MDP policy using observations from the AUV simulator.

4.3.1 Actions, observations, and states

A POMDP model comprises of actions, states and observations. A group is used to demonstrate a set of values that are mutually exclusive to one another that describe an action, state or observation. For example, a group pertaining to the altitude could include altitude low (meaning the vehicle is too close to the seabed), altitude ok (meaning the altitude of the vehicle is within the desired range), and altitude high (meaning the vehicle is too far from the seabed). This group of values cannot exist simultaneously (i.e. altitude is low and high). Combinations of the various groups

describe the extent of the observation, state, and action space, these are referred to as joint-actions, join-observations, and joint-states.

Each AUV sub-system was modelled with its own groups for actions, observations and states. These were then combined in later simulations by created conditional dependencies between the probability functions and reward functions of the model. In Table 4-1, Table 4-2 and Table 4-3 these groups are described.

Table 4-1 Possible *action* variables (groups)

sub-system	action group	possible actions	description
depth	fin position	DEFLECT_NONE DEFLECT_DOWN DEFLECT_UP	When there is *no* deflection the AUV's pitch is constant. Deflection *up* increases pitch while deflection *down* decreases pitch.
power-management	power-management mode	POWER_NORMAL POWER_SAVING_MODE ABORT	When in *power saving mode*. Energy consumption is reduced by a set percentage from the initialization file. When the mode is set to *abort* the vehicle will end the mission by surfacing (unless no depth sub-system is present in which case it will end the simulation upon calling abort).

Table 4-2 Possible *observation* variables (groups)

sub-system	observation group	possible observations	description
depth	altitude	ALTITUDE_OK ALTITUDE_LOW ALTITUDE_HIGH ALTITUDE_UNKNOWN	The vehicle operates using altitude keeping. In the initialization file the minimum and maximum altitude thresholds are set. The vehicle is observed as *low* when the altitude is less than the minimum and *high* when greater than the maximum. The altitude is *ok* when within these thresholds. The altitude is *unknown* when: ● The vehicle cannot achieve DVL-lock due to the altitude being greater than the DVL range (set in initialization file). ● The vehicle cannot achieve bottom-

sub-system	observation group	possible observations	description
			lock due to the pitch angle being too great in either direction (i.e. greatly up or greatly down). ● A cascade failure from *very low/ critical* energy capacity has caused the sensor 'failure'.
	vehicle depth	DEPTH_GOOD DEPTH_SHALLOW DEPTH_DEEP	Depth is *shallow* when vehicle is less than a minimum depth value set in the initialization file. Vehicle is too *deep* when it exceeds its crush depth (also set in the initialization file).
power-management	pitch change	PITCH_UNCHANGING PITCH_INCREASING PITCH_DECREASING	The pitch is *unchanging* when the fin mode has no deflect. The pitch is increasing when *deflected up* and is *decreasing* when deflected down.
	pitch	PITCH_GREATLY_UP PITCH_UP PITCH_LEVEL PITCH_DOWN PITCH_GREATLY_DOWN	When the pitch has an angle of 0 it is considered *level*. If pitch is greater than 0 it is *up*. A threshold is set on the maximum pitch to ensure bottom-lock is not lost. If the vehicle's pitch exceeds this threshold (set in the initialization file) it is considered *greatly up*. Negative pitch is *down* and follows the same logic as up for *maximum* and *greatly down*.
	energy capacity	CAPACITY_OK CAPACITY_LOW CAPACITY_VERYLOW CAPACITY_CRITICAL	Energy is *ok* when capacity exceeds a specified percentage (set in initialization file). Energy is *low* when less than the threshold. Energy is *very low* when less than half the threshold. Energy is *critical* when less than a quarter the threshold.
	power usage	HOTEL_LOW HOTEL_OK HOTEL_HIGH	The hotel load is the energy consumption per timestep. A lower and higher rate is set in the initialization file. The power usage or energy consumption is *low* when below the lower rate and *high* when above the high rate. The fault manager uses this to increase the likelihood of a low energy state over time.
	est. mission time	FIRST_QUARTER SECOUND_QUARTER THIRD_QUATER ALMOST_DONE	Estimates the mission into 1/4 portions. If low energy is detected in early mission time (i.e. first 1/4) the vehicle will abort as this is likely to be an issue with the on-board power supply.

sub-system	observation group	possible observations	description
			If the vehicle is in the last quarter, it will try and remain in normal energy usage unless it hits the very low energy threshold as this is seen as being close to being finished.
	power-man. mode	USAGE_NORMAL	Current power usage mode: the vehicle is set to. *Power saving* reduces energy used at each timestep while *aborting* causes the vehicle to surface and end simulation.
		POWER_SAVING	
		ABORTED	

Table 4-3 Possible state variables (groups)

sub-system	state group	possible states	description
depth	vehicle depth	DEPTH_GOOD	Depth is *deep* when either the crush depth is exceeded, or the altitude is measured to be too low. Depth is *good* when altitude is within thresholds, depth is within range, and low energy state is achieved. Depth is *shallow* when the vehicle either observes that the vehicle is too close to the surface, or the altitude is too high. The depth is also considered shallow if the altitude is unknown and the pitch is within its threshold limits as to not cause loss of bottom-lock and a cascade failure has not caused sensor failure.
		DEPTH_SHALLOW	
		DEPTH_DEEP	
	vehicle pitch	PITCH_GREATLY_UP	When pitch has an angle of 0 it is considered *level*. If pitch is greater than 0 it is *up*. If at the threshold for the pitch the vehicle's pitch is *maxed*. The threshold is set in the initialization file and is determined by the fault manager as the maximum angle the vehicle can achieve before losing bottom-lock due to excess pitch angle. If the vehicle's pitch exceeds this threshold it is considered *greatly up*. Negative pitch is *down* and follows the same logic as up for *maximum* and *greatly down*.
		PITCH_UP_MAX	
		PITCH_UP	
		PITCH_LEVEL	
		PITCH_DOWN	
		PITCH_DOWN_MAX	
		PITCH_GREATLY_DOWN	
power-man.	energy capacity	POWER_GOOD	Energy is *good* when capacity exceeds a specified percentage (set in initialization file).
		POWER_LOW	

sub-system	state group	possible states	description
		POWER_VERYLOW POWER_CRITICAL	Energy is *low* when less than the percentage threshold. Energy is *very low* when less than half the percentage threshold. Energy is *critical* when less than a quarter the percentage threshold.
	power-man. mode	USAGE_NORMAL POWER_SAVING ABORTED	Current power usage mode the vehicle is set to. Power saving reduces energy used each timestep while aborted causes the vehicle to surface and end simulation.
	est. mission time	FIRST_QUARTER SECOUND_QUARTER THIRD_QUATER ALMOST_DONE	Estimates the mission into quarter portions. If low energy is detected in early mission time (i.e. first quarter) the vehicle will abort as this, it is likely there is an issue with the on-board power supply. If the vehicle is in the last quarter, it will try and remain in normal power usage mode unless it hits the 'very low' energy threshold as this is seen as being close to being finished. Note: in a real mission completion would be measured by completed tasks but for our purposes this was simulated as time. The mission time is calculated through an elapsed time that is measured at each timestep. This is to better simulate a continuous time vehicle despite the simulator and fault manager working in discrete time.

4.3.2 Building the POMDP

The fault manager builds the POMDP model when an input model file is received from the mission control. This model is separated into three parts (Figure 4-3). The first part is basic model information such as the name (to differentiate between other models in the results folders), horizon, analysis type, and reward function discount. Horizon is set to '1' meaning at each timestep the vehicle chooses a new action, rather than plan for actions in advance. The Q-MDP solver was deemed sufficient for the analysis, but the system could be expanded to include other solvers for future development. The discount is for the calculation of the utility function in the Bellman equations (see section 2.10 and equation 2-9).

Another parameter is the modification value for the probability of *transitioning-between-states* function and probability of *making-an-observation* function. These multiply the value against all probabilities set for these functions. For example, a value of 90% would reduce all probabilities set by the statements by 10%. Later, these are used for simulations with reduced probability functions (see section 5.1.4).

Next, the action, observation and state groups are read and used to generate all possible combined joint-actions, joint-observations and joint-states.

#Comment designated by #
Model: model_name
Horizon: 1
Discount: 0.9
Analysis: Q-MDP
ModTrans: 1.0
ModObservation: 0.9

Action Groups: U
AG_1: A_1, .. , A_i
...
AG_U: B_1, . ., B_j

State Groups: V
SG_1: C_1, .. , C_k
...
SG_V: D_1, . ., D_l

Observation Groups: W
OG_1: E_1, .. , E_m
...
OG_W: F_1, . ., F_n

Observation Probabilities statements
O_1 : Joint Actions : Joint States : Joint Observations : Probability
…
O_x : Joint Actions : Joint States : Joint Observations : Probability

Transition Probabilities statements
T_1: Joint Actions : Joint Start State : Joint End State : Probability
…
T_y: Joint Actions : Joint Start State : Joint End State : Probability

Rewards statements
R_1: Joint Actions : Joint State : Reward Value
…
R_z: Joint Actions : Joint State : Reward Value

Figure 4-3 Example POMDP Model File

The final part of the POMDP model set the probability and reward functions. The probability functions probability of *making-an-observation* and the probability of *transitioning-between-states* must sum to one (see equations 4-1, and 4-2 respectively).

The values for the probabilities and reward functions are set via statements in the POMDP model file (see Figure 4-4).

$$\sum_{N}^{i=1} o_i | \, s, a \; == 1.0 \tag{4-1}$$

$$\sum_{N}^{i=1} s'_i | \, s, a \; == 1.0 \tag{4-2}$$

Updating Probability of Observation Function

For Each Statement of Observations:
    For Each Joint-Action that is True
        For Each Joint-State that is True
        For Each Observation
        If True
            O(o| s, a) *= probability
        If False
            O(o| s, a) *= 1 – probability

Updating Probability of Transition Function

For Each Statement of Transitions:
    For Each Joint-Action that is True
        For Each Joint-Start-State that is True
        For Each Joint-End-State
        If True
            T(s'| s, a) *= probability
        If False
            T(a'| s, a) *= 1 – probability

Updating Reward Function

For Each Statement of Rewards:
    For Each Joint-Action that is True
        For Each Joint -State that is True
            R(S,A) += reward

Figure 4-4 Pseudo-code for POMDP probability of observation, transition and reward functions update for given statements in the POMDP file

Once the probability of observation, transition, and reward functions are set, the model is sent to the solver to generate a policy.

4.3.3 Solving the POMDP

The built POMDP model is then passed to the Q-MDP solver [64] [80] [89] [85] . This solver solves the model as an approximated MDP rather than a POMDP. It can only

do so by assuming the state is known. The MDP is then solved using a value iteration method. The resulting policy and value function are stored for later use by the Q-MDP.

4.3.4 Execution of the POMDP

Once an MDP policy is successfully attained, the POMDP informs the mission control that it is ready, and then waits for observations from the AUV simulator. These observations are used to update the belief probability distribution of the vehicle's joint-state space. Initially, the state belief distribution is uniform with all possible joint-states being equally probable. In each subsequent timestep, equations 4-3 and 4-4 (Section 2.11) update the belief distribution.

$$b'(s') = \eta\, O(o|s', a) \sum_{s \epsilon S} T(s'|s, a) b(s) \tag{4-3}$$

$$\eta = \frac{1}{\sum_{s' \epsilon S} O(o|s', a) \sum_{s \epsilon S} T(s'|s, a) b(s)} \tag{4-4}$$

Once the belief distribution is updated the previously solved MDP policy is run through the Q-MDP solver that updates the policy to account for the belief space of the vehicle (see Figure 4-5). An action is chosen based on the updated Q-Policy and sent back to the AUV simulator node for execution.

Algorithm Q-MDP:

R= reward function
T= Transition function
b(s) = Belief
Q= Utility function
N = number of joint-states
M = number of joint-actions
V = MDP Value Iterative Function
For each joint-state s_i:
For each joint-actions a_j

$$Q(s_i, a_j) = r(s_i, a_j) + \sum_{k=1}^{N} V(s_i) T(s'_i | s_i, a_j)$$

$$\text{Policy} = \underset{a}{\operatorname{argmax}} \sum_{j=1}^{M} b(s_i') Q(s_i, a_j)$$

Figure 4-5 Pseudo-code for the Q-MDP solver

4.4 AUV simulator node

The AUV Simulator node begins by reading in the initialization file (see Figure 4-6 and Appendix C – Table) passed to it by the mission control. This file has parameters for each sub-system modelled. Once the vehicle is initialized it asks the sub-systems to make observations. These observations are generated using simulated sensor measurements from static pre-set log files for each sub-system.

These observations are then sent to the fault manager. The AUV simulator waits until new actions are returned from the fault manager and then sends these to the appropriate sub-system for execution. Once the actions have been executed the vehicle will update its state and reads in the new sensor measurements from the log files and perform new observations.

```
#This is a AUV that has a depth system and power system
AUV_NAME: AUV_PowerDepth_Integrated
MISSION_TIME: 6000
NOISE: 0
ABORT_AT_COMMAND: 1
CASCADE_FAILURE: 1

#sub-system depth
DEPTH_FILE: /basicshallow.log
ALTITUDE_MIN: 6
ALTITUDE_MAX: 10
DVL_RANGE: 25
MAX_DEPTH: 35
MIN_DEPTH: 5
START_DEPTH: 2
NOISE_DEPTH: 2
MOVEMENT_MAG: 5
HIGH_ANGLE: 15
LOW_ANGLE: 2
ANGLE_CHANGE: 5


#sub-system power
POWER_FILE:  /power_basic63.log
POWER_STORED: 4600
POWER_RATES: 0.4 0.6
LOW_POWER_MODE: 0.75
```

Figure 4-6 AUV simulation initialization file example

The simulation ends when either the log files have no new input, the power management system aborts due to critical power and the depth sub-system is not modelled, when the vehicle has surfaced, or when the energy has been exhausted (value of 0).

4.4.1 Depth Sub-system

The depth sub-system predominantly handles changes to the vehicle depth and pitch angle. First, the vehicle depth must be within an acceptable range. This range spans the vehicle's minimum depth under the surface for safe operations and the

maximum depth before it reaches its crush depth (set in the initialization file). The seabed depth measurements are simulations and injected via a log file. The depth and other measurements are done via simulation; and are without a typical sensor model. This was done to simplify the system. In the cases of a additive noise a variance to the AUV depth are run through a Gaussian distribution with a variance typical of 100 meter depth sensor.

The vehicle is designed to maintain constant altitude above the seabed. The minimum altitude is set to avoid running into navigation hazards (rocks, debris, marine structures, wrecks, etc.) on the seabed. The maximum altitude is the distance the vehicle can be from the sea bottom for its sensors to still work as intended (i.e. altimeter, DVL, etc.). The vehicle model also accounts for DVL-lock (or bottom-lock) by modelling the altitude as unknown when the vehicle pitch or roll exceeds a maximum angle, the altitude exceeds the DVL range, or a cascade failure has caused the sensor to 'fail'. The maximum angle and DVL-range are pre-set in the initialization file.

The depth sub-system also considers the state of the vehicle pitch (Figure 4-7). The pitch can be nose-up, -down, or level. It also checks whether the pitch has a magnitude greater than a set maximum value. The system additionally observes if the pitch is changing over time. The vehicle pitch determines how the vehicle's depth changes over time. If the pitch becomes too large it can cause problems with the sensors since the vehicle (and sensor) is no longer aligned with the seabed. This is modelled by assuming that the vehicle loses DVL-lock.

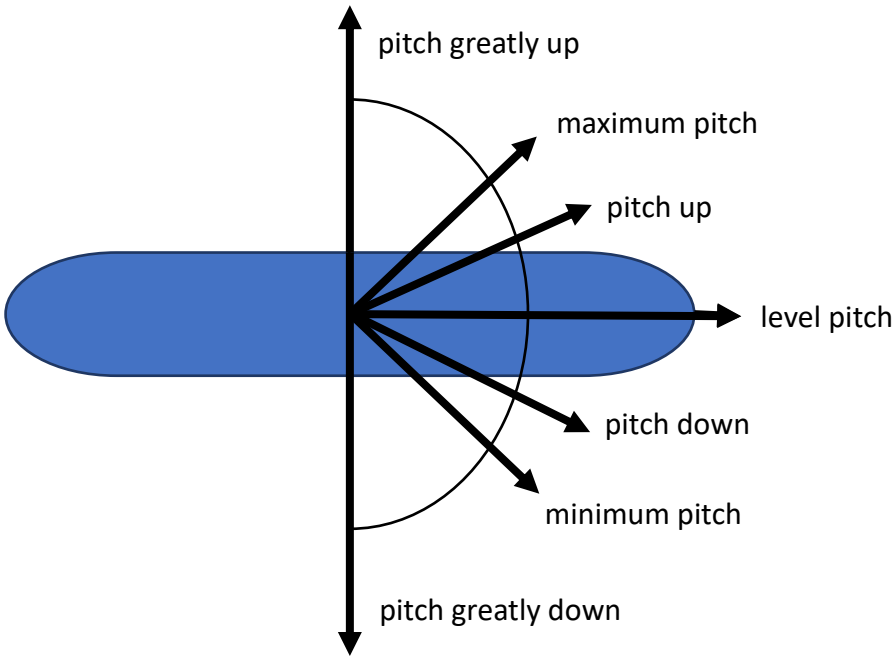

Figure 4-7 AUV pitch observation definitions. 'Up' is in the positive direction, 'down' is in the negative direction and 'level' is pitch = 0.

The AUV simulator assumes constant pre-set values for pitch angle changes; this is set in the initialization file (see Appendix C – Table). The depth sub-system has only one action group, changing its fin's deflection: up, down or no deflection (zero angle-of-attack). This mode is changed by the fault manager which updates the pitch accordingly (see Figure 4-8).

```
fin mode = 1 then pitch += change in pitch
         = 0 then pitch does not change
         = -1 then pitch -= change in pitch
```

Figure 4-8 Pseudo-code for pitch change

The AUV depth is changed by the vehicle 'flying' and the pitch affecting how the vehicle changes its depth in the water column. The depth change is affected by a fixed

incremental distance (set in initialization file) traveled each timestep scaled by the sin of the pitch angle (see equation 4-5).

$$vehicle\ depth\ +=\ -fixed_{distance}\ x\ sin(pitch\ angle) \qquad (4\text{-}5)$$

Additive Gaussian noise can be applied to the vehicle depth measurements to simulate noisy measurements (see equation 4-6). Although depth measurements in real systems are generally of good fidelity this was to demonstrate the POMDP's ability to account for noise in the sensing, actuation or interaction with the environment (sea states). The environmental seabed depth is read from an external file that is used to generate 'measurements' for the seabed depth at each timestep. The standard deviation for the simulations in the next section were given the range of 2 meters this is well within what can be attained for an off-shelf 100-meter depth sensor. The signal to noise ratio could not be modelled as a consequence of the noise being prescribed rather than measured.

$$measured\ depth\ = Gaussian(mean = actual\ depth\ of\ vehicle, stddev) \qquad (4\text{-}6)$$

Once a new depth has been calculated, observations are made on the vehicle state which includes the pitch, depth, and altitude (see Table 4-1, Table 4-2, Table 4-3 for more information). The speed is approximately 2 knots with timesteps being 2s (depending on initialization file).

The other sub-system in the AUV simulator was the power-management sub-system.

4.4.2 Power-Management Sub-system

The power management sub-system monitors the energy capacity and consumption along with estimating the remaining mission time and when to abort the mission if there is an energy shortage

The power management sub-system first determines how far along the mission is. It achieves this by estimating the remaining mission time and comparing it to estimated mission time set in the initialization file. This is a simplification, see Section 2.2.3 for typical energy modelling. The power management sub-system here uses the total from summing all three (3) power types (propulsion, payload sensor, vehicle equipment (equations 2-1 and 2-2)). For future consideration, in *power saving mode*, items like data logger, embedded processor and communications could be turned off or reduced to achieve lower power states.

The power management sub-system updates itself by reading two measurements from the power log for the simulation. These measurements are the energy consumed for the timestep and the elapsed time since the previous timestep (this can be greater than a single timestep). These measurements were generated pragmatically using a random distribution relative to an overall distribution.

The remaining energy on the vehicle is calculated based on the initial energy capacity (set in the initialization file) and the cumulative energy consumed over the simulation. The overall remaining energy, relative to requirements for the remaining mission, is assessed to be one of four observations: *ok*, *low*, *very low*, and *critical* low'. The AUV remaining energy observations are based on whether the percentage of remaining energy (remaining / total energy) (Table 4-4) is below a pre-set rate (for example rate of 40% means *low* 40%, *very low* 20%, *critical* 10%).

Table 4-4 Energy capacity observation definitions

ok	>	rate x total capacity
low	≤	rate x total capacity
very Low	≤	rate / 2 x total capacity
critically low	≤	rate / 4 x total capacity

The rate of energy consumption is determined by how much energy was consumed during the timestep (measured from the simulation log file) compared to the time elapsed (see equation 4-7). Then, the energy rate is assessed to be high, normal, or low compared to pre-set rates in the AUV initialization file.

$$lower\ rate\ x\ \frac{total\_capacity}{total\_time} \leq \frac{used_{capacity}}{elapsed_{time}} \leq\ upper\ rate\ x \frac{total\_capacity}{total\_time} \qquad (4\text{-}7)$$

The mission phase is determined by the estimated total mission time (set in the initialization file) and elapsed time. The elapsed time is determined from an input value in the log file that gives the time elapsed over the timestep. The elapsed time measurement was to simulate a real-world vehicle that would monitor elapsed time between discrete timesteps. The vehicle then determines if the mission is currently in it's *first, second, third,* or *final quarter* of the mission phase.

The fault manager can change the power-management mode of the vehicle. The first mode is for *normal* operations. The vehicle nominally operates in this mode. The second mode is *power-saving* which reduces the energy consumed each timestep at a fixed (set in the initialization file) rate. For example, if the low power rate is 75% then when this mode is engaged all energy consumption is reduced by 25% of what it was originally. This simulates the vehicle turning off unnecessary sensors or limiting computation processes. The last mode is for the vehicle to *abort* the mission.

When the simulator only runs the power management sub-system (i.e. depth sub-system not initialized and therefore, not engaged) simulation, once the capacity reaches *critical* the vehicle aborts and the simulation terminate. In the jointly-dependent combined power management and depth simulations, the vehicle starts to rise in the water column when the energy is *very low* and will surface when it reaches *critically low*. The maximum pitch restriction is lifted so the vehicle can rise faster.

The vehicle engages the *power-saving* mode at critically low energies. It will not engage this mode if the energy is only *low* and the vehicle is in the last quarter of its mission (since there is a good likelihood it can complete its mission before the energy capacity drops further). This *power-saving* mode simulates the vehicle turning off unnecessary processes/sensors or limiting their power consumption. The *power-saving* mode will always engage for *very-low* and *critical* observations.

To test the proof-of-concept fault manager and AUV simulator framework several scenarios and POMDP models were developed. Each sub-system was tested independently. Several scenarios spanning different conditions (environmental and vehicle configurations) were developed and simulated.

A subset of simulations for the depth sub-system have additive Gaussian noise on the depth measurements. These were paired with modifications to the POMDP models where the probabilities of the transitioning-between-states and making-an-observation functions had their pre-set probabilities in the model build reduced by 10%, 20% and 40%. The objective was to explore the effects of noise on the system and how the fault management system responds to loss of certainty in its states given actions and observations from the noisy sensors. The actions and reward function remained unchanged.

The depth and power-management POMDP models were combined, without modification, for a simulation that solved for their joint-states, joint-observations, and joint-actions but were non-interacting with each other. This was to show that the combined POMDP model works the same as the individual models if the actions, observations, and states are independent.

A POMDP model of the two sub-systems was developed where the two sub-systems were dependent on one another. The scenario of dependency selected was such that

given the state of the power management system the depth sub-system would change its desired actions. This demonstrates how the model addresses interactions between sub-systems for an integrated fault management system.

Finally, an interdependent model was generated that had the depth and altitude measurements fail when the power capacity reached *very low* or *critical* simulating a cascade failure.

5 SIMULATIONS

To validate the POMDP fault management model, three simulation sets were performed through the AUV simulator. The first set was with just the depth sub-system. These simulations were performed for three environments with simulated and real data. Additionally, a comparison between the reduction of the POMDP probability functions for *transitioning-between-states* and *making-an-observation* were performed with one of the depth sub-system's simulation environment model. The second set was with just the power-management sub-system. This model was run with different initial energy capacities to assess response to different energy levels. The final simulation set was a combined model with both the depth and power-management sub-systems engaged.

For a complete list of simulations, their figures and initialization files please see Appendix C – Table.

For the depth and power-management independent sub-systems, a Figure 5-1 type plot is produced to highlight, for each simulation, the belief states changing with time. This was not done for the combined models of depth and power management due to the joint-state space being too large to easily graph (> 1000 possible states for all permutations).

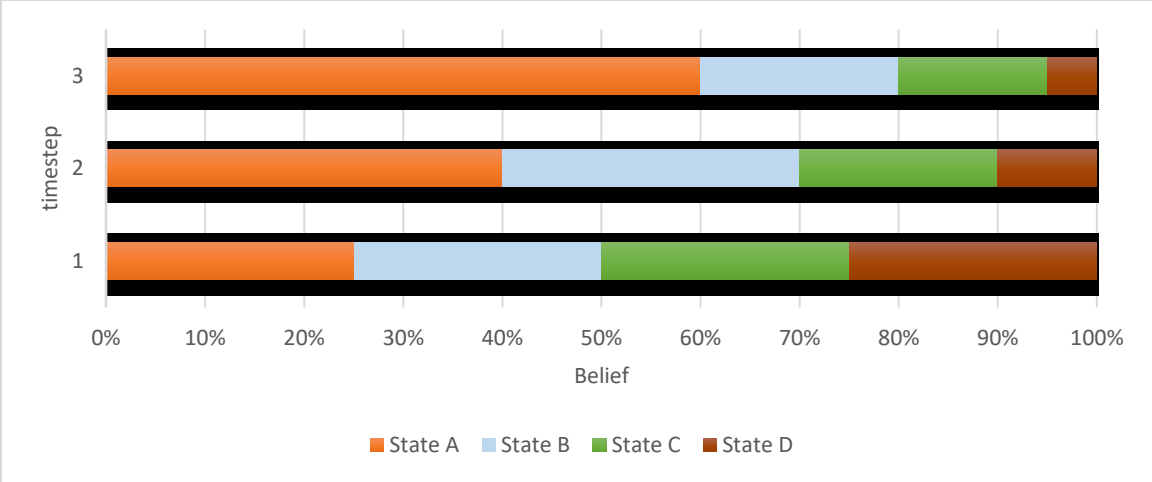

Figure 5-1 Simplified belief state depiction example. There are four possible states: A, B, C and D. At timestep 1 the probability distribution is uniform over all states (i.e. they are all equally 25% probable). At timestep 2 state A has the highest probability at 40%, state B at 30%, state C at 20 % and state D at 10%. At the final timestep state A has a 60% probability of being the actual state followed by state B at 20%, state C at 15% and state D at 5%.

The first set of simulations were conducted with the depth sub-system only.

5.1 Depth sub-system simulations

The depth sub-system tests were conducted for three environment types: shallow water, deep water that exceeds the AUV's operational depth, and waters over an uneven seabed that varied rapidly compared to the AUV's response time. These three sets span limiting cases of underwater bathymetry that an AUV may encounter to test the POMDP fault manager depth sub-system. The same POMDP depth model was used for all simulations. For each of the environment models using real-world measurements the data used was obtained from Bedford Basin in Halifax, Nova Scotia.

Another series of tests were conducted where the probabilities of making-observations and transitioning-between-states functions were augmented and another with additive Gaussian noise to the depth sensor measurements was combined with the augmentation.

To demonstrate the vehicle's responses in the water column from the fault management system a series of four (4) graphs are presented. They represent the vehicle and seabed depths as the vehicle senses the seabed, vehicle altitude-keeping, vehicle pitch angle, and fin mode. All depth simulations were run with the vehicle in altitude-keeping mode and depth changes were realized with the vehicle 'flying' (i.e. using changes in its pitch to change its depth). The altitude is lost when the vehicle loses DVL-lock which is represented by a DVL sensor. From here on bottom-lock will refer to DVL-lock. When the DVL cannot sense the bottom due to the vehicle's altitude or pitch the vehicle loses bottom-lock. Changes in pitch are affected by the fin mode which is driven by the fault manager. The change in pitch, if required at any timestep, is a set incremental angle change (initialization file, Appendix C – Table) dictated by the fin mode. Additionally, for each simulation the evolving belief state over the course of the mission (see Figure 5-1) is plotted.

The first group of tests were for the shallow water environment.

5.1.1 Shallow water

The first series of tests were for shallow water where the AUV's DVL can sense the seabed and achieve bottom-lock. The first was a simple shallow water simulated environment model with a gradual incline in the seabed.

5.1.1.1 SIMULATED SHALLOW-WATER ENVIRONMENT MODEL WITH GRADUAL INCLINE SEABED

The environment model had gradual depth incline (approximately 20 °) although slight variation to the incline occurs in the seabed which drives the altitude-keeping AUV to rise. The AUV simulation was given a DVL range of 20 meters, and a high pitch angle of 15°. The altitude-keeping range is 8-12 meters. See Appendix C – Table for the initialization file. These parameters were chosen so the vehicle can generally see the seabed but is limited in pitch angle making it critical that it responds quickly to changes in the seabed to avoid collisions with it.

In Figure 5-2 (a), the vehicle initially dives until it reaches the desired altitude however; the incline of the seabed quickly forces the vehicle upwards. In Figure 5-2 (b) the altitude is initially unknown due to the seabed being outside the DVL's range however, as it dives it is able to get altitude readings when it comes within range of the DVL. The fin mode can be seen in Figure 5-2 (d) correlating to changes in pitch as it is driven by the fault manager. During timesteps 30 – 40, the seabed forces the vehicle upwards and the response can be seen in the fin mode correctly pointing upwards. During this time there is slight oscillation in the pitch as the vehicle tries to rise faster to avoid the seabed while maintaining bottom-lock by not having the pitch exceed its maximum (15°).

This simulation demonstrated the fault manager could drive the vehicle and prevent the vehicle from colliding with the seabed – which would be a failure. It was able to achieve altitude, and despite some oscillations, prevent the pitch from exceeding its limits.

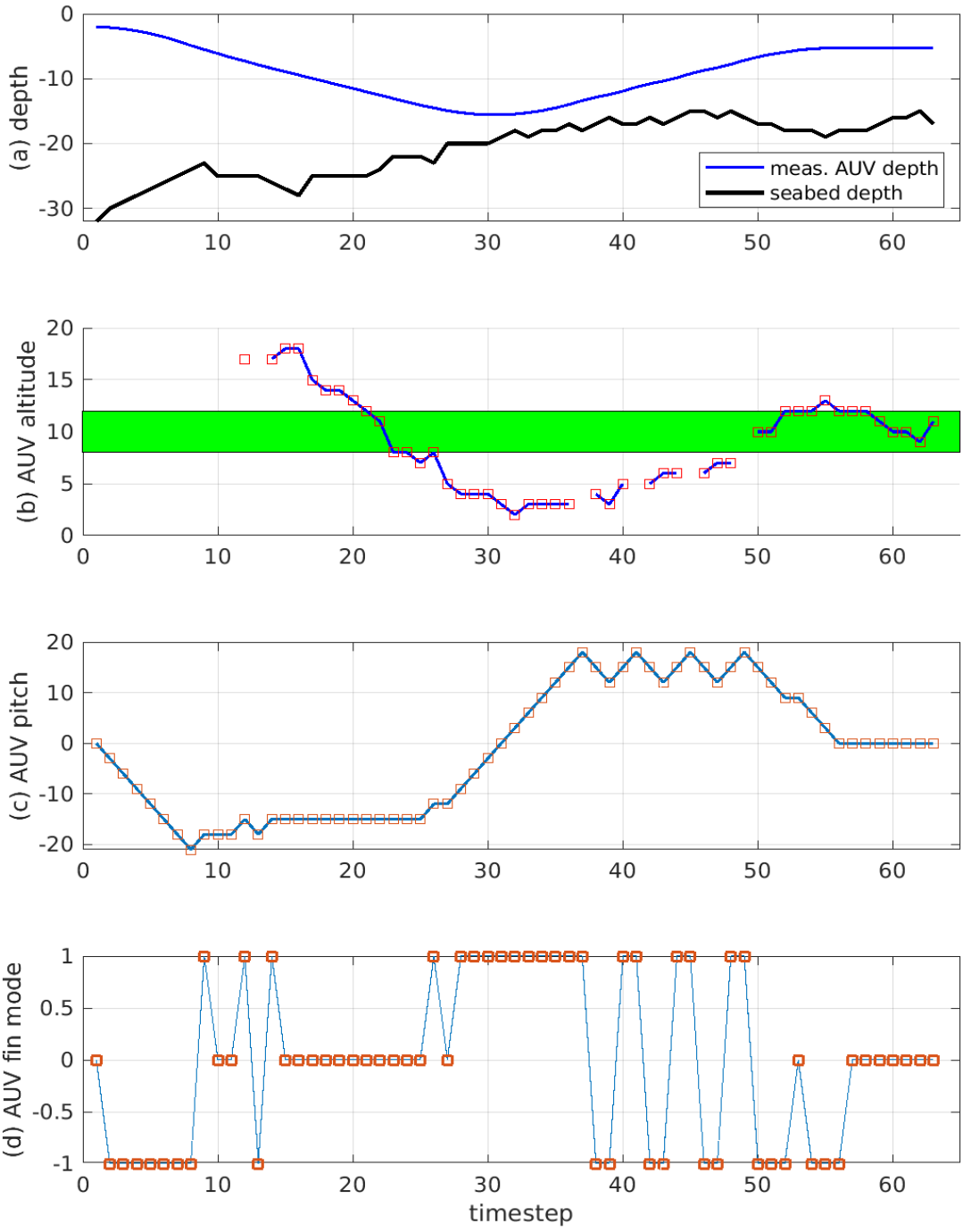

Figure 5-2 Simulated shallow-water environment model with gradual incline seabed: AUV performance with fault manager (DVL range = 20m, altitude-keeping range = 8 – 12 m, high pitch = 15°). (a) The altitude-keeping AUV depth correctly tracks the seabed and changes depth based on vehicle pitch changes. (b) When the vehicle DVL is unable to achieve bottom-lock no altitude value is plotted. (c) Pitch is as expected given the depth changes. When pitch is < 0 the vehicle dives and when > 0 it rises. (d) Fin mode in driven by the fault manager to increases (+1), decrease (-1) or hold pitch (0). The desired altitude range is given by the green band in (b).

As shown in Figure 5-3 initially, the probability for all belief states are uniform distributed. During the oscillations (timesteps 35-45) the edge case of the vehicle trying to rise faster causes some oscillations in the pitch. We can see this in Figure 5-3 from the most confident state changing over a cycle of about 4 timesteps for each oscillation. From timesteps 56+, with each timestep the vehicle becomes more confident of its state.

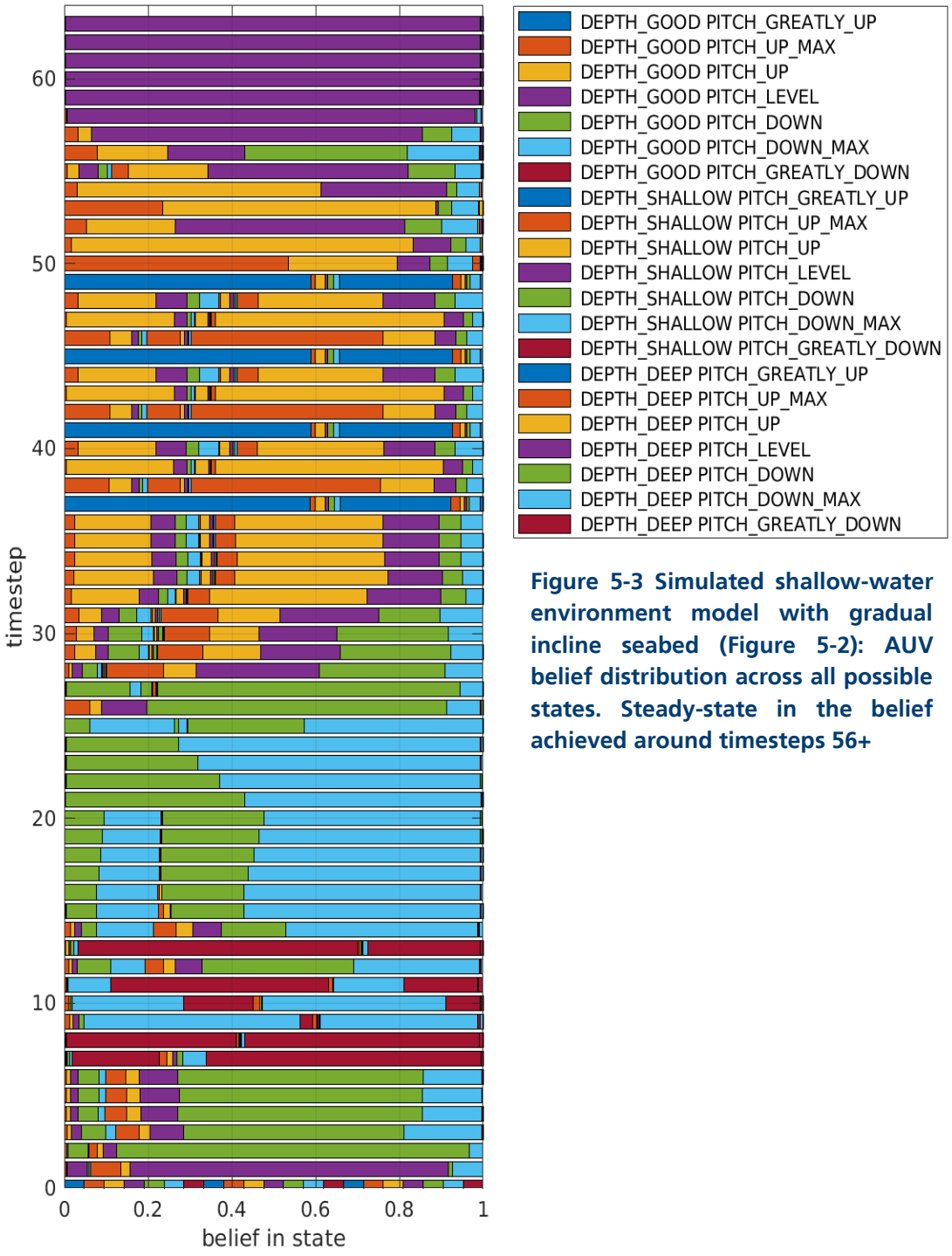

Figure 5-3 Simulated shallow-water environment model with gradual incline seabed (Figure 5-2): AUV belief distribution across all possible states. Steady-state in the belief achieved around timesteps 56+

This simulation showed the fault manager successfully controlling the vehicle and avoiding collision with the seabed. There are some issues with oscillations in the pitch due to conflicting diving parameters which prevent the pitch from becoming too great. However, this did not impede the vehicle from completing its mission.

The second shallow water environment simulation added noise to the depth sensor measurement.

5.1.1.2 SIMULATED SHALLOW-WATER ENVIRONMENT MODEL WITH GRADUAL INCLINE SEABED WITH ADDITIVE NOISE APPLIED TO DEPTH SENSOR MEASUREMENT

The shallow-water environment model from the previous simulation was repeated with an additive Gaussian noise applied to the depth sensor measurement. The noise was given a ±2meter standard deviation from the actual measurement which is line with real-world depth sensors depending on quality of sensor (for example, a hundred-meter depth sensor can have up to ±5-meter accuracy) . For more information please see section 4.4.1. The AUV simulation was given a DVL range of 20 meters, and a high pitch angle of 15 °. The altitude-keeping range is 8-12 meters. See Appendix C – Table for the initialization file. These parameters were chosen to match the previous simulation.

In Figure 5-4 (a), the depth measurements have an additive noise induced. This noise is given a Gaussian distribution with a ±2meter standard deviation. In Figure 5-4 (b) the altitude is initially unknown due to the seabed being outside the DVL's range however, as it dives it is able to get altitude readings when it comes within range of the DVL. During timesteps 30 – 40, the seabed forces the vehicle upwards and the response can be seen in the fin mode correctly pointing upwards.

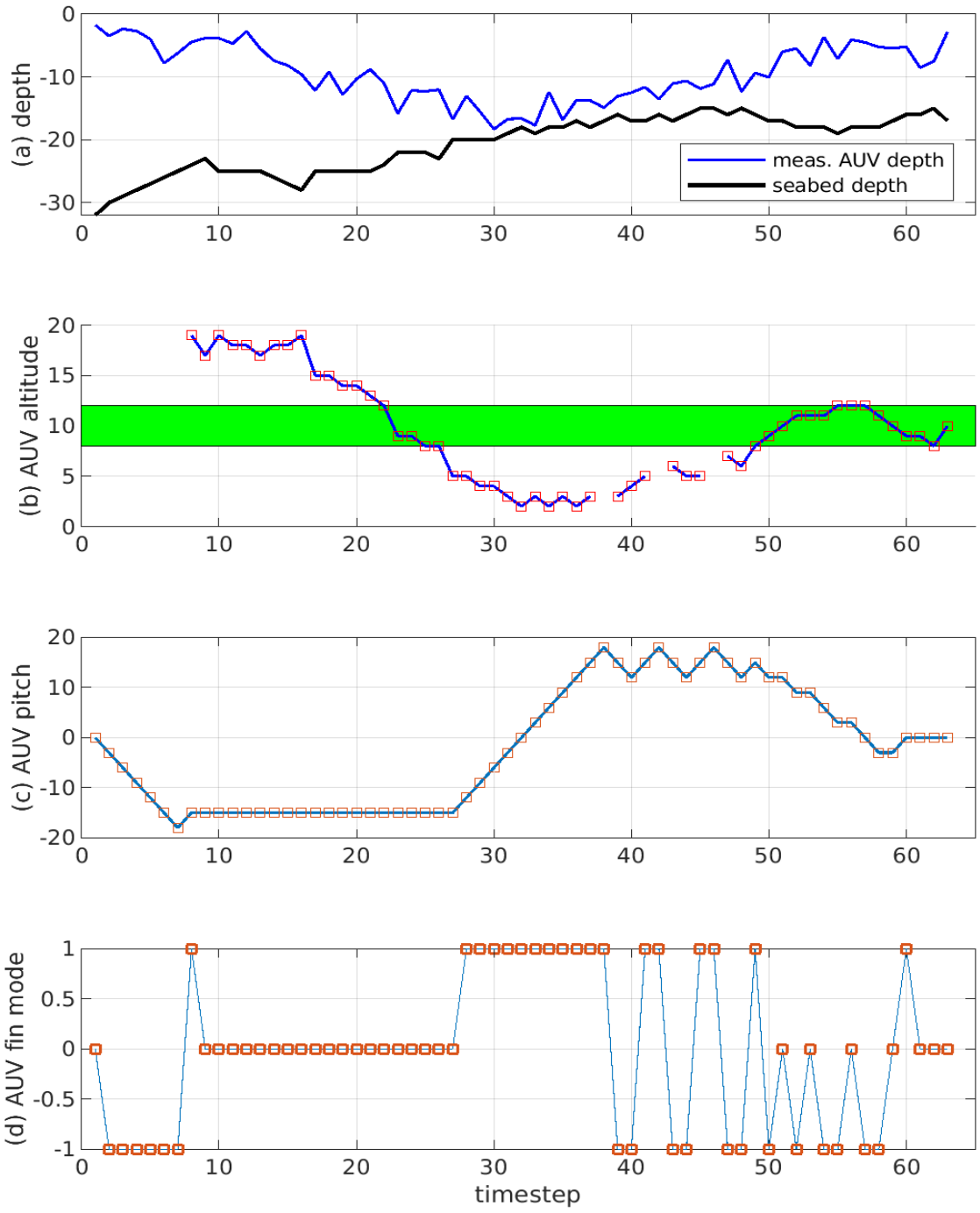

Figure 5-4 Simulated shallow-water environment model with gradual incline seabed: AUV performance with fault manager (DVL range = 20m, altitude-keeping range = 8 – 12 m, high pitch = 15°, additive noise ±2 meters). (a) The altitude-keeping AUV depth correctly tracks the seabed and changes depth based on vehicle pitch changes. (b) When the vehicle DVL is unable to achieve bottom-lock no altitude value is plotted. (c) Pitch is as expected given the depth changes. When pitch is < 0 the vehicle dives and when > 0 it rises. (d) Fin mode in driven by the fault manager to increases (+1), decrease (-1) or hold pitch (0). The desired altitude range is given by the green band in (b).

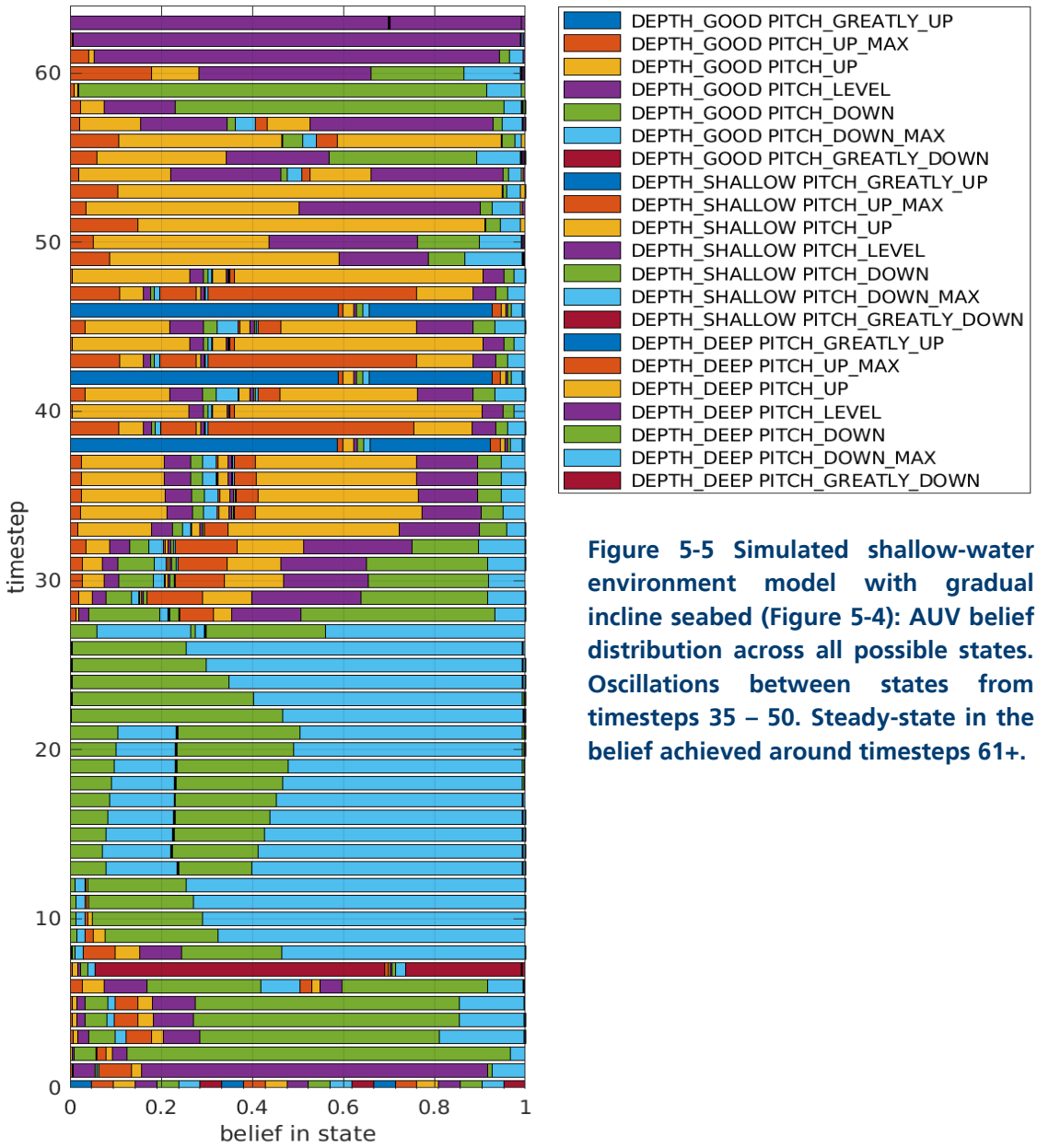

Figure 5-5 Simulated shallow-water environment model with gradual incline seabed (Figure 5-4): AUV belief distribution across all possible states. Oscillations between states from timesteps 35 – 50. Steady-state in the belief achieved around timesteps 61+.

It can be seen that Figure 5-5 there is a lower confidence in the state and slower build of confidence compared to the previous model's belief (Figure 5-3) without an induced Gaussian noise.

This simulation demonstrated that the additive noise induces observations that may not be true for the state resulting in a reduction of the confidence in the state of the POMDP model.

The third shallow water environment simulation used real-world bathymetry.

5.1.1.3 SHALLOW-WATER ENVIRONMENT MODEL WITH ACTUAL BATHYMETRY

The shallow-water environment model with actual bathymetry used real AUV measurements from Bedford Basin, Halifax, Nova Scotia. This AUV simulation was given a DVL range of 20 meters, and a high pitch angle of 15° (see Appendix C – Table for the initialization file. The altitude-keeping range is 7-10 meters. These parameters were chosen so the vehicle can generally see the seabed but is limited in pitch making it critical that it responds quickly to seabed changes.

Figure 5-6 (a) shows the environment model has a fairly consistent depth resulting in the vehicle reaching a steady-state altitude fairly quickly.

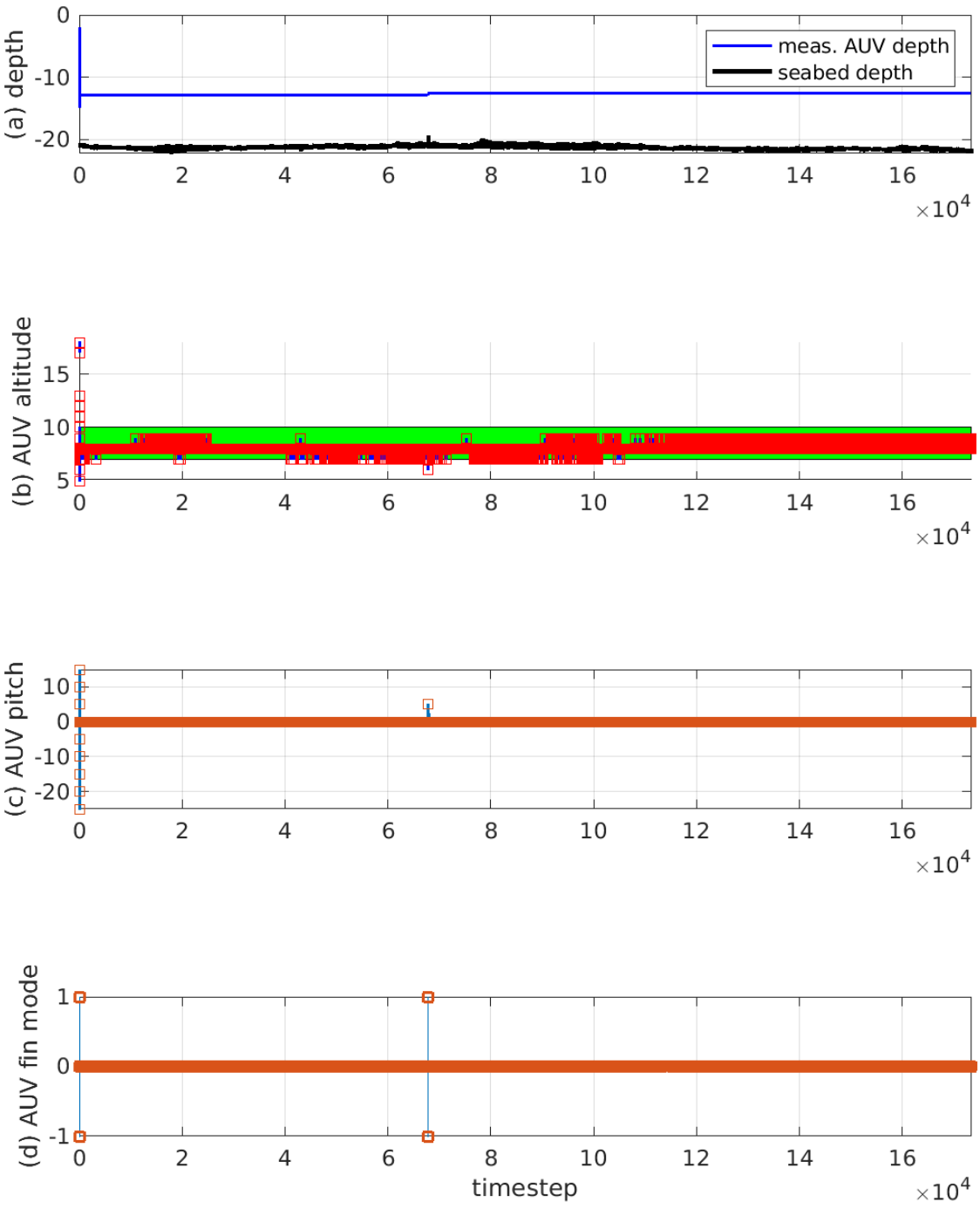

Figure 5-6 Shallow-water environment model with actual bathymetry: AUV performance with the fault manager (DVL range = 20 m, altitude-keeping range = 7 – 10 m (green band (b)), high pitch = 15°). (a) The measured AUV depth tracks the seabed and changes depth based on the vehicle pitch. (b) When the vehicle DVL is unable to achieve bottom-lock, no altitude value is plotted – not the case here. (c) The vehicle pitch correctly does not change since the seabed depth barely changes. (d) Fin mode is driven by the fault manager which increases (+1), decreases (-1), or holds the pitch constant (0). The desired altitude range is given by the green band in (b).

In Figure 5-7 after the initial (approximately 35) timesteps the vehicle reaches a steady-state and the state becomes known due to the high belief in its state. The initial timesteps are from the vehicle diving, resulting in the state changing during the dive. Due to size, only the first 70 timesteps are plotted in Figure 5-7.

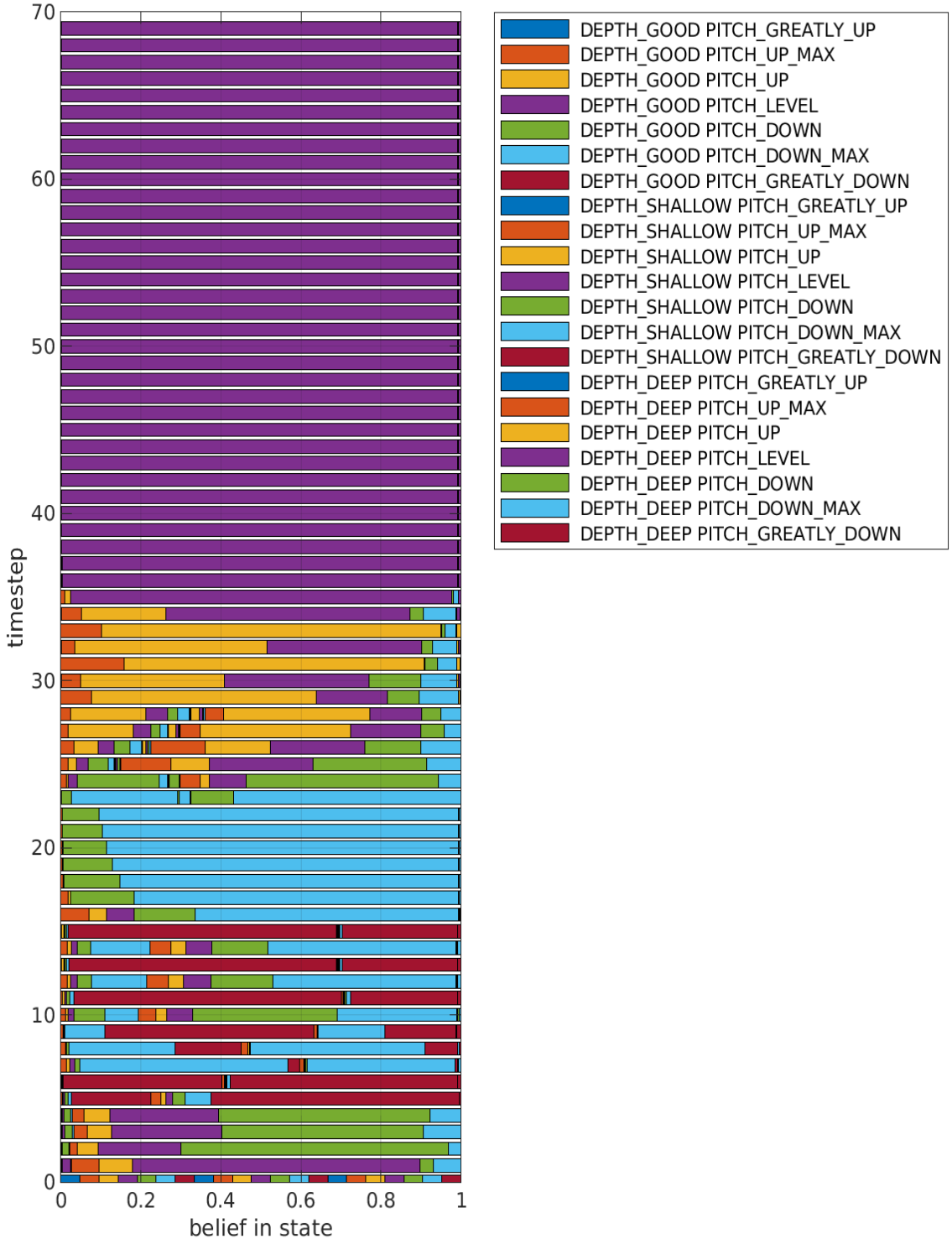

Figure 5-7 Shallow-water environment model with actual bathymetry (Figure 5-6): AUV belief distribution across all possible states. Towards timestep 35 the belief in the state of depth good pitch level becomes high (approximately 98%). Only the initial 70 timesteps are plotted since after timestep 35 the vehicle enters a steady state.

This simulation ran for approximately 173 000 timesteps. It demonstrated that with the Markov assumption the POMDP fault manager can solve for large time scales since it does not need to retain history information beyond the previous timestep. It also shows that if a system reaches a steady-state in observations, the state can be approximated as known (a POMDP requirement). This as reflected in the high confidence in its belief state.

The shallow water cases, here, with both simulated and real bathymetry represent only one type of AUV environment. Another common environment that AUVs encounter is deep water (i.e. the seabed depth exceeded the vehicle's crush depth). This is studied next.

5.1.2 Deep-water

The second series of tests were for deep water environments where the AUV cannot sense the seabed and achieve bottom-lock. The water depth at some points exceed the vehicle's crush depth. In that case, the fault manager had the vehicle maintain its depth until bottom-lock could be achieved again. The first simulation was a simple deep water simulated environment model with a seabed incline (approximately 30° with variations in the incline).

5.1.2.1 DEEP-WATER SIMULATED ENVIRONMENT MODEL

The deep-water simulation had an environment model with a gradual depth decline and incline of (approximately 30°) in the seabed where the depth reaches a maximum of 80 meters. The AUV had a DVL range of 18 meters, a high pitch angle of 15 °, a starting depth of 40 meters, and a crush depth of 50 meters. The altitude-keeping range is 8-12 meters, and the vehicle's speed is 2 knots with the timestep being 2s. See Appendix C – Table for the initialization file. The DVL range was chosen to sense the seabed initially but be unable to sense it during the deeper sections. The crush depth was chosen so the vehicle would track the seabed until it reaches the deeper portions of the simulations (in excess of 72 m).

In Figure 5-8 (a) at approximately timestep 40 the vehicle is unable to dive and maintain altitude-keeping and bottom-lock. The vehicle instead, correctly, maintains its depth above the crush depth until bottom-lock is re-acquired. The altitude is lost due to the seabed depth. In Figure 5-8 (b), during the timesteps where altitude cannot be attained the pitch oscillates (Figure 5-8 (c)) due to the conflicting requirements of seabed tracking and maximum allowable depth, the fault manager (controlling the fin mode, Figure 5-8 (d)) oscillates between diving and ascending. This was not ideal; however, it did not affect the vehicle functionality for the depth sub-system. This could result in issues with other sub-systems such as power loss from fin changes. An addition of dampening and dead-banding for edge cases such as this would solve the issue.

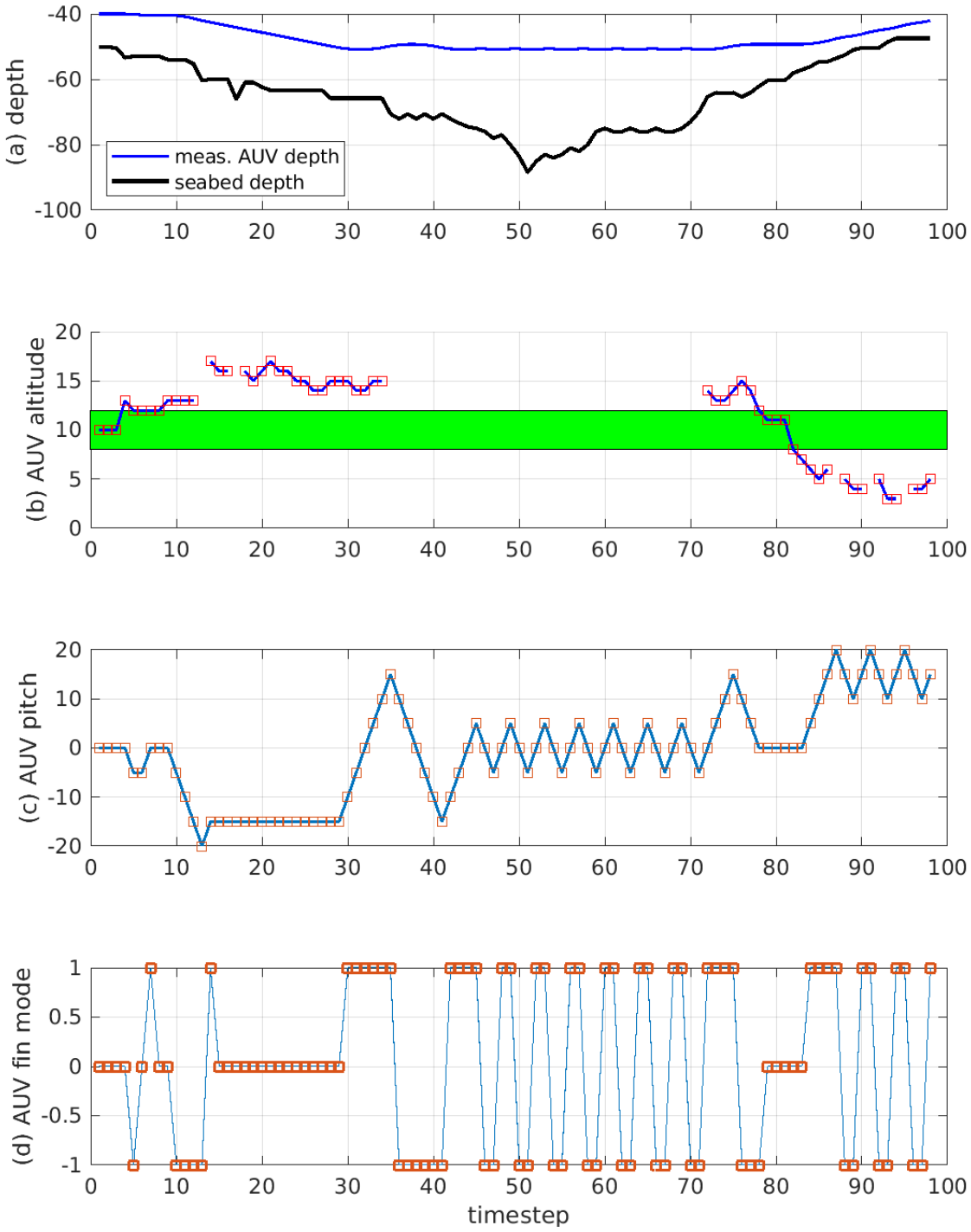

Figure 5-8 Deep-water simulated environment model: AUV performance with fault manager (DVL range = 22 m, altitude-keeping range = 10 – 20 m, high pitch = 15°, crush depth = 50 m) (a) The measured AUV depth tracks the seabed until it loses bottom-lock (timesteps 35 – 70) due to seabed depth. (b) When the vehicle loses bottom-lock (timesteps 35 – 70) no altitude value is plotted. (c) Vehicle pitch correctly oscillates when there is no bottom-lock (timesteps 41 – 70). (d) Fin mode oscillates for the same reason as in (c). The desired altitude range is given by the green band in (b).

The vehicle becomes unsure of its state during the loss of bottom-lock due to the seabed depth. This results in the vehicle oscillating between depth states (see timesteps 41 – 70 in Figure 5-9). Once the seabed depth decreases the state evens out. Between timesteps 80 – 85 the fault manager is confident of its state as the vehicle operates at its ideal depth and altitude (Figure 5-8 (a) and (b)). The vehicle becomes less sure of its state between 85+ due to the rapidly rising seabed forcing the vehicle upwards. The vehicle is fairly confident of its state between timesteps 2 – 5 (Figure 5-9) with the state being DEPTH_ GOOD, PITCH_LEVEL. At timesteps 88, 91 and 98 the pitch goes to -20° (Figure 5-8 (c) and can be seen that the state (Figure 5-9) becomes DEPTH_DEEP, PITCH_DOWN_GREATLY which pitch angle is nose-down past the threshold. The DEPTH_DEEP is due to the rapidly rising seabed causing the vehicle having a lower altitude than desired.

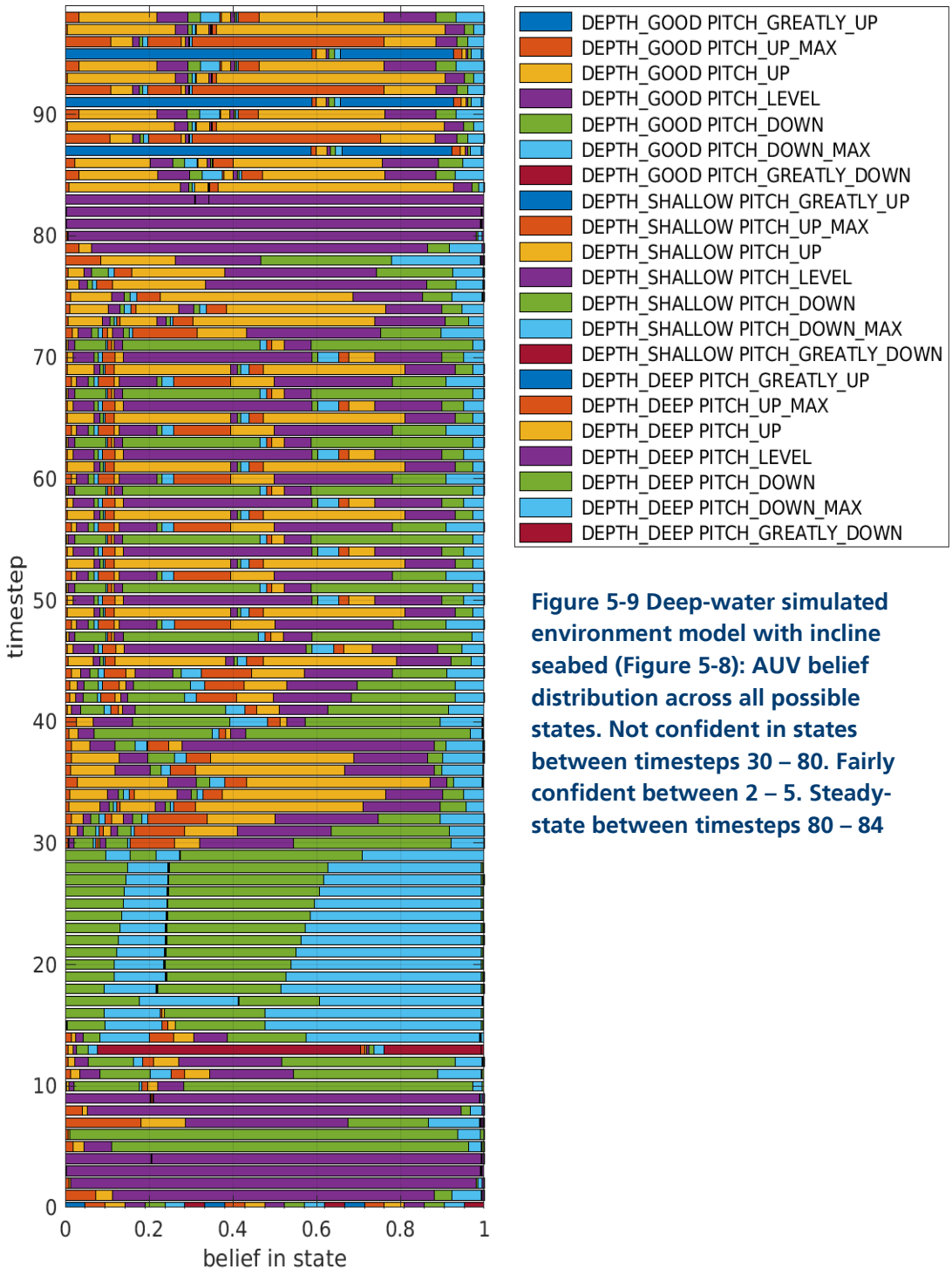

Figure 5-9 Deep-water simulated environment model with incline seabed (Figure 5-8): AUV belief distribution across all possible states. Not confident in states between timesteps 30 – 80. Fairly confident between 2 – 5. Steady-state between timesteps 80 – 84

This simulation shows the fault manager successfully keeping the vehicle from exceeding its crush depth due to loss of altitude (conflicting requirements) when the seabed becomes too deep to track. Another deep-water simulation was conducted using actual bathymetry measurements.

5.1.2.2 DEEP-WATER WITH REAL-WORLD BATHYMETRY ENVIRONMENT MODEL

The deep-water simulation has an environment model that was from an AUV collecting data from Bedford Basin, Halifax, Nova Scotia. The AUV was given a DVL range of 10 meters, a high pitch angle of 20 °, a crush depth of 15 meters, and a minimum depth of 3 meters. (see Appendix C – Table for the initialization file). The altitude-keeping range was from 8 to 12 meters. The vehicle's speed was 2 knots and the timestep was 2s. The DVL range was chosen so the vehicle could usually sense the seabed in order to track it. The crush depth was set to 15 meters, so the vehicle is unable to maintain altitude over some portion (the depth for this environment model was shallow so it was necessary to set shallow crush depths). The minimum depth of the vehicle was decreased to 3 meters since the environment model is fairly shallow.

In Figure 5-10 (a), the vehicle and seabed depths are shown as a function of time. During timesteps 70 – 80, the seabed is very shallow, the fault manager oscillates between avoiding the seabed and achieving the minimum vehicle depth (set in the initialization file). This oscillation is caused by the edge case of trying to dive to avoid minimum depth while trying to attain the altitude range. During timesteps 380 – 400, the seabed depth becomes too deep and the vehicle can no longer achieve bottom-lock and loses altitude measurements (Figure 5-10 (b)). This results again in some oscillation due to the conflicting requirements of diving to achieve altitude range and maintaining the vehicle above the crush depth.

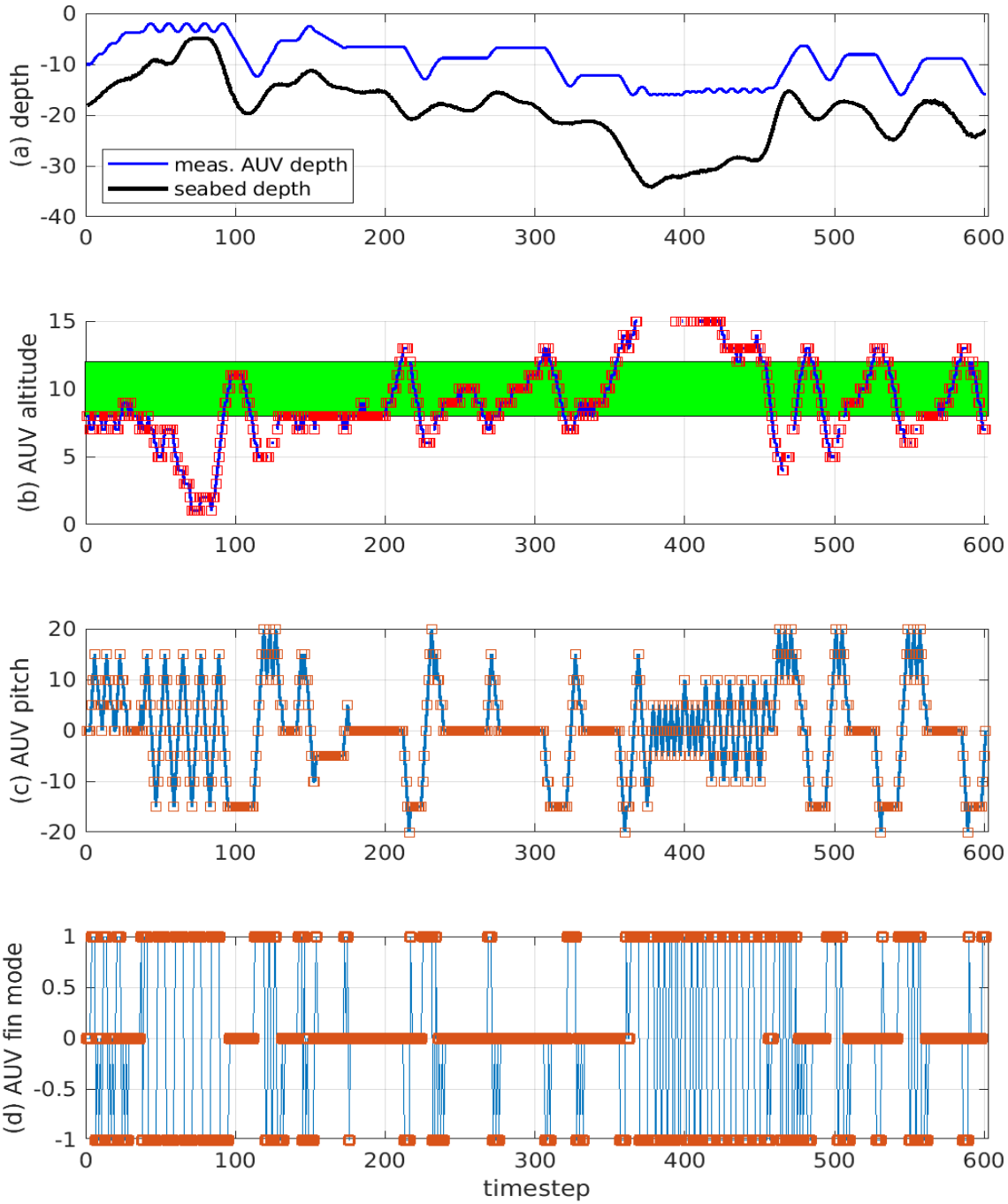

Figure 5-10 Deep-water with real-world bathymetry environment model: AUV performance with fault manager: (a) The AUV has some oscillation around timestep 40-80 due to conflict between minimum depth and minimum altitude. The vehicle achieves (b) altitude-keeping (green band) for most of the mission although it loses altitude and depth measurements around timesteps 370-400 due to increased seabed depth beyond vehicle crush depth. (c) When the pitch < 0 the vehicle dives and when > 0 it rises. Pitch oscillations for reasons in (a). (d) Fin mode is driven by the fault manager which increases (+1), decreases (-1), or holds the pitch constant (0). The desired altitude range is given by the green band in (b).

When the vehicle has both the desired altitude range and depth range the belief in its state becomes confident (see Figure 5-11) approximately in timesteps 20-35, 130-140, 155-210, 240-270, 275-305, 330-355, 510-525, and 560-585. This is due to the vehicle's subsequent observations agreeing with past ones resulting in the vehicle entering a steady state. During timesteps where it is unable to achieve depth and altitude-keep or when the seabed is rapidly changing it becomes less confident in its states (see Figure 5-9).

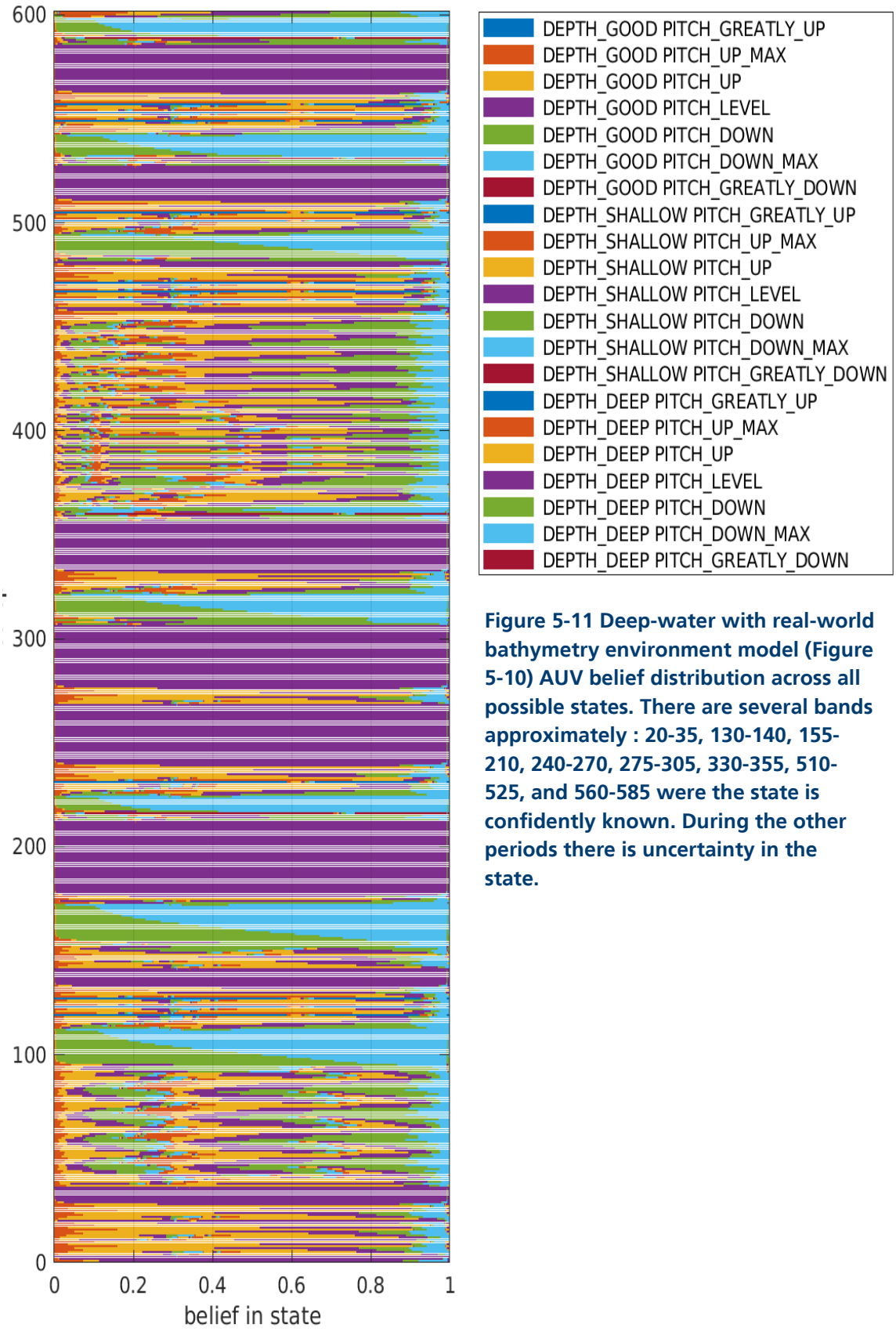

Figure 5-11 Deep-water with real-world bathymetry environment model (Figure 5-10) AUV belief distribution across all possible states. There are several bands approximately : 20-35, 130-140, 155-210, 240-270, 275-305, 330-355, 510-525, and 560-585 were the state is confidently known. During the other periods there is uncertainty in the state.

This simulation show that the fault manager was able to successfully keep the vehicle from exceeding its crush depth when the seabed was too deep for the vehicle to altitude-keep using an environment with actual bathymetry. It also shows that the belief in the state becomes more variable when it has conflicting parameters (i.e. keep the AUV above the maximum depth and achieving bottom-lock on a too deep seabed). This results in it prioritizing one set of parameters (minimum depth) to ensure vehicle health.

The next series of tests used rapidly changing seabed depth to demonstrate the fault manager's response to a variable environment.

5.1.3 Variable seabed

The third test series was for a variable seabed which has a rapidly changing depth. The AUV must respond quickly to avoid colliding with the seabed and, simultaneously, maintain bottom-lock. The first test used a simulated random environment model.

5.1.3.1 VARIABLE SEABED WITH SIMULATED RANDOM ENVIRONMENT MODEL

The first variable seabed depth test had a simulated random environment model. The AUV was initialized with a DVL range of 30 meters, a high pitch angle of 20 °, and a speed of 2 knots and timestep of 2s. It maintains an altitude-keeping range of 12 to 16 meters, this was increased from previous models due to the rapid changes of the environment. (Appendix C – Table for the initialization file). The DVL range was chosen so the vehicle DVL could sense the seabed. The crush depth was set to 35 m, so the vehicle would have stretches where it is unable to altitude-keep.

In Figure 5-12(a), the vehicle reacts to the rapid changes in the seabed depth. Around timestep 15, the vehicle narrowly avoids collision by rapidly increasing its pitch from the initial dive, due to the decrease in seabed depth. Later, around timestep 55, the vehicle scrapes the seabed because of an anomalous peak (simulating a shoal). These types of collision are difficult to avoid beyond setting the vehicle to a higher altitude-tracking threshold (which means poorer resolution sonar sensor

measurements). If this is an expected concern, the remedy is an obstacle avoidance (front-facing) sensor. In a real-world model, depending on the vehicle speed and structural material, this could result in vehicle damage. However, it is assumed only minor damage resulted from this collision and the vehicle was able to continue its mission. Between timesteps 20-45, the vehicle DVL loses bottom-lock due to the pitch of the vehicle exceeding the maximum pitch angle tolerated for bottom-lock (15°) .The vehicle loses bottom-lock when the seabed depth rapidly deceases, and the vehicle is unable to follow fast enough to keep within range.

In Figure 5-12 (c) and (d), the pitch and fin when the pitch becomes too large in response to trying to change vehicle depth to avoid colliding with the seabed (timesteps 20-45).

Here is shows how the speed of the vehicle effects tracking the seabed, if the seabed changes faster than the vehicle can follow the vehicle is unable to maintain altitude keeping.

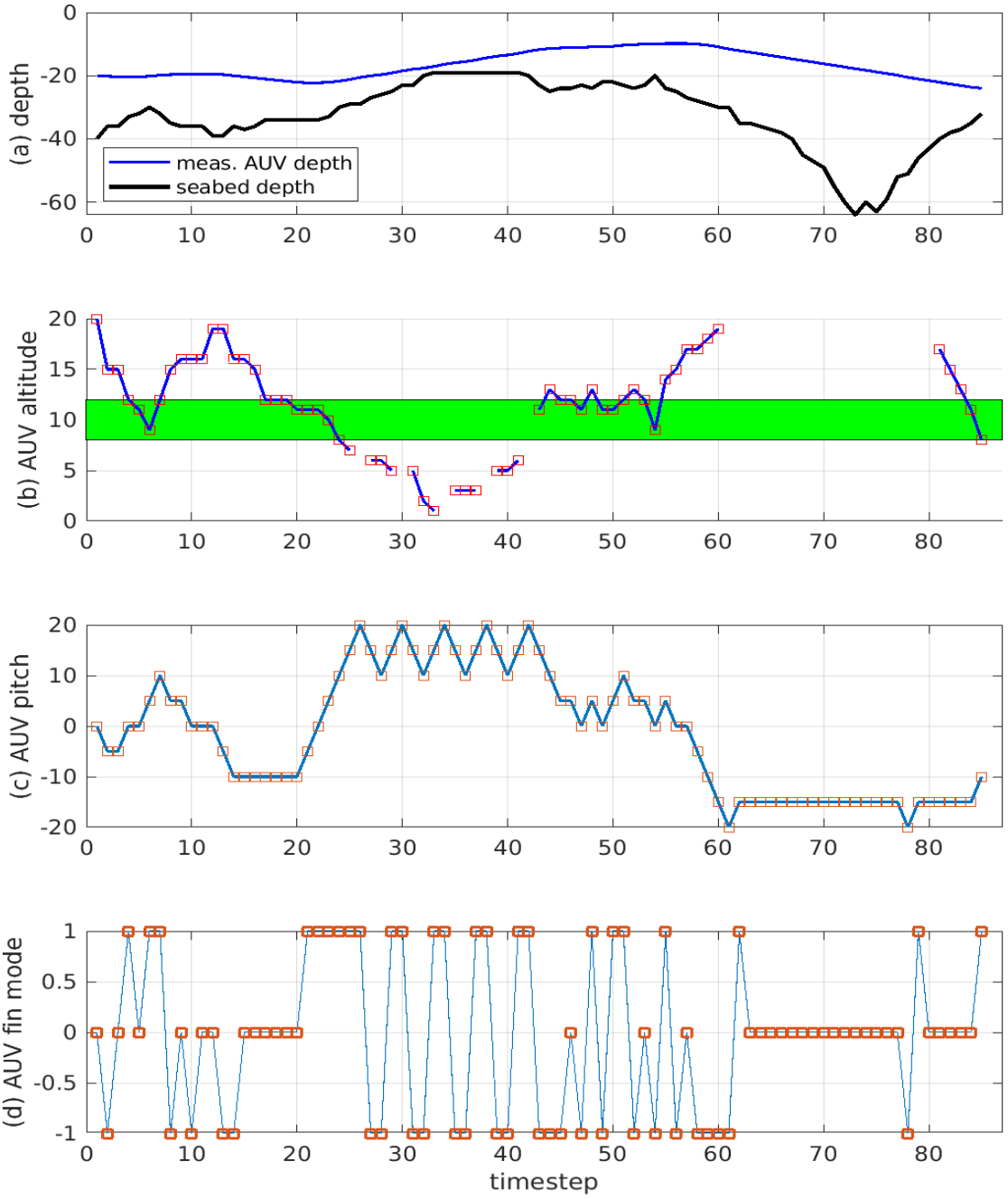

Figure 5-12 Variable seabed with simulated random environment model: AUV performance with fault manager (DVL range is 35m, high pitch = 20°, altitude-keeping range = 12 – 16 m, crush depth is 35m). (a) The measured AUV depth tracks the seabed depth until the seabed rises quickly at timesteps 25 and 35 (near collisions). (b) When the vehicle DVL loses bottom-lock (timesteps 25-45) due to high pitch, and when the seabed has descended beyond the DVL range and the vehicle's speed is unable to match (timesteps 60 -80) there is no altitude value plotted. (c) When the pitch < 0 the vehicle dives and when > 0 it rises – vehicle pitch oscillates between timesteps 20-45 due conflicting requirements of rising and descending. (d) Fin mode is driven by the fault manager and increases (+1), decreases (-1), or holds the pitch constant (0). The desired altitude range is given by the green band in (b).

As shown in Figure 5-13, with rapid changes in the seabed (compared to the vehicle speed and its ability to respond) the vehicle never enters a steady-state condition. Due to this rapidly changing environment, the fault manager is less confident in any one state. The beliefs rapidly change as conflicting observations are made.

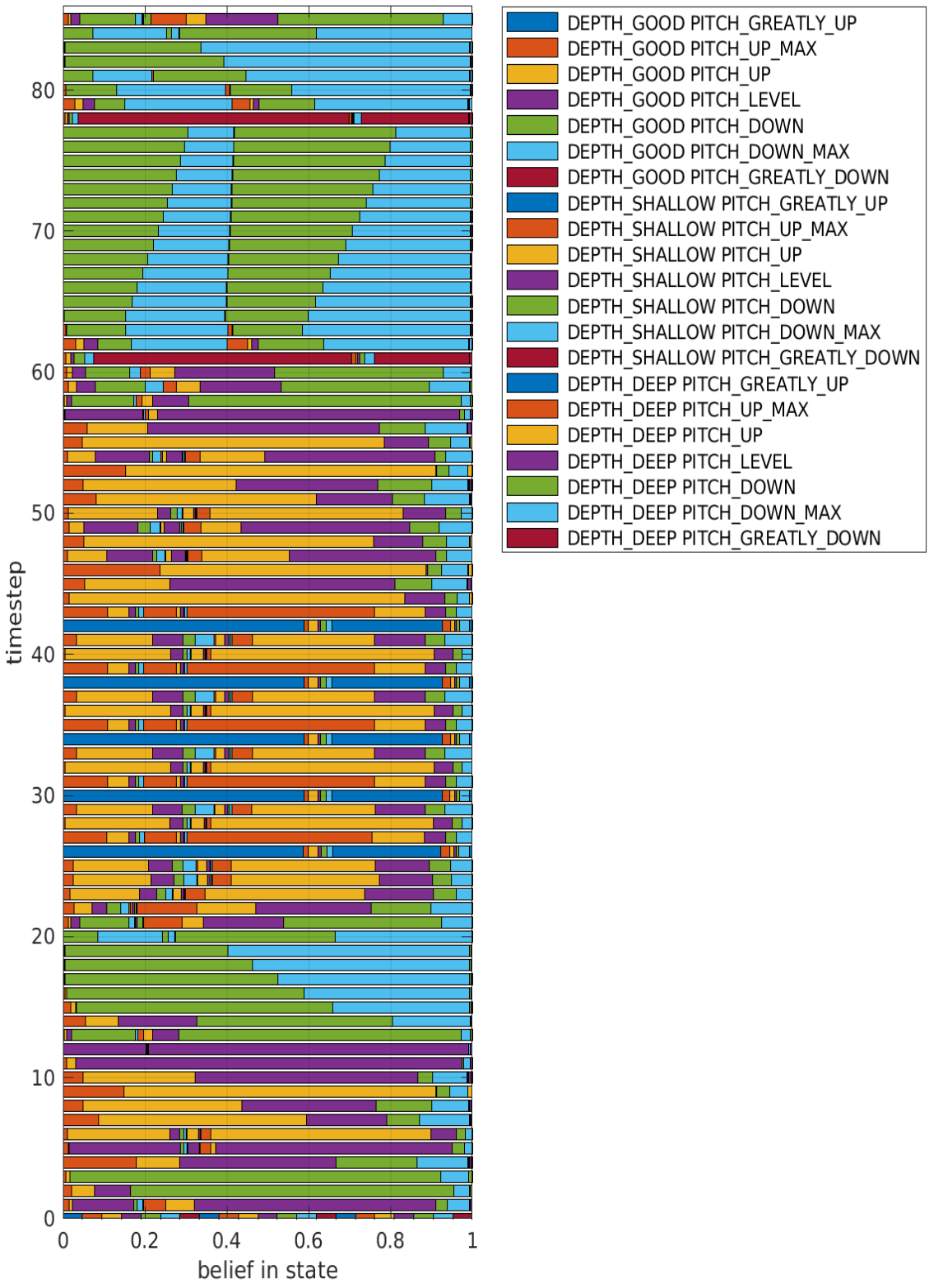

Figure 5-13 Variable seabed with simulated random environment model (Figure 5-12): AUV belief distribution across all possible states. The states fluctuate, and the system never enters a steady state.

This simulation shows the fault manager can respond to a variable seabed environment. A variable environment can cause lower confidence in any one belief state due to the rapid changes between timesteps. The AUV vehicle is limited by its speed and angle change, this results in the possibility of the vehicle being unable to completely track the seabed where it is rapidly changing.

A second simulation was performed on a variable seabed this time with real-world bathymetry.

5.1.3.2 VARIABLE SEABED REAL-WORLD BATHYMETRY ENVIRONMENT MODEL

This variable seabed simulation has an environment model from real-world bathymetry of Bedford Basin, Halifax, Nova Scotia. The AUV was given a DVL range of 30 meters, a high pitch angle of 20 °, a crush depth of 35 meters, and a minimum depth of 3 meters. The altitude range was between 12 and 16 meters. (See Appendix C – Table for the initialization file.). The DVL range was chosen so the vehicle could always sense the seabed and have bottom-lock. The minimum depth of the vehicle was decreased to 3 meters since the environment model is shallow.

In Figure 5-14(a), the vehicle and seabed depth were plotted on the same time scale to show the vehicle fault manager's response to the seabed. From timesteps 70 – 80 the vehicle's response has some oscillations in very shallow water. The fault manager is attempting to avoid the seabed and achieve the minimum vehicle depth (set in the initialization file). At this time, the seabed depth reaches 4.9 meters. Near timestep 450 the depth of the seabed rapidly increases like a cliff face. The vehicle must ascend rapidly to avoid collision. Due to the rapid changes in the seabed the fault manager is constantly changing the fin mode (d) to alter the pitch Figure 5-14 (c) so the vehicle avoids the seabed Figure 5-14 (a) but maintains altitude (b).

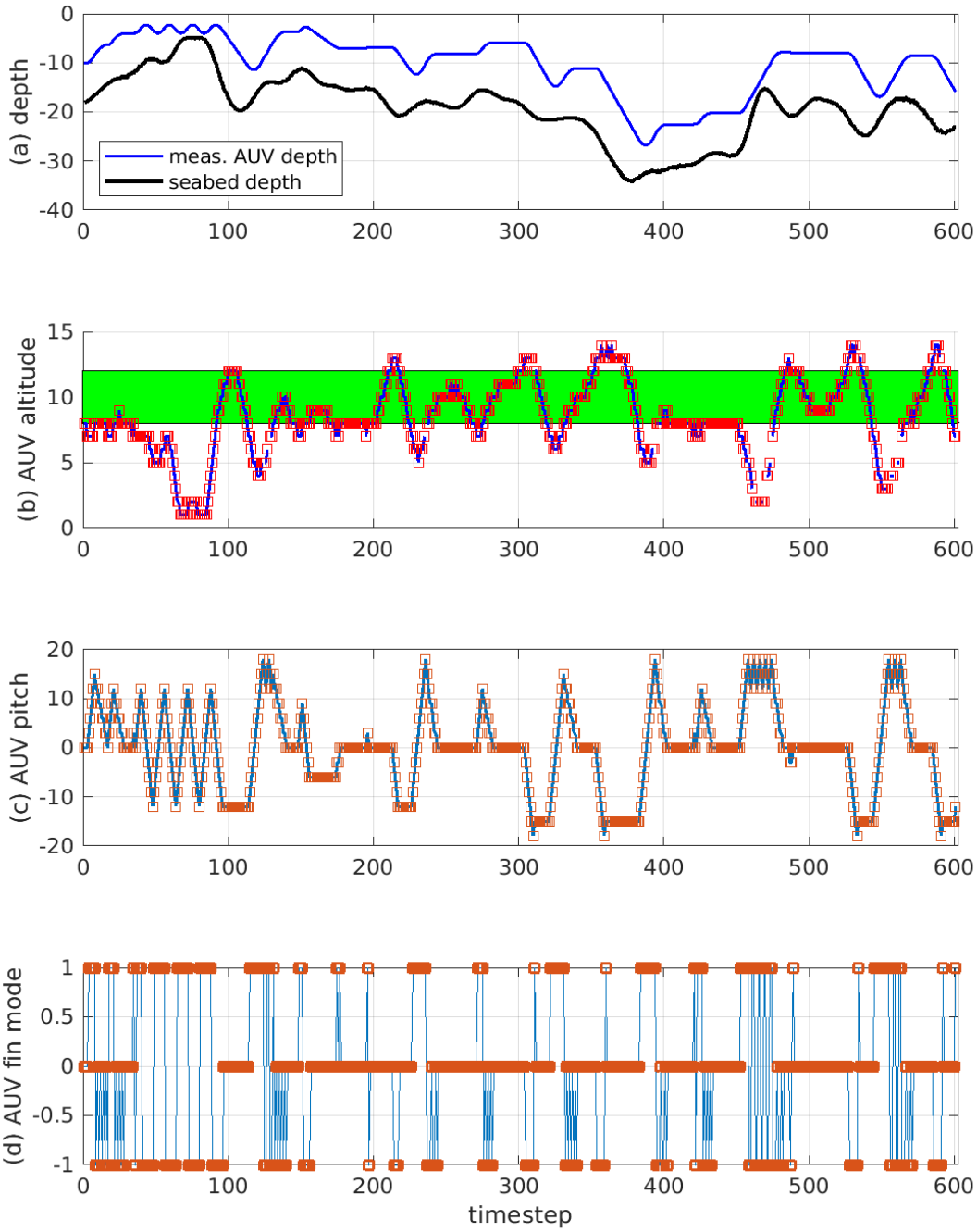

Figure 5-14 Variable seabed real-world bathymetry environment model: AUV performance with fault manager (DVL range = 30m, high pitch = 20°, crush depth = 35m, min vehicle depth = 4m). (a) The AUV some oscillations during the edge case of trying to maintain minimum depth and altitude range during timesteps 40-70). A near miss is located around timestep 470. (b) Given a DVL range of 20 m in 32 m of water, the vehicle DVL achieves bottom-lock throughout the missions. (c). When the pitch < 0 the vehicle dives and when > 0 it rises. vehicle rises. Pitch oscillations early in the run for the reasons in (a). (d) Fin mode is driven by the fault manager which increases (+1), decreases (-1), or holds the pitch constant (0). The desired altitude range is given by the green band in (b). The desired altitude range is given by the green band in (b).

Figure 5-15 shows that with high frequency variability in the seabed depth, compared to the vehicle's ability to respond, there are only specific stretches where the vehicle is certain of its state (timesteps 20 – 30, 20 – 25, 145 – 150, 165 – 205, 230 – 260, 270 – 305, 320 – 350, 370 – 420, 440 – 460, 500 – 540, and 570 – 585). During the rapid changes the vehicle's state distribution is varied and changes quickly from new observations being made.

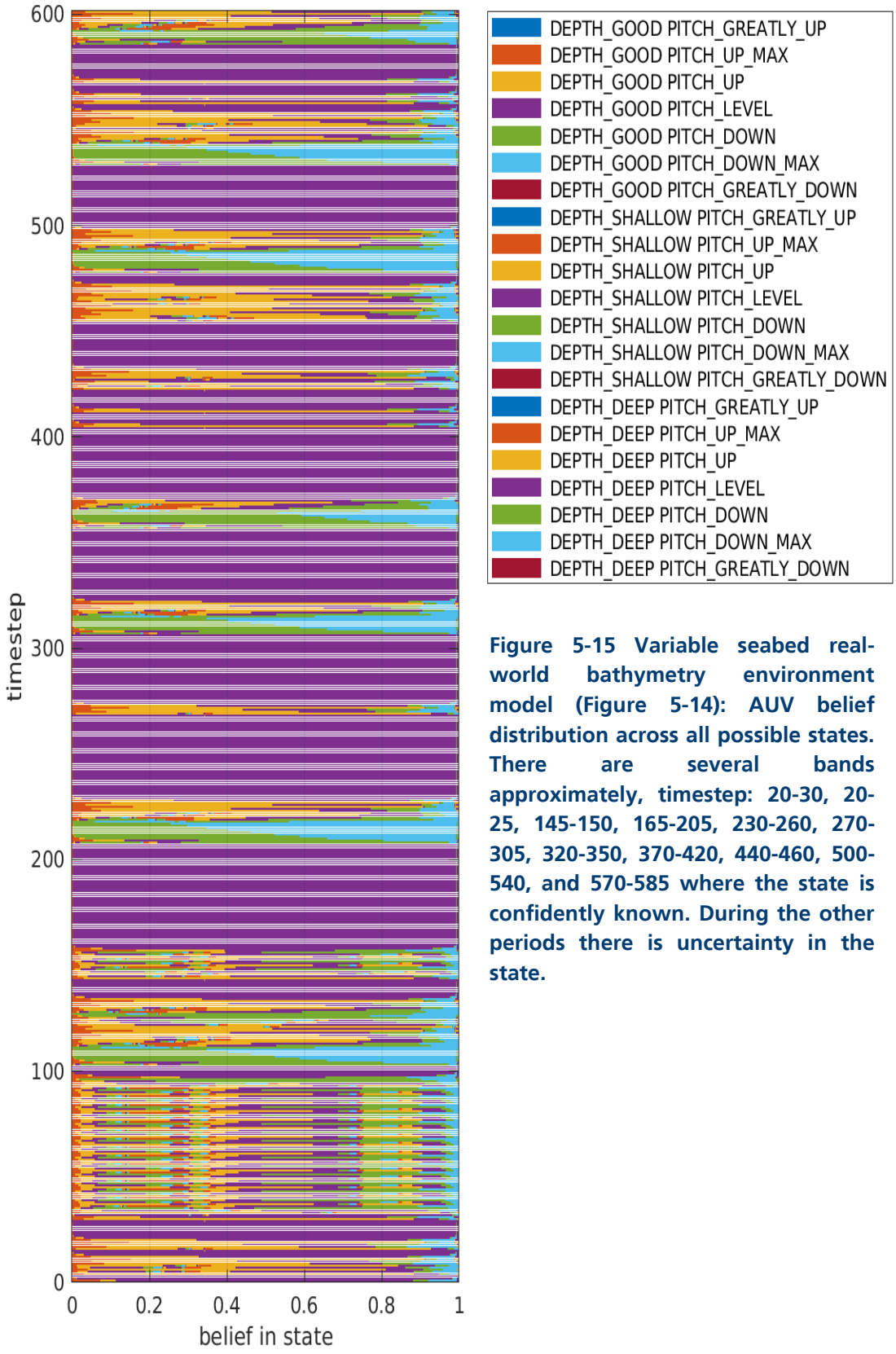

Figure 5-15 Variable seabed real-world bathymetry environment model (Figure 5-14): AUV belief distribution across all possible states. There are several bands approximately, timestep: 20-30, 20-25, 145-150, 165-205, 230-260, 270-305, 320-350, 370-420, 440-460, 500-540, and 570-585 where the state is confidently known. During the other periods there is uncertainty in the state.

Like the previous simulated environment model (Section 5.1.3.1), the vehicle belief of its state fluctuates as the seabed rapidly changes depth. This results in observations to vary a lot between consecutive timesteps (i.e. observations may be that the vehicle is in shallow water to be followed by very deep water in the next timestep). This is due to the nature of the rapidly changing seabed. For example, at timestep 460 the seabed is about 20 meters but at timestep 470 the seabed is 15 meters. This rapid change in the environment causes rapid change in the prediction of the state.

The variation in the belief state is due to rapid changes in the seabed causing reduced confidence. The vehicle however, was able to successfully navigate and achieve bottom-lock while avoiding collisions with the seabed. The next set of tests demonstrate the effect of modifying the POMDP model itself and how the vehicle fault manager responds.

5.1.4 POMDP probability function modified and inclusion of noise

The fourth set of simulations explore the reduction of confidence in the POMDP model and the addition of a noisy sensor model. These tests use the shallow-water simulated environment with a gradual seabed incline (see 5.1.1.1) and with additive Gaussian noise on the depth measurements(5.1.1.2). The parameters for these tests were all the same with the altitude range being between 4 and 8 meters. The DVL range was set to 20 meters, altitude range of 8-12 meters, speed of 2 knots, timesteps of 2s, and depth range between 5 and 35 with maximum pitch being 15°. These were chosen to match the parameters of the unmodified simulation in 5.1.1.1.

These tests reduce the probabilities assigned to the POMDP model functions for the probability of *making-an-observation* and probability of *transitioning-between-states*, $O(o|s,a)$ *and* $T(s'|s,a)$, respectively.

The reward function was not altered since the rewards are arbitrary and their values are only relevant compared to each other. If reward values are all reduced by a specific

percentage, the system will not alter the model's behaviour since the rewards relative to one another are unchanged.

The first set of tests explores a reduction in the probability of *making-an-observation* function.

5.1.4.1 OBSERVATION FUNCTION PROBABILITIES REDUCED BY 10%, 20% AND 40%

These tests explore the impact of the probability of *making-an-observation* function, $O(o|\ s,a)$, in the POMDP. It reduces the set probabilities in the POMDP model by a controlled amount. Three tests were conducted with reductions of 10%, 20%, and 40%.

As shown in Figure 5-28 The three different simulations of reduced observation functions have the same trends. The 10% reduced is able to attain altitude keeping (b). The 20% reduced follows the 10% reduced although it is slower to react to changes. A 40% reduction in the probability has the most dramatic effect with the vehicle having very little confidence in the observations made of the rising seabed and consequently, colliding with the seabed. Similarly, due to the high pitch angles from the few altitude measurements that were achieved, there are much more instances of fin mode changes resulting in an unstable pitch. This is due to the fault manager's uncertainty in its state being reflected in its responses.

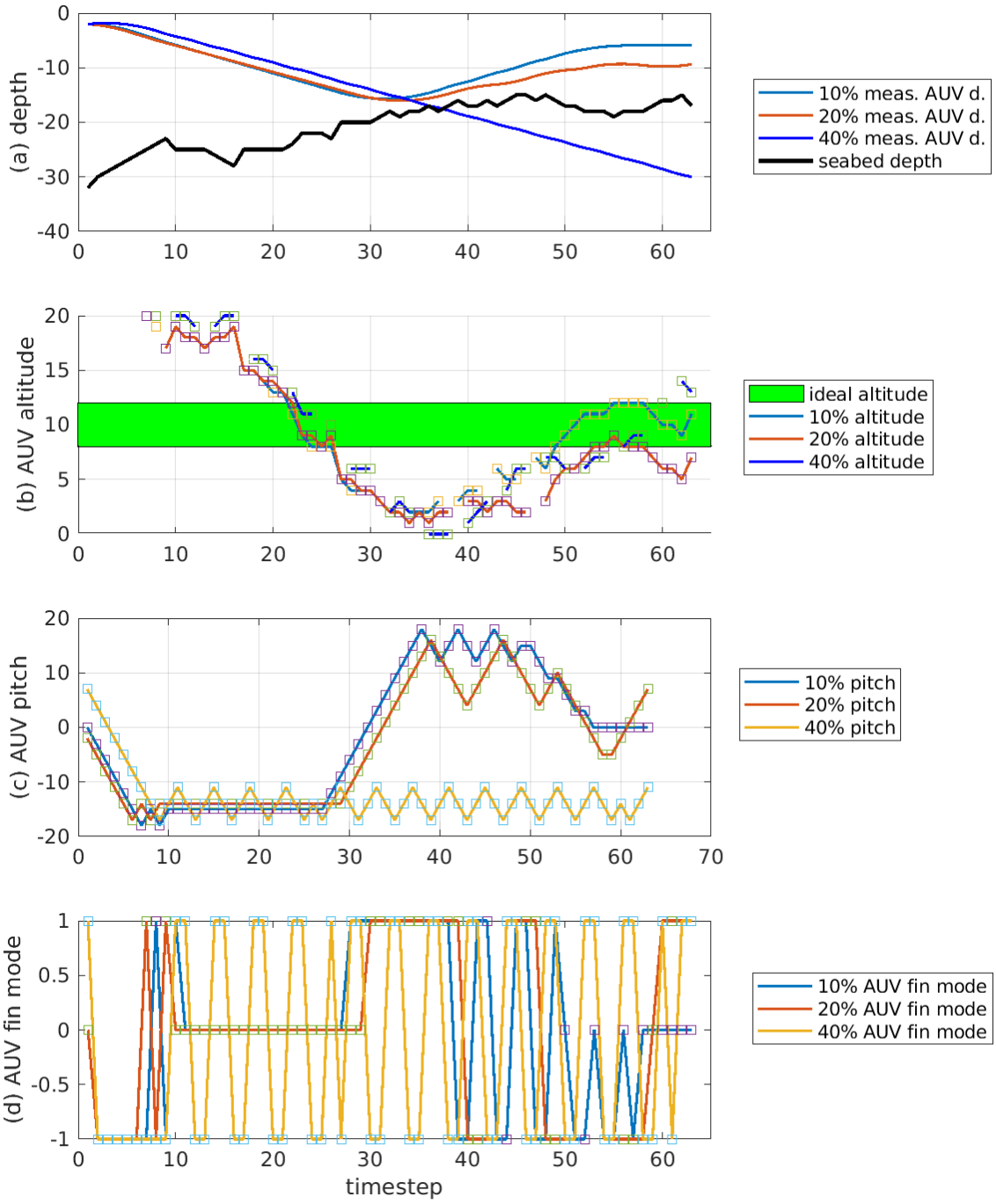

Figure 5-16 Observation probabilities reduced by 10%, 20% and 40%: AUV performance with the fault manager over a shallow gradual seabed incline. In all cases, the results are poorer with increase reductions in the observation probabilities. The desired altitude range is given by the green band in (b).

In the 10% reduced model there is not much change from the original model in section 5.1.1.1. The vehicle is confident of its state throughout the simulation run (Figure 5-29).

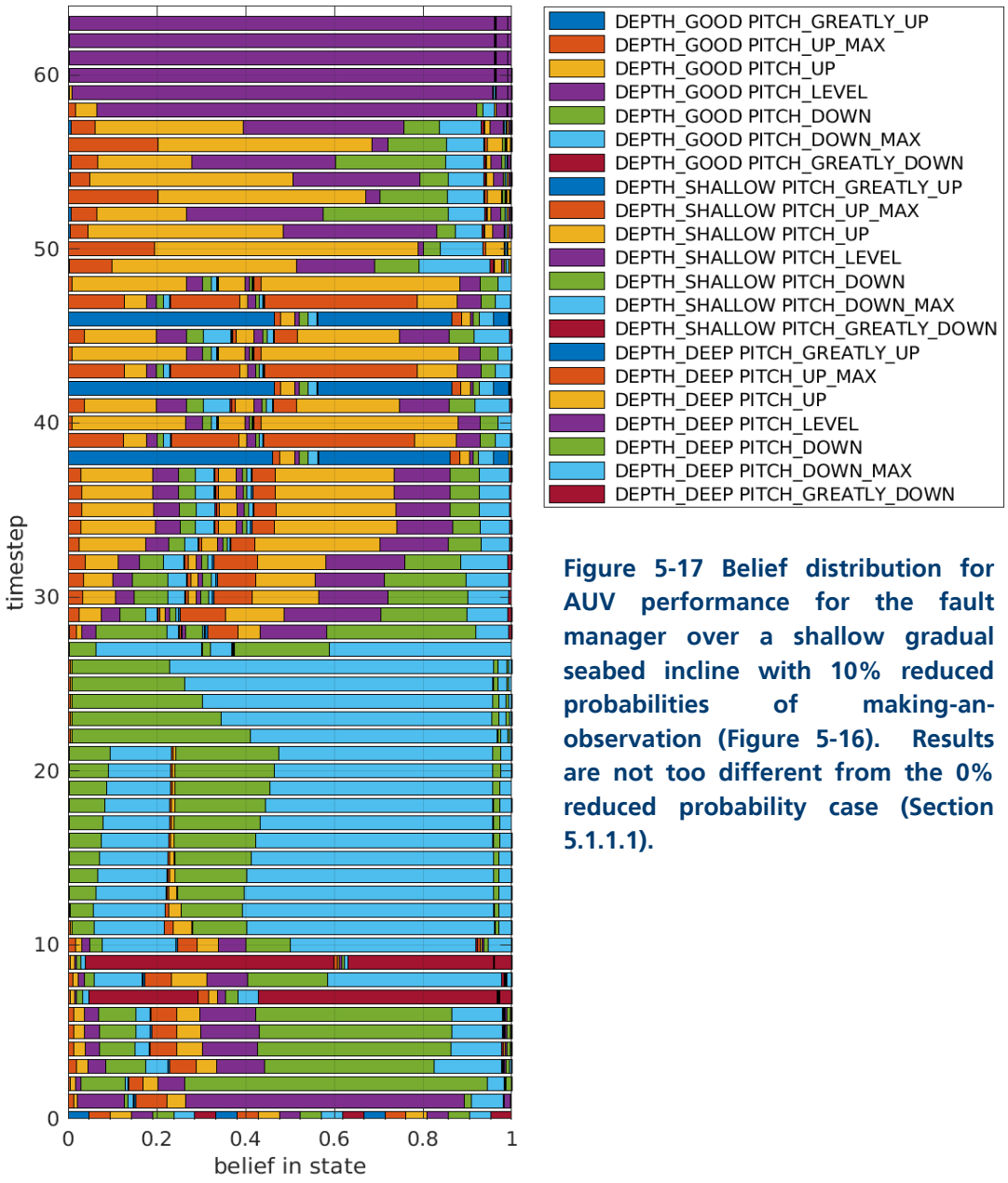

Figure 5-17 Belief distribution for AUV performance for the fault manager over a shallow gradual seabed incline with 10% reduced probabilities of making-an-observation (Figure 5-16). Results are not too different from the 0% reduced probability case (Section 5.1.1.1).

In the 20% reduced model (Figure 5-18) the fault management system still has confidence in its belief state in the first half of the simulation however; as the seabed rises and forces the vehicle to adapt the fault manager becomes less sure of its state. During timestep 54 there is a large change in the belief state due to a highly erroneous depth measurement. This is interesting since it shows the noisy data results in a fault causing the vehicle to attempt to rectify its depth only to receive more accurate data and have to rapidly change its belief state and actions.

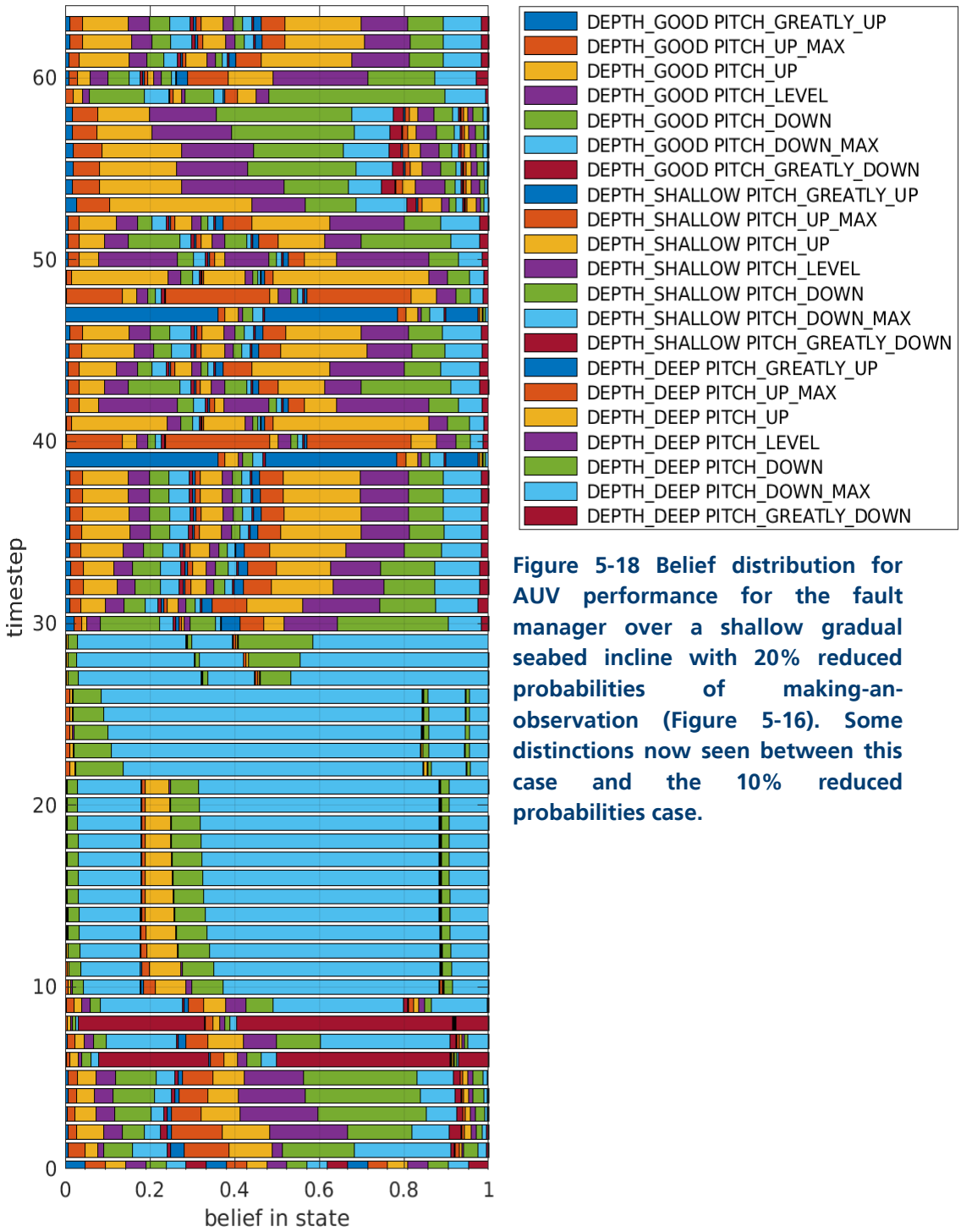

Figure 5-18 Belief distribution for AUV performance for the fault manager over a shallow gradual seabed incline with 20% reduced probabilities of making-an-observation (Figure 5-16). Some distinctions now seen between this case and the 10% reduced probabilities case.

In the 40% reduced model the fault management system lacks confidence in the vehicle state. This is due to POMDP model not being able to place confidence in the observations due to the highly reduced probability of making those observations given any state or prior actions.

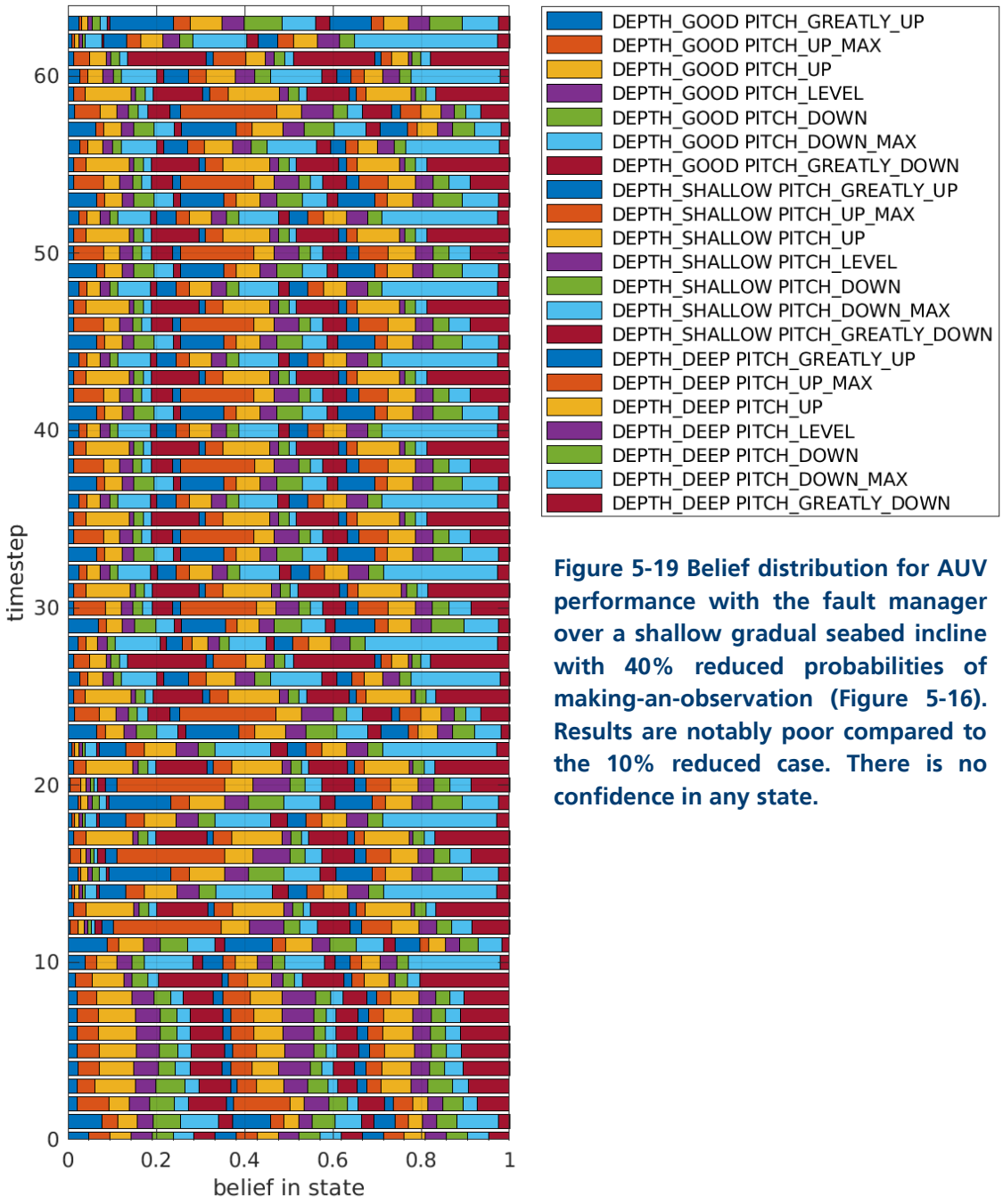

Figure 5-19 Belief distribution for AUV performance with the fault manager over a shallow gradual seabed incline with 40% reduced probabilities of making-an-observation (Figure 5-16). Results are notably poor compared to the 10% reduced case. There is no confidence in any state.

These simulations clearly demonstrate how reducing the probability of observation causes the vehicle fault manager to have less confidence in its state. In the 40% case, the vehicle is unable to have confidence in any state.

The next set of tests have a reduction in probabilities were applied to the *transitioning-between-states* function instead.

5.1.4.2 TRANSITION FUNCTION PROBABILITIES REDUCED BY 10%, 20% AND 40%

These tests explore the impact of the probability of transitioning-between-states, *T(s'| s,a),* POMDP function. It reduces the probabilities in the POMDP model by a set rate, namely 10%, 20%, and 40%.

In Figure 5-32 (a), the simulations for reductions of 10% and 20% perform similarly. The 20% reduction simulation has more fin mode changes and greater variation in the pitch. The simulation with 40% reduction is unable to prevent collisions with the seabed or achieve bottom-lock for altitude readings. The lack of confidence in its state has the consequence of being unable to predict states. Similarly, the lack confidence in its actions means it is unable to change the state to a desired outcome. The fin mode varies rapidly for the 40% reduced model and the change in pitch to rise above the seabed comes far too late.

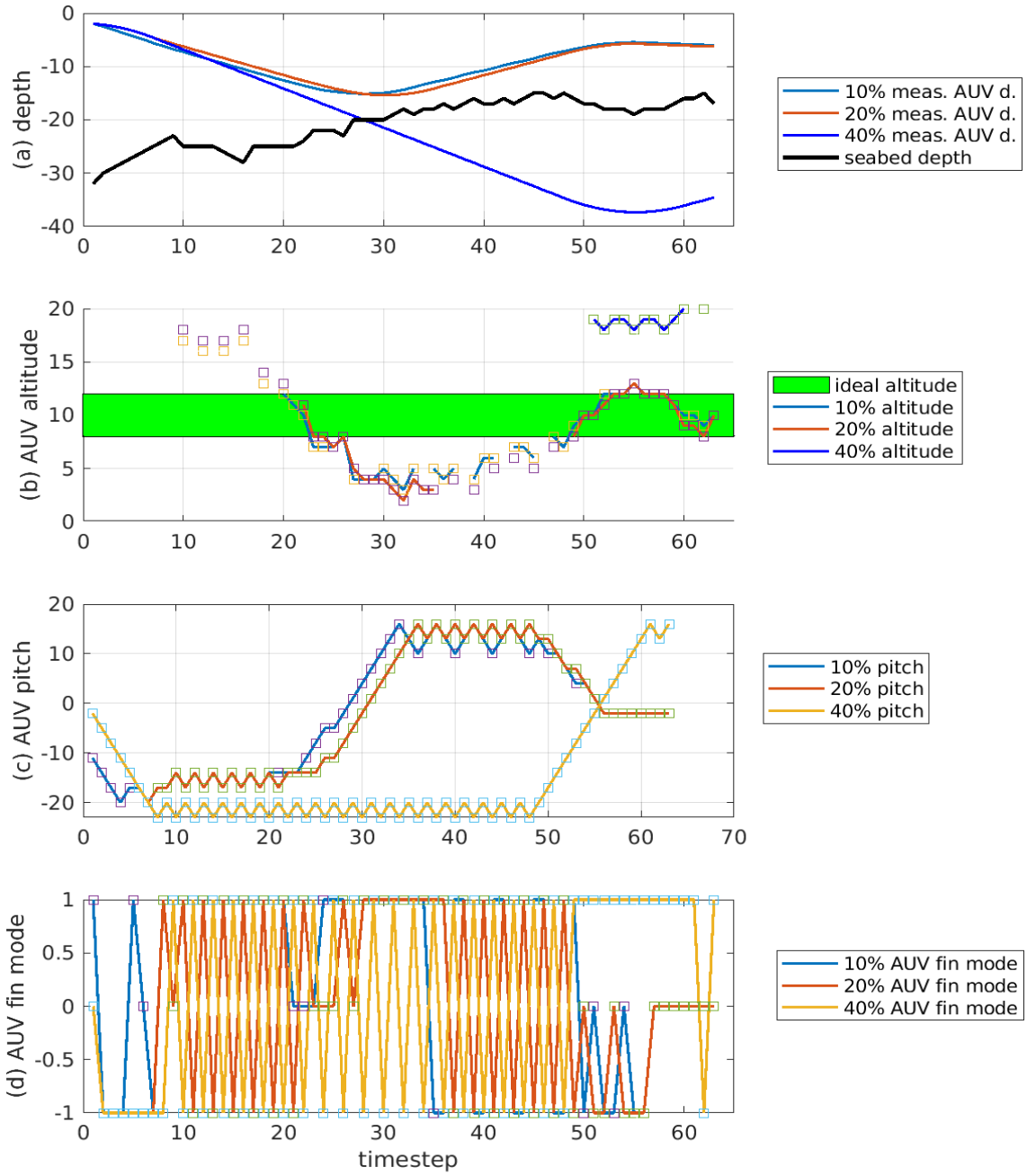

Figure 5-20 State transition probabilities reduced by 10%, 20% and 40%: AUV performance with the fault manager over a shallow gradual incline environment. Generally, the AUV performance is poorer and more uncertain with increasingly reduced probabilities where the fault manager is unable to direct the AUV towards the desired outcomes. The desired altitude range is given by the green band in (b).

In the 10% reduction (Figure 5-21) to the probability of transitioning-between-states model there is little change from the original model in section 5.1.1.1. The vehicle can still have confidence in its state throughout the simulation.

Figure 5-21 Belief distribution for AUV performance with the fault manager over a shallow gradual seabed incline with 10% reduced probability of transitioning-between-states (Figure 5-20). Results are not remarkable beyond the 0% reduced probability case.

The difference between a reduction of 20% (Figure 5-22) and 10% (Figure 5-21) is not significant although it is noticeable that the vehicle is slower to have confidence in new states. This results in lower confidence for any one state during most of the simulation where the states rapidly change.

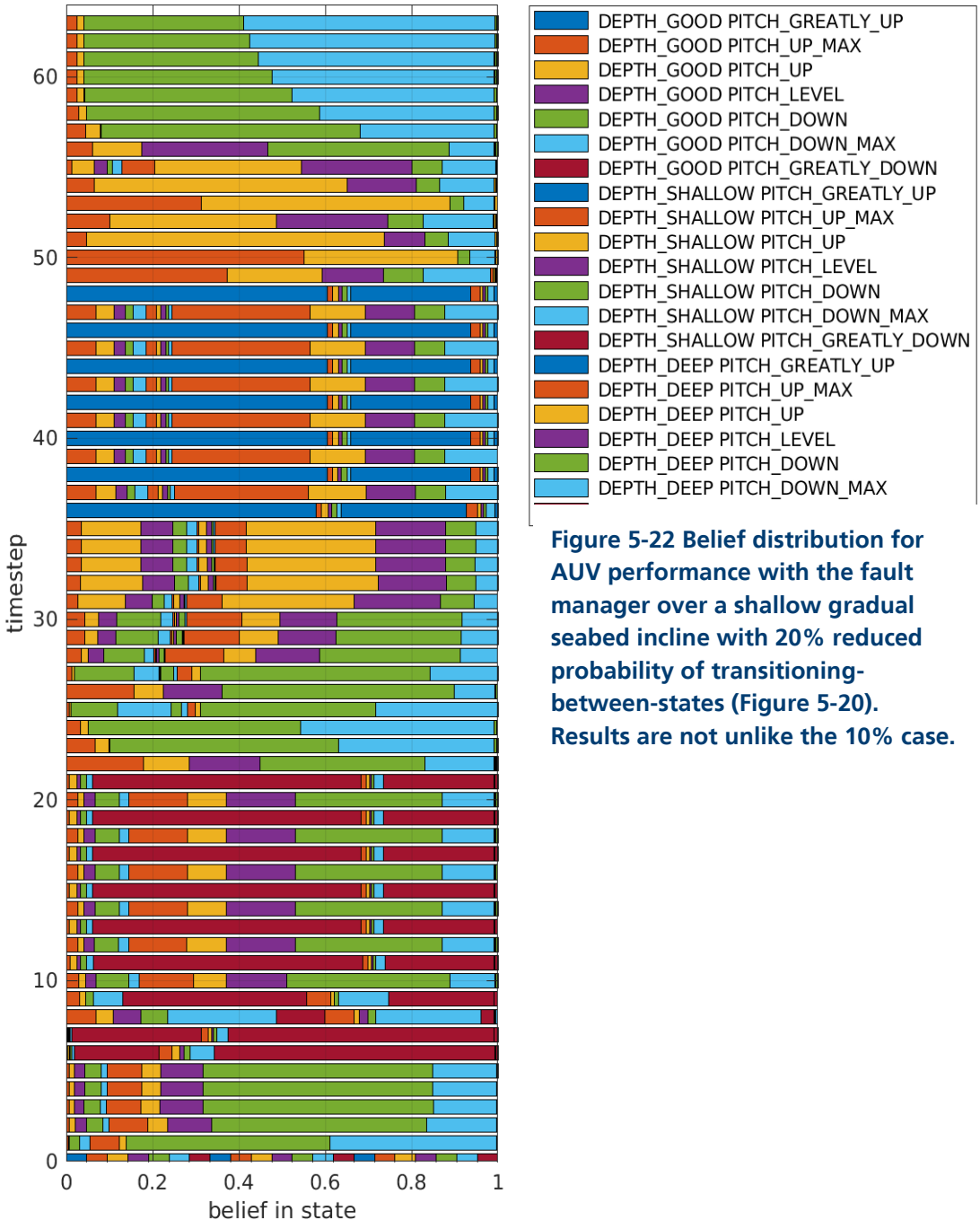

Figure 5-22 Belief distribution for AUV performance with the fault manager over a shallow gradual seabed incline with 20% reduced probability of transitioning-between-states (Figure 5-20). Results are not unlike the 10% case.

In the 40% (Figure 5-23) reduced model the fault management system lacks confidence in the states, although it does have more confidence than it did for the 40% reduction to the observation function (see section 5.1.4.4). This makes sense as the observations drive the belief in the state although the fault manager determining the best actions would be problematic.

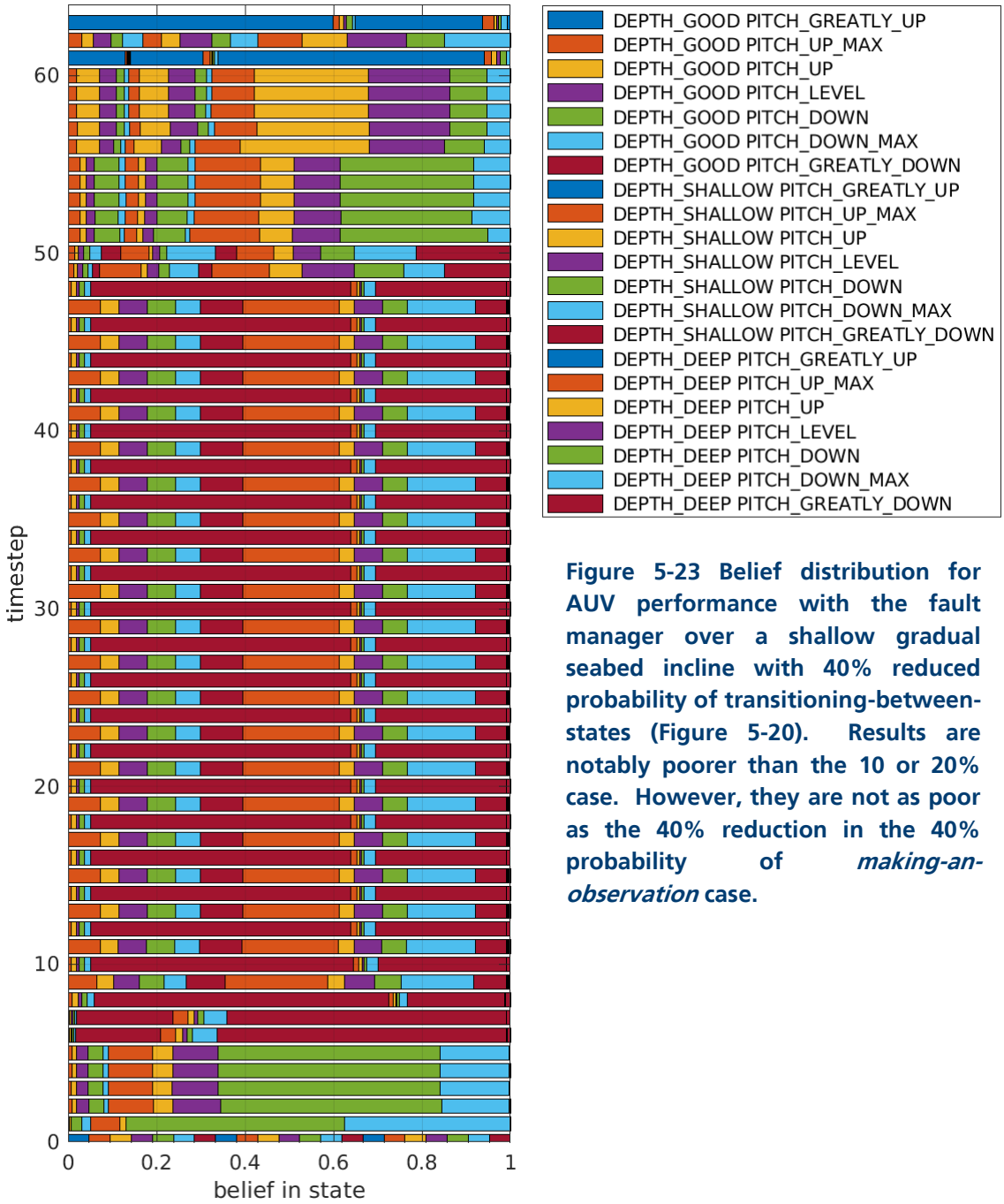

Figure 5-23 Belief distribution for AUV performance with the fault manager over a shallow gradual seabed incline with 40% reduced probability of transitioning-between-states (Figure 5-20). Results are notably poorer than the 10 or 20% case. However, they are not as poor as the 40% reduction in the 40% probability of *making-an-observation* case.

This set of simulations show that reductions in the probabilities of transitioning-between-states has less impact on the vehicle belief state and how the fault manager chooses actions. The fault manager when the probabilities were reduced by 10% and 20% was still able to function, however at 40% while still maintaining a better belief

distribution than 40% reduction in the observation function (section 5.1.4.4) the fault manager was unable to choose actions and the vehicle collided with the seabed.

The next set of tests highlight the impact of reductions in probabilities for the transition-between-states and making-an-observation functions and the addition of Gaussian additive noise.

5.1.4.3 OBSERVATION AND TRANSITION FUNCTIONS' PROBABILITIES REDUCED BY 10%, 20%, AND 40%

The next series of tests explore the combined effects of the probability of transitioning-between-states, $T(s' | s,a)$, and of making-an-observation, $O(o | s,a)$, on the POMDP. It reduces the probabilities in the POMDP model at a set rate. Note, these probabilities are independent – not a joint-probability. Three tests were conducted using rates of 10%, 20%, and 40%.

In Figure 5-36(a), the two simulations of reduced probabilities of 10% and 20% were similar. The reduced simulation of 20% had more fin mode (d) changes which causes more variation in vehicle pitch (Figure 5-36 (c)). The 40% reduction meant the vehicle was unable to avoid the seabed (Figure 5-36 (a)) and an extremely low pitch (Figure 5-36 (c)) which results in complete loss of altitude (Figure 5-36 (b)). This makes sense given the results in sections 5.1.4.4 and 5.1.4.5. With both probability functions so greatly reduced, the fault manager was unable to determine its state or choose actions to properly drive the vehicle.

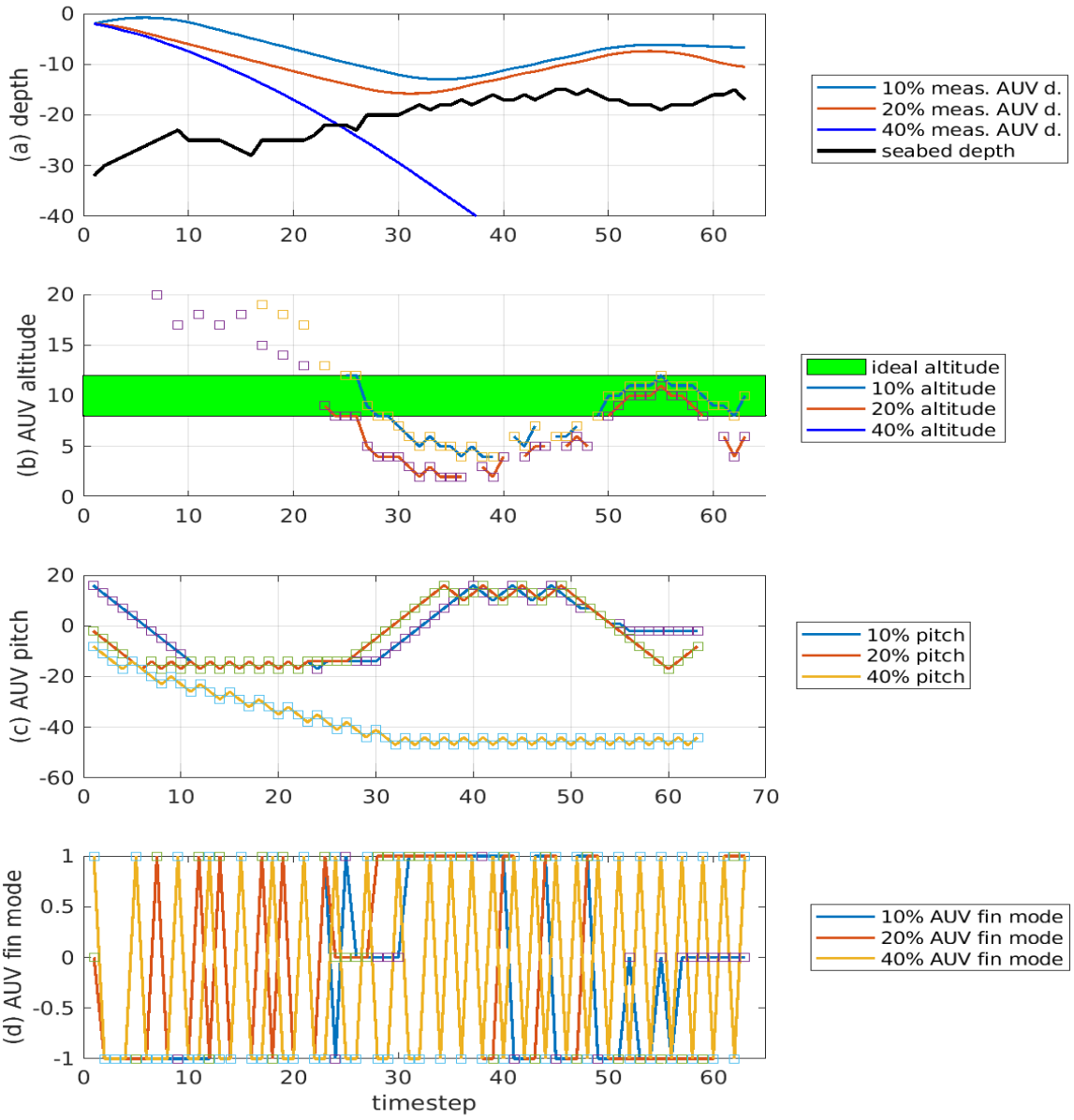

Figure 5-24 The probabilities of *making-an-observation* and *transitioning-between-states* functions are both eroded by 10, 20 and 40%: AUV performance with the fault manager driving the fin mode over a shallow gradual incline environment. (a) The 10% and 20% track the same general path. The 40% case has the AUV collide with the seabed. (b) AUV altitude follows the trends in (a). (c) Pitch oscillations increase at 20% reduction. (d) Fin mode trends, expectedly, follow (c). The desired altitude range is given by the green band in (b).

Similar to results in sections 5.1.4.4 and 5.1.4.5 the reduction of both probability (Figure 5-25) functions by 10% has minimal impact on the state belief. There is overall less confidence in any one state for each timestep compared to the original (un-reduced) simulations in section 5.1.1.1.

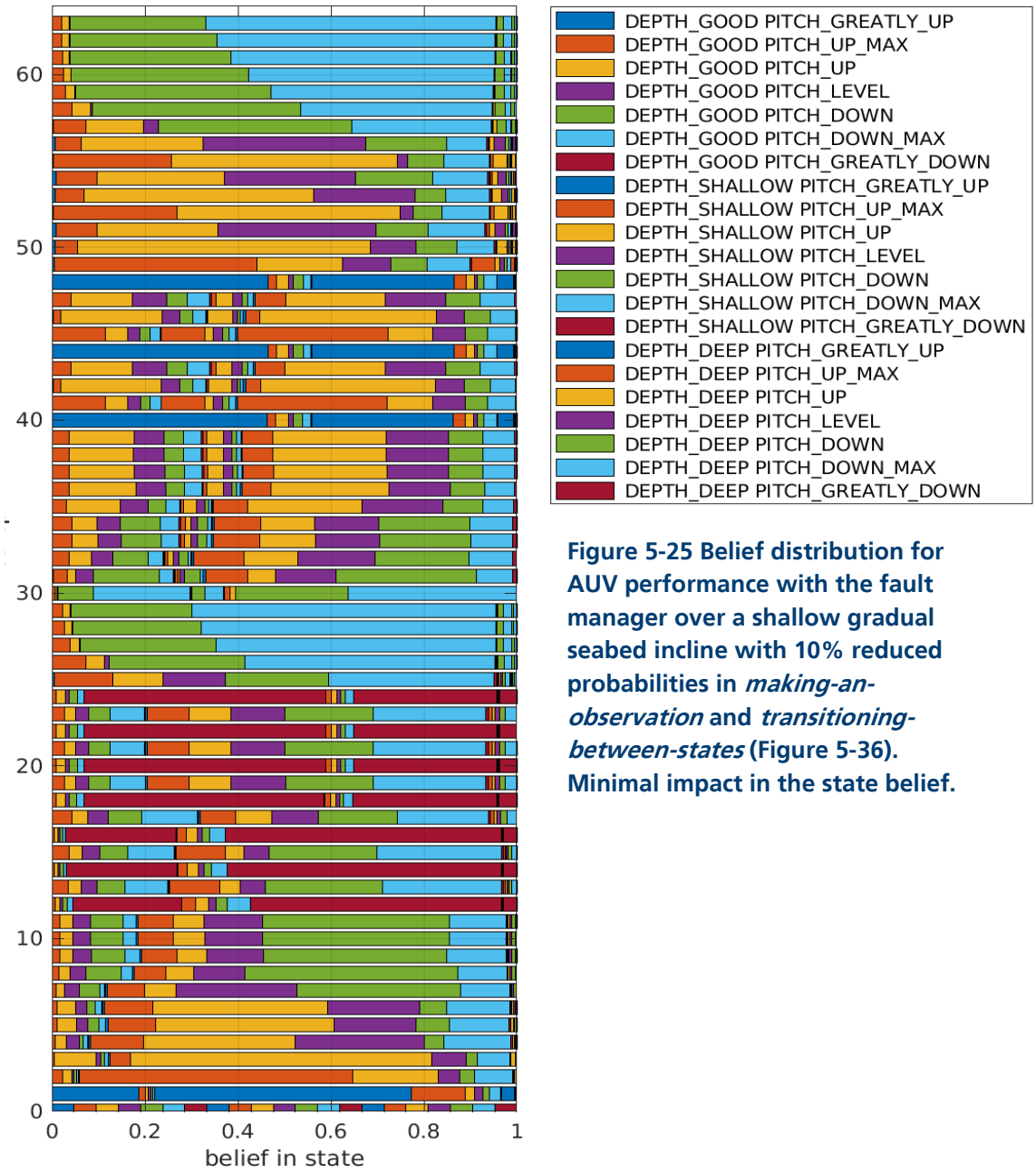

Figure 5-25 Belief distribution for AUV performance with the fault manager over a shallow gradual seabed incline with 10% reduced probabilities in *making-an-observation* and *transitioning-between-states* (Figure 5-36). Minimal impact in the state belief.

The reduction of 20% for both probability functions results in a much less confident fault manager. This creates a more variable belief state as shown in Figure 5-26.

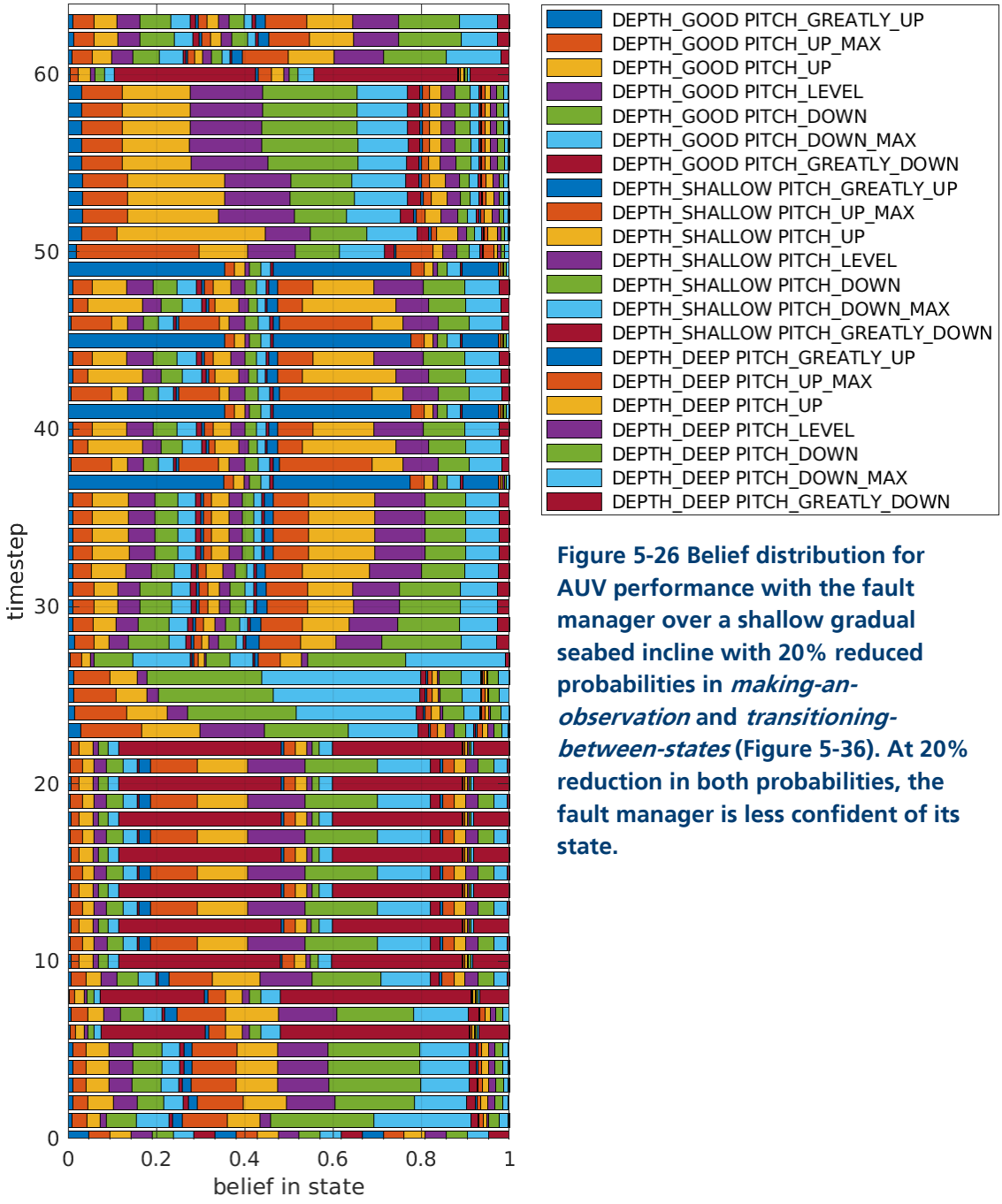

Figure 5-26 Belief distribution for AUV performance with the fault manager over a shallow gradual seabed incline with 20% reduced probabilities in *making-an-observation* and *transitioning-between-states* (Figure 5-36). At 20% reduction in both probabilities, the fault manager is less confident of its state.

Finally, similar to the results in sections 5.1.4.4 and 5.1.4.5, the reduction of 40% (Figure 5-27) results in a fault manager incapable of having confidence in any one state.

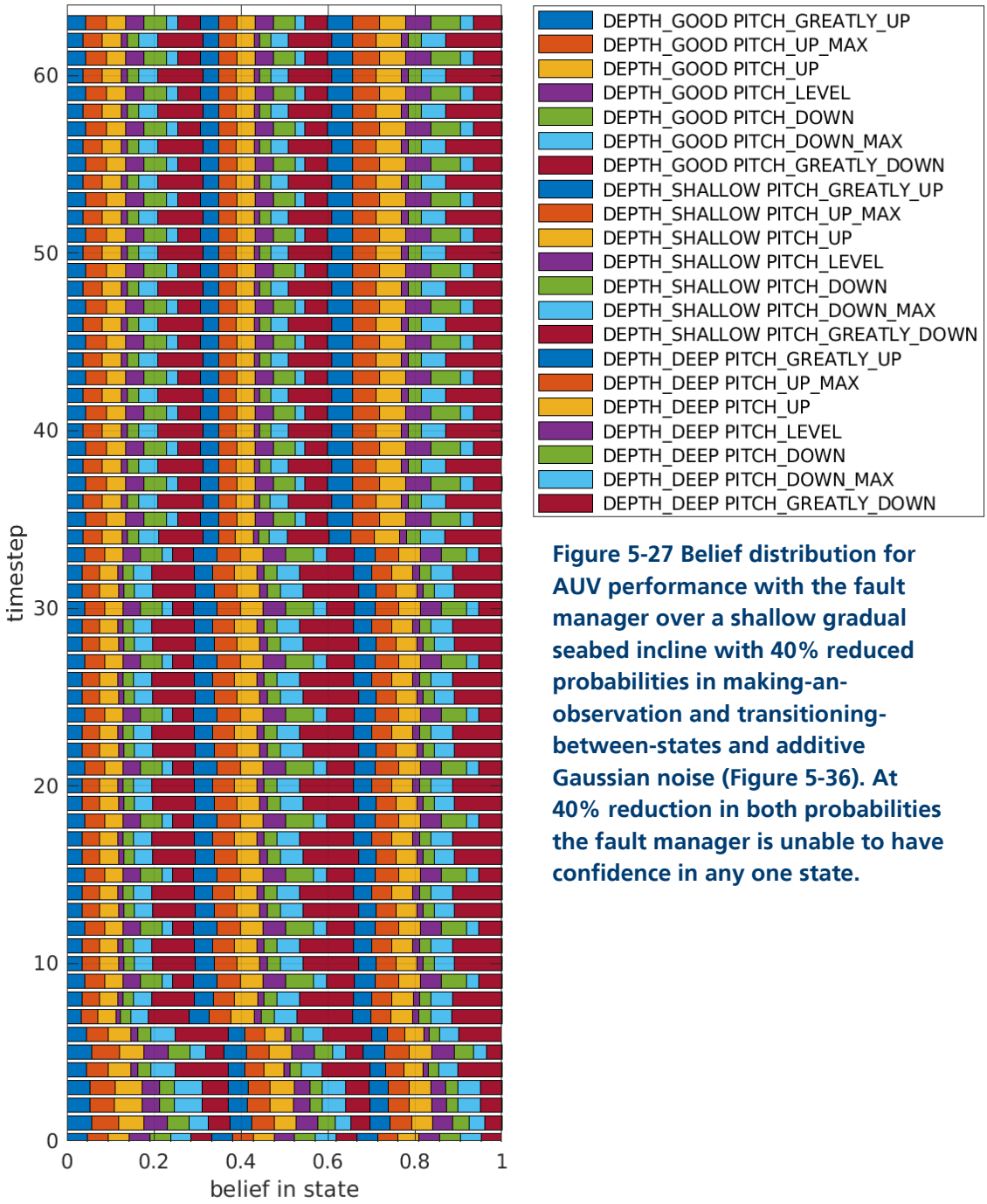

Figure 5-27 Belief distribution for AUV performance with the fault manager over a shallow gradual seabed incline with 40% reduced probabilities in making-an-observation and transitioning-between-states and additive Gaussian noise (Figure 5-36). At 40% reduction in both probabilities the fault manager is unable to have confidence in any one state.

These series of simulations demonstrated that reduction of the probability functions result in the system less sure of its measurements and the actions required. This is further explored in the next few simulations that repeat the reduction along with an additive Gaussian noise induced in the measurements.

5.1.4.4 OBSERVATION FUNCTION PROBABILITIES REDUCED BY 10%, 20% AND 40% WITH ADDITIVE NOISE

These tests explore the impact of the probability of *making-an-observation* function, *O(o| s,a)*, in the POMDP. It reduces the set probabilities in the POMDP model by a controlled amount. Three tests were conducted with reductions of 10%, 20%, and 40%. Additive Gaussian noise was induced in the AUV depth measurement, similar to that of the simulation in section 5.1.1.2. This was given a standard deviation of 2 meters (see equation 4-6 in section 4.4.1) from the actual measurement set in the initialization file (see Appendix C – Table).

As shown in Figure 5-28(a) the measured AUV depth is noisy from the additive noise. The altitude follows and is likewise noisy. The three different simulations of reduced observation functions have the same trends as the previous simulation without noise (5.1.4.1). The again 40% reduction in the probability has the most dramatic effect with the vehicle having very little confidence in the observations made of the rising seabed and consequently, colliding with the seabed. Similarly, due to the high pitch angles from the few altitude measurements that were achieved, there are much more instances of fin mode changes resulting in an unstable pitch. This is due to the fault manager's uncertainty in its state being reflected in its responses. The noise increases the uncertainty in the observations.

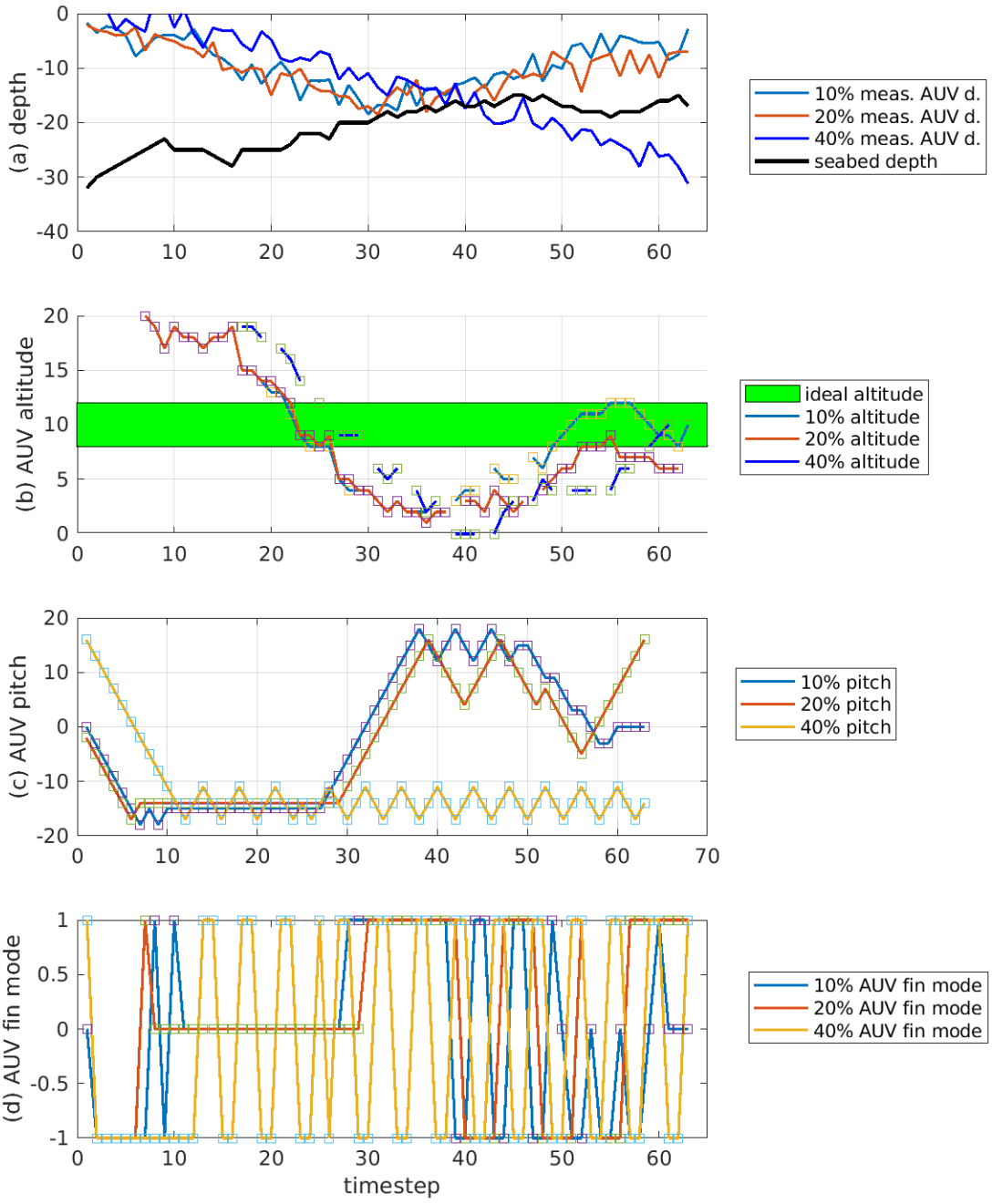

Figure 5-28 Observation probabilities reduced by 10%, 20% and 40% and additive Gaussian noise in depth measurements: AUV performance with the fault manager over a shallow gradual seabed incline. In all cases, the results are poorer with increase reductions in the observation probabilities. The desired altitude range is given by the green band in (b).

In the 10% reduced model there is not much change from the original model in section 5.1.1.2. There is a reduction in the confidence of its state compared to the noise-free model (section 5.1.4.1)The vehicle is confident of its state throughout the simulation run (Figure 5-29).

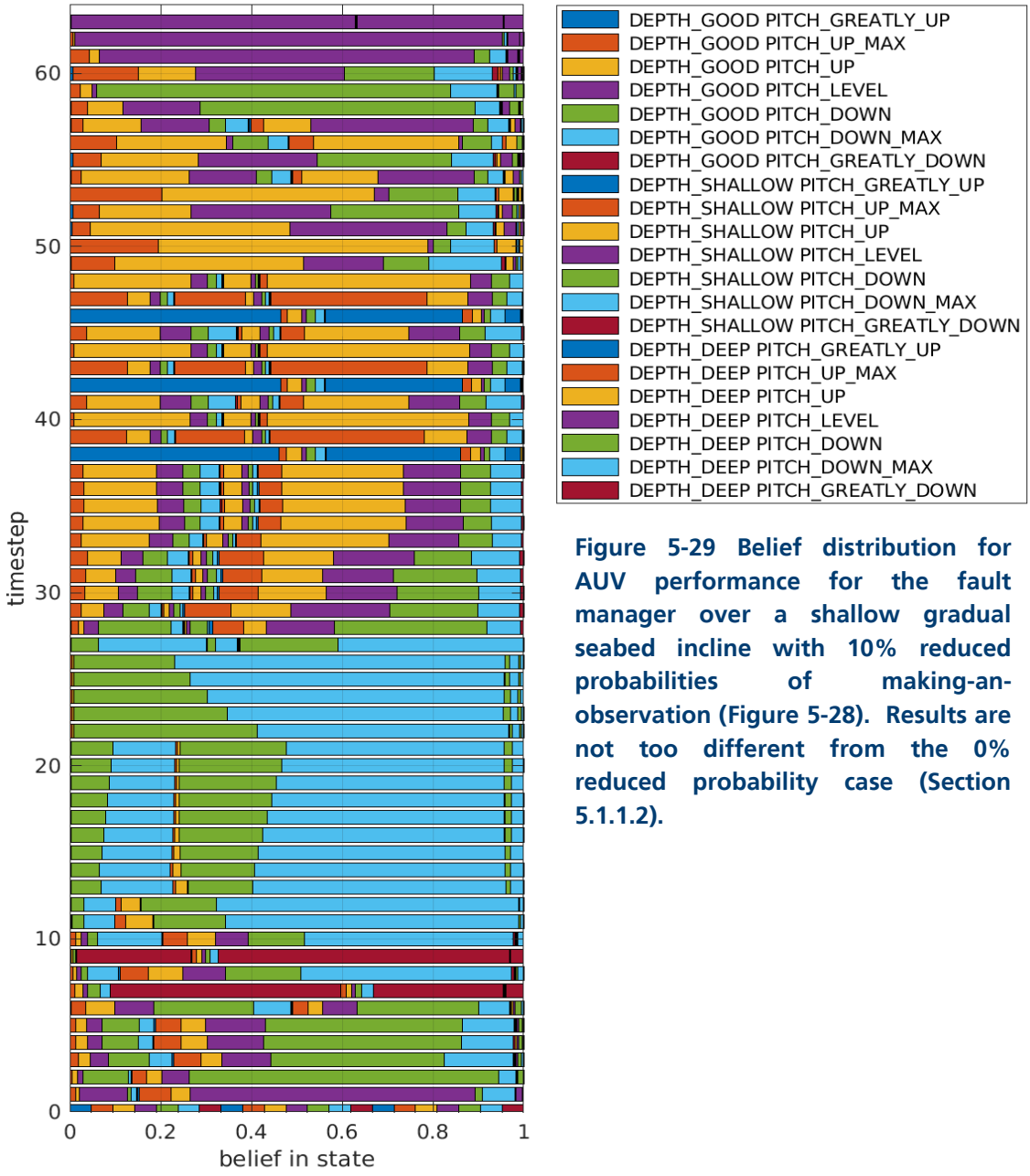

Figure 5-29 Belief distribution for AUV performance for the fault manager over a shallow gradual seabed incline with 10% reduced probabilities of making-an-observation (Figure 5-28). Results are not too different from the 0% reduced probability case (Section 5.1.1.2).

In the 20% reduced model (Figure 5-30) the fault management system still has confidence in its belief state in the first half of the simulation however; as the seabed rises and forces the vehicle to adapt the fault manager becomes less sure of its state. There is a decrease in the confidence compared to the previous section 5.1.4.1.

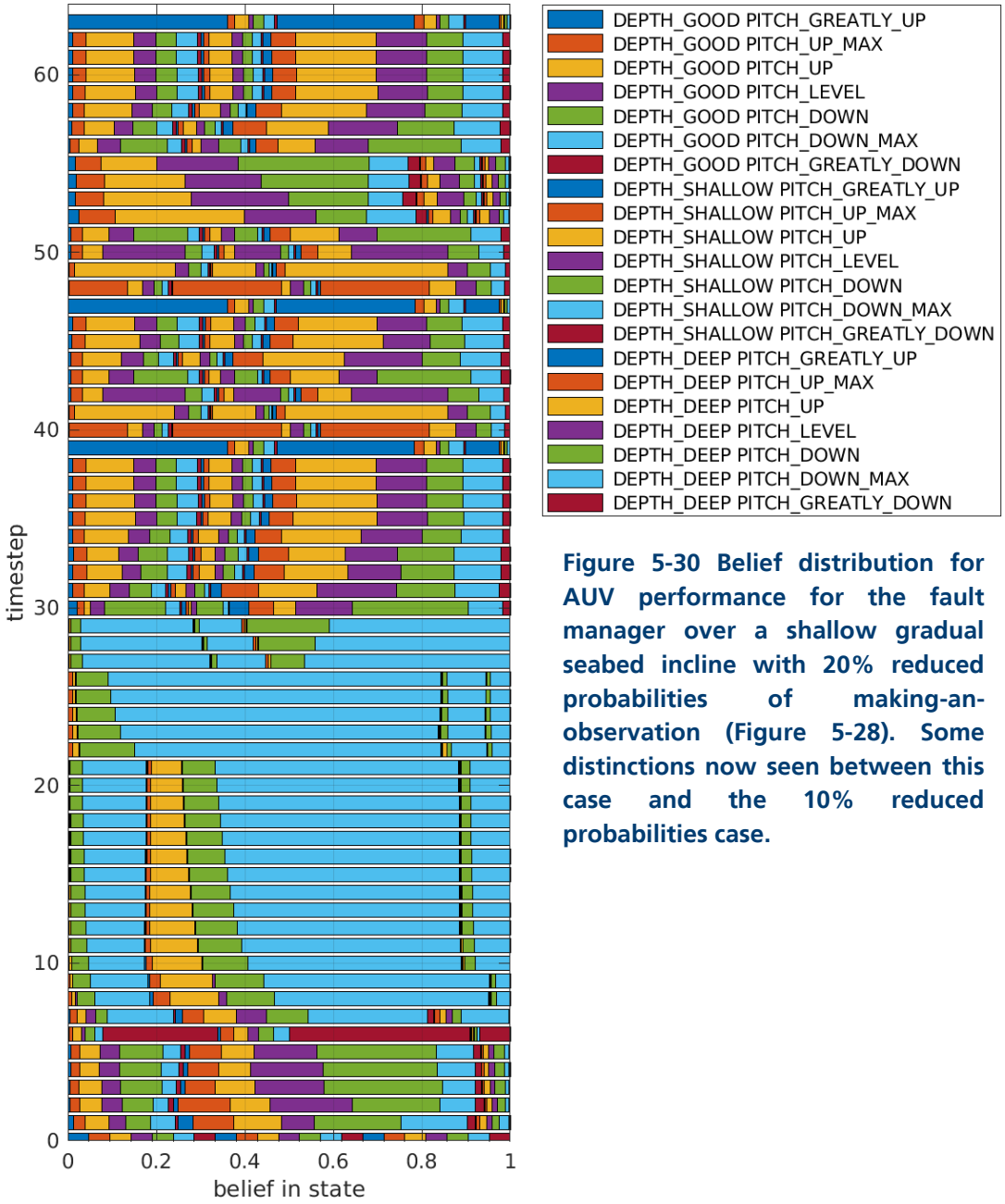

Figure 5-30 Belief distribution for AUV performance for the fault manager over a shallow gradual seabed incline with 20% reduced probabilities of making-an-observation (Figure 5-28). Some distinctions now seen between this case and the 10% reduced probabilities case.

In the 40% reduced model (Figure 5-31) the fault management system lacks confidence in the vehicle state. This is due to POMDP model not being able to place confidence in the observations due to the highly reduced probability of making those observations given any state or prior actions.

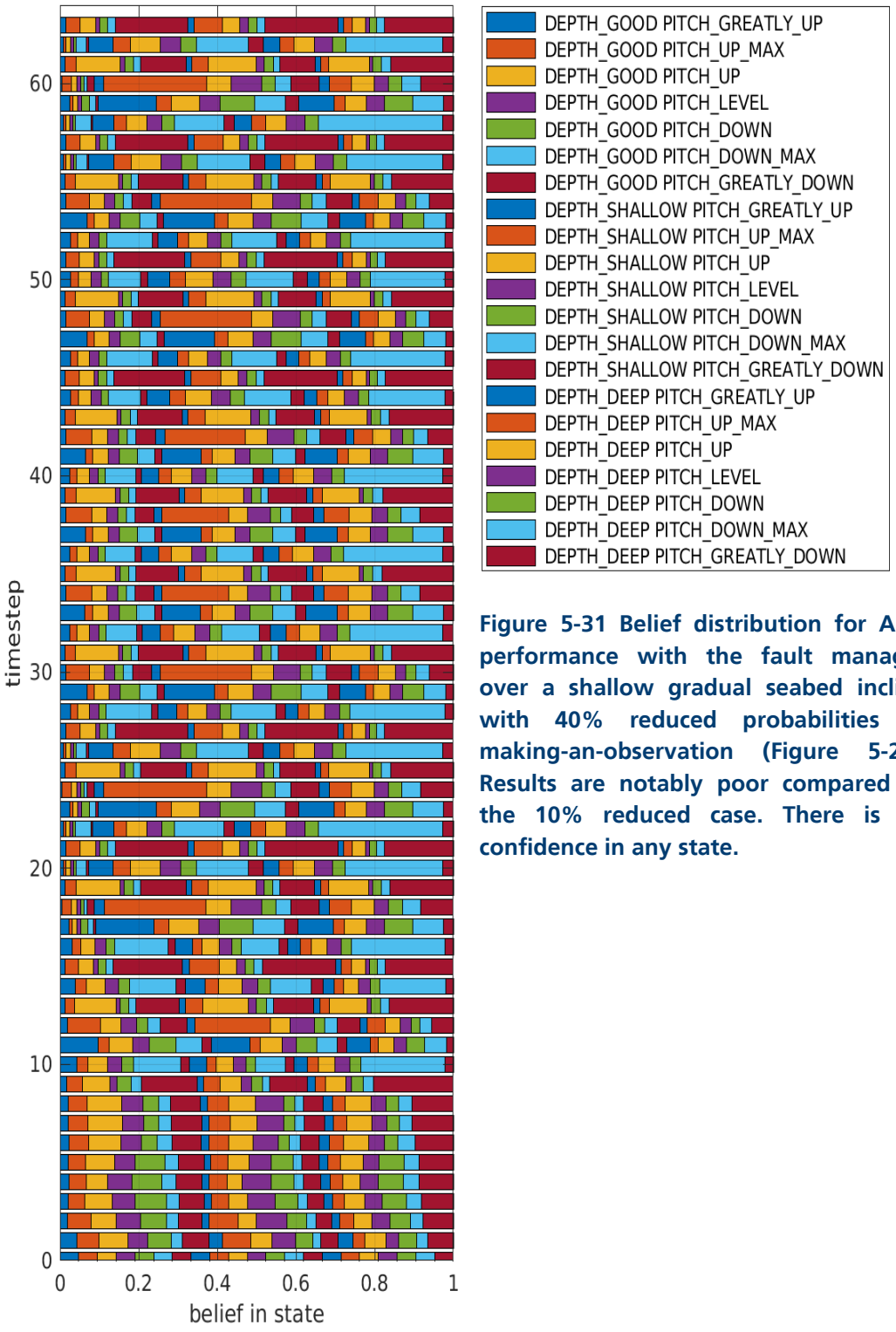

Figure 5-31 Belief distribution for AUV performance with the fault manager over a shallow gradual seabed incline with 40% reduced probabilities of making-an-observation (Figure 5-28). Results are notably poor compared to the 10% reduced case. There is no confidence in any state.

These simulations clearly demonstrate how reducing the probability of observation and additive Gaussian noise on the sensor measurement causes the vehicle fault manager to have less confidence in its state. In the 40% case, the vehicle is unable to

have confidence in any state. The additive noise can also result in observations that are false causing the vehicle to perform unnecessary or counter-productive actions. It also results in a lower confidence of state due to conflicting observations.

The next set of tests likewise have additive Gaussian noise however; the reduction in probabilities were applied to the *transitioning-between-states* function instead.

5.1.4.5 TRANSITION FUNCTION PROBABILITIES REDUCED BY 10%, 20% AND 40% WITH ADDITIVE NOISE

These tests explore the impact of the probability of transitioning-between-states, $T(s' | s,a)$, POMDP function. It reduces the probabilities in the POMDP model by a set rate, namely 10%, 20%, and 40%. Additive Gaussian noise was also added to the AUV simulation depth measurements, similar to section 5.1.1.2. The depth measurement was assigned a standard deviation of 2 meters (see equation 4-6 in section 4.4.1) from the actual measurement (set in the initialization file, see Appendix C – Table).

In Figure 5-32 (a), the measured AUV depth is sporadic due to the additive noise. The altitude likewise follows suit from the additive Gaussian noise. The simulations for reductions of 10% and 20% perform similarly. The 20% reduction simulation has more fin mode changes and greater variation in the pitch. The simulation with 40% reduction is unable to prevent collisions with the seabed or achieve bottom-lock for altitude readings. The lack of confidence in its state has the consequence of being unable to predict states. Similarly, the lack confidence in its actions means it is unable to change the state to a desired outcome. The fin mode varies rapidly for the 40% reduced model and the change in pitch to rise above the seabed comes far too late.

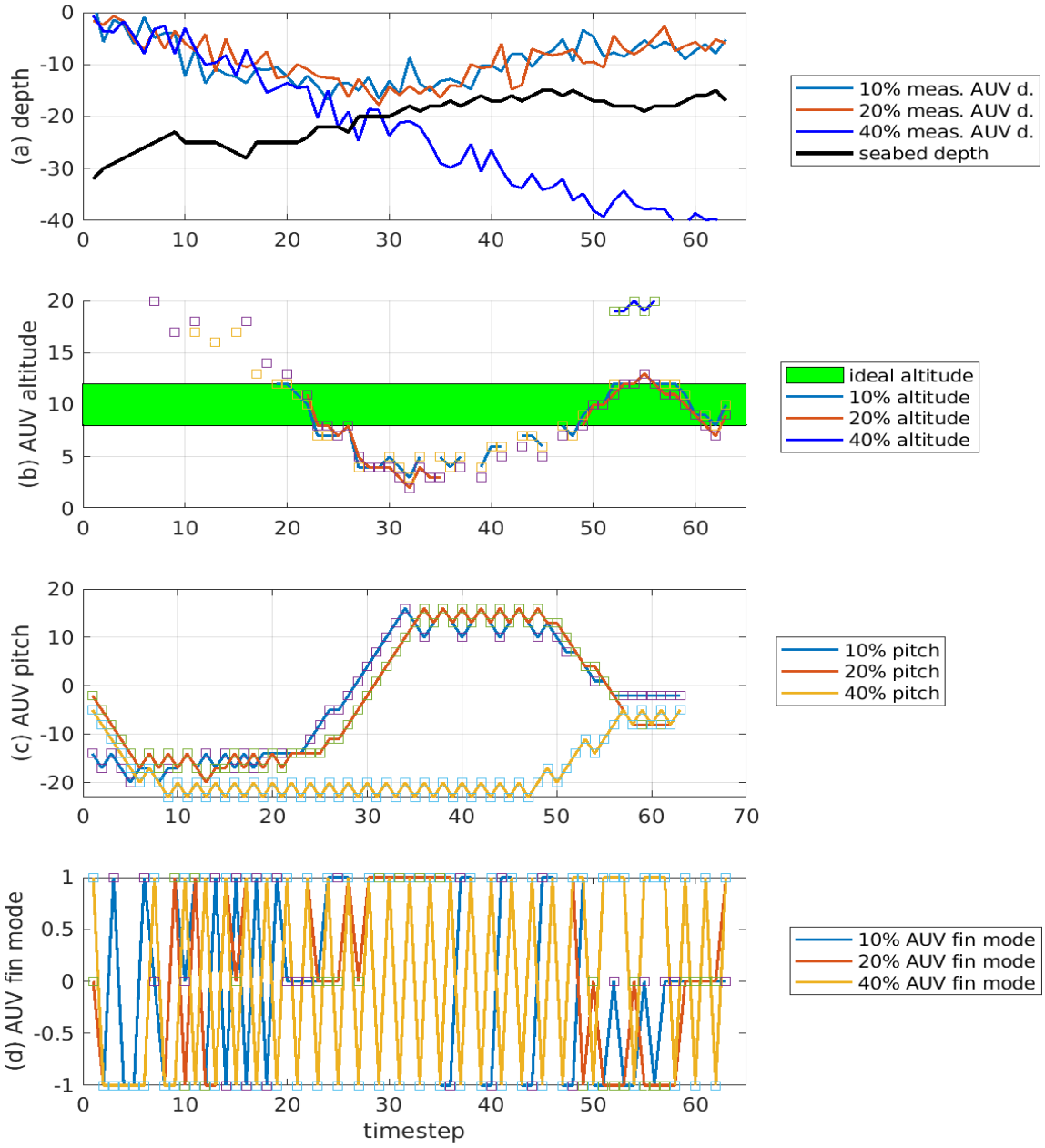

Figure 5-32 State transition probabilities reduced by 10%, 20% and 40%: AUV performance with the fault manager over a shallow gradual incline environment. Generally, the AUV performance is poorer and more uncertain with increasingly reduced probabilities where the fault manager is unable to direct the AUV towards the desired outcomes. The desired altitude range is given by the green band in (b).

In the 10% reduction (Figure 5-33) to the probability of *transitioning-between-states* model there is little change from the original model in section 5.1.1.2, and an decrease in confidence compared to the noise-free model (section 5.1.4.2). The vehicle can still have confidence in its state throughout the simulation.

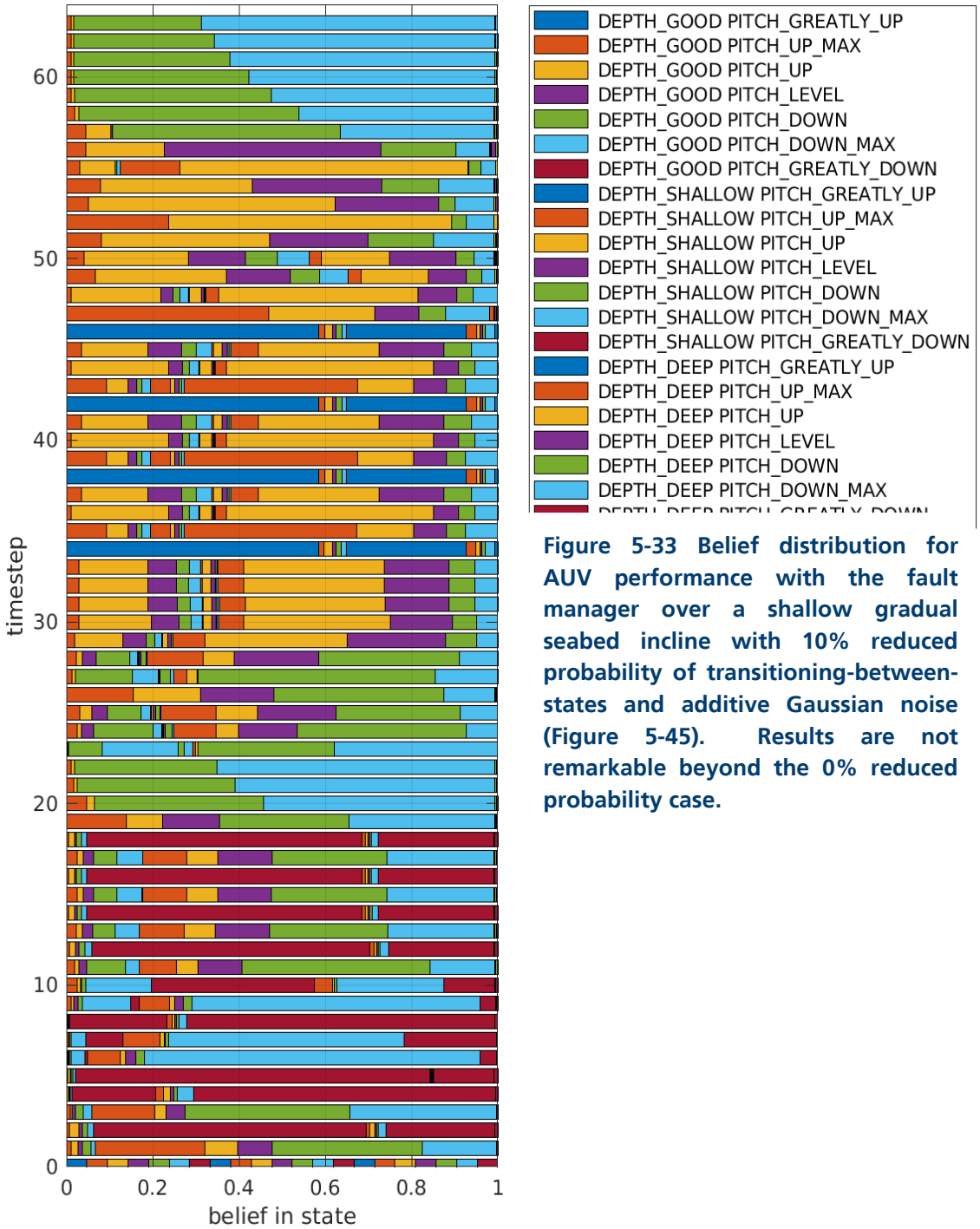

Figure 5-33 Belief distribution for AUV performance with the fault manager over a shallow gradual seabed incline with 10% reduced probability of transitioning-between-states and additive Gaussian noise (Figure 5-45). Results are not remarkable beyond the 0% reduced probability case.

The difference between a reduction of 20% (Figure 5-34) and 10% (Figure 5-33) is not significant although it is noticeable that the vehicle is slower to have confidence in new states. This results in lower confidence for any one state during most of the simulation where the states rapidly change.

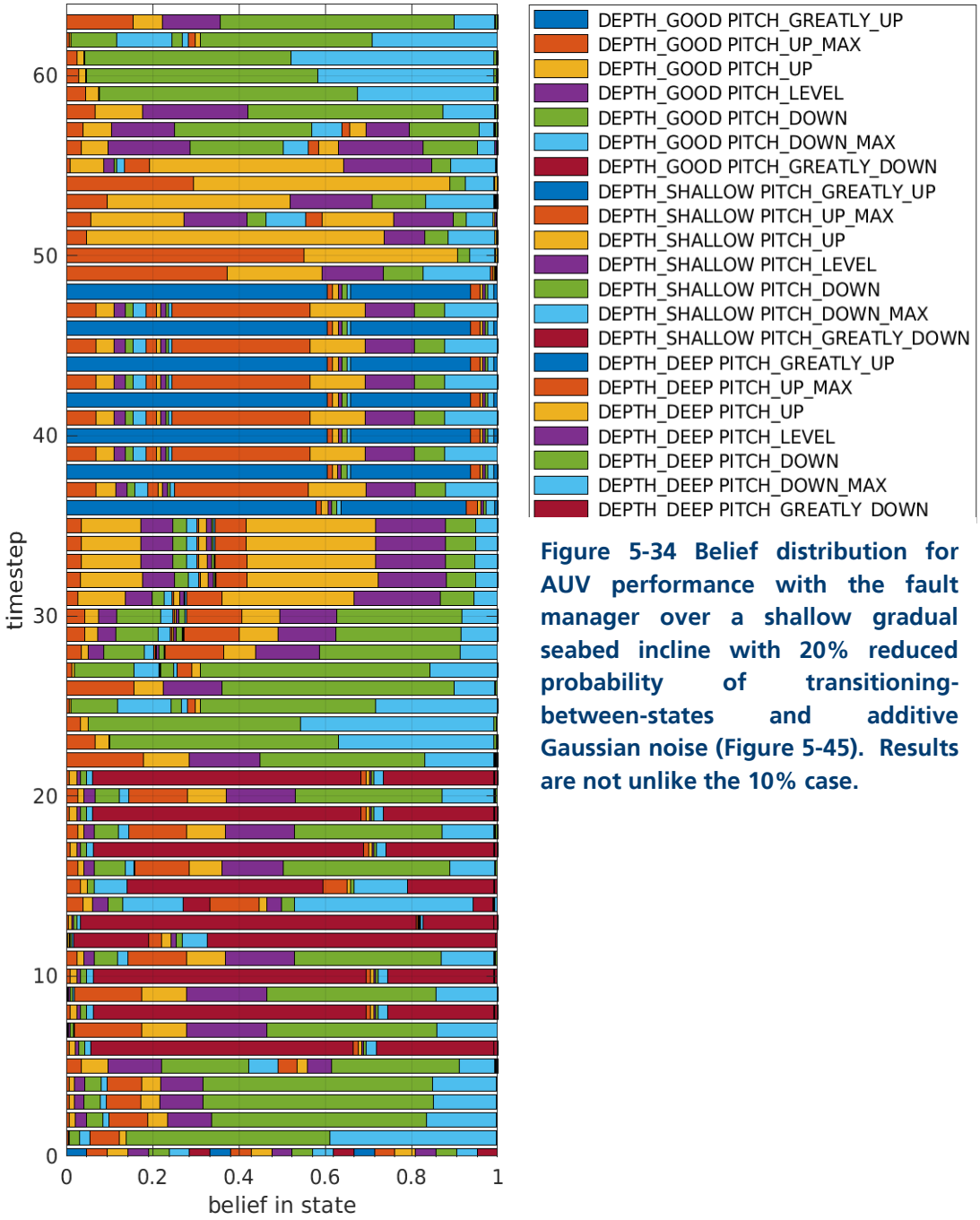

Figure 5-34 Belief distribution for AUV performance with the fault manager over a shallow gradual seabed incline with 20% reduced probability of transitioning-between-states and additive Gaussian noise (Figure 5-45). Results are not unlike the 10% case.

In the 40% reduced model (Figure 5-35) the fault management system lacks confidence in the states, although it does have more confidence than it did for the 40% reduction to the observation function (see section 5.1.4.4). This makes sense as the observations drive the belief in the state although the fault manager determining the best actions would be problematic.

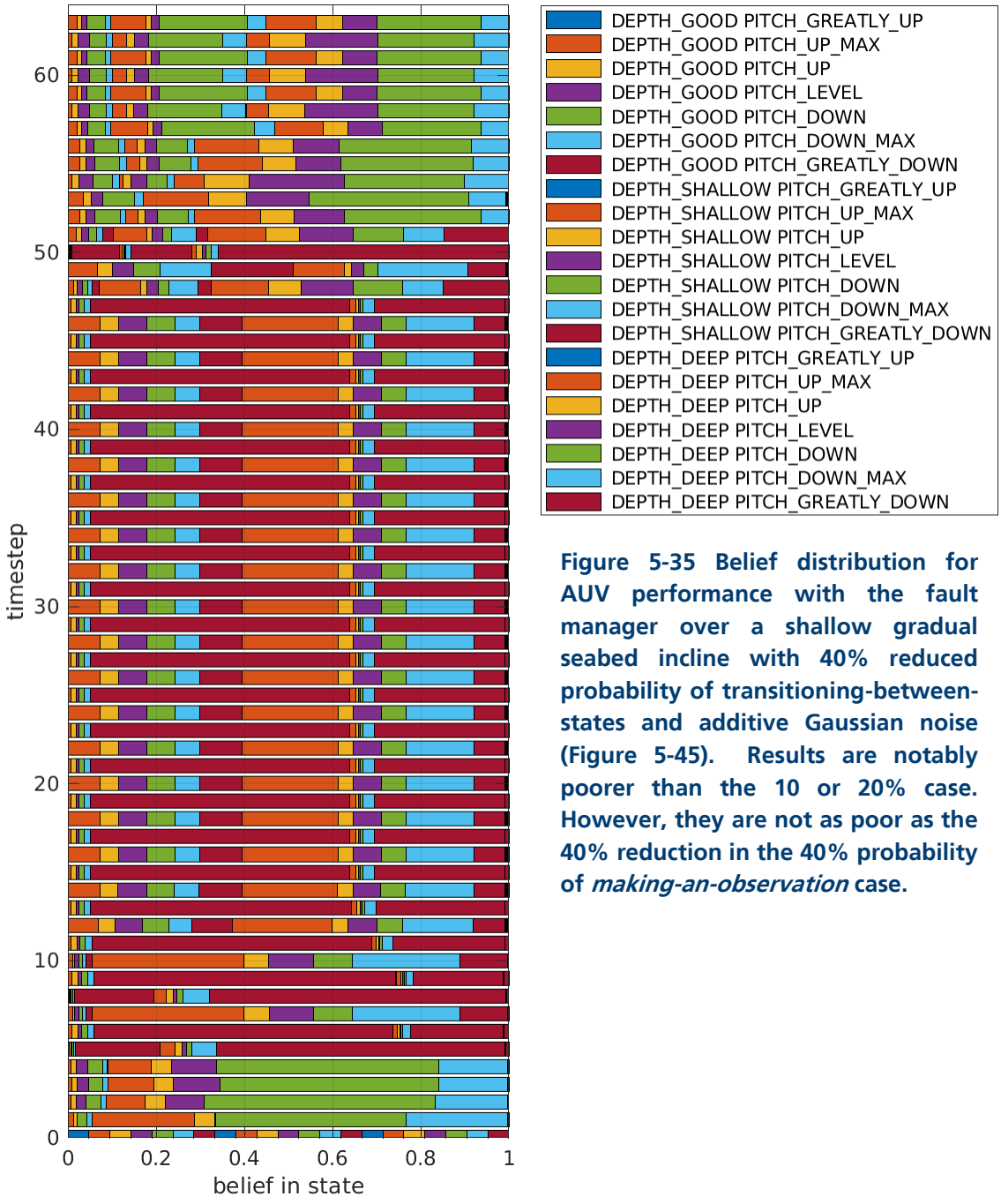

Figure 5-35 Belief distribution for AUV performance with the fault manager over a shallow gradual seabed incline with 40% reduced probability of transitioning-between-states and additive Gaussian noise (Figure 5-45). Results are notably poorer than the 10 or 20% case. However, they are not as poor as the 40% reduction in the 40% probability of *making-an-observation* case.

This set of simulations show that reductions in the probabilities of transitioning-between-states and a noisy sensor measurement has less impact on the vehicle belief state and how the fault manager chooses actions. The fault manager when the probabilities were reduced by 10% and 20% was still able to function, however at 40% while still maintaining a better belief distribution than 40% reduction in the observation function (section 5.1.4.4) the fault manager was unable to choose actions and the vehicle collided with the seabed. The Gaussian noise resulted in less confidence in state due to conflict between observations as compared to the noise-free model 5.1.4.2.

The next set of tests highlight the impact of reductions in probabilities for the transition-between-states and making-an-observation functions and the addition of Gaussian additive noise.

5.1.4.6 OBSERVATION AND TRANSITION FUNCTIONS' PROBABILITIES REDUCED BY 10%, 20%, AND 40%

These series of tests explore the combined effects of the probability of transitioning-between-states, $T(s' | s,a)$, and of making-an-observation, $O(o | s,a)$, on the POMDP. It reduces the probabilities in the POMDP model at a set rate. Note, these probabilities are distinct – not a joint-probability. Three tests were conducted using rates of 10%, 20%, and 40%. Additive Gaussian noise was also added to the AUV simulation's depth measurements. This was assigned a standard deviation of 2 meters (see equation 4-6 in section 4.4.1) from the actual measurement set in the initialization file (see Appendix C – Table).

In Figure 5-36(a), the measured AUV depth is sporadic with the additive noise. The altitude likewise follows suit from the additive noise. These two simulations of reduced probabilities of 10% and 20% were similar. The reduced simulation of 20% had more fin mode (d) changes which causes more variation in vehicle pitch (Figure 5-36 (c)). The 40% reduction meant the vehicle was unable to avoid the seabed (Figure 5-36 (a)) and an extremely low pitch (Figure 5-36 (c)) which results in complete loss of altitude (Figure

5-36 (b)). This makes sense given the results in sections 5.1.4.4 and 5.1.4.5. With both probability functions so greatly reduced, the fault manager was unable to determine its state or choose actions to properly drive the vehicle.

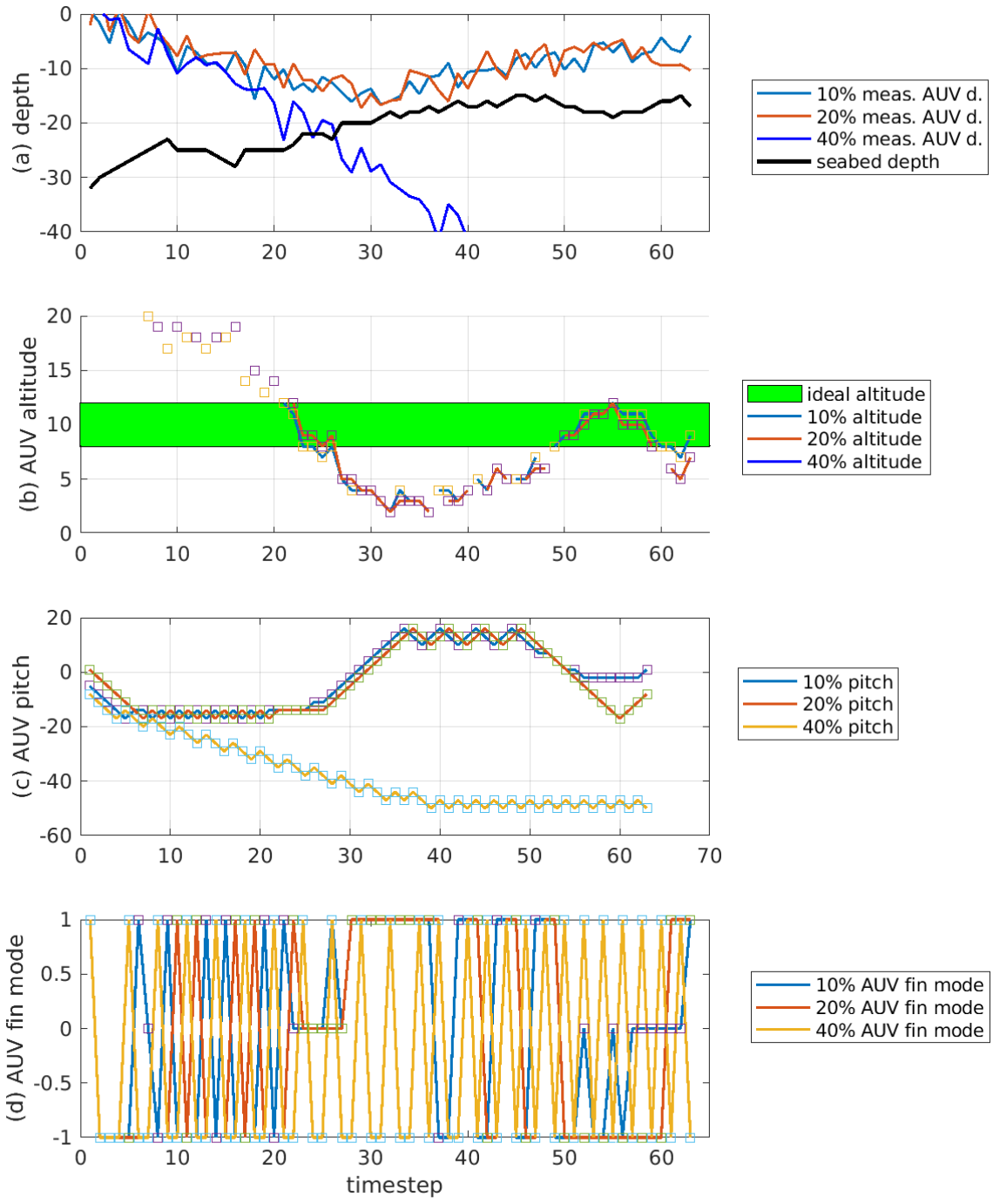

Figure 5-36 The probabilities of *making-an-observation* and *transitioning-between-states* functions are both eroded by 10, 20 and 40%: AUV performance with the fault manager driving the fin mode over a shallow gradual incline environment. (a) The AUV depth is sporadic due to the additive Gaussian noise. The 40% case has the AUV collide with the seabed. (b) AUV altitude follows the trends in (a). (c) Pitch oscillations increase at 20% reduction. (d) Fin mode trends, expectedly, follow (c). The desired altitude range is given by the green band in (b).

Similar to results in sections 5.1.4.4 and 5.1.4.5 the reduction of both probability functions by 10% (Figure 5-37) has minimal impact on the state belief. There is overall less confidence in any one state for each timestep compared to the original (un-reduced) simulations in section 5.1.1.2. The noise lowers the confidence of the belief in state; however, the fault manager is still able to act similarly to previous noise-free models (section 5.1.4.3).

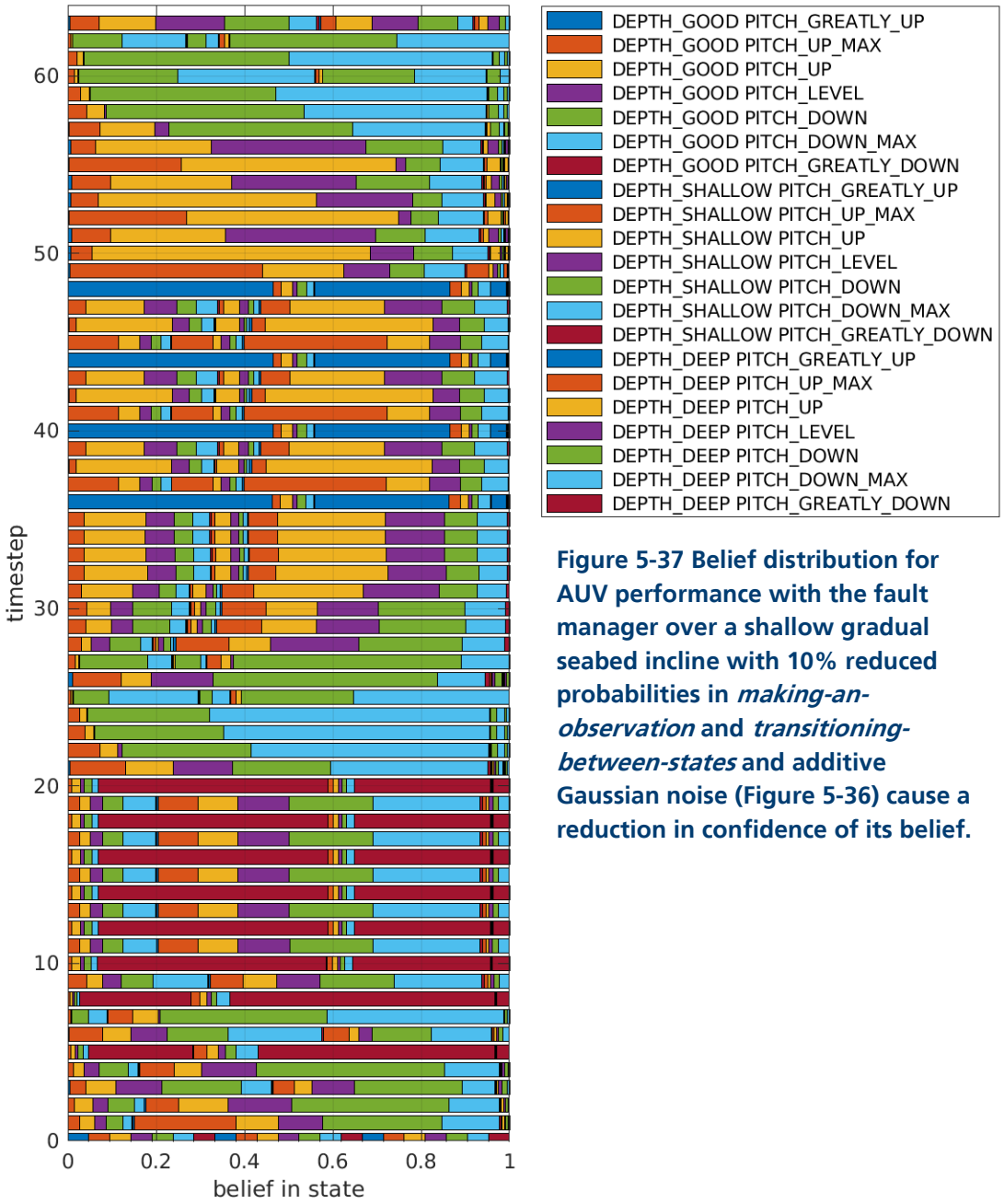

Figure 5-37 Belief distribution for AUV performance with the fault manager over a shallow gradual seabed incline with 10% reduced probabilities in *making-an-observation* and *transitioning-between-states* and additive Gaussian noise (Figure 5-36) cause a reduction in confidence of its belief.

The reduction of 20% for both probability functions results in a much less confident fault manager. This creates a more variable belief state as shown in Figure 5-38.

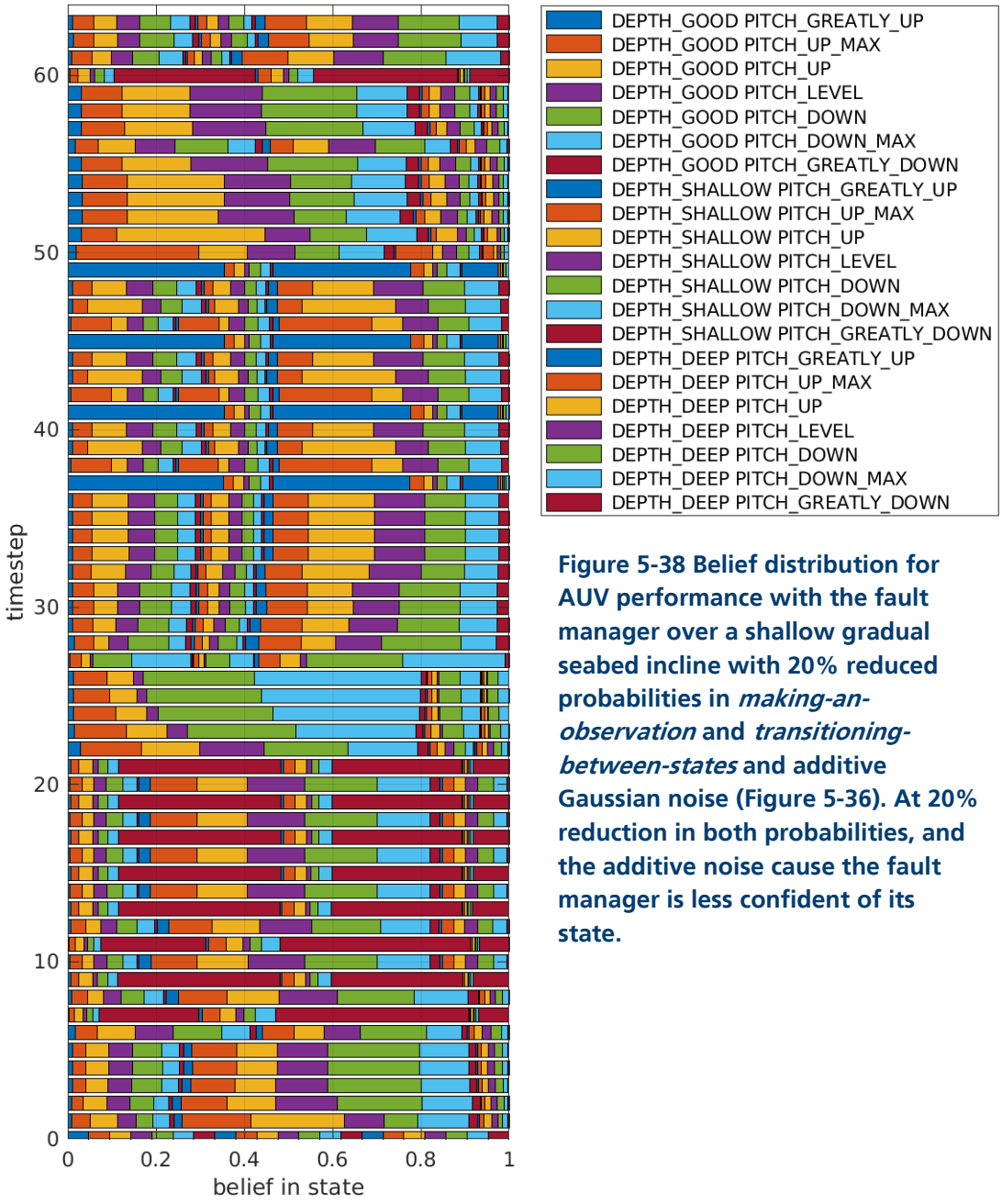

Figure 5-38 Belief distribution for AUV performance with the fault manager over a shallow gradual seabed incline with 20% reduced probabilities in *making-an-observation* and *transitioning-between-states* and additive Gaussian noise (Figure 5-36). At 20% reduction in both probabilities, and the additive noise cause the fault manager is less confident of its state.

Finally, similar to the results in sections 5.1.4.4 and 5.1.4.5, the reduction of 40% (Figure 5-39) results in a fault manager incapable of having confidence in any one state.

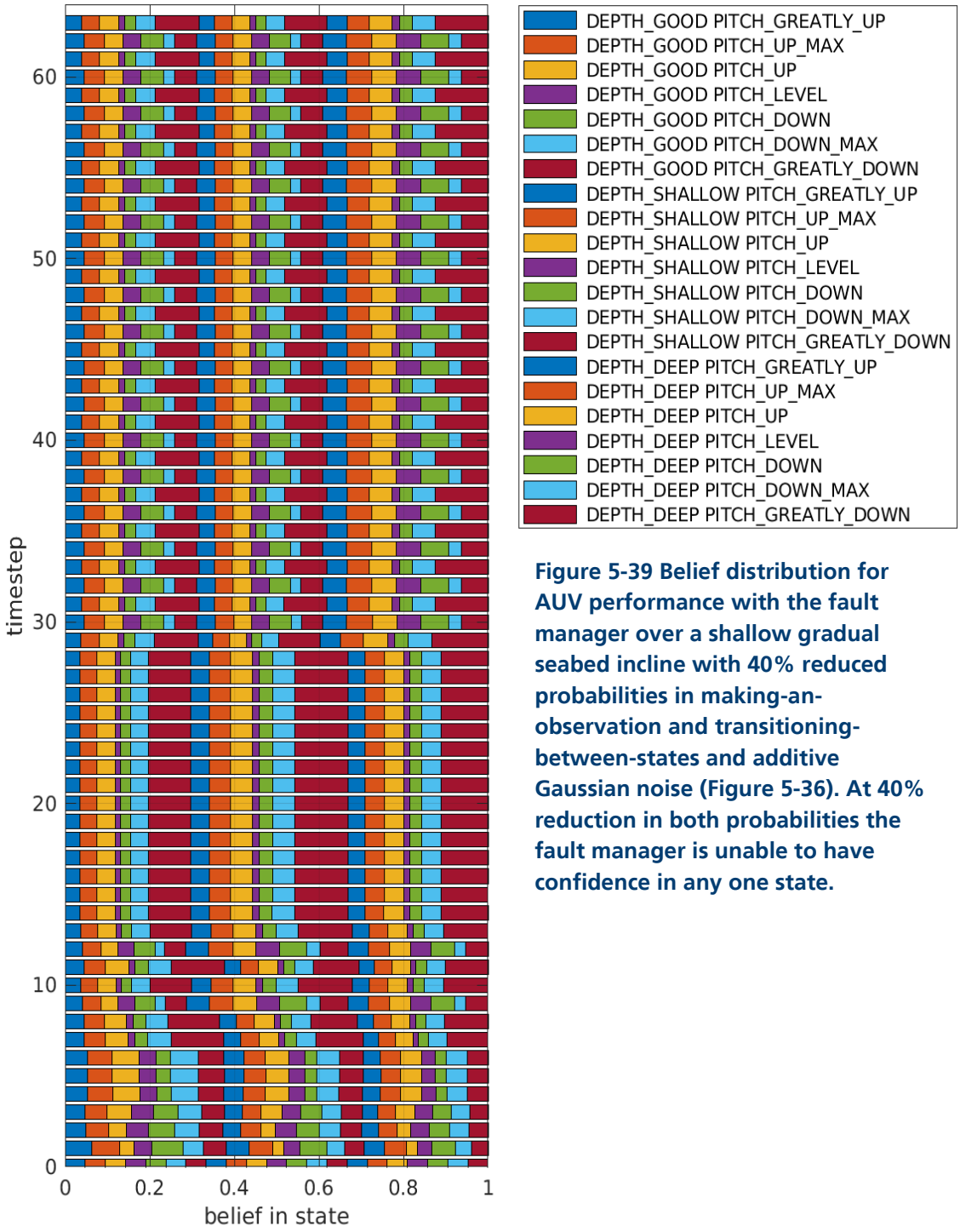

Figure 5-39 Belief distribution for AUV performance with the fault manager over a shallow gradual seabed incline with 40% reduced probabilities in making-an-observation and transitioning-between-states and additive Gaussian noise (Figure 5-36). At 40% reduction in both probabilities the fault manager is unable to have confidence in any one state.

It was demonstrated that the combined reduction of probabilities for *transitioning-between-states*, $T(s'|\ s,a)$ and *making-an-observation*, $O(o|\ s,a)$, on the POMDP fault manager led to a reduction in the confidence of the states at each timestep. In the 40%

case, the fault manager is unable to properly control the vehicle. The higher the probability of these functions, the less confident the fault manager is of the state and actions it needs to perform to ensure the vehicle's performance. The additive noise increased the lack of confidence

These tests concluded the depth sub-system only simulations, the next series of tests were for the power management sub-system.

5.2 Power management sub-system simulations

The second set of tests were with the power-management sub-system only. These tests were conducted for three energy levels to show the fault manager's response to sufficient energy, insufficient energy (requires a power usage reduction), or critical capacity (requires the vehicle to abort its mission). Both the initial energy capacity and total estimated mission time are set in the initialization file (see Appendix C – Table). The energy consumption is a random distribution read from a log file. For more information see section 4.4.2.

The vehicle's energy usage is captured in a series of plots for each test. The first plot, (a), shows the energy capacity (Joules) over the mission. The second plot, (b), shows the estimated mission time left. Plot (c) shows the effect of the power management mode controlled by the fault manager. A value of 0 means energy usage is normal, 1 means the vehicle has entered a power-saving-mode where power usage is reduced by a set incremental rate defined in the initialization file (see Appendix C – Table), and 2 means the mission has been aborted. Plot (d) shows the energy consumed at each timestep. This was a variable input since energy consumption by sensors, actuators and processors is rarely constant.

The first of these tests had sufficient energy to complete the mission.

5.2.1 Sufficient capacity with simulated energy consumption

The sufficient-energy capacity test began with a capacity of 8000 Joules and finished with about 2500 Joules of energy. Since the energy was only measured to be *low* (as opposed to *very low* or *critical*) in the final quarter of the mission, the *power-saving* mode was not engaged since the fault manager determines there is likely sufficient energy to finish the simulation. The *low* energy rate was set to 40% (set in initialization file, see Appendix C – Table). This results in the *low* energy value to be under 3200 Joules. A *very low* energy capacity is defined as under 1600 Joules and a *critical* capacity is 800 Joules.

In Figure 5-40 (a) the AUV energy capacity decreases over the course of the mission. It does reach below 3200 Joules around timestep 55, but never drops below the *very-low* or *critical* thresholds. The estimated time to completion is in (b). The total estimated mission time (set in initialization file) is inaccurate causing the mission to take longer than estimated. The power mode is shown in Figure 5-40 (c) – it never switches from normal usage. Although the energy capacity does reach a *low* energy threshold the usage remains normal since the vehicle is in the last quarter of its mission and the fault manager determines there is enough energy to finish the mission. Figure 5-40 (d) shows the energy usage measurement at each timestep.

The belief state (Figure 5-41) is fairly steady with the main changes being the mission time.

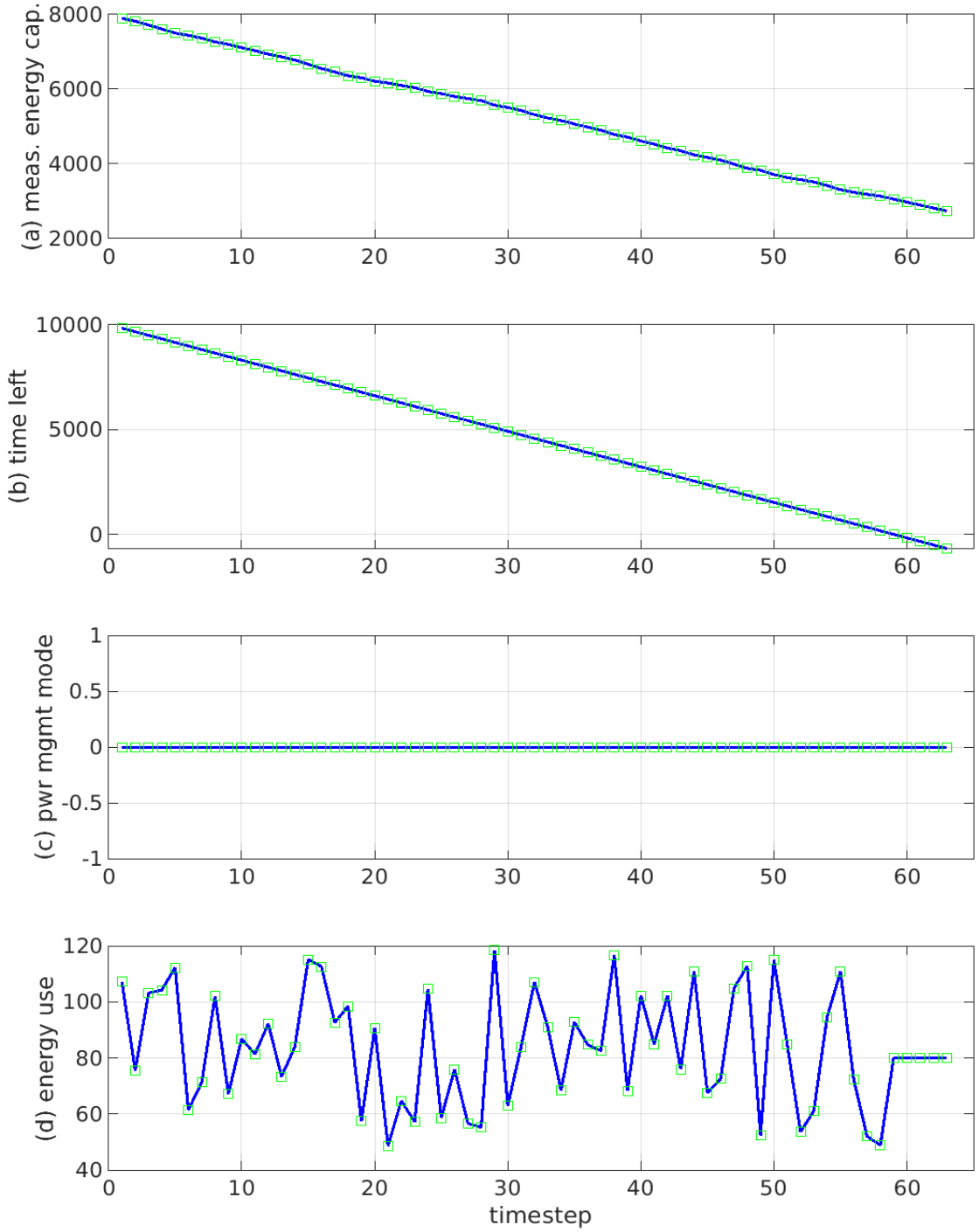

Figure 5-40 Sufficient energy for given simulated energy consumption in (d) with the estimated mission time of (b). The fault manager drives the power sub-system with thresholds: low=3.2 kJ, very low = 1.6 kJ, and critical = 0.8 kJ. (a) Energy (J) decreases as expected, it does not go below the *low energy* state. (b) The estimated time to mission competition is longer than expected. (c) The fault manager driven power mode shows all zeros which means energy is consumed at the normal rate which is expected given sufficient energy.

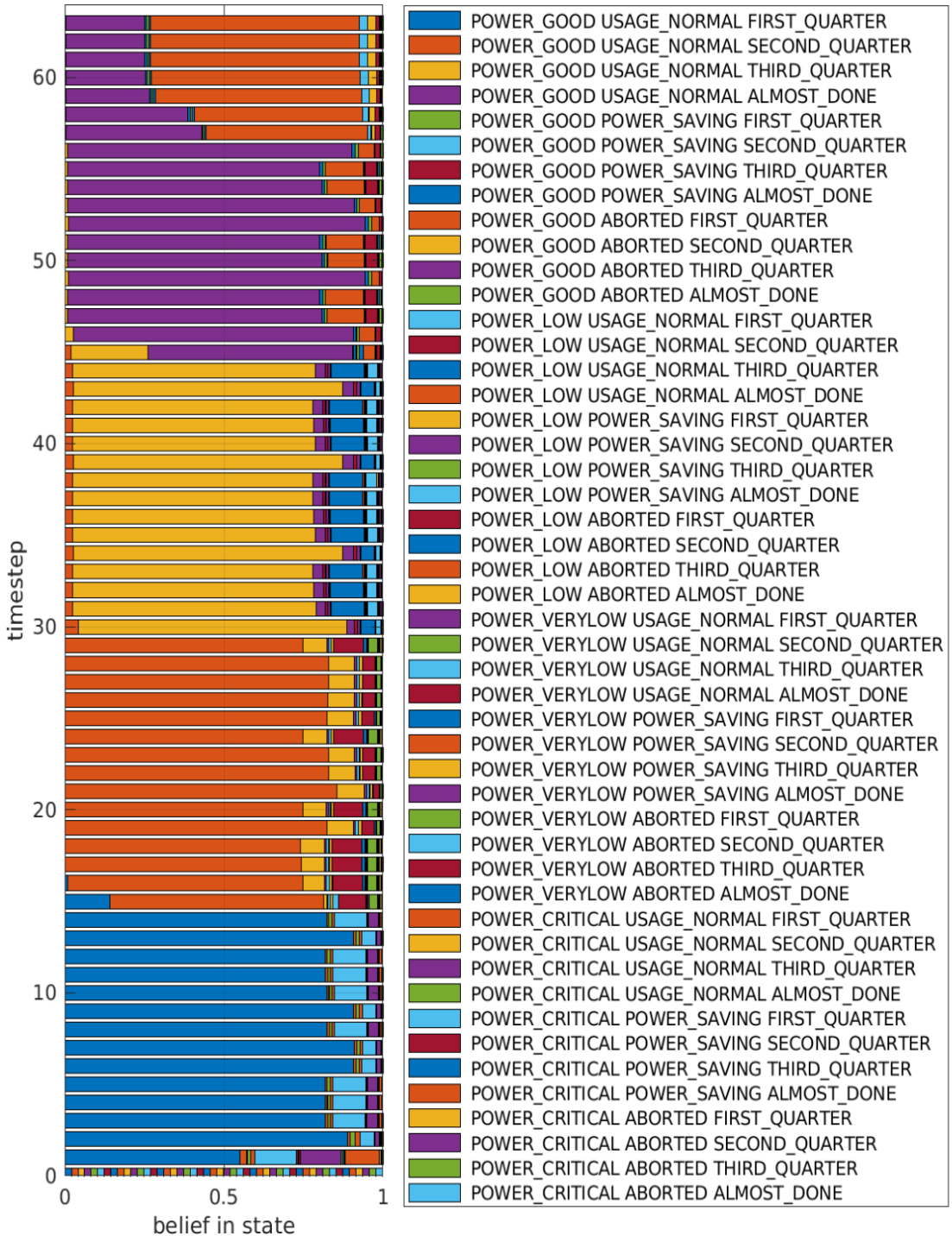

Figure 5-41 Belief distribution for AUV performance with the fault manager (Figure 5-40): sufficient energy given simulated energy consumption. Belief state progresses with confidence as expected.

This model shows the vehicle operating with sufficient energy and will be used as a benchmark for the next two tests that operate at a reduced energy state.

The initial energy capacity of the sufficient model was reduced by 30% for an insufficient energy model.

5.2.2 Insufficient capacity with simulated energy consumption

The insufficient energy capacity test was initialized with a capacity of 5750 Joules and finished with 630 Joules of energy. The *low* energy rate was set to 40% (set in initialization file see Appendix C – Table). This means the low energy value is under 2300 Joules. A *very low* energy capacity is under 1150 Joules and a *critical* capacity is 575 Joules. The power-saving-mode reduces the energy consumed by 25% (set in the initialization file).

In Figure 5-42(a) the AUV energy capacity decreases over the course of the mission. It goes below 1150 Joules (*very low*) at around timestep 55 which activates the power-saving-mode seen in Figure 5-42 (c). The vehicle manages to finish the simulation before crossing the *critical* energy level threshold. The estimated time to completion is shown in Figure 5-42 (b). The total estimated mission time (set in initialization file) is inaccurate causing the mission to take longer than estimated. Figure 5-42 (d) shows the energy usage measurement for each timestep. Once the power-saving-mode is engaged (timestep 55), there is a noticeable reduction in the energy consumed at each timestep (Figure 5-42 (d)).

The vehicle is fairly confident (Figure 5-43) of its state in the first 85% of the mission, with the main changes coming from the mission time states. However, towards the end of the simulation when the energy capacity begins to reach the *low* and then *very low* levels the vehicle becomes less confident of its state.

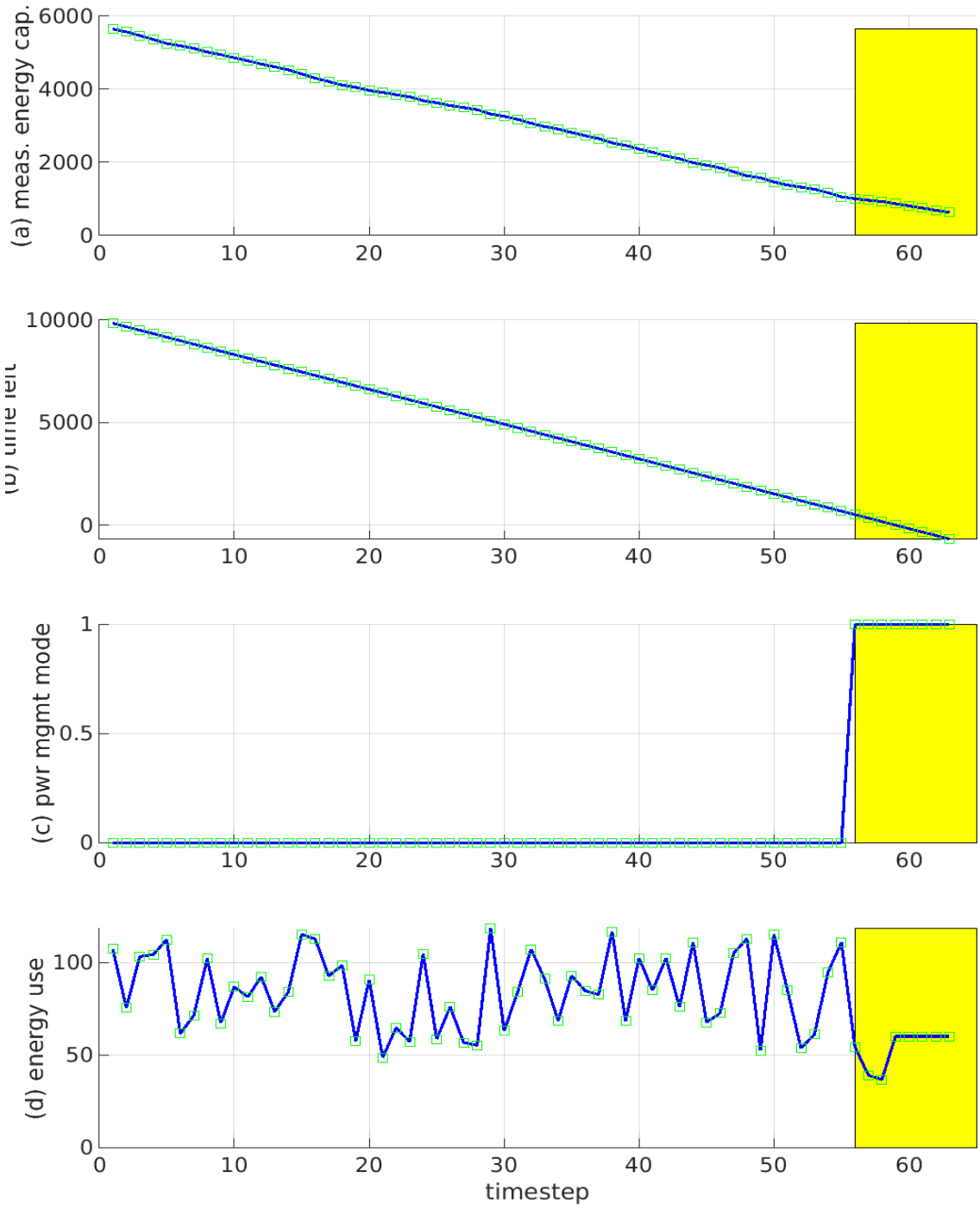

Figure 5-42 Insufficient energy for given simulated energy consumption in (d) with the estimated mission time of (b). The fault manager drives the power sub-system with thresholds: low= 5.75 kJ, very low = 2.3 kJ, and critical = 1.15 kJ). (a) The total energy (Joules) decreases as expected over the mission. At timestep = 55, the vehicle enters the *power-saving* mode. (c) The fault manager driven power mode (energy consumption rate: 0 = normal rate, 1 = lower power rate) switches to *power-saving* mode at timestep = 55 given insufficient energy. The yellow band is when the *power-saving* mode is engaged.

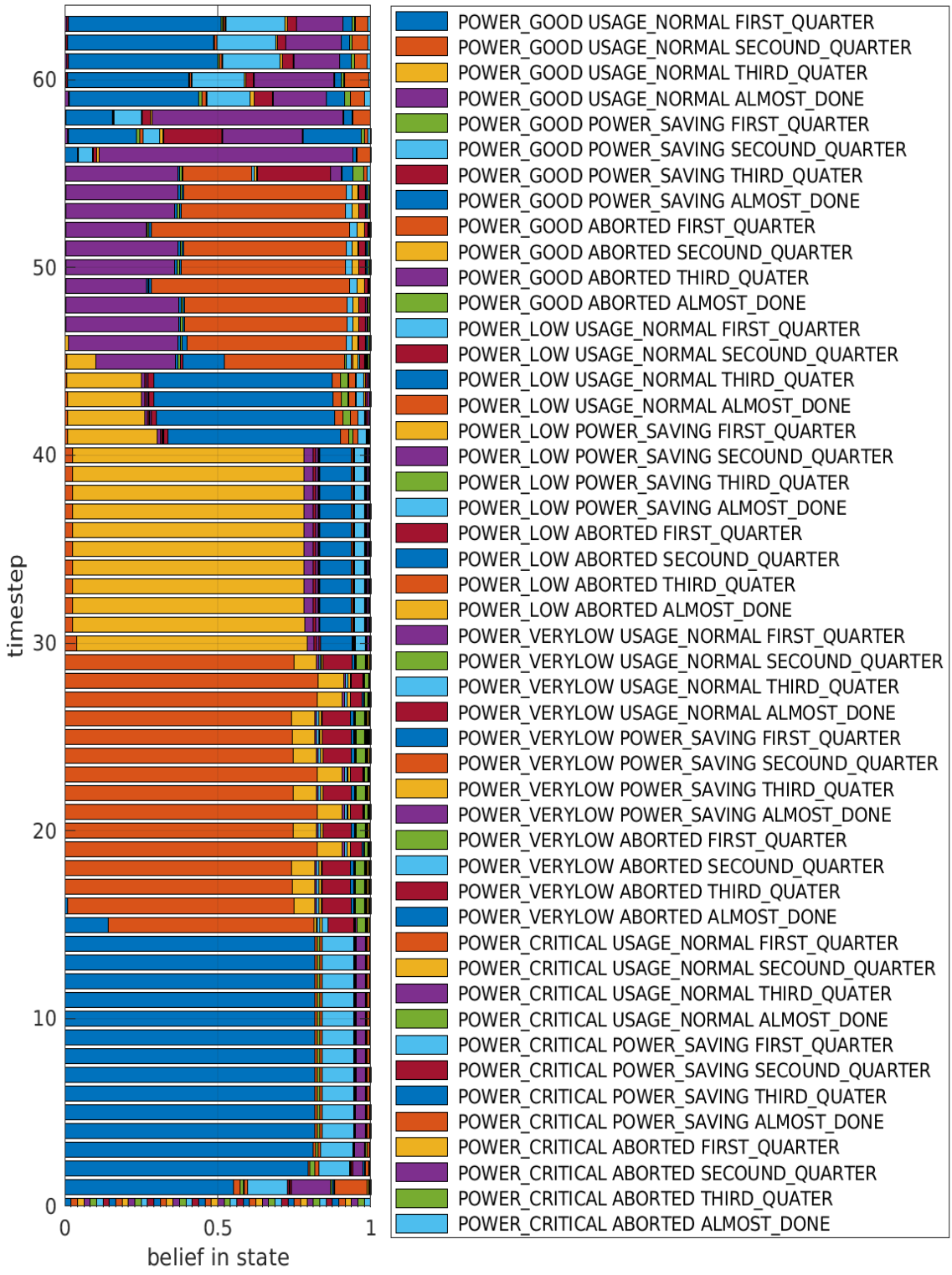

Figure 5-43 Belief distribution for AUV performance with the fault manager: insufficient (Figure 5-42) energy given simulated energy consumption. Belief state progresses with confidence as expected until the last 15% of the mission where the state becomes more uncertain due to the lower energy modes.

This simulation demonstrated the vehicle successfully responding to a *very low* energy state by reducing its energy consumption. This is further demonstrated in the next simulation where the vehicle reaches a *critical* energy level and must abort the mission.

5.2.3 Critical capacity with simulated energy consumption

The critical energy capacity test began with 5000 Joules and finished with about 420 joules of energy. The low energy rate was set to 40% (set in initialization file, see Appendix C – Table). This results in the *low* energy value to be under 2000 Joules, the *very low* energy capacity to be under 1000 Joules and a *critical* capacity of 500 joules. The power-saving-mode reduces the energy consumed by 25% which is set in the initialization file.

As shown in Figure 5-44(a) the AUV energy capacity decreases over the course of the mission as expected. It reaches below 1000 Joules around timestep 48 which activates the *power-saving* mode seen in Figure 5-44 (c). The energy capacity reaches *critical* around timestep 55 resulting in the fault manager calling for a mission abort and the simulation ending. The estimated time to completion is shown Figure 5-44 (b). The total estimated mission time (set in initialization file) is inaccurate causing the mission to take longer than estimated. Figure 5-44 (d) shows the energy consumed at each timestep. It is highly variable to simulate actual energy usage by different sensors, actuators and processors during the timestep. Around timestep 48 the energy consumption, while still variable, is reduced due to the power-saving-mode being engaged.

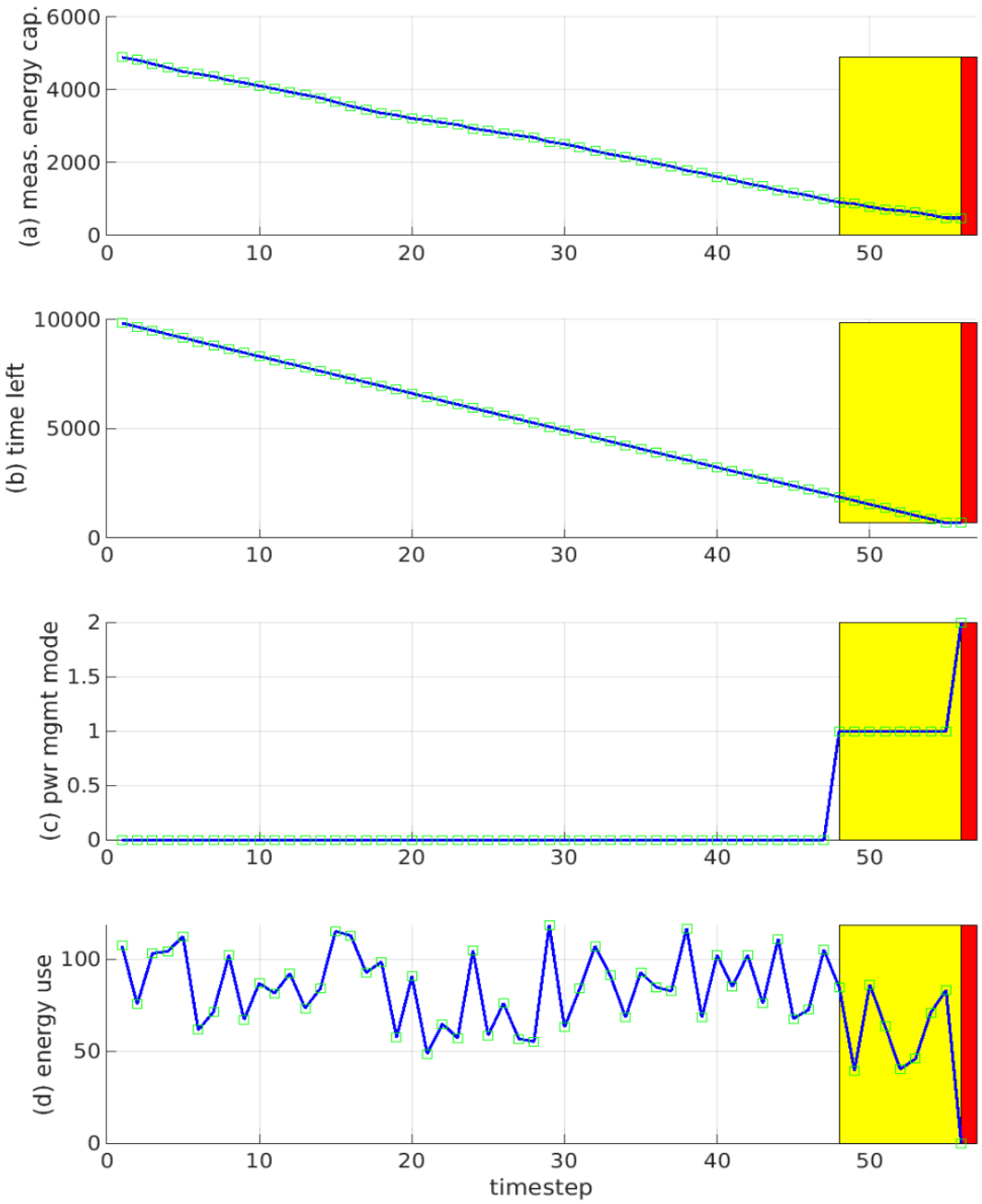

Figure 5-44 Critically low energy for given simulated energy consumption in (d) with the mission time in (b). The fault manager drives the power sub-system with thresholds: *low*= 2.0 kJ, *very low* = 1.0 kJ, and *critical* = 0.5 kJ). (a) The total energy (J) decreases as expected over the mission (simulation). (b) The total estimated mission time is inaccurate from the start. (c) The fault manager driven power mode (energy consumption rate: 0 = normal, 1 = lower power) switches at timestep = 48 to *power-saving* mode and at timestep = 55 critical mode since it crossed that threshold. The yellow band is when the *power-saving* mode is engaged. The red band is when the abort has been triggered due to critical energy levels.

In Figure 5-45 , the vehicle is fairly confident of its state at the beginning with the main changes coming from the mission time quarter. However; towards the end of the simulation (timestep 35) when the energy capacity begins to reach *low*, *very low* and then *critical* capacities levels the vehicle becomes less sure of its state. From timestep 46 onwards there are a few different states for each timestep that contend as the actual state.

This simulation successfully shows how the fault manager attempts to complete the mission by reducing the energy consumed at each timestep but ultimately resorts to aborting to prevent vehicle failure. This becomes more significant in the next series of tests where the power-management and depth sub-systems are modelled together.

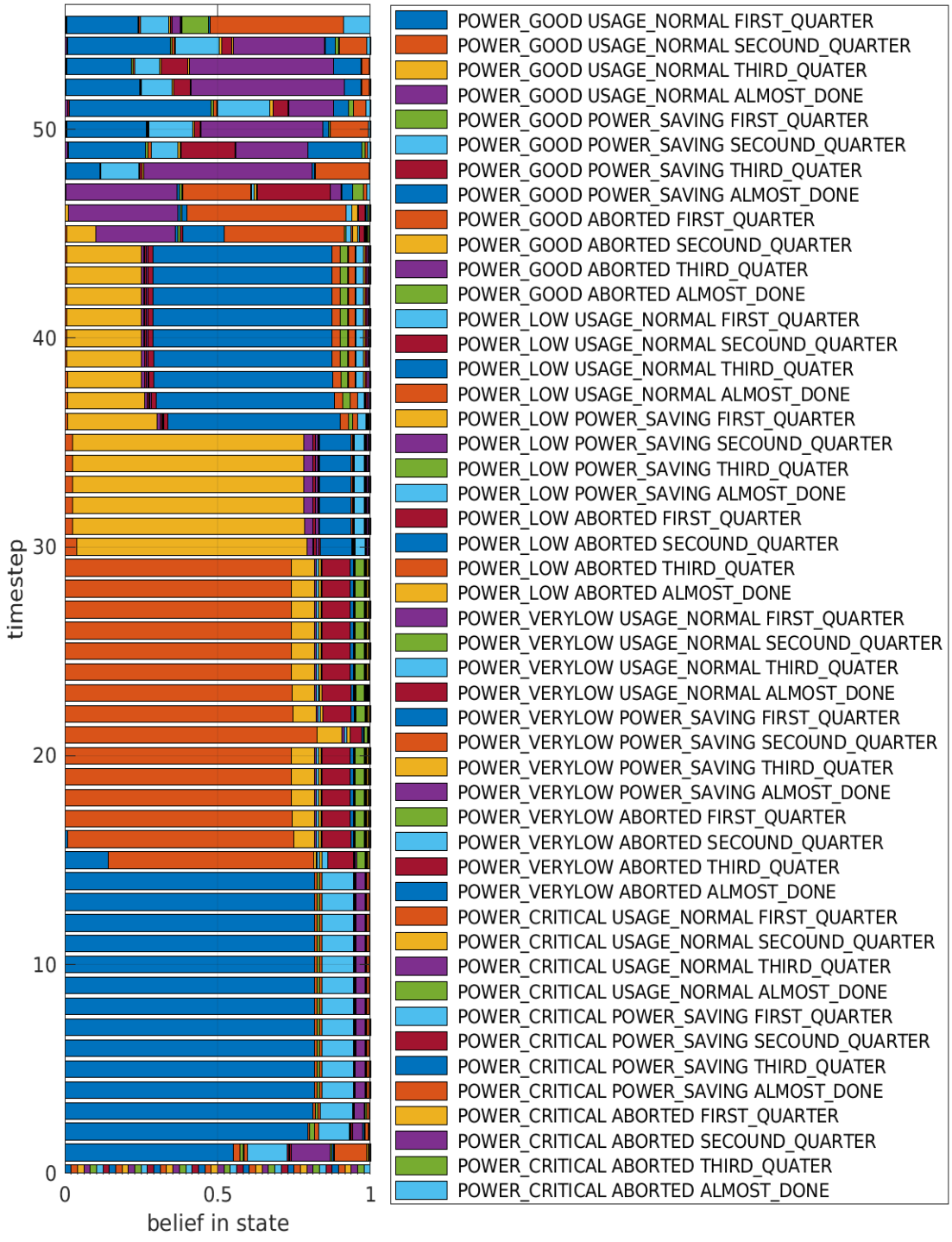

Figure 5-45 Belief distribution for AUV performance with the fault manager: critically low energy given simulated energy consumption (Figure 5-44). Belief state progresses with confidence as expected until timestep = 35 and from that point the state becomes progressively more uncertain due to the critically low energy. At timestep 46 the fault manager becomes even less confident in its state.

5.3 Combined depth and power-management sub-systems simulations

The last series of tests were with the depth and power management sub-systems combined into a single POMDP model. Four separate tests were conducted. The first was of the vehicle operating with the sub-systems independent of one another. The second was of the sub-systems operating interactively with dependent operations. The third model is of the interacting sub-systems where a cascade model caused in the power-management sub-system resulted in failures in the depth sub-system. The final was the same model as the previous simulation, however the seabed became deeper towards the end of the simulation while the AUV was rising to surface. These models have both figures for the power-management and depth sub-system as in the previous sections however; due to the large belief state space (approximately 1000 possible states) the belief state figures are not shown.

The first test was of the independent sub-system models.

5.3.1 Non-interacting sub-systems POMDP model

This model has each sub-system independent of the other. It combines the depth and power models into one POMDP model although no conditional probabilities or joint-rewards (i.e. connections) were added between the two sub-systems. The objective was to show how a combined model can operate like two separate ones. The altitude range was 8-12 meters, DVL of 20 meters, max and min depth ratings of 35m and 5 meters, speed was 2 knots, timesteps 2s and maximum angle 15° with a pitch change of 3°. These values were chosen to match those from section 5.1.1.1.

The depth measurement simulation data used was the shallow water simulated environment model from section 5.1.1.1. The power management energy model (8000 Joules stored) used was similar to the sufficient energy capacity model 5.2.1.

As shown in Figure 5-46 and Figure 5-47 their actions are similar to the previous simulations of independently modelled sub-systems. This is due to the lack of interaction between them. One of the drawbacks of POMDP is the large state,

observation, and action spaces that occur, thus if sub-systems that operate independently can be parsed into separate models it can reduce the size of any one POMDP model and still perform similarly to a combined non-interacting model.

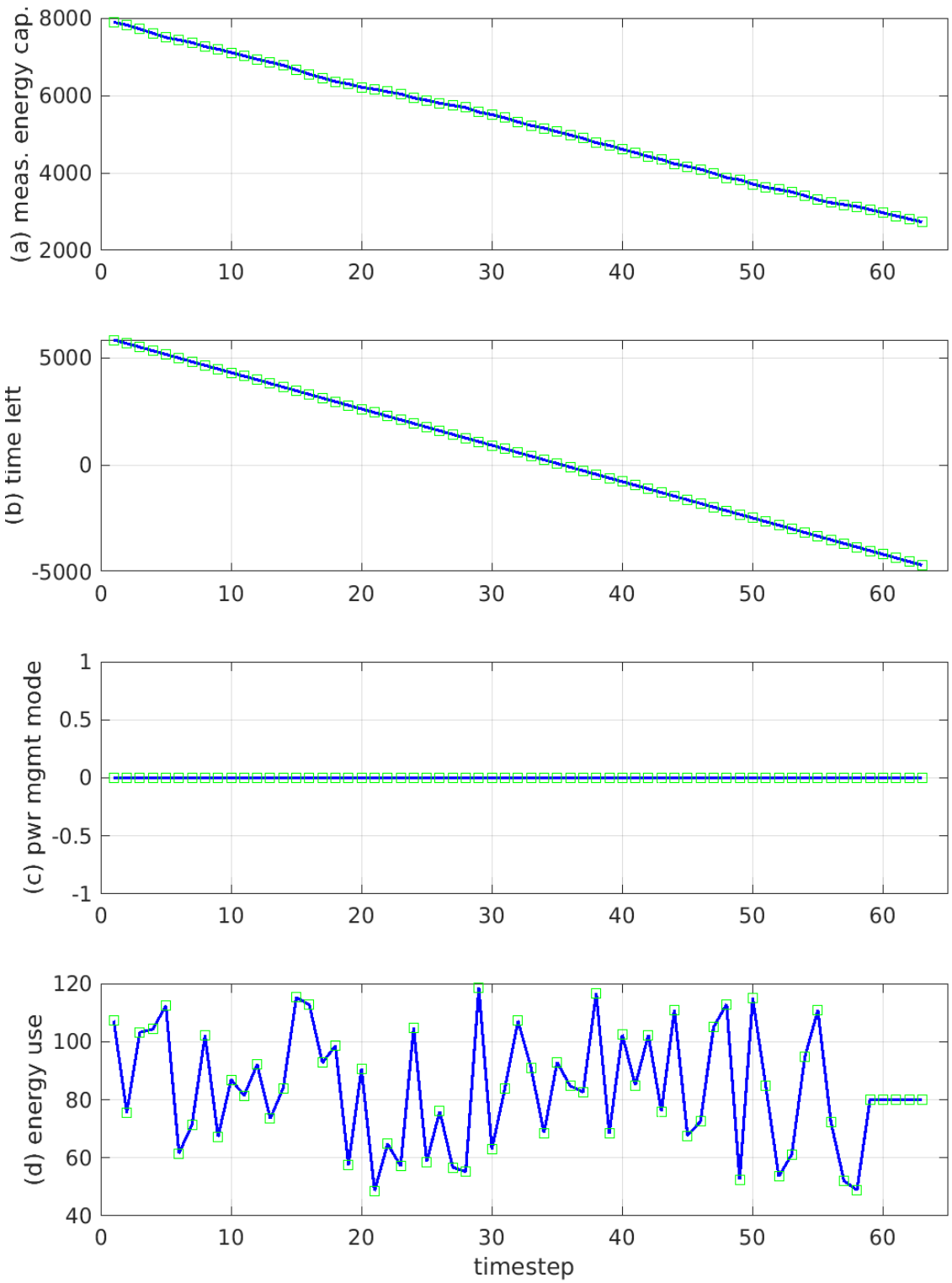

Figure 5-46 Non-interacting power management sub-systems POMDP model for given simulated energy consumption in (d) with the mission time in (b). The fault manager drives the power sub-system with 8 kJ energy to start. Similar responses to the system of section 5.2.1.

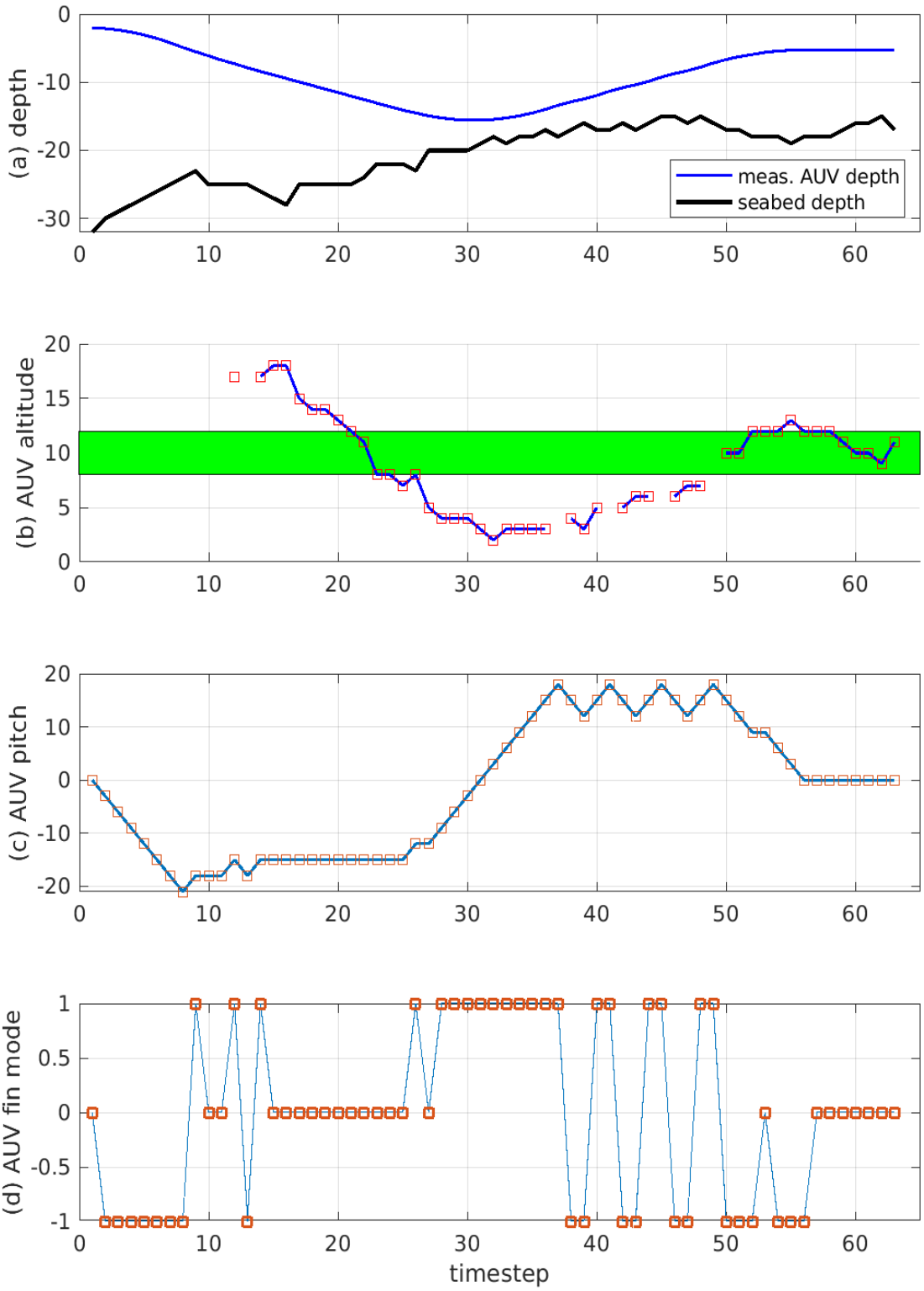

Figure 5-47 Non-interacting depth sub-systems POMDP model with simulated shallow-water environment a with gradual incline seabed. Near identical responses to the system of section 5.1.1.1. The desired altitude range is given by the green band in (b).

This model shows that when the sub-systems are independent there is little difference whether the fault manager uses separate models or combined ones since the sub-systems do not interact. This is useful to know since more sub-systems or larger state/actions/observation spaces can quickly increase the complexity of the POMDP and future implementations could reduce complexity by parsing out independencies of the sub-systems.

This simulation also serves as a baseline for the next simulation that involves dependencies between the sub-systems.

5.3.2 Interdependent sub-systems model

This simulation combined the depth and power management sub-systems so that they were dependent on one another. It is proposed that when the AUV reaches *very-low* energy capacity it begins to move up in the water column. When it reaches *critical* it would attempt to surface. This uses the state and observations of the power-management sub-system to effect change actions in the depth sub-system. The altitude range was 8-12 meters, DVL of 20 meters, max and min depth ratings of 35m and 5 meters, speed was 2 knots, timesteps 2s and maximum angle 15° with a pitch change of 3°. These values were chosen to match those from section 5.1.1.1.

The shallow-water simulated-environment model from section 5.1.1.1 was used. The power-management model used was the one with an initial energy capacity of 5200 Joules and similar the critical capacity model from section 5.2.3 (65% of 5.2.1). This results in a *low* energy value to be under 2040 Joules. A *very low* energy value to be under 1040 Joules and a *critical* capacity of 520 Joules.

In Figure 5-48 and Figure 5-49 the energy capacity reaches *very low* around timestep 49 triggering the power-saving-mode to be engaged in timestep 50 (Figure 5-48(c), yellow band). The AUV begins a gradual decrease in depth (slow ascent) (Figure 5-49(a)) at the same time with the pitch angle beginning to increase (Figure 5-49(c)). However; around timestep 58 when the vehicle reaches critical energy capacity (Figure

5-48 (a)) the mission is aborted in timestep 59 (Figure 5-48 (c), red band) which causes the vehicle to rapidly ascend (Figure 5-49 (a)) reaching the surface around timestep 62. The energy consumed during each timestep is reduced once the power-saving mode is engaged and similarly when the abort is specified (Figure 5-49(d)). The vehicle successfully surfaces at timestep 62 with approximately 260 joules remaining.

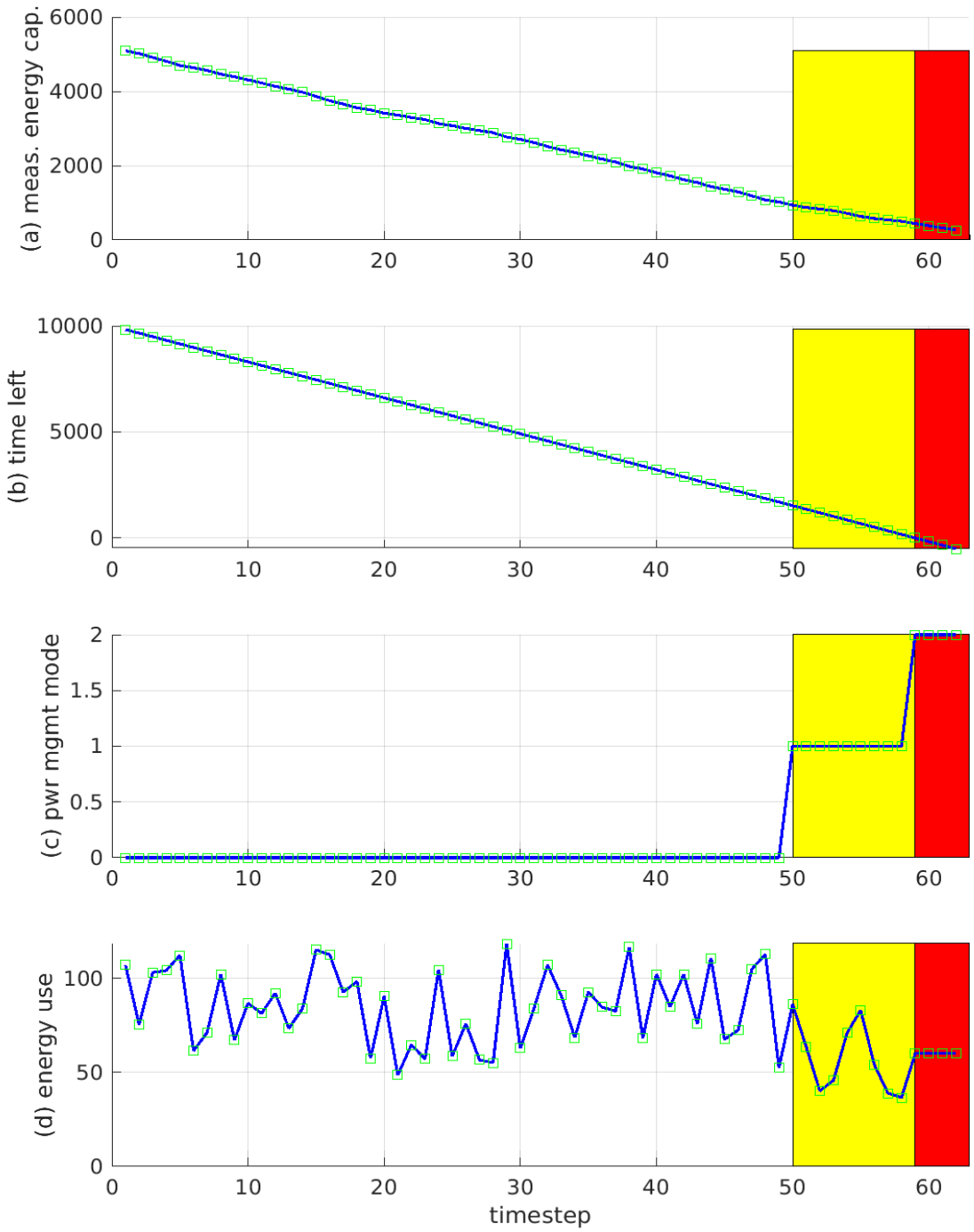

Figure 5-48 Interdependent sub-systems model: AUV performance with fault manager driving the power sub-system with initial energy = 5.2 kJ and defined thresholds: *low* power = 2.08 kJ, *very low* power = 1.04 kJ, *critically low* power = 0.52 kJ) and given energy consumption in (d). Responses like system in section 5.2.3. © The fault manager driven power rate (0=normal, 1= low power, 2=vehicle aborting the mission) with transition at timestep 50 to *very low* power and timestep 59 to *critically low* power and aborting mission. The yellow band is when the *power-saving* mode is engaged. The red band is when the abort has been triggered due to critical energy levels.

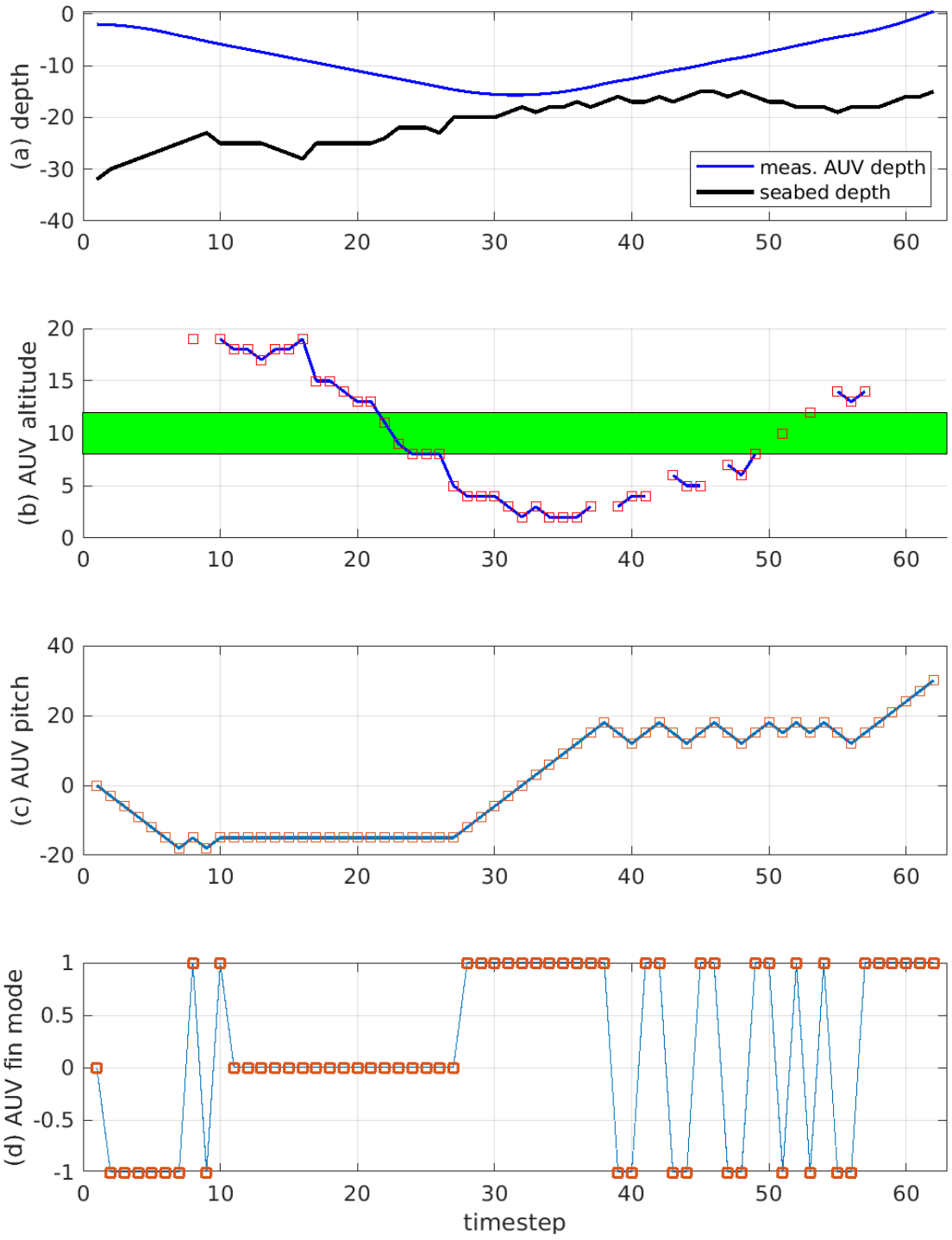

Figure 5-49 Interdependent sub-systems model: AUV performance with fault manager driving the fin mode for pitch changes. (a) When *power-saving* mode is triggered at timestep 50 (Figure 5-48), the AUV starts a gradual rise. At timestep 59, the energy is *critically low* which triggers an abort (fast rise), (c) This is achieved by pitching the vehicle

up. (d) The fin mode correctly enables all this. The desired altitude range is given by the green band in (b).

This simulation is useful as it demonstrates the strength of a fault manager implementing a POMDP model. The POMDP is able to model the dependencies between the sub-systems and allow for more complicated reasoning and fault solving. This is further explored in the last simulation in which a cascade failure is modelled.

5.3.3 Interdependent with cascade failure sub-systems model

This simulation, like the previous one (see section 5.3.2), combines the depth and power management sub-systems so that they were dependent on one another. It is proposed that when the AUV reaches *very low* energy capacity it would begin to move up in the water column. When it reaches *critical* the vehicle would abort the mission and surface. This uses the state and observations of the energy capacity to effect change actions in the depth. The altitude range was 8-12 meters, DVL of 20 meters, max and min depth ratings of 35m and 5 meters, speed was 2 knots, timesteps 2s and maximum angle 15° with a pitch change of 3°. These values were chosen to match those from section 5.1.1.1.

However, a cascade failure was introduced such that once the energy capacity had reached *very low* or *critical* the altitude and depth measurements would be lost due to sensor failure resulting in the vehicle being unable to measure its depth or altitude. The depth simulation data used was the shallow-water simulated-environment model from section 5.1.1.1. The power management model used was one with an initial energy capacity of 5200 Joules and similar to the critical capacity model from section 5.2.3 (57% of 5.2.1). This results in the *low* energy value to be under 2080 Joules. A *very low* energy capacity would be under 1040 Joules and *critical* capacity is 520 Joules.

In Figure 5-50 and Figure 5-51, the energy level reaches very low around timestep 50 triggering the *power-saving* mode to be engaged (Figure 5-50(c)). The AUV begins a gradual ascent (Figure 5-51(a)) at the same time with the pitch angle increases

(Figure 5-51(c)). Due to the *very lo*w energy capacity, a cascade failure occurs where the sensors are not powered thus the vehicle does not know its depth and altitude (Figure 5-51(a, b)). In Figure 5-51(a), the actual vehicle depth is the ground truth vehicle depth while the measured AUV depth is what the vehicle is able to sense. This actual depth was to demonstrate the vehicle's trajectory after the depth measurements have been lost. At timestep 60, when the vehicle reaches *critical* (Figure 5-50(a)), the mission is aborted (Figure 5-50(c)) with the vehicle rapidly ascends(Figure 5-51(a)) to the surface around timestep 62. The energy consumed at each timestep is notably reduced once the power-saving-mode is engaged and similarly when the abort is specified as seen in Figure 5-51(d). The vehicle successfully surfaces at timestep 62 with approximately 260 Joules remaining.

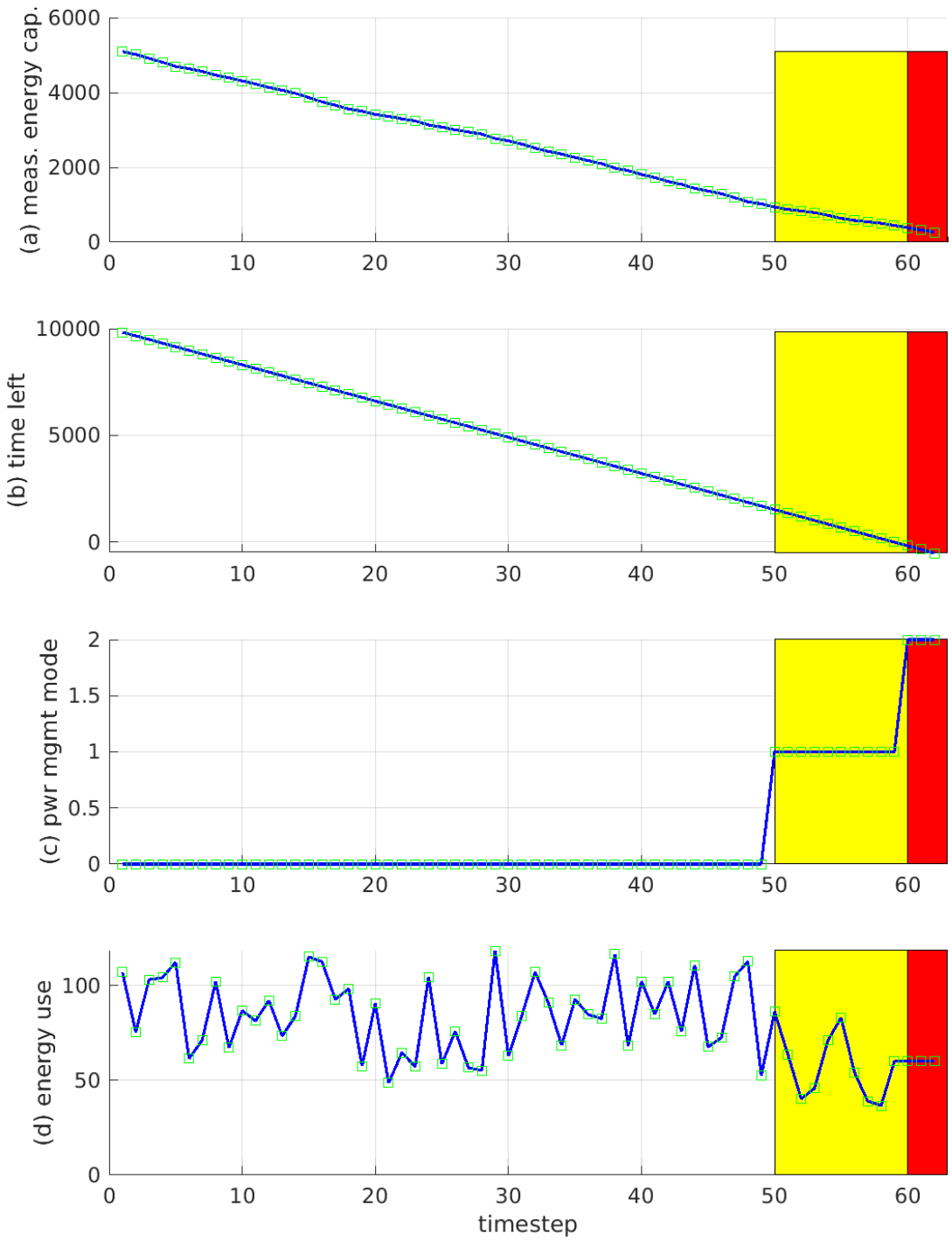

Figure 5-50 Interdependent with cascade failure power-management sub-system POMDP model: AUV performance with fault manager driving the power sub-system with initial energy = 5.2 kJ and defined thresholds: low power = 2.08 kJ, very low power = 1.04 kJ, critically low power = 0.52 kJ). (a) At timestep 50, the *power-saving* mode is triggered and the AUV rises gradually. At timestep 60 the energy drops to critical and the mission is aborted and the AUV rises rapidly. The yellow band is when the *power-saving* mode is engaged. The red band is when the abort has been triggered due to critical energy levels.

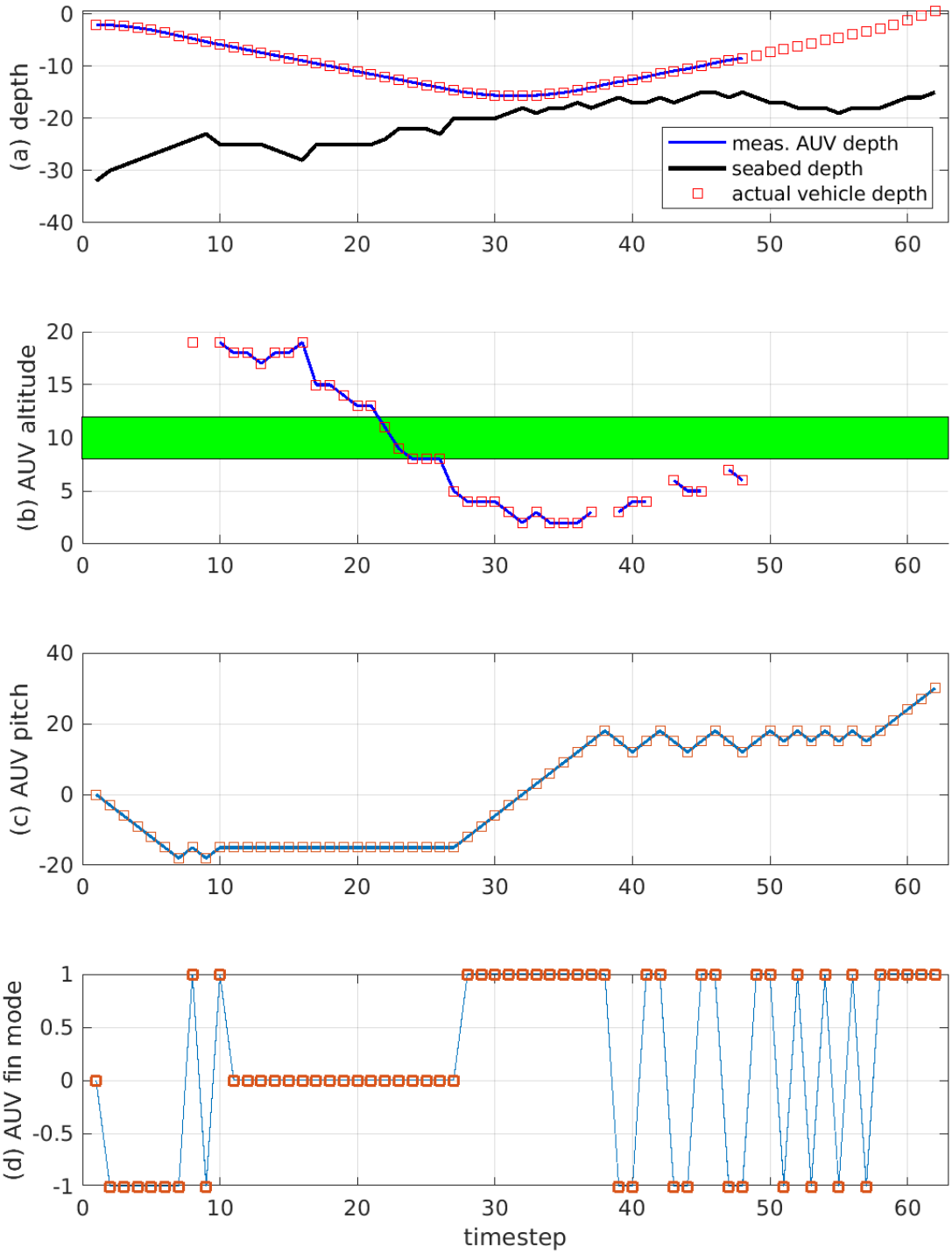

Figure 5-51 Interdependent with cascade failure depth sub-system POMDP model: AUV performance with fault manager driving the fin mode for pitch changes. At timestep 50 the energy capacity is less than 20% resulting in the vehicle DVL is no longer powered and loss of depth and altitude measurements. The actual depth (a) is given to demonstrate the vehicle's path. At timestep 50 the *power-saving* mode is triggered. The desired altitude range is given by the green band in (b).

5.3.4 Interdependent with cascade failure sub-systems model with declining seabed

This simulation, like the previous one (see section 5.3.3), combines the depth and power management sub-systems with a cascade fault. This simulation however, used a seabed that became deeper towards the end of the simulation when the vehicle is undergoing failure. The altitude range was 8-12 meters, DVL of 20 meters, max and min depth ratings of 35m and 5 meters, speed was 2 knots, timesteps 2s and maximum angle 15° with a pitch change of 3°. These values were chosen to match those from section 5.1.1.1 and 5.3.3. The power management model used was one with an initial energy capacity of 5200 Joules and similar to the critical capacity model from section 5.2.3 (57% of 5.2.1). This results in the *low* energy value to be under 2080 Joules. A *very low* energy capacity would be under 1040 Joules and *critical* capacity is 520 Joules the same as 5.3.3.

In Figure 5-52 and Figure 5-53, the energy level reaches very low around timestep 43 triggering the *power-saving* mode to be engaged (Figure 5-52 (c)). The AUV begins a gradual ascent (Figure 5-53 (a)) at the same time with the pitch angle increases (Figure 5-53 (c)). Due to the *very lo*w energy capacity, a cascade failure occurs where the sensors are not powered thus the vehicle does not know its depth and altitude (Figure 5-53 (a, b)). In Figure 5-52 (a), the actual vehicle depth is the ground truth vehicle depth while the measured AUV depth is what the vehicle is able to sense. This actual depth was to demonstrate the vehicle's trajectory after the depth measurements have been lost. At timestep 60, when the vehicle reaches *critical* (Figure 5-52 (a)), the mission is aborted (Figure 5-52(c)) with the vehicle rapidly ascends Figure 5-53 (a)) to the surface around timestep 62. The energy consumed at each timestep is notably reduced once the power-saving-mode is engaged and similarly when the abort is specified as seen in Figure 5-53 (d). The vehicle successfully surfaces at timestep 62 with approximately 260 Joules remaining.

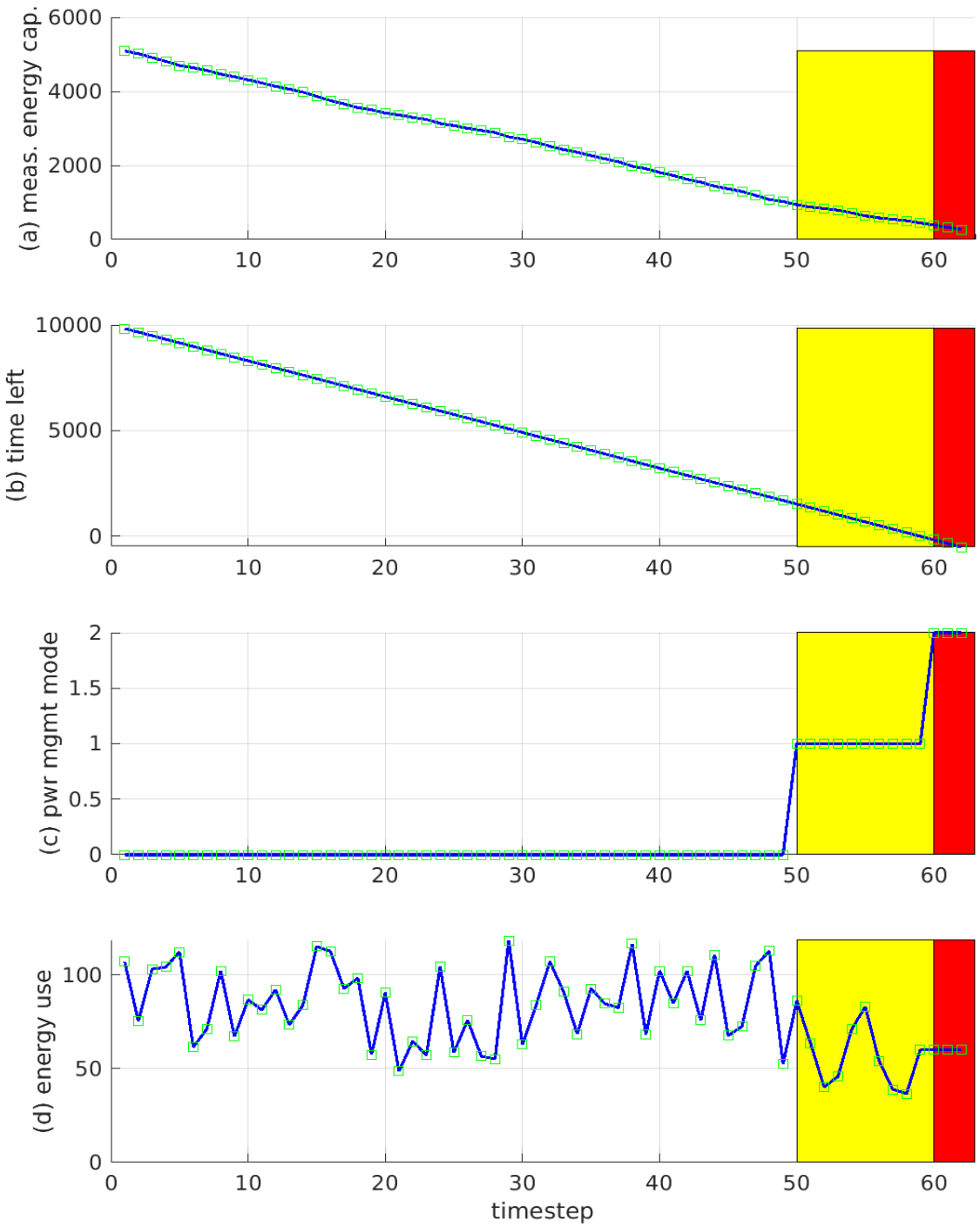

Figure 5-52 Interdependent with cascade failure power-management sub-system POMDP model: AUV performance with fault manager driving the power sub-system with initial energy = 5.2 kJ and defined thresholds: low power = 2.08 kJ, very low power = 1.04 kJ, critically low power = 0.52 kJ). (a) At timestep 50, the *power-saving* mode is triggered and the AUV rises gradually. At timestep 60 the energy drops to critical and the mission is aborted and the AUV rises rapidly. The yellow band is when the *power-saving* mode is engaged. The red band is when the abort has been triggered due to critical energy levels.

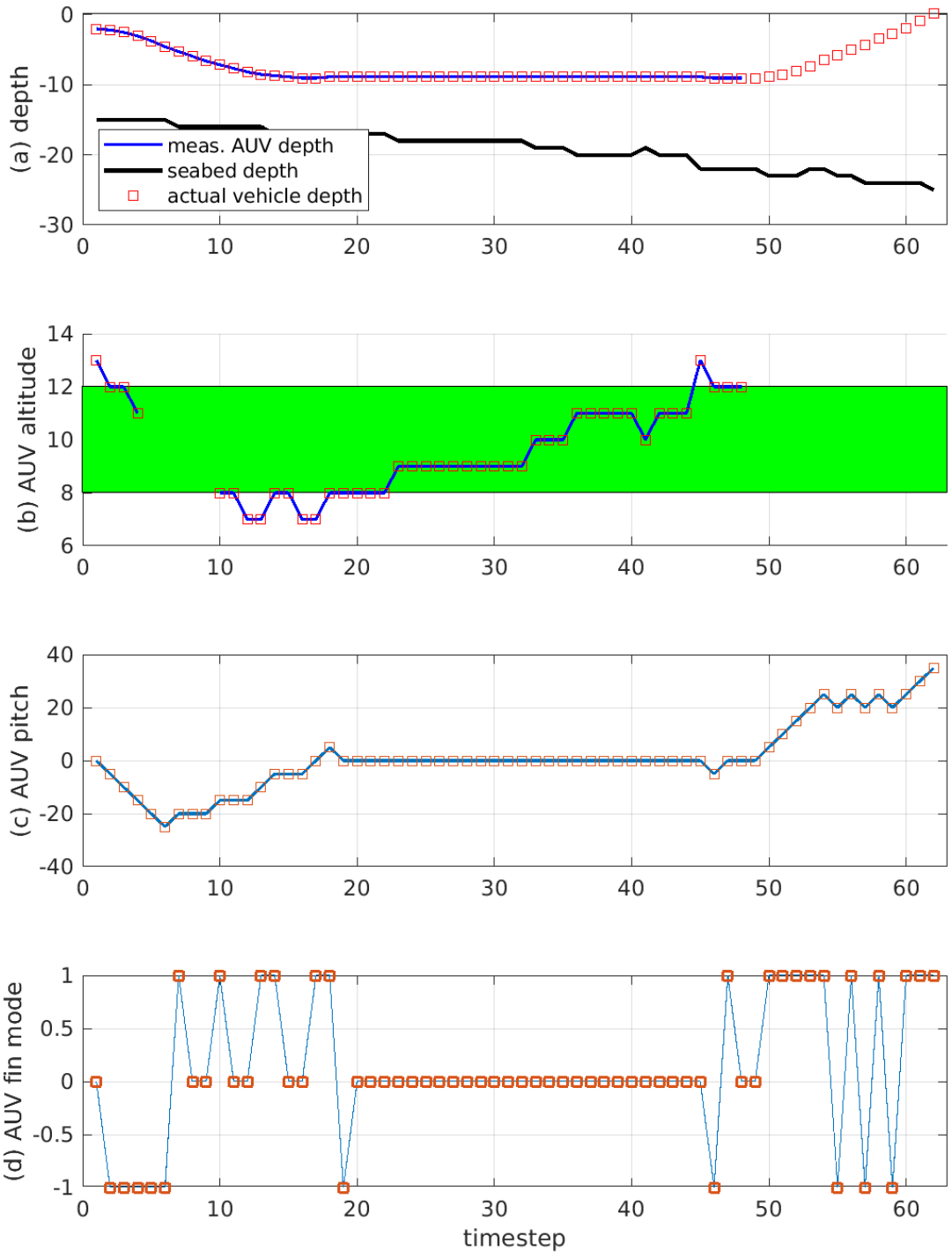

Figure 5-53 Interdependent with cascade failure depth sub-system POMDP model: AUV performance with fault manager driving the fin mode for pitch changes. At timestep 50 the energy capacity is less than 20% resulting in the vehicle DVL is no longer powered and loss of depth and altitude measurements. The actual depth (a) is given to demonstrate the vehicle's path. At timestep 50 the power-saving mode is triggered. The desired altitude range is given by the green band in (b).

In summary, each sub-system was tested separately. The depth sub-system was tested for three environment models (shallow water, deep water, and variable seabed) with each environment having a simulated environment model and a real-world model. The depth sub-system was also tested to show how changing the probabilities of making-an-observation and transitioning-between-states function changes the response of the vehicle fault manager and the effect of noise in the measurements.

The power-management sub-system was tested with three initial energy capacities representing a mission with sufficient energy, insufficient energy requiring a reduction in energy consumption, and an insufficient energy requiring the vehicle to abort the mission.

The sub-systems were combined into a single model and were additionally tested for three scenarios. The first was the sub-systems combined but acting independently. The second scenario was with both sub-systems combined and interacting so that when a *very low* or *critical* energy capacity is measured the vehicle attempts to surface. Finally, the third scenario is a combined model that was interdependent and had the additional complexity of a cascade failure causing sensor failures – loss of depth and altitude measurements.

The fault manager successfully navigated the vehicle through all the simulations with the exception of the tests sets involving a reduction to the probability of functions by 40%. The fault manager was able to complete three different seabed topographies, a noise induced simulation with reductions of 10%, and 20% to one or both functions and all three energy capacity conditions. The fault manager was able to demonstrate that independent models that do not have interactions between states, observations, and actions, correctly, respond the same as two separate models. A fault manager model with: (1) non-interacting sub-systems and (2) interdependent sub-systems with a cascade failure, both successfully surfaced the vehicle.

CONCLUSIONS

This thesis presents the development of a novel proof-of-concept fault management system using a partially observable Markov decision process, deliberative, and model-based, implementation for autonomous underwater vehicles. The POMDP fault management system was modelled in discrete time with each timestep simulating new sensor measurements.

The POMDP was modelled for two sub-systems and were tested with an AUV simulator to demonstrate their efficacy as fault-manager models. The three groups of models tested were: (1) single depth sub-system, (2) a power management sub-system and (3) a combined model of the depth and power-management sub-systems.

The depth sub-system tests show the vehicle operating in three types of environments: shallow water, deep water, and rapidly variable seabed. These were tested with simulated and real-world sample data. These models show the POMDP model was able to successfully provide consistent actions to prevent the vehicle from grounding itself and maintain a desired altitude to perform a DVL-lock/bottom-lock.

Additional tests were performed with the shallow simulated environment model and with additive Gaussian noise to its depth measurement and reductions in the conditional probability functions. These models show the impact of the probability functions on the POMDP model. For reductions of 10% and 20% the fault manager could still operate, albeit in a less confident belief-state however, at a reduction of 40% it was unable to maintain a belief-state distribution and correctly choose actions to drive the vehicle.

The power-management sub-systems show the vehicle operating at three energy levels, sufficient energy for the mission, low energy capacity that requires a power-saving-mode to be engaged, and insufficient energy capacity that requires the vehicle to abort the mission. These models show the POMDP model was able to successfully

determine when the power-saving mode of mission abort should be engaged based upon the remaining energy capacity, energy usage, and estimated remaining mission time.

Finally, combinations of the depth and power-management sub-systems were demonstrated. The first was of non-interacting depth and power-management model that showed while both sub-systems were driving actions in the vehicle, they could do so without impact on the other's states, observations and actions. Therefore, the combined independent model operates in the same manner as the separate independent models – not unexpectedly. This allows independent sub-systems to be modelled and independently controlled.

The second was of an interdependent model that had the depth sub-system actions conditional on the power-management sub-systems state and observation. The fault manager was able to surface the vehicle when the energy capacity became too low. This integration is important for AUV fault management. While sub-systems may be concerned with different aspects of vehicle control, the overall vehicle state and health is dependent on all sub-systems. The last of the tests was of a cascade failure in which *very low* and *critical* energy levels caused the depth and altitude sensors to fail. This model showed how the vehicle is able to maintain actions while under cascading failures, and safely drive the vehicle to the surface.

The fault manager was successful in driving a simulated AUV given a POMDP model for each of the sub-systems independently and with interactions. The Q-MDP solver allows the POMDP to be solved in polynomial time while still able to support larger joint-state, joint-action, and joint-observation spaces. The POMDP model provides intelligent decision making that captures the limited observability of the vehicle in its environment, probability of actions resulting in the desired state change, and

integration of different sub-system to provide actions that can successfully manipulate the vehicle.

To conclude, a POMDP allows for intelligent modelling of interactive AUV sub-systems leading to a more informed and autonomous fault management. This was validated against challenging bathymetry and power-management requirements with additional injected faults. In this thesis it has been demonstrated that the novel application of a partially observable Markov decision process model can be successfully applied to fault management for an autonomous underwater vehicle.

6.1 Recommendations

Eight recommendations are made for further development of the fault management system implemented for this thesis.

1. Expand the AUV simulator to include more sub-systems and a more accurate model based on an actual AUV (i.e. include yaw and roll, accelerations, variable speed, etc.) and a fully integrated sensor model.

2. Expand the power-management sub-system to more accurately model an AUV and different forms of energy consumption integrated over different sources each timestep.

3. Add damping or a dead band in the POMDP to reduce oscillations from cases where there are conflicting needs to dive and rise (i.e. seabed is too deep to follow). This does not change the efficacy of the POMDP method.

4. Explore alternative POMDP solvers for more efficient solutions. The Q-MDP solver used was deemed sufficient for the analysis. However, there is a plethora of solvers that could have actions to gather more data. Future work could investigate pros and cons of alternative solvers to address this. Of these, point-based value/policy iteration and Perseus appear to have potential for the work here.

5. Explore control usage combining dependent POMDP models working in tandem with non-interacting POMDP models. Due to the nature of POMDPs, an increase in observations, states, and actions creates exponential growth in joint-state, joint-observation, and joint-actions space size. It was concluded that non-interacting models work similarly when combined. Therefore, non-interacting sub-systems could run separately to reduce POMDP complexities.

6. Machine learning can be applied to learn the reward functions over the course of simulations to improve the probability functions and rewards.

7. Test fault manager on an AUV in-the-loop simulator.

8. Values for the transition, observation and reward functions were set arbitrarily based upon prior knowledge. A more concerted effort could be applied to analyze existing AUV behaviours to refine these probabilities.

REFERENCES

[1] M. Quigley, B. Gerkey, J. F. K. Conle, J. L. T. Foote, R. W. E. Berger and A. Ng, "ROS: an open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, Kobe, Japan, 2009.

[2] M. Palmer, "Lloyd's Register Code for Unmanned Marine Systems," in *Warship 2017: Naval Submarines & UUVs*, Bath, UK, 2017.

[3] M. P. Brito, G. Griffiths and P. Challenor, "Risk Analysis for Autonomous Underwater Vehicle Operations in Extreme Environments," *Risk Analysis,* vol. 30, no. 12, pp. 1771-1788, 2010.

[4] C. Kaminski, T. Crees, J. Ferguson, A. Forrest, J. Williams, D. Hopkin and G. Heard, "12 Days Under Ice – An Historic AUV Deployment in the Canadian High Arctic," in *IEEE/OES Autonomous Underwater Vehicles*, Monterey, CA, USA, 2010.

[5] The Organization for Economic Co-operation and Development (OECD), "Ocean's Supercluster," The Organization for Economic Co-operation and Development (OECD), 2019. [Online]. Available: https://oceansupercluster.ca/. [Accessed 12 June 2019].

[6] M. W. Atherton, Echoes and Images: The encyclopedia of Side-Scan and Scanning Sonar Operations, Vancouver, BC, Canada: OysterInk Publications, 2011.

[7] M. Seto and A. Z. Bashir, "Fault tolerance considerations for long endurance AUVs," in *Reliability and Maintainability Symposium*, Orlando, FL, USA, 2017.

[8] M. Seto, "On-Line Learning with Evolutionary Algorithms towards Adaptation of Underwater Vehicle Missions to Dynamic Ocean Environments," in *IEEE Machine Learning and Applications*, Honolulu, HI, USA, 2011.

[9] M. Seto, Marine Robot Autonomy, Halifax, NS, Canada: Springer, 2013.

[10] Merriam-Webster, Incorporated, Merriam-Webster Dictionary, 2019.

[11] C. L. R. McGhan, R. M. Murray, R. Serra, M. D. Ingham, M. Ono and T. Estlin, "A risk-aware architecture for resilient spacecraft operations," in *IEEE Aerospace Conference*, Big Sky, MT, USA, 2015.

[12] R. Dearden and J. Ernits, "Automated Fault Diagnosis for an Autonomous Underwater Vehicle," *Journal of Oceanic Engineering,* pp. 484-499, 2018.

[13] V. Myers and D. P. Williams, "A POMDP for Multi-view Target Classification with an Autonomous Underwater Vehicle," in *IEEE Oceans*, Seattle, WA, USA, 2010.

[14] T. Crees, C. Kaminski, J. Ferguson, J. M. Laframboise, A. Forrest, J. Williams, E. MacNeil, D. Hopkin and R. Pederson, "UNCLOS Under Ice Survey – An Historic AUV Deployment in the Canadian High Arctic," in *IEEE Oceans*, Seattle, WA, USA, 2010.

[15] D. Thompson, D. Caress, D. Clague, D. Conlin, J. Harvey, E. Martin, J. Paduan, C. Paull, J. Ryan, H. Thomas and Y. Zhang, "MBARI Dorado AUV's Scientific Results," in *IEEE Oceans*, San Diego, CA, USA, 2013.

[16] J. Manley and S. Willcox, "The Wave Glider: A New Concept for Deploying Ocean Instrumentation," *IEEE Instrumentation & Measurement Magazine,* vol. 13, no. 6,

pp. 8 - 13, 2010.

[17] M. Conry, A. Keefe, W. Ober, M. Rufo and D. Shane, "BIOSwimmer: Enabling technology for port security," in *IEEE International Conference on Technologies for Homeland Security (HST)*, Waltham, MA, USA, 2013.

[18] E. E. Allmendinger, "The Basic Design Process," in *Submersible Vehicle Design*, Jersey City, N. J., USA, THE SOCIETY OF NAVAL ARCHITECTS AND MARINE ENGINEERS, 1990, pp. 1-70.

[19] J. David, R. Bauer and M. Seto, "Coupled Hydroplane and Variable Ballast Control System for Autonomous Underwater Vehicle Altitude-Keeping to Variable Seabed," *IEEE Journal of Oceanic Engineering,* vol. 43, no. 4, pp. 873 - 887, 2017.

[20] Y. Yang, S. Jie, L. Tongxu and H. Chong, "A variable buoyancy control system for large UUV," in *Chinese Control Conference (CCC)*, Xi'an, 2013.

[21] X. Lurton, An Introduction to Underwater Acoustics Principles and Applications, Chichester, UK: Sprinder, 2002.

[22] S. Blouin, G. J. Heard and S. Pecknold, "Autonomy and Networking Challenges of Future Underwater Systems," in *Canadian Conference on Electrical and Computer Engineering*, Halifax, 2015.

[23] D. McLeod, J. Jacobson, M. Hardy and C. Embry, "Autonomous inspection using an underwater 3D LiDAR," IEEE, San Diego, CA, 2013.

[24] J. Liu, A. Jakas, A. Al-Obaidi and Y. Liu, "Practical issues and development of underwater 3D laser scanners," in *IEEE Conference on Emerging Technologies and Factory Automation* , Bilbao, 2010.

[25] H. Kaushal and G. Kaddoum, "Underwater Optical Wireless Communication," *IEEE Access ,* vol. 4, no. 1, pp. 1518 - 1547, 2016.

[26] D. Stronger and P. Stone, "Maximum likelihood estimation of sensor and action model functions on a mobile robot," in *IEEE International Conference on Robotics and Automation*, Pasadena, CA, USA, 2008.

[27] G. Reader, J. Potter and J. Hawle, "The evolution of AUV power systems," in *IEEE Oceans*, Biloxi, MI, USA, USA, 2002.

[28] D. Fenucci, A. Caffaz, R. Costanzi, E. Fontanesi, V. Manzari, L. Sani, M. Stifani, D. Tricarico, A. Turetta and A. Caiti, "WAVE: A wave energy recovery module for long endurance gliders and AUVs," in *IEEE Oceans*, Monterey, CA, USA, 2016.

[29] D. Codetta-Raiteri and L. Prtinale, "Dynamic Bayesian Networks for Fault Detection, Identification, and Recovery in Autonomous Spacecraft," *Transactions on Systems, Man, and Cybernetics,* vol. 45, no. 1, pp. 13-24, 2015.

[30] M. L. Seto and H. Li, "On-Board AUV Autonomy through Adaptive Fins Control," in *Automation Science and Engineering*, Toronto, Ontario, Canada, 2010.

[31] K. Svendsen, P. MacLeod, M. Lichodzijewski, M. Seto and J. Covill, "Development of Third Party Behaviours for Autonomous Marine Vehicles," in *Unmanned Systems Canada*, Halifax, NS, Canada, 2015.

[32] M. K. Seto and K. Svendsen, "Advanced AUV Fault Management," in *Autonomous Underwater Vehicles: Design and Practice*, Stevenage, United Kingdom, Institution of Engineering and Technology, 2019, pp. 1-27.

[33] C. Harris and R. Dearden, "Contingency Planning for Long-Duration AUV Missions," in *IEEE/OES Autonomous Underwater Vehicles (AUV)*, Southampton, UK, 2012.

[34] M. Blanke, M. Kinnaert, J. Lunze and M. Staroswiecki, Diagnosis and Fault-Tolerant Control, Berlin, Germany: Springer, 2006.

[35] D. Koller and N. Friedman, Probabilistical Graphical Models, Cambridge, MA, USA: Massachusetts Institute of Technology, 2009.

[36] J. Emits, R. Dearden and M. Pebody, "Automatic Fault Detection and Execution Monitoring for AUV Missions," in *IEEE/OES Autonomous Underwater Vehicles*, Monterey, CA, USA, 2010.

[37] G. A. Bekey, Autonomous Robots, Boston: MIT

[38] L. Portinale, D. Codetta-Raiteri, A. Guiotto and S. DiNolfo, "ARPHA: a software prototype for fault detection, identification and recovery in autonomous spacecrafts ACTA FUTURA," *Acta Futura ,* vol. 5, no. 1, pp. 99-110, 2012.

[39] S. C. Hayden, A. J. Sweet and S. Shulman, "Lessons Learned in the Livingstone 2 on Earth Observing One Flight Experiment," NASA, Arlington, Virginia, USA, 2005.

[40] B. Williams, M. Ingham, S. Chung and P. Elliott, "Model-based programming of intelligent embedded systems and robotic space explorers," *Proceedings of the IEEE,* vol. 91, no. 1, pp. 212-237, 2003.

[41] R. Dearden, "Bayesian Fault Diagnosis: Common Approaches and Challenges," in *Bayesian Fault Diagnosis: Common Approaches and Challenges*, lba, Italy, 2010.

[42] J. d. Kleer and B. C. Williams, "Diagnosing Multiple Faults," *Artificial Intelligence,* pp. 97-130, 1987.

[43] R. Grossman and R. Larson, "Viewing hybrid systems as products of control systems and automata," in *Decision and Control*, Tucson, AZ, 1992.

[44] B. Meddins, Introduction to digital signal processing, Woburn, MA, USA: Elsevier Ltd, 2000.

[45] M. Seto, L. Paull and S. Saeedi, "Introduction to Autonomy for Marine Robots," in *Marine Robot Autonomy*, Halifax, NS, Springer New York Heidelberg Dordrecht London, 2013, pp. 1-46.

[46] H. Park, A. Barrett, E. Baumann, M. Grage and S. Narasimhan, "Modular architecture for hybrid diagnostic reasoners," in *2nd IEEE International Conference on Space Mission Challenges for Information Technology (SMC-IT'06)*, Pasadena, CA, USA, 2006.

[47] E. Demaine, "Undecidable and P-Complete Ep. 20," MIT OpenCourseWare, Cambridge, Massachusetts, USA, 2015.

[48] E. Demaine, "Overview Ep. 1," MIT OpenCourseWare, Cambridge, Massachusetts, USA, 2013.

[49] G. Griffiths and M. Brito, "Predicting risk in missions under sea ice with Autonomous Underwater Vehicles," in *IEEE/OES Autonomous Underwater Vehicles*, Woods Hole, MA, 2008.

[50] D. Codetta-Raiteri and L. Portinale, "Modeling and Analysis of Dependable Systems through Generalized Continuous Time Bayesian Networks," in *015 Annual Reliability and Maintainability Symposium (RAMS)*, Palm Harbor, FL, USA, 2015.

[51] M. Brito and G. Griffiths, "A Bayesian approach for predicting risk of autonomous underwater vehicle loss during their missions," *Reliability Engineering and System Safety,* pp. 55-67, 2016.

[52] M. Brito, G. Griffths and P. Challenor, "Risk Analysis for Autonomous Underwater Vehicle Operations in Extream Evnironments," *Risk Analysis,* vol. 30, no. 12, pp. 1771- 1781, 2010.

[53] G. Griffiths and A. Trembanis, "Eliciting expert judgement for the probability of AUV loss in contrasting operational environments," in *15th International Symposium on Unmanned Untethered Submersible Technology*, Durham, NH, USA, 2007.

[54] M. P. Brito, G. Griffiths and A. Trembranis, "Eliciting Expert Judgment on the Probability of Loss of an AUV Operating in Four Environments," National Oceanography Centre, Southampton, 2008.

[55] M. P. Brito and G. Griffiths, "Updating Autonomous Underwater Vehicle Risk Based on the Effectiveness of Failure Prevention and Correction," *Journal of Atmospheric and Oceanic Technology,* pp. 797-807, 2018.

[56] A. Bobbio, D. Codetta-Raiteri, S. Montani and L. Portinal, "Modeling Cascading Failure Propagation through Dynamic Bayesian Networks," in *IFAC Workshop on Dependable Control of Discrete Systems*, Bari, Italy, 2009.

[57] M. Brito and G. Griffiths, "A Markov Chain State Transition Approach to Establishing Critical Phases for AUV Reliability," *IEEE Journal of Oceanic Engineering,* vol. 36, no. 1, pp. 139-149, 2011.

[58] L. Portinale, D. Codetta-Raiteri and R. Terruggia, "Selecting Failure Countermeasures through Decision Network Analysis," in *Reliability and Maintainability Symposium*, Colorado Springs, CO, USA, 2014.

[59] D. Codetta-Raiteri, L. Portinale and R. Terruggia, "Quantitative evaluation of attack/defense scenarios through Decision Network modelling and analysis," in *International Carnahan Conference on Security Technology*, Rome, Italy, 2014.

[60] L. Portinale and D. Codetta-Raiteri, "Using Dynamic Decision Networks and Extended Fault Trees for Autonomous FDIR," in *IEEE 23rd International Conference on Tools with Artificial Intelligence*, Boca Raton, FL, USA, 2011.

[61] X. Boyen and D. Koller, "Approximate learning of dynamic models," in *Advances in Neural Information Processing Systems*, Denver, Colorado, USA, 1998.

[62] A. R. Cassandra, L. P. Kaelbling and J. A. Kurien, "Acting under Uncertainty: Discrete Bayesian Models for Mobile-Robot Navigation," in *International Conference on*

Intelligent Robots and Systems, Osaka, Japan, 1996.

[63] L. P. Kaelbling, M. L. Littman and A. R. Cassandra, "Planning and Acting in Partially Observable Stochastic Domains," Brown University, Providence, Rhode Island, USA, 1996.

[64] S. Thurn, W. Burgard and D. Fox, Probabilistical Robotics, Cambridge, MA, USA: MIT Press, 2006.

[65] C. H. Papadimitriou and J. N. Tsitsiklis, "The Complexity of Markov Decision Processes," *Mathematics of Operations Research,* vol. 12, no. 4, pp. 441-450, 1987.

[66] M. L. Littman, T. L. Dean and L. P. Kaelbling, "On the complexity of solving Markov decision problems," in *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, Montréal, Qué, Canada, 1995.

[67] H. Kawano, "Real-time Obstacle Avoidance for Underactuated Autonomous Underwater Vehicles in Unknown Vortex Sea Flow by the MDP Approach," in *International Conference on Intelligent Robots and Systems*, Beijing, China, 2006.

[68] M. J. Bays and S. A. Redfield, "Towards tracking a row of discrete targets with AUVs using Markov Decision Processes and probabilistic techniques," in *IEEE/OES Autonomous Underwater Vehicles*, Woods Hole, MA, USA, 2008.

[69] M. Li, H. Bai and N. Krishnamurthi, "A Markov Decision Process for the Interaction between Autonomous Collision Avoidance and Delayed Pilot Commands," in *IFAC Conference on Cyber-Physical and Human Systems*, Miami, Florida, USA, 2018.

[70] X. Yu, X. Zhou and Y. Zhang, "Collision-free trajectory generation for UAVs using Markov decision process," in *International Conference on Unmanned Aircraft Systems (ICUAS)*, Miami, FL, USA, USA, 2017.

[71] M. L. Littman, "The Witness Algorithm: Solving Partially Observable Markov Decision Processes," Brown University, Providence, Rhode Island, USA, 1994.

[72] A. Cassandra, M. Nodine, S. Bondale, S. Ford and D. Wells, "Using POMDP-Based State Estimation to Enhance Agent System Survivability," in *IEEE Symposium on Multi-Agent Security and Survivability*, Philadelphia, PA, USA, 2005.

[73] G. Shani, R. I. Brafman and S. E. Shimony, "Model-Based Online Learning of POMDPs," in *European Conference on Machine Learning*, Porto, Portugal, 2005.

[74] N. Meuleau, K.-E. Kim, L. P. Kaelbling and A. R. Cassandra, "Solving POMDPs by Searching the Space of Finite Policies," Cornell University, Ithaca, New York, USA, 2013.

[75] S. Ross, J. Pineau, S. Paquet and B. Chaib-draa, "Online Planning Algorithms for POMDPs," *Journal of Artificial Intelligence Research,* vol. 32, no. 2, p. 663–704, 2008.

[76] D. S. Bernstein, S. Zilberstein and N. Immerman, "The Complexity of Decentralized Control of Markov Decision Processes," in *Uncertainty in artificial intelligence*, San Francisco, CA, USA , 2000.

[77] A. R. Cassandra, M. Littman and L. Kaelbling, "POMDP Tutorial," 31 01 1999.

[Online]. Available: http://cs.brown.edu/research/ai/pomdp/tutorial/index.html. [Accessed 03 06 2019].

[78] E. J. Sondik, "The Optimal Control of Partially Observable Markov Processes Over the Infinite Horizon: Discounted Costs," *Operations Research,* vol. 26, no. 2, pp. 282-304, 1978.

[79] E. A. Hansen, "Solving POMDPs by Searching in Policy Space," in *Conference on Uncertainty in Artificial Intelligence* , Madison, Wisconsin, USA, 1998.

[80] A. Cassandra, "Exact and Approximate Algorithms for Partially Observable Markov Decision Processes," Brown University, Providence, Rhode Island, USA, 1998.

[81] J. Pineau, G. Gordon and S. Thrun, "Point-based value iteration: An anytime algorithm for POMDPs," in *International Joint Conferences on Artificial Intelligence*, Acapulco, Mexico, 2003.

[82] M. T. J. Spaan and N. Vlassis, "Perseus: Randomized Point-based Value Iteration for POMDPs," *Journal Of Artificial Intelligence Research,* vol. 24, no. 1, pp. 1995-220, 2005.

[83] S. Ji, R. Parr, H. Li, X. Liao and L. Carin, "Point-Based Policy Iteration," in *Conference on Artificial intelligence*, Vancouver, British Columbia, Canada, 2007.

[84] P. H. Lommel, "An Extended Kalman Filter Extension of the Augmented Markov Decision Process," Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, 2005.

[85] M. Hauskre, "Value-function approximations for partially observable Markov decision processes," *Journal of Artificial Intelligence Research,* vol. 13, no. 1, pp. 33-94 , 2000.

[86] A. Cassandra, M. Nodine, S. Bondale, S. Ford and D. Wells, "Using POMDP-Based State Estimation to Enhance Agent System Survivability," in *Symposium on Multi-Agent Security and Survivability*, Philadelphia, PA, USA, 2005.

[87] V. Myers and D. P. Williams, "Adaptive Multiview Target Classification in Synthetic Aperture Sonar Images Using a Partially Observable Markov Decision Process," *IEEE Journal of Oceanic Engineering ,* vol. 37, no. 1, pp. 45-55, 2012.

[88] Q. Yang, J. Zhang and G. Shi, "Path planning for unmanned aerial vehicle passive detection under the framework of partially observable Markov decision process," in *Chinese Control And Decision Conference (CCDC)*, Shenyang, China, 2018.

[89] M. L. Littman, A. R. Cassandra and L. P. Kaelbling, "Learning policies for partially observable environments: Scaling up," in *International Conference on Machine Learning*, Tahoe City, California, USA, 1995.

[90] O. Hassanein, S. A. Salman, S. G. Anavatti and T. Ray, "ANFN controller based on differential evolution for Autonomous Underwater Vehicles," in *IEEE International Conference on Innovative Engineering Systems*, Alexandria, Egypt, 2012.

[91] D. L. Poole and A. K. Mackworth, Artificial intelligence : foundations of computational agents, New York, NY, USA: Cambridge University Press, 2010.

APPENDIX A – POMDP MODEL FILE

```
# Kathleen Svendsen
# Depth and Power System

Model: basic_PowerDepth
horizon: 1
discount: 0.9

#options for analysis: None MDP QMDP
analysis: QMDP

#list my actions, i can list my actions into groups, these become joint actions
NUM_ACTION_GROUPS: 2
#depth 1
AG: DEFLECT_NONE DEFLECT_DOWN DEFLECT_UP
#power 1
AG: POWER_NORMAL POWER_SAVING_MODE  ABORT

#list my states------------------------------------------------------------------
NUM_STATE_GROUPS: 5
# Depth, 2
SG: DEPTH_GOOD DEPTH_SHALLOW DEPTH_DEEP
SG: PITCH_GREATLY_UP PITCH_UP_MAX PITCH_UP PITCH_LEVEL PITCH_DOWN PITCH_DOWN_MAX
PITCH_GREATLY_DOWN

#POWER 3
SG: POWER_GOOD POWER_LOW POWER_VERYLOW POWER_CRITICAL
SG: USAGE_NORMAL POWER_SAVING ABORTED
SG: FIRST_QUARTER SECOND_QUARTER THIRD_QUARTER ALMOST_DONE


#list my observations------------------------------------------------------------------
NUM_OBSERVATION_GROUPS: 8
#depth 4
OG: ALTITUDE_OK ALTITUDE_LOW ALTITUDE_HIGH ALTITUDE_UNKNOWN
OG: DEPTH_GOOD DEPTH_SHALLOW DEPTH_DEEP DEPTH_UNKNOWN
OG: PITCH_UNCHANGING PITCH_INCREASING PITCH_DECREASING
OG: PITCH_GREATLY_UP PITCH_UP PITCH_LEVEL PITCH_DOWN PITCH_GREATLY_DOWN

#power 4
OG: CAPACITY_OK CAPACITY_LOW CAPACITY_VERYLOW CAPACITY_CRITICAL
OG: HOTEL_LOW HOTEL_OK HOTEL_HIGH
OG: FIRST_QUARTER SECOND_QUARTER THIRD_QUARTER ALMOST_DONE
OG: USAGE_NORMAL POWER_SAVING ABORTED




#-------------------------------------------------------------------------------------------------------------
#Observation probabilities
# O: <a1 a2...an> : <state> : <o1 o2 ... om> : %f
```

#Depth Observations----------
#how does no deflection affect the PITCH
O: * : PITCH_GREATLY_UP : PITCH_GREATLY_UP PITCH_INCREASING : 0.99
O: * : PITCH_GREATLY_UP : PITCH_GREATLY_UP : 0.9
O: * : PITCH_GREATLY_DOWN : PITCH_GREATLY_DOWN PITCH_DECREASING : 0.99
O: * : PITCH_GREATLY_DOWN : PITCH_GREATLY_DOWN : 0.9

O: * : PITCH_UP : PITCH_UP : 0.7
O: * : PITCH_DOWN : PITCH_DOWN : 0.7
O: * : PITCH_UP_MAX : PITCH_UP PITCH_UNCHANGING : 0.85
O: * : PITCH_DOWN_MAX : PITCH_DOWN PITCH_UNCHANGING : 0.85

O: DEFLECT_NONE : PITCH_LEVEL : PITCH_LEVEL : 0.9
O: DEFLECT_NONE : PITCH_LEVEL : PITCH_LEVEL PITCH_UNCHANGING : 0.9

O: DEFLECT_NONE : * : PITCH_UNCHANGING : 0.9
O: DEFLECT_NONE : PITCH_LEVEL : PITCH_LEVEL PITCH_UNCHANGING : 0.9
O: DEFLECT_NONE : PITCH_UP : PITCH_UP PITCH_UNCHANGING : 0.8
O: DEFLECT_NONE : PITCH_GREATLY_UP : PITCH_GREATLY_UP PITCH_UNCHANGING : 0.8
O: DEFLECT_NONE : PITCH_DOWN : PITCH_DOWN PITCH_UNCHANGING : 0.8
O: DEFLECT_NONE : PITCH_GREATLY_DOWN : PITCH_GREATLY_DOWN PITCH_UNCHANGING : 0.8

O: DEFLECT_UP : * : PITCH_INCREASING : 0.9
O: DEFLECT_UP : PITCH_LEVEL : PITCH_UP PITCH_INCREASING : 0.7
O: DEFLECT_UP : PITCH_LEVEL : PITCH_GREATLY_UP PITCH_INCREASING : 0.4
O: DEFLECT_UP : PITCH_UP : PITCH_UP PITCH_INCREASING : 0.7
O: DEFLECT_UP : PITCH_UP : PITCH_GREATLY_UP PITCH_INCREASING : 0.6

O: DEFLECT_DOWN : * : PITCH_DECREASING : 0.9
O: DEFLECT_DOWN : PITCH_LEVEL : PITCH_DOWN PITCH_DECREASING : 0.7
O: DEFLECT_DOWN : PITCH_LEVEL : PITCH_GREATLY_DOWN PITCH_DECREASING : 0.4
O: DEFLECT_DOWN : PITCH_DOWN : PITCH_DOWN PITCH_DECREASING : 0.7
O: DEFLECT_DOWN : PITCH_DOWN : PITCH_GREATLY_DOWN PITCH_DECREASING : 0.6

O: * : PITCH_GREATLY_UP : PITCH_GREATLY_UP : 0.9
O: * : PITCH_GREATLY_DOWN : PITCH_GREATLY_DOWN : 0.9
O: * : PITCH_GREATLY_UP : ALTITUDE_UNKNOWN PITCH_GREATLY_UP : 0.9
O: * : PITCH_GREATLY_DOWN : ALTITUDE_UNKNOWN PITCH_GREATLY_DOWN : 0.9

O: DEFLECT_DOWN : PITCH_UP_MAX : PITCH_GREATLY_UP : 0.8
O: DEFLECT_NONE : PITCH_UP_MAX : PITCH_UP_MAX : 0.9

O: DEFLECT_UP : PITCH_DOWN_MAX : PITCH_GREATLY_DOWN : 0.8
O: DEFLECT_NONE : PITCH_DOWN_MAX : PITCH_DOWN_MAX : 0.9

#depth related observations
O: * : DEPTH_SHALLOW : ALTITUDE_UNKNOWN : 0.8
O: * : DEPTH_SHALLOW PITCH_UP_MAX : ALTITUDE_UNKNOWN PITCH_UP_MAX : 0.8
O: * : DEPTH_SHALLOW PITCH_DOWN_MAX : ALTITUDE_UNKNOWN PITCH_DOWN_MAX : 0.8
O: * : DEPTH_SHALLOW PITCH_LEVEL : ALTITUDE_UNKNOWN PITCH_LEVEL : 0.8
O: * : DEPTH_SHALLOW PITCH_DOWN : ALTITUDE_UNKNOWN PITCH_DOWN : 0.8
O: * : DEPTH_SHALLOW PITCH_UP : ALTITUDE_UNKNOWN PITCH_UP: 0.8

O: * : DEPTH_SHALLOW : DEPTH_SHALLOW : 0.8
O: * : DEPTH_SHALLOW : ALTITUDE_HIGH : 0.8
O: * : DEPTH_SHALLOW : ALTITUDE_HIGH DEPTH_GOOD : 0.8
O: * : DEPTH_SHALLOW : ALTITUDE_HIGH DEPTH_SHALLOW : 0.9

O: * : DEPTH_GOOD : ALTITUDE_OK : 0.7
O: * : DEPTH_GOOD : DEPTH_GOOD : 0.7
O: * : DEPTH_GOOD : ALTITUDE_OK DEPTH_GOOD : 0.9
O: * : DEPTH_GOOD : ALTITUDE_HIGH DEPTH_GOOD : 0.3

O: * : DEPTH_DEEP : DEPTH_DEEP : 0.9
O: * : DEPTH_DEEP : ALTITUDE_LOW : 0.9
O: * : DEPTH_DEEP : DEPTH_DEEP ALTITUDE_LOW : 0.95
O: * : DEPTH_DEEP : DEPTH_GOOD ALTITUDE_LOW : 0.95
O: * : DEPTH_DEEP : DEPTH_DEEP ALTITUDE_UNKNOWN : 0.95
O: * : DEPTH_DEEP : DEPTH_DEEP ALTITUDE_HIGH : 0.95
O: * : DEPTH_DEEP : DEPTH_DEEP ALTITUDE_OK : 0.95
#---
#Observation probabilities POWER

O: * : USAGE_NORMAL : USAGE_NORMAL : 0.95
O: * : POWER_SAVING : POWER_SAVING : 0.95
O: * : ABORTED : ABORTED : 0.95

O: * : POWER_GOOD : CAPACITY_OK : 0.7
O: * : POWER_GOOD : CAPACITY_OK HOTEL_LOW : 0.9
O: * : POWER_GOOD : CAPACITY_OK HOTEL_HIGH : 0.6
O: * : POWER_GOOD : CAPACITY_OK HOTEL_OK : 0.8

O: * : POWER_GOOD : CAPACITY_LOW HOTEL_HIGH : 0.5
O: * : POWER_GOOD : CAPACITY_LOW THIRD_QUARTER : 0.7
O: * : POWER_GOOD : CAPACITY_LOW ALMOST_DONE : 0.8

O: * : POWER_LOW : CAPACITY_LOW : 0.8
O: * : POWER_LOW : CAPACITY_LOW HOTEL_OK : 0.8
O: * : POWER_LOW : CAPACITY_LOW HOTEL_LOW : 0.7
O: * : POWER_LOW : CAPACITY_LOW HOTEL_HIGH : 0.7
O: * : POWER_VERYLOW : CAPACITY_LOW HOTEL_HIGH : 0.6

O: * : POWER_VERYLOW : CAPACITY_VERYLOW : 0.8
O: * : POWER_VERYLOW : CAPACITY_VERYLOW HOTEL_OK : 0.8
O: * : POWER_VERYLOW : CAPACITY_VERYLOW HOTEL_LOW : 0.7
O: * : POWER_VERYLOW : CAPACITY_VERYLOW HOTEL_HIGH : 0.5
O: * : POWER_CRITICAL : CAPACITY_VERYLOW HOTEL_HIGH : 0.6

O: * : POWER_CRITICAL : CAPACITY_CRITICAL : 0.7
O: * : POWER_CRITICAL : CAPACITY_CRITICAL HOTEL_HIGH : 0.9
O: * : POWER_CRITICAL : CAPACITY_CRITICAL HOTEL_OK : 0.8
O: * : POWER_CRITICAL : CAPACITY_CRITICAL HOTEL_LOW : 0.7

#time observations
O: * : FIRST_QUARTER : SECOND_QUARTER : 0.2

```
O: * : FIRST_QUARTER : THIRD_QUARTER : 0.05
O: * : FIRST_QUARTER : ALMOST_DONE : 0.01
O: * : FIRST_QUARTER : FIRST_QUARTER : 0.9

O: * : SECOND_QUARTER : FIRST_QUARTER : 0.2
O: * : SECOND_QUARTER : THIRD_QUARTER : 0.2
O: * : SECOND_QUARTER : ALMOST_DONE : 0.05
O: * : SECOND_QUARTER : SECOND_QUARTER : 0.8

O: * : THIRD_QUARTER : FIRST_QUARTER : 0.01
O: * : THIRD_QUARTER : SECOND_QUARTER : 0.2
O: * : THIRD_QUARTER : ALMOST_DONE : 0.2
O: * : THIRD_QUARTER : THIRD_QUARTER : 0.75

O: * : ALMOST_DONE : FIRST_QUARTER : 0.001
O: * : ALMOST_DONE : SECOND_QUARTER : 0.1
O: * : ALMOST_DONE : THIRD_QUARTER : 0.3
O: * : ALMOST_DONE : ALMOST_DONE : 0.7

#cascade

O: * : DEPTH_DEEP : DEPTH_UNKNOWN : 0.9
O: * : POWER_VERYLOW DEPTH_DEEP : DEPTH_UNKNOWN POWER_VERYLOW : 0.9
O: * : POWER_CRITICAL DEPTH_DEEP : DEPTH_UNKNOWN POWER_CRITICAL : 0.9
O: * : POWER_VERYLOW DEPTH_DEEP : DEPTH_UNKNOWN POWER_VERYLOW ALTITUDE_UNKNOWN :
0.9
O: * : POWER_CRITICAL DEPTH_DEEP : DEPTH_UNKNOWN POWER_CRITICAL ALTITUDE_UNKNOWN : 0.9

#-----------------------------------------------------------------------------------------------------
#Transition probabilities
#   T: <a1 a2...an> : <start-state> : <end-state> : %f  P(S' : A, S)

#Depth Transitions-----------------------------------------------------------------------------
# no change in deflect no change in PITCH
T: DEFLECT_NONE : PITCH_GREATLY_UP : PITCH_GREATLY_UP : 0.9
T: DEFLECT_NONE : PITCH_UP : PITCH_UP : 0.9
T: DEFLECT_NONE : PITCH_LEVEL : PITCH_LEVEL : 0.9
T: DEFLECT_NONE : PITCH_DOWN : PITCH_DOWN : 0.9
T: DEFLECT_NONE : PITCH_GREATLY_DOWN : PITCH_GREATLY_DOWN : 0.9
T: DEFLECT_NONE : PITCH_UP_MAX : PITCH_UP_MAX : 0.9
T: DEFLECT_NONE : PITCH_DOWN_MAX : PITCH_DOWN_MAX : 0.9

#my PITCH changes my depth (regardless of my action)
T: * : PITCH_UP DEPTH_DEEP : DEPTH_DEEP : 0.4
T: * : PITCH_UP DEPTH_DEEP : DEPTH_GOOD : 0.6
T: * : PITCH_UP DEPTH_GOOD : DEPTH_GOOD : 0.4
T: * : PITCH_UP DEPTH_GOOD : DEPTH_SHALLOW : 0.6
T: * : PITCH_UP DEPTH_SHALLOW : DEPTH_SHALLOW : 0.9

T: * : PITCH_GREATLY_UP DEPTH_DEEP : DEPTH_DEEP : 0.3
T: * : PITCH_GREATLY_UP DEPTH_DEEP : DEPTH_GOOD : 0.7
T: * : PITCH_GREATLY_UP DEPTH_GOOD : DEPTH_GOOD : 0.3
T: * : PITCH_GREATLY_UP DEPTH_GOOD : DEPTH_SHALLOW : 0.7
```

T: * : PITCH_GREATLY_UP DEPTH_SHALLOW : DEPTH_SHALLOW : 0.95

T: * : PITCH_DOWN DEPTH_SHALLOW : DEPTH_SHALLOW : 0.4
T: * : PITCH_DOWN DEPTH_SHALLOW : DEPTH_GOOD : 0.6
T: * : PITCH_DOWN DEPTH_GOOD : DEPTH_GOOD : 0.4
T: * : PITCH_DOWN DEPTH_GOOD : DEPTH_DEEP : 0.6
T: * : PITCH_DOWN DEPTH_DEEP : DEPTH_DEEP : 0.9

T: * : PITCH_GREATLY_DOWN DEPTH_SHALLOW : DEPTH_SHALLOW : 0.3
T: * : PITCH_GREATLY_DOWN DEPTH_SHALLOW : DEPTH_GOOD : 0.7
T: * : PITCH_GREATLY_DOWN DEPTH_GOOD : DEPTH_GOOD : 0.3
T: * : PITCH_GREATLY_DOWN DEPTH_GOOD : DEPTH_DEEP : 0.7
T: * : PITCH_GREATLY_DOWN DEPTH_DEEP : DEPTH_DEEP : 0.95

#SG: PITCH_GREATLY_UP PITCH_UP_MAX PITCH_UP PITCH_LEVEL PITCH_DOWN PITCH_DOWN_MAX PITCH_GREATLY_DOWN

#opposite deflect changes PITCH
T: DEFLECT_DOWN : PITCH_GREATLY_UP : PITCH_GREATLY_UP : 0.4
T: DEFLECT_DOWN : PITCH_GREATLY_UP : PITCH_UP : 0.3
T: DEFLECT_DOWN : PITCH_GREATLY_UP : PITCH_UP_MAX : 0.8
T: DEFLECT_DOWN : PITCH_UP_MAX : PITCH_UP_MAX : 0.4
T: DEFLECT_DOWN : PITCH_UP_MAX : PITCH_LEVEL : 0.3
T: DEFLECT_DOWN : PITCH_UP_MAX : PITCH_UP : 0.8
T: DEFLECT_DOWN : PITCH_UP : PITCH_LEVEL : 0.8
T: DEFLECT_DOWN : PITCH_UP : PITCH_DOWN : 0.4

T: DEFLECT_UP : PITCH_GREATLY_DOWN : PITCH_GREATLY_DOWN : 0.4
T: DEFLECT_UP : PITCH_GREATLY_DOWN : PITCH_DOWN_MAX : 0.3
T: DEFLECT_UP : PITCH_GREATLY_DOWN : PITCH_DOWN_MAX : 0.8
T: DEFLECT_UP : PITCH_DOWN_MAX : PITCH_DOWN_MAX : 0.4
T: DEFLECT_UP : PITCH_DOWN_MAX : PITCH_LEVEL : 0.3
T: DEFLECT_UP : PITCH_DOWN_MAX : PITCH_DOWN_MAX : 0.8
T: DEFLECT_UP : PITCH_DOWN_MAX : PITCH_LEVEL : 0.8
T: DEFLECT_UP : PITCH_DOWN_MAX : PITCH_UP : 0.4

#SAME deflect INCREASE PITCH
T: DEFLECT_UP : PITCH_GREATLY_UP : PITCH_GREATLY_UP : 0.9
T: DEFLECT_UP : PITCH_DOWN_MAX : PITCH_GREATLY_UP : 0.9
T: DEFLECT_UP : PITCH_UP : PITCH_DOWN_MAX : 0.7
T: DEFLECT_UP : PITCH_UP : PITCH_UP : 0.6
T: DEFLECT_UP : PITCH_LEVEL : PITCH_UP : 0.9
T: DEFLECT_UP : PITCH_DOWN : PITCH_LEVEL : 0.7
T: DEFLECT_UP : PITCH_DOWN_MAX : PITCH_DOWN : 0.9
T: DEFLECT_UP : PITCH_GREATLY_DOWN: PITCH_DOWN_MAX : 0.9

T: DEFLECT_DOWN : PITCH_GREATLY_UP : PITCH_UP_MAX : 0.9
T: DEFLECT_DOWN : PITCH_UP_MAX : PITCH_UP : 0.9
T: DEFLECT_DOWN : PITCH_UP : PITCH_LEVEL : 0.7
T: DEFLECT_DOWN : PITCH_LEVEL : PITCH_DOWN : 0.9
T: DEFLECT_DOWN : PITCH_DOWN : PITCH_DOWN : 0.6
T: DEFLECT_DOWN : PITCH_DOWN : PITCH_DOWN_MAX : 0.7

T: DEFLECT_DOWN : PITCH_DOWN_MAX : PITCH_GREATLY_DOWN : 0.9
T: DEFLECT_DOWN : PITCH_GREATLY_DOWN: PITCH_GREATLY_DOWN : 0.9

#NO deflect
T: DEFLECT_NONE : PITCH_GREATLY_UP : PITCH_GREATLY_UP : 0.9
T: DEFLECT_NONE : PITCH_UP_MAX : PITCH_UP_MAX : 0.9
T: DEFLECT_NONE : PITCH_UP : PITCH_UP : 0.9
T: DEFLECT_NONE : PITCH_LEVEL : PITCH_LEVEL : 0.9
T: DEFLECT_NONE : PITCH_DOWN : PITCH_DOWN : 0.9
T: DEFLECT_NONE : PITCH_DOWN_MAX : PITCH_DOWN_MAX : 0.9
T: DEFLECT_NONE : PITCH_GREATLY_DOWN: PITCH_GREATLY_DOWN : 0.9

#Power Transition

T: POWER_NORMAL : * : USAGE_NORMAL : 0.8
T: POWER_SAVING_MODE : * : POWER_SAVING : 0.8
T: ABORT : * : ABORTED : 0.8

T: POWER_NORMAL : POWER_NORMAL USAGE_NORMAL : POWER_NORMAL USAGE_NORMAL : 0.6
T: POWER_NORMAL : POWER_NORMAL USAGE_NORMAL : POWER_LOW USAGE_NORMAL : 0.3
T: POWER_NORMAL : POWER_NORMAL USAGE_NORMAL : POWER_VERYLOW USAGE_NORMAL: 0.1
T: POWER_NORMAL : POWER_NORMAL USAGE_NORMAL : POWER_CRITICAL USAGE_NORMAL: 0.01

T: POWER_SAVING_MODE : POWER_NORMAL POWER_SAVING : POWER_NORMAL POWER_SAVING : 0.7
T: POWER_SAVING_MODE : POWER_NORMAL POWER_SAVING : POWER_LOW POWER_SAVING : 0.2
T: POWER_SAVING_MODE : POWER_NORMAL POWER_SAVING : POWER_VERYLOW POWER_SAVING:
0.01
T: POWER_SAVING_MODE : POWER_NORMAL POWER_SAVING : POWER_CRITICAL POWER_SAVING:
0.001

T: POWER_NORMAL : POWER_LOW USAGE_NORMAL : POWER_NORMAL USAGE_NORMAL : 0.2
T: POWER_NORMAL : POWER_LOW USAGE_NORMAL : POWER_LOW USAGE_NORMAL : 0.6
T: POWER_NORMAL : POWER_LOW USAGE_NORMAL : POWER_VERYLOW USAGE_NORMAL: 0.2
T: POWER_NORMAL : POWER_LOW USAGE_NORMAL : POWER_CRITICAL USAGE_NORMAL: 0.1

T: POWER_SAVING_MODE : POWER_LOW POWER_SAVING : POWER_NORMAL POWER_SAVING : 0.3
T: POWER_SAVING_MODE : POWER_LOW POWER_SAVING : POWER_LOW POWER_SAVING : 0.7
T: POWER_SAVING_MODE : POWER_LOW POWER_SAVING : POWER_VERYLOW POWER_SAVING: 0.1
T: POWER_SAVING_MODE : POWER_LOW POWER_SAVING : POWER_CRITICAL POWER_SAVING: 0.05

T: POWER_NORMAL : POWER_VERYLOW USAGE_NORMAL : POWER_NORMAL USAGE_NORMAL : 0.05
T: POWER_NORMAL : POWER_VERYLOW USAGE_NORMAL : POWER_LOW USAGE_NORMAL : 0.1
T: POWER_NORMAL : POWER_VERYLOW USAGE_NORMAL : POWER_VERYLOW USAGE_NORMAL: 0.6
T: POWER_NORMAL : POWER_VERYLOW USAGE_NORMAL : POWER_CRITICAL USAGE_NORMAL: 0.2

T: POWER_SAVING_MODE : POWER_VERYLOW POWER_SAVING : POWER_NORMAL POWER_SAVING : 0.1
T: POWER_SAVING_MODE : POWER_VERYLOW POWER_SAVING : POWER_LOW POWER_SAVING : 0.3
T: POWER_SAVING_MODE : POWER_VERYLOW POWER_SAVING : POWER_VERYLOW POWER_SAVING:
0.7
T: POWER_SAVING_MODE : POWER_VERYLOW POWER_SAVING : POWER_CRITICAL POWER_SAVING: 0.1

T: POWER_NORMAL : POWER_CRITICAL USAGE_NORMAL : POWER_NORMAL USAGE_NORMAL : 0.001
T: POWER_NORMAL : POWER_CRITICAL USAGE_NORMAL : POWER_LOW USAGE_NORMAL : 0.01
T: POWER_NORMAL : POWER_CRITICAL USAGE_NORMAL : POWER_VERYLOW USAGE_NORMAL: 0.2
T: POWER_NORMAL : POWER_CRITICAL USAGE_NORMAL : POWER_CRITICAL USAGE_NORMAL: 0.8

T: POWER_SAVING_MODE : POWER_CRITICAL POWER_SAVING : POWER_NORMAL POWER_SAVING : 0.002
T: POWER_SAVING_MODE : POWER_CRITICAL POWER_SAVING : POWER_LOW POWER_SAVING : 0.02
T: POWER_SAVING_MODE : POWER_CRITICAL POWER_SAVING : POWER_VERYLOW POWER_SAVING: 0.3
T: POWER_SAVING_MODE : POWER_CRITICAL POWER_SAVING : POWER_CRITICAL POWER_SAVING: 0.7

#clock goes forward regardless (if had a navigation system this would be more in depth)
T: * : FIRST_QUARTER : FIRST_QUARTER : 0.7
T: * : FIRST_QUARTER : SECOND_QUARTER : 0.25
T: * : FIRST_QUARTER : THIRD_QUARTER : 0.05
T: * : FIRST_QUARTER : ALMOST_DONE : 0.01

T: * : SECOND_QUARTER : FIRST_QUARTER : 0.1
T: * : SECOND_QUARTER : SECOND_QUARTER : 0.7
T: * : SECOND_QUARTER : THIRD_QUARTER : 0.25
T: * : SECOND_QUARTER : ALMOST_DONE : 0.05

T: * : SECOND_QUARTER : FIRST_QUARTER : 0.001
T: * : SECOND_QUARTER : SECOND_QUARTER : 0.1
T: * : THIRD_QUARTER : THIRD_QUARTER : 0.7
T: * : THIRD_QUARTER : ALMOST_DONE : 0.3

T: * : ALMOST_DONE : THIRD_QUARTER : 0.1
T: * : ALMOST_DONE : FIRST_QUARTER : 0.001
T: * : ALMOST_DONE : SECOND_QUARTER : 0.001

#cascade----------

T: * : POWER_CRITICAL : DEPTH_UNKNOWN : 0.8
T: * : POWER_CRITICAL : ALTITUDE_UNKNOWN : 0.8
T: * : POWER_VERYLOW : DEPTH_UNKNOWN : 0.7
T: * : POWER_VERYLOW : ALTITUDE_UNKNOWN : 0.7
T: * : POWER_CRITICAL : DEPTH_UNKNOWN ALTITUDE_UNKNOWN : 0.8
T: * : POWER_VERYLOW : DEPTH_UNKNOWN ALTITUDE_UNKNOWN : 0.8
T: * : POWER_CRITICAL : POWER_CRITICAL DEPTH_UNKNOWN ALTITUDE_UNKNOWN : 0.99
T: * : POWER_VERYLOW : POWER_VERYLOW DEPTH_UNKNOWN ALTITUDE_UNKNOWN : 0.99

#myRewards
#The rewards
R: <a1 a2...an> : <start-state> : %f

#Depth rewards----------

#first lets cover deflection
R: DEFLECT_NONE : DEPTH_GOOD : 5
R: DEFLECT_NONE : DEPTH_GOOD PITCH_LEVEL : 35
R: DEFLECT_NONE : PITCH_GREATLY_DOWN : -30
R: DEFLECT_NONE : PITCH_GREATLY_UP : -30
R: DEFLECT_NONE : DEPTH_SHALLOW PITCH_DOWN_MAX : 20
R: DEFLECT_NONE : DEPTH_DEEP PITCH_UP_MAX : 20
R: DEFLECT_DOWN : PITCH_DOWN_MAX : -25
R: DEFLECT_UP : PITCH_UP_MAX : -25

R: DEFLECT_UP : DEPTH_DEEP : 20
R: DEFLECT_DOWN : DEPTH_SHALLOW : 20
R: DEFLECT_DOWN : DEPTH_SHALLOW PITCH_LEVEL : 10
R: DEFLECT_UP : DEPTH_DEEP PITCH_LEVEL : 5
R: DEFLECT_NONE : DEPTH_DEEP PITCH_DOWN_MAX : 10

R: DEFLECT_DOWN : DEPTH_GOOD PITCH_LEVEL : -10
R: DEFLECT_DOWN : DEPTH_GOOD PITCH_UP : 10
R: DEFLECT_DOWN : PITCH_GREATLY_DOWN : -25
R: DEFLECT_DOWN : PITCH_GREATLY_UP : 10
R: DEFLECT_DOWN : ALTITUDE_UNKNOWN PITCH_LEVEL : 20
R: DEFLECT_DOWN : ALTITUDE_UNKNOWN PITCH_UP : 20
R: DEFLECT_DOWN : ALTITUDE_UNKNOWN PITCH_UP_MAX : 20
R: DEFLECT_NONE : ALTITUDE_UNKNOWN PITCH_DOWN_MAX : 20

R: DEFLECT_UP : DEPTH_GOOD PITCH_DOWN : 10
R: DEFLECT_UP : DEPTH_GOOD PITCH_LEVEL : -10
R: DEFLECT_UP : PITCH_GREATLY_UP : -25
R: DEFLECT_UP : PITCH_GREATLY_DOWN : 10

R: DEFLECT_UP : DEPTH_DEEP : 20
R: DEFLECT_DOWN : DEPTH_SHALLOW : 20

R: * : DEPTH_SHALLOW : -30
R: * : DEPTH_DEEP : -30
R: * : DEPTH_GOOD : 20

#Power
R: POWER_NORMAL : * : 1
R: POWER_NORMAL : POWER_GOOD USAGE_NORMAL : 10
R: POWER_NORMAL : POWER_LOW THIRD_QUARTER : 5
R: POWER_NORMAL : POWER_LOW ALMOST_DONE : 10

R: POWER_SAVING_MODE : * : -1
R: POWER_SAVING_MODE : POWER_GOOD : -10
R: POWER_SAVING_MODE : POWER_LOW SECOND_QUARTER : 10
R: POWER_SAVING_MODE : POWER_LOW THIRD_QUARTER : 5
R: POWER_SAVING_MODE : POWER_VERYLOW : 20

R: * : POWER_LOW : -5
R: * : POWER_CRITICAL : -20
R: ABORT : POWER_GOOD : -20

R: ABORT : POWER_LOW : -5
R: ABORT : POWER_LOW FIRST_QUARTER : 10
R: ABORT : POWER_LOW POWER_SAVING SECOND_QUARTER : 10

R: ABORT : POWER_VERYLOW ALMOST_DONE: -5
R: ABORT : POWER_VERYLOW USAGE_NORMAL THIRD_QUARTER: -5
R: ABORT : POWER_VERYLOW POWER_SAVING THIRD_QUARTER: 10
R: ABORT : POWER_VERYLOW FIRST_QUARTER: 20
R: ABORT : POWER_VERYLOW SECOND_QUARTER: 10

R: ABORT : POWER_CRITICAL POWER_SAVING : 20
R: ABORT : POWER_CRITICAL USAGE_NORMAL : 5
R: ABORT : POWER_CRITICAL FIRST_QUARTER : 10
R: ABORT : POWER_CRITICAL SECOND_QUARTER : 10
R: ABORT : POWER_CRITICAL THIRD_QUARTER : 10

R: ABORT : ABORTED : 1
R: POWER_NORMAL : ABORTED : -40
R: POWER_SAVING_MODE : ABORTED : -40

R: DEFLECT_UP : ABORTED : 30
R: DEFLECT_DOWN : ABORTED : -40
R: DEFLECT_NONE : ABORTED : -40
R: DEFLECT_DOWN : POWER_CRITICAL : -40
R: DEFLECT_NONE : POWER_CRITICAL : -40
R: PITCH_DOWN : POWER_VERYLOW : -40
R: PITCH_LEVEL : POWER_VERYLOW : -40

R: * : DEPTH_GOOD POWER_VERYLOW : -30
R: * : DEPTH_SHALLOW POWER_CRITICAL PITCH_UP : 20
R: DEFLECT_UP : POWER_LOW PITCH_UP : 10
R: DEFLECT_UP : POWER_VERYLOW PITCH_UP : 15
R: DEFLECT_UP : POWER_VERYLOW PITCH_UP_MAX : 25
R: DEFLECT_UP : POWER_VERYLOW PITCH_GREATLY_UP : -10
R: DEFLECT_UP : POWER_VERYLOW DEPTH_SHALLOW PITCH_UP : 20
R: DEFLECT_UP : POWER_VERYLOW DEPTH_SHALLOW PITCH_UP_MAX : 30
R: DEFLECT_UP : POWER_VERYLOW DEPTH_SHALLOW PITCH_GREATLY_UP : -10
R: DEFLECT_UP : POWER_CRITICAL PITCH_UP : 30
R: DEFLECT_UP : POWER_CRITICAL PITCH_UP_MAX : 40
R: DEFLECT_UP : POWER_CRITICAL PITCH_GREATLY_UP : 20
R: DEFLECT_UP ABORT : POWER_CRITICAL PITCH_UP ABORTED : 40
R: DEFLECT_UP ABORT : POWER_CRITICAL PITCH_UP_MAX ABORTED : 60
R: DEFLECT_UP ABORT : POWER_CRITICAL PITCH_GREATLY_UP ABORTED : 60

R: * : DEPTH_UNKNOWN POWER_VERYLOW : -30
R: * : DEPTH_UNKNOWN POWER_CRITICAL PITCH_UP : 10
R: DEFLECT_UP ABORT : DEPTH_UNKNOWN POWER_CRITICAL PITCH_UP ABORTED : 40
R: DEFLECT_UP ABORT : DEPTH_UNKNOWN POWER_CRITICAL PITCH_UP_MAX ABORTED : 60
R: DEFLECT_UP ABORT : DEPTH_UNKNOWN POWER_CRITICAL PITCH_GREATLY_UP ABORTED : 40

APPENDIX B - MISSION CONTROL FILE

```
# depth
# shallow
/Analysis/Models/POMDP/basic_Depth.pomdp      :  /Analysis/Models/AUV/depth_shallow.auv
:  /Analysis/Simulations/Sim_Depth_Shallow

/Analysis/Models/POMDP/basic_Depth.pomdp      : /Analysis/Models/AUV/depth_bedfordbasin.auv
: /Analysis/Simulations/Sim_Depth_befordBasin

/Analysis/Models/POMDP/basic_Depth_noise.pomdp  :
/Analysis/Models/AUV/depth_shallow_noise.auv  : Analysis/Simulations/Sim_Depth_Shallow_Noise

# Deep
/Analysis/Models/POMDP/basic_Depth.pomdp  : /Analysis/Models/AUV/depth_deep.auv
: /Analysis/Simulations/Sim_Depth_Deep

# Erratic
/Analysis/Models/POMDP/basic_Depth.pomdp  : /Analysis/Models/AUV/depth_erratic.auv    :
/Analysis/Simulations/Sim_Depth_Erratic

# Power
/Models/POMDP/basic_Power.pomdp       : /Analysis/Models/AUV/power_Basic.auv    :
/Analysis/Simulations/Sim_Power_Basic

/Analysis/Models/POMDP/basic_Power.pomdp   : /Analysis/Models/AUV/power_lowCapacity.auv
: /Analysis/Simulations/Sim_Power_LowCapacity

/Analysis/Models/POMDP/basic_Power.pomdp    : /Analysis/Models/AUV/power_critical_abort.auv
: /Analysis/Simulations/Sim_Power_Critical_Abort

/Analysis/Models/POMDP/basic_Power_noise.pomdp : Analysis/Models/AUV/power_BasicNoise.auv
: Analysis/Simulations/Sim_Power_Noise

#  2 sub-systems depth and power
/Analysis/Models/POMDP/basic_PowerDepth.pomdp     :
/Analysis/Models/AUV/2PowerDepth_Basic.auv  : /Analysis/Simulations/Sim_PowerDepth_Basic

/Analysis/Models/POMDP/integrated_PowerDepth.pomdp :
/Analysis/Models/AUV/2PowerDepth_Basic_integrated.auv :
/Analysis/Simulations/Sim_PowerDepth_Integrated


# depth
# shallow
/Analysis/Models/POMDP/basic_Depth.pomdp      :  /Analysis/Models/AUV/depth_shallow.auv
:  /Analysis/Simulations/Sim_Depth_Shallow

/Analysis/Models/POMDP/basic_Depth.pomdp      : /Analysis/Models/AUV/depth_bedfordbasin.auv
: /Analysis/Simulations/Sim_Depth_befordBasin

/Analysis/Models/POMDP/basic_Depth_noise.pomdp  :
```

APPENDIX C – TABLE OF SIMULATIONS PERFORMED

<table>
<tr><th>thesis section - Depth sub-system simulations</th><th>figure</th><th>generated time histories</th><th>Initialization file for AUV simulator</th></tr>
<tr><td rowspan="5">Section 5.1.1.1
shallow water (simulated gradual incline, approx. 20 deg</td><td rowspan="4">Fig. 5-2</td><td>measured AUV depth, seabed depth</td><td rowspan="5">#This is a simple AUV that only has a depth system
AUV_NAME: AUV_Shallow
MISSION_TIME: 10000
NOISE: 0

#subsystem depth
DEPTH_FILE:
/home/kat/Dropbox/Kathleen_Thesis/Analysis/Models/DepthLogs/basicshallow.log
ALTITUDE_MIN: 8
ALTITUDE_MAX: 12
DVL_RANGE: 20
MAX_DEPTH: 35
MIN_DEPTH: 5
START_DEPTH: 2
NOISE_DEPTH: 2
MOVEMENT_MAG: 2
HIGH_ANGLE: 15
LOW_ANGLE: 2
ANGLE_CHANGE: 3</td></tr>
<tr><td>AUV altitude</td></tr>
<tr><td>AUV pitch</td></tr>
<tr><td>fin mode (-1 nose-down, +1 nose-up, 0 level)</td></tr>
<tr><td>Fig 5-3</td><td>belief distribution of states</td></tr>
<tr><td rowspan="5">Section 5.1.1.2
shallow water with additive noise(simulated gradual incline, approx. 20 deg</td><td rowspan="4">Fig. 5-4</td><td>measured AUV depth, seabed depth</td><td rowspan="5">#This is a simple AUV that only has a depth system
AUV_NAME: AUV_ShallowNoise
MISSION_TIME: 10000
NOISE: 1

#subsystem depth
DEPTH_FILE:
/home/kat/Dropbox/Kathleen_Thesis/Analysis/Models/DepthLogs/basicshallow.log
ALTITUDE_MIN: 8
ALTITUDE_MAX: 12
DVL_RANGE: 20
MAX_DEPTH: 35
MIN_DEPTH: 5
START_DEPTH: 2
NOISE_DEPTH: 2
MOVEMENT_MAG: 2
HIGH_ANGLE: 15
LOW_ANGLE: 2
ANGLE_CHANGE: 3</td></tr>
<tr><td>AUV altitude</td></tr>
<tr><td>AUV pitch</td></tr>
<tr><td>fin mode (-1 nose-down, +1 nose-up, 0 level)</td></tr>
<tr><td>Fig. 5-5</td><td>belief distribution of states</td></tr>
</table>

<table>
<tr><td rowspan="5">Section 5.1.1.3
shallow water (actual bathymetry)</td><td rowspan="4">Fig. 5-6</td><td>measured AUV depth, seabed depth</td><td rowspan="5"><pre>#This is a simple AUV that only has a depth system
AUV_NAME: Model_bedfordbasin_shallow
MISSION_TIME: 10000
NOISE: 0

#subsystem depth
DEPTH_FILE:
/home/kat/Dropbox/Kathleen_Thesis/Analysis/Models/DepthLogs
/bedfordBasin.log
ALTITUDE_MIN: 7
ALTITUDE_MAX: 10
DVL_RANGE: 20
MAX_DEPTH: 35
MIN_DEPTH: 5
START_DEPTH: 2
NOISE_DEPTH: 2
MOVEMENT_MAG: 2
HIGH_ANGLE: 15
LOW_ANGLE: 2
ANGLE_CHANGE: 5</pre></td></tr>
<tr><td>AUV altitude</td></tr>
<tr><td>AUV pitch</td></tr>
<tr><td>fin mode (-1 nose-down, +1 nose-up, 0 level)</td></tr>
<tr><td>Fig. 5-7</td><td>belief distribution of states</td></tr>
<tr><td rowspan="5">Section 5.1.2.1
deep water (simulated gradual decline and incline, approx. 30 deg, depth maximum 80 meters)</td><td rowspan="4">Fig. 5-8</td><td>measured AUV depth, seabed depth</td><td rowspan="5"><pre>#This is a simple AUV that only has a depth system
AUV_NAME: AUV_DeepWaters
MISSION_TIME: 10000
NOISE: 0
ABORT_AT_COMMAND: 0

#subsystem depth
DEPTH_FILE:
/home/kat/Dropbox/Kathleen_Thesis/Analysis/Models/Dept
hLogs/deep_simulation.log
ALTITUDE_MIN: 8
ALTITUDE_MAX: 12
DVL_RANGE: 18
MAX_DEPTH: 50
MIN_DEPTH: 15
START_DEPTH: 40
NOISE_DEPTH: 2
MOVEMENT_MAG: 2
HIGH_ANGLE: 15
LOW_ANGLE: 2
ANGLE_CHANGE: 5</pre></td></tr>
<tr><td>AUV altitude</td></tr>
<tr><td>AUV pitch</td></tr>
<tr><td>fin mode (-1 nose-down, +1 nose-up, 0 level)</td></tr>
<tr><td>Fig. 5-9</td><td>belief distribution of states</td></tr>
</table>

<table>
<tr><td rowspan="5">Section 5.1.2.2
deep water (actual bathymetry)</td><td rowspan="4">Fig. 5-10</td><td>measured AUV depth, seabed depth</td><td rowspan="5">#This is a simple AUV that only has a depth system
AUV_NAME: AUV_DeepReal
MISSION_TIME: 10000
NOISE: 0

#subsystem depth
DEPTH_FILE:
/home/kat/Dropbox/Kathleen_Thesis/Analysis/Models/DepthLogs/erratic_real.log
ALTITUDE_MIN: 8
ALTITUDE_MAX: 12
DVL_RANGE: 20
MAX_DEPTH: 15
MIN_DEPTH: 3
START_DEPTH: 10
NOISE_DEPTH: 2
MOVEMENT_MAG: 2
HIGH_ANGLE: 15
LOW_ANGLE: 2
ANGLE_CHANGE: 5</td></tr>
<tr><td>AUV altitude</td></tr>
<tr><td>AUV pitch</td></tr>
<tr><td>fin mode (-1 nose-down, +1 nose-up, 0 level)</td></tr>
<tr><td>Fig. 5-11</td><td>belief distribution of states</td></tr>
<tr><td rowspan="5">Section 5.1.3.1
variable seabed (simulated random)</td><td rowspan="4">Fig. 5-12</td><td>measured AUV depth, seabed depth</td><td rowspan="5">#This is a simple AUV that only has a depth system
AUV_NAME: AUV_Erratic
MISSION_TIME: 10000
NOISE: 0

#subsystem depth
DEPTH_FILE:
/home/kat/Dropbox/Kathleen_Thesis/Analysis/Models/DepthLogs/erratic_simulation.log
ALTITUDE_MIN: 12
ALTITUDE_MAX: 16
DVL_RANGE: 20
MAX_DEPTH: 35
MIN_DEPTH: 3
START_DEPTH: 20
NOISE_DEPTH: 2
MOVEMENT_MAG: 2
HIGH_ANGLE: 15
LOW_ANGLE: 2
ANGLE_CHANGE: 5</td></tr>
<tr><td>AUV altitude</td></tr>
<tr><td>AUV pitch</td></tr>
<tr><td>fin position(-1 nose-down, +1 nose-up, 0 level)</td></tr>
<tr><td>Fig. 5-13</td><td>belief distribution of states</td></tr>
</table>

<table>
<tr>
<td rowspan="4">Section 5.1.3.2
variable seabed (actual bathymetry)</td>
<td rowspan="4">Fig. 5-14</td>
<td>measured AUV depth, seabed depth</td>
<td rowspan="5">#This is a simple AUV that only has a depth system
AUV_NAME: AUV_ErraticReal
MISSION_TIME: 10000
NOISE: 0

#subsystem depth
DEPTH_FILE:
/home/kat/Dropbox/Kathleen_Thesis/Analysis/Models/DepthLogs/erratic_real.log
ALTITUDE_MIN: 8
ALTITUDE_MAX: 12
DVL_RANGE: 30
MAX_DEPTH: 35
MIN_DEPTH: 3
START_DEPTH: 10
NOISE_DEPTH: 2
MOVEMENT_MAG: 2
HIGH_ANGLE: 15
LOW_ANGLE: 2
ANGLE_CHANGE: 3</td>
</tr>
<tr><td>AUV altitude</td></tr>
<tr><td>AUV pitch</td></tr>
<tr><td>fin mode (-1 nose-down, +1 nose-up, 0 level)</td></tr>
<tr><td>Fig. 5-15</td><td>belief distribution of states</td></tr>
<tr>
<td rowspan="4">Section 5.1.4.1
shallow water with a +10% increase in uncertainty for the probability of observation function
O(o| a, s)</td>
<td rowspan="4">Fig. 5-16</td>
<td>measured AUV depth, seabed depth</td>
<td rowspan="5">#This is a simple AUV that only has a depth system
AUV_NAME: AUV_Shallow
MISSION_TIME: 10000
NOISE: 0

#subsystem depth
DEPTH_FILE:
/home/kat/Dropbox/Kathleen_Thesis/Analysis/Models/DepthLogs/basicshallow.log
ALTITUDE_MIN: 8
ALTITUDE_MAX: 12
DVL_RANGE: 20
MAX_DEPTH: 35
MIN_DEPTH: 5
START_DEPTH: 2
NOISE_DEPTH: 2
MOVEMENT_MAG: 2
HIGH_ANGLE: 15
LOW_ANGLE: 2
ANGLE_CHANGE: 3</td>
</tr>
<tr><td>AUV altitude</td></tr>
<tr><td>AUV pitch</td></tr>
<tr><td>fin mode (-1 nose-down, +1 nose-up, 0 level)</td></tr>
<tr><td>Fig. 5-17</td><td>belief distribution of states</td></tr>
</table>

<table>
<tr>
<td rowspan="5">Section 5.1.4.1
shallow water with +20% increase in uncertainty for the probability of observation function
O(| a, s)</td>
<td rowspan="4">Fig. 5-16</td>
<td>measured AUV depth, seabed depth</td>
<td rowspan="5">#This is a simple AUV that only has a depth system
AUV_NAME: AUV_Shallow
MISSION_TIME: 10000
NOISE: 0

#subsystem depth
DEPTH_FILE:
/home/kat/Dropbox/Kathleen_Thesis/Analysis/Models/Dept
hLogs/basicshallow.log
ALTITUDE_MIN: 8
ALTITUDE_MAX: 12
DVL_RANGE: 20
MAX_DEPTH: 35
MIN_DEPTH: 5
START_DEPTH: 2
NOISE_DEPTH: 2
MOVEMENT_MAG: 2
HIGH_ANGLE: 15
LOW_ANGLE: 2
ANGLE_CHANGE: 3</td>
</tr>
<tr><td>AUV altitude</td></tr>
<tr><td>AUV pitch</td></tr>
<tr><td>fin mode (-1 nose-down, +1 nose-up, 0 level)</td></tr>
<tr><td>Fig. 5-18</td><td>belief distribution of states</td></tr>
<tr>
<td rowspan="5">Section 5.1.4.1
shallow water with +40% increase in uncertainty for the probability of observation function
O(o | a, s)</td>
<td rowspan="4">Fig. 5-16</td>
<td>measured AUV depth, seabed depth</td>
<td rowspan="5">#This is a simple AUV that only has a depth system
AUV_NAME: AUV_Shallow
MISSION_TIME: 10000
NOISE: 0

#subsystem depth
DEPTH_FILE:
/home/kat/Dropbox/Kathleen_Thesis/Analysis/Models/Dept
hLogs/basicshallow.log
ALTITUDE_MIN: 8
ALTITUDE_MAX: 12
DVL_RANGE: 20
MAX_DEPTH: 35
MIN_DEPTH: 5
START_DEPTH: 2
NOISE_DEPTH: 2
MOVEMENT_MAG: 2
HIGH_ANGLE: 15
LOW_ANGLE: 2
ANGLE_CHANGE: 3</td>
</tr>
<tr><td>altitude</td></tr>
<tr><td>pitch</td></tr>
<tr><td>fin position (-1 down, +1 up, 0 level)</td></tr>
<tr><td>Fig. 5-19</td><td>belief distribution of states</td></tr>
</table>

<table>
<tr>
<td rowspan="5">Section 5.1.4.2
shallow water with +10% increase in uncertainty for the probability of transitions function
T(s’ | a, s)</td>
<td rowspan="4">Fig. 5-20</td>
<td>Measured AUV depth, seabed depth</td>
<td rowspan="5">#This is a simple AUV that only has a depth system
AUV_NAME: AUV_Shallow
MISSION_TIME: 10000
NOISE: 0

#subsystem depth
DEPTH_FILE:
/home/kat/Dropbox/Kathleen_Thesis/Analysis/Models/Dept
hLogs/basicshallow.log
ALTITUDE_MIN: 8
ALTITUDE_MAX: 12
DVL_RANGE: 20
MAX_DEPTH: 35
MIN_DEPTH: 5
START_DEPTH: 2
NOISE_DEPTH: 2
MOVEMENT_MAG: 2
HIGH_ANGLE: 15
LOW_ANGLE: 2
ANGLE_CHANGE: 3</td>
</tr>
<tr><td>altitude</td></tr>
<tr><td>pitch</td></tr>
<tr><td>fin position (-1 down, +1 up, 0 level)</td></tr>
<tr><td>Fig. 5-21</td><td>belief distribution of states</td></tr>
<tr>
<td rowspan="5">Section 5.1.4.2
shallow water with +20% increase in uncertainty for the probability of transitions function
T(s’ | a, s)</td>
<td rowspan="4">Fig. 5-20</td>
<td>measured AUV depth, seabed depth</td>
<td rowspan="5">#This is a simple AUV that only has a depth system
AUV_NAME: AUV_Shallow
MISSION_TIME: 10000
NOISE: 0

#subsystem depth
DEPTH_FILE:
/home/kat/Dropbox/Kathleen_Thesis/Analysis/Models/Dept
hLogs/basicshallow.log
ALTITUDE_MIN: 8
ALTITUDE_MAX: 12
DVL_RANGE: 20
MAX_DEPTH: 35
MIN_DEPTH: 5
START_DEPTH: 2
NOISE_DEPTH: 2
MOVEMENT_MAG: 2
HIGH_ANGLE: 15
LOW_ANGLE: 2
ANGLE_CHANGE: 3</td>
</tr>
<tr><td>altitude</td></tr>
<tr><td>pitch</td></tr>
<tr><td>fin position (-1 down, +1 up, 0 level)</td></tr>
<tr><td>Fig. 5-22</td><td>belief distribution of states</td></tr>
</table>

<table>
<tr>
<td rowspan="5">Section 5.1.4.2
shallow water with +40% increase in uncertainty for the probability of transitions function
T(s’ | a, s)</td>
<td rowspan="4">Fig. 5-20</td>
<td>measured AUV depth, seabed depth</td>
<td rowspan="5">#This is a simple AUV that only has a depth system
AUV_NAME: AUV_Shallow
MISSION_TIME: 10000
NOISE: 0

#subsystem depth
DEPTH_FILE:
/home/kat/Dropbox/Kathleen_Thesis/Analysis/Models/Dept
hLogs/basicshallow.log
ALTITUDE_MIN: 8
ALTITUDE_MAX: 12
DVL_RANGE: 20
MAX_DEPTH: 35
MIN_DEPTH: 5
START_DEPTH: 2
NOISE_DEPTH: 2
MOVEMENT_MAG: 2
HIGH_ANGLE: 15
LOW_ANGLE: 2
ANGLE_CHANGE: 3</td>
</tr>
<tr><td>altitude</td></tr>
<tr><td>pitch</td></tr>
<tr><td>fin position (-1 down, +1 up, 0 level)</td></tr>
<tr><td>Fig. 5-23</td><td>belief distribution of states</td></tr>
<tr>
<td rowspan="5">Section 5.1.4.3
shallow water with +10% increase in uncertainty for the probability of transitions function
T(s’ | a, s) and +10% increase in uncertainty for the probability of observation function
O(o | a, s)</td>
<td rowspan="4">Fig. 5-24</td>
<td>measured AUV depth, seabed depth</td>
<td rowspan="5">#This is a simple AUV that only has a depth system
AUV_NAME: AUV_Shallow
MISSION_TIME: 10000
NOISE: 0

#subsystem depth
DEPTH_FILE:
/home/kat/Dropbox/Kathleen_Thesis/Analysis/Models/Dept
hLogs/basicshallow.log
ALTITUDE_MIN: 8
ALTITUDE_MAX: 12
DVL_RANGE: 20
MAX_DEPTH: 35
MIN_DEPTH: 5
START_DEPTH: 2
NOISE_DEPTH: 2
MOVEMENT_MAG: 2
HIGH_ANGLE: 15
LOW_ANGLE: 2
ANGLE_CHANGE: 3</td>
</tr>
<tr><td>altitude</td></tr>
<tr><td>pitch</td></tr>
<tr><td>fin position (-1 down, +1 up, 0 level)</td></tr>
<tr><td>Fig. 5-25</td><td>belief distribution of states</td></tr>
</table>

<table>
<tr>
<td rowspan="5">Section 5.1.4.3
shallow water with +20% increase in uncertainty for the probability of transitions function
T(s’ | a, s) and +20% increase in uncertainty for the probability of observation function
O(o | a, s)</td>
<td rowspan="4">Fig. 5-24</td>
<td>measured AUV depth, seabed depth</td>
<td rowspan="5">#This is a simple AUV that only has a depth system
AUV_NAME: AUV_Shallow
MISSION_TIME: 10000
NOISE: 0

#subsystem depth
DEPTH_FILE:
/home/kat/Dropbox/Kathleen_Thesis/Analysis/Models/Dept
hLogs/basicshallow.log
ALTITUDE_MIN: 8
ALTITUDE_MAX: 12
DVL_RANGE: 20
MAX_DEPTH: 35
MIN_DEPTH: 5
START_DEPTH: 2
NOISE_DEPTH: 2
MOVEMENT_MAG: 2
HIGH_ANGLE: 15
LOW_ANGLE: 2
ANGLE_CHANGE: 3</td>
</tr>
<tr><td>altitude</td></tr>
<tr><td>pitch</td></tr>
<tr><td>fin position (-1 down, +1 up, 0 level)</td></tr>
<tr><td>Fig. 5-26</td><td>belief distribution of states</td></tr>
<tr>
<td rowspan="5">Section 5.1.4.3
shallow water with +40% increase in uncertainty for the probability of transitions function
T(s’ | a, s) and +40% increase in uncertainty for the probability of observation function
O(o | a, s)</td>
<td rowspan="4">Fig. 5-24</td>
<td>measured AUV depth, seabed depth</td>
<td rowspan="5">#This is a simple AUV that only has a depth system
AUV_NAME: AUV_Shallow
MISSION_TIME: 10000
NOISE: 0

#subsystem depth
DEPTH_FILE:
/home/kat/Dropbox/Kathleen_Thesis/Analysis/Models/Dept
hLogs/basicshallow.log
ALTITUDE_MIN: 8
ALTITUDE_MAX: 12
DVL_RANGE: 20
MAX_DEPTH: 35
MIN_DEPTH: 5
START_DEPTH: 2
NOISE_DEPTH: 2
MOVEMENT_MAG: 2
HIGH_ANGLE: 15
LOW_ANGLE: 2
ANGLE_CHANGE: 3</td>
</tr>
<tr><td>altitude</td></tr>
<tr><td>pitch</td></tr>
<tr><td>fin position (-1 down, +1 up, 0 level)</td></tr>
<tr><td>Fig. 5-27</td><td>belief distribution of states</td></tr>
</table>

<table>
<tr>
<td rowspan="4">Section 5.1.4.4
shallow water with additive noise (std = 2) +10% increase in uncertainty for the probability of observation function
O(o| a, s)</td>
<td rowspan="3">Fig. 5-28</td>
<td>measured AUV depth, seabed depth</td>
<td rowspan="4">#This is a simple AUV that only has a depth system
AUV_NAME: AUV_ShallowNoise
MISSION_TIME: 10000
NOISE: 1

#subsystem depth
DEPTH_FILE:
/home/kat/Dropbox/Kathleen_Thesis/Analysis/Models/DepthLogs/basicshallow.log
ALTITUDE_MIN: 8
ALTITUDE_MAX: 12
DVL_RANGE: 20
MAX_DEPTH: 35
MIN_DEPTH: 5
START_DEPTH: 2
NOISE_DEPTH: 2
MOVEMENT_MAG: 2
HIGH_ANGLE: 15
LOW_ANGLE: 2
ANGLE_CHANGE: 3</td>
<td rowspan="4"></td>
</tr>
<tr>
<td>AUV altitude</td>
</tr>
<tr>
<td>AUV pitch</td>
</tr>
<tr>
<td>fin mode (-1 nose-down, +1 nose-up, 0 level)</td>
</tr>
<tr>
<td>Fig. 5-29</td>
<td>belief distribution of states</td>
</tr>
<tr>
<td rowspan="4">Section 5.1.4.4
shallow water with additive noise (std = 2) +20% increase in uncertainty for the probability of observation function
O(| a, s)</td>
<td rowspan="3">Fig. 5-28</td>
<td>measured AUV depth, seabed depth</td>
<td rowspan="4">#This is a simple AUV that only has a depth system
AUV_NAME: AUV_ShallowNoise
MISSION_TIME: 10000
NOISE: 1

#subsystem depth
DEPTH_FILE:
/home/kat/Dropbox/Kathleen_Thesis/Analysis/Models/DepthLogs/basicshallow.log
ALTITUDE_MIN: 8
ALTITUDE_MAX: 12
DVL_RANGE: 20
MAX_DEPTH: 35
MIN_DEPTH: 5
START_DEPTH: 2
NOISE_DEPTH: 2
MOVEMENT_MAG: 2
HIGH_ANGLE: 15
LOW_ANGLE: 2
ANGLE_CHANGE: 3</td>
<td rowspan="4"></td>
</tr>
<tr>
<td>AUV altitude</td>
</tr>
<tr>
<td>AUV pitch</td>
</tr>
<tr>
<td>fin mode (-1 nose-down, +1 nose-up, 0 level)</td>
</tr>
<tr>
<td>Fig. 5-30</td>
<td>belief distribution of states</td>
</tr>
</table>

<table>
<tr>
<td rowspan="4">Section 5.1.4.4
shallow water with additive noise (std = 2) +40% increase in uncertainty for the probability of observation function
O(o | a, s)</td>
<td rowspan="3">Fig. 5-28</td>
<td>measured AUV depth, seabed depth</td>
<td rowspan="4">#This is a simple AUV that only has a depth system
AUV_NAME: AUV_ShallowNoise
MISSION_TIME: 10000
NOISE: 1

#subsystem depth
DEPTH_FILE:
/home/kat/Dropbox/Kathleen_Thesis/Analysis/Models/DepthLogs/basicshallow.log
ALTITUDE_MIN: 8
ALTITUDE_MAX: 12
DVL_RANGE: 20
MAX_DEPTH: 35
MIN_DEPTH: 5
START_DEPTH: 2
NOISE_DEPTH: 2
MOVEMENT_MAG: 2
HIGH_ANGLE: 15
LOW_ANGLE: 2
ANGLE_CHANGE: 3</td>
</tr>
<tr><td>altitude</td></tr>
<tr><td>pitch</td></tr>
<tr><td>fin position (-1 down, +1 up, 0 level)</td></tr>
<tr><td>Fig. 5-31</td><td>belief distribution of states</td></tr>
<tr>
<td rowspan="4">Section 5.1.4.5
shallow water with additive noise (std = 2) +10% increase in uncertainty for the probability of transitions function
T(s’ | a, s)</td>
<td rowspan="3">Fig. 5-32</td>
<td>Measured AUV depth, seabed depth</td>
<td rowspan="4">#This is a simple AUV that only has a depth system
AUV_NAME: AUV_ShallowNoise
MISSION_TIME: 10000
NOISE: 1

#subsystem depth
DEPTH_FILE:
/home/kat/Dropbox/Kathleen_Thesis/Analysis/Models/DepthLogs/basicshallow.log
ALTITUDE_MIN: 8
ALTITUDE_MAX: 12
DVL_RANGE: 20
MAX_DEPTH: 35
MIN_DEPTH: 5
START_DEPTH: 2
NOISE_DEPTH: 2
MOVEMENT_MAG: 2
HIGH_ANGLE: 15
LOW_ANGLE: 2
ANGLE_CHANGE: 3</td>
</tr>
<tr><td>altitude</td></tr>
<tr><td>pitch</td></tr>
<tr><td>fin position (-1 down, +1 up, 0 level)</td></tr>
<tr><td>Fig. 5-33</td><td>belief distribution of states</td></tr>
</table>

<table>
<tr>
<td rowspan="4">Section 5.1.4.5
shallow water with additive noise (std = 2) +20% increase in uncertainty for the probability of transitions function
T(s’ | a, s)</td>
<td rowspan="3">Fig. 5-32</td>
<td>measured AUV depth, seabed depth</td>
<td rowspan="4">#This is a simple AUV that only has a depth system
AUV_NAME: AUV_ShallowNoise
MISSION_TIME: 10000
NOISE: 1

#subsystem depth
DEPTH_FILE:
/home/kat/Dropbox/Kathleen_Thesis/Analysis/Models/DepthLogs/basicshallow.log
ALTITUDE_MIN: 8
ALTITUDE_MAX: 12
DVL_RANGE: 20
MAX_DEPTH: 35
MIN_DEPTH: 5
START_DEPTH: 2
NOISE_DEPTH: 2
MOVEMENT_MAG: 2
HIGH_ANGLE: 15
LOW_ANGLE: 2
ANGLE_CHANGE: 3</td>
<td rowspan="4"></td>
</tr>
<tr>
<td>altitude</td>
</tr>
<tr>
<td>pitch</td>
</tr>
<tr>
<td>fin position (-1 down, +1 up, 0 level)</td>
</tr>
<tr>
<td>Fig. 5-34</td>
<td>belief distribution of states</td>
</tr>
<tr>
<td rowspan="4">Section 5.1.4.5
shallow water with additive noise (std = 2) +40% increase in uncertainty for the probability of transitions function
T(s’ | a, s)</td>
<td rowspan="3">Fig. 5-32</td>
<td>measured AUV depth, seabed depth</td>
<td rowspan="4">#This is a simple AUV that only has a depth system
AUV_NAME: AUV_ShallowNoise
MISSION_TIME: 10000
NOISE: 1

#subsystem depth
DEPTH_FILE:
/home/kat/Dropbox/Kathleen_Thesis/Analysis/Models/DepthLogs/basicshallow.log
ALTITUDE_MIN: 8
ALTITUDE_MAX: 12
DVL_RANGE: 20
MAX_DEPTH: 35
MIN_DEPTH: 5
START_DEPTH: 2
NOISE_DEPTH: 2
MOVEMENT_MAG: 2
HIGH_ANGLE: 15
LOW_ANGLE: 2
ANGLE_CHANGE: 3</td>
<td rowspan="4"></td>
</tr>
<tr>
<td>altitude</td>
</tr>
<tr>
<td>pitch</td>
</tr>
<tr>
<td>fin position (-1 down, +1 up, 0 level)</td>
</tr>
<tr>
<td>Fig. 5-35</td>
<td>belief distribution of states</td>
</tr>
</table>

<table>
<tr>
<td rowspan="4">Section 5.1.4.6
shallow water with additive noise (std = 2) +10% increase in uncertainty for the probability of transitions function
T(s’ | a, s) and +10% increase in uncertainty for the probability of observation function
O(o | a, s)</td>
<td rowspan="3">Fig. 5-36</td>
<td>measured AUV depth, seabed depth</td>
<td rowspan="4">#This is a simple AUV that only has a depth system
AUV_NAME: AUV_ShallowNoise
MISSION_TIME: 10000
NOISE: 1

#subsystem depth
DEPTH_FILE:
/home/kat/Dropbox/Kathleen_Thesis/Analysis/Models/DepthLogs/basicshallow.log
ALTITUDE_MIN: 8
ALTITUDE_MAX: 12
DVL_RANGE: 20
MAX_DEPTH: 35
MIN_DEPTH: 5
START_DEPTH: 2
NOISE_DEPTH: 2
MOVEMENT_MAG: 2
HIGH_ANGLE: 15
LOW_ANGLE: 2
ANGLE_CHANGE: 3</td>
<td rowspan="4"></td>
</tr>
<tr>
<td>altitude</td>
</tr>
<tr>
<td>pitch</td>
</tr>
<tr>
<td>fin position (-1 down, +1 up, 0 level)</td>
</tr>
<tr>
<td>Fig. 5-37</td>
<td>belief distribution of states</td>
</tr>
<tr>
<td rowspan="4">Section 5.1.4.6
shallow water with additive noise (std = 2) +20% increase in uncertainty for the probability of transitions function
T(s’ | a, s) and +20% increase in uncertainty for the probability of observation function
O(o | a, s)</td>
<td rowspan="3">Fig. 5-36</td>
<td>measured AUV depth, seabed depth</td>
<td rowspan="4">#This is a simple AUV that only has a depth system
AUV_NAME: AUV_ShallowNoise
MISSION_TIME: 10000
NOISE: 1

#subsystem depth
DEPTH_FILE:
/home/kat/Dropbox/Kathleen_Thesis/Analysis/Models/DepthLogs/basicshallow.log
ALTITUDE_MIN: 8
ALTITUDE_MAX: 12
DVL_RANGE: 20
MAX_DEPTH: 35
MIN_DEPTH: 5
START_DEPTH: 2
NOISE_DEPTH: 2
MOVEMENT_MAG: 2
HIGH_ANGLE: 15
LOW_ANGLE: 2
ANGLE_CHANGE: 3</td>
<td rowspan="4"></td>
</tr>
<tr>
<td>altitude</td>
</tr>
<tr>
<td>pitch</td>
</tr>
<tr>
<td>fin position (-1 down, +1 up, 0 level)</td>
</tr>
<tr>
<td>Fig. 5-38</td>
<td>belief distribution of states</td>
</tr>
</table>

<table>
<tr>
<td rowspan="4">Section 5.1.4.6
shallow water with additive noise (std = 2) +40% increase in uncertainty for the probability of transitions function
T(s’ | a, s) and +40% increase in uncertainty for the probability of observation function
O(o | a, s)</td>
<td rowspan="3">Fig. 5-36</td>
<td>measured AUV depth, seabed depth</td>
<td rowspan="4">#This is a simple AUV that only has a depth system
AUV_NAME: AUV_ShallowNoise
MISSION_TIME: 10000
NOISE: 1

#subsystem depth
DEPTH_FILE:
/home/kat/Dropbox/Kathleen_Thesis/Analysis/Models/DepthLogs/basicshallow.log
ALTITUDE_MIN: 8
ALTITUDE_MAX: 12
DVL_RANGE: 20
MAX_DEPTH: 35
MIN_DEPTH: 5
START_DEPTH: 2
NOISE_DEPTH: 2
MOVEMENT_MAG: 2
HIGH_ANGLE: 15
LOW_ANGLE: 2
ANGLE_CHANGE: 3</td>
</tr>
<tr>
<td>AUV altitude</td>
</tr>
<tr>
<td>AUV pitch</td>
</tr>
<tr>
<td>fin position(-1 nose-down, +1 nose-up, 0 level)</td>
</tr>
<tr>
<td>Fig. 5-39</td>
<td>belief distribution of states</td>
</tr>
</table>

thesis section - power sub-system simulations	figure	generated time histories	Initialization file for AUV simulator

Section 5.2.1 sufficient capacity with 8000 joules stored.	Fig. 5-40	energy capacity	`#This is a simple AUV that only has a depth system`
			`AUV_NAME: AUV_Power_GoodCapacity`
			`MISSION_TIME: 10000`
			`NOISE: 0`
			`CASCADE_FAILURE: 0`
			
			`#subsystem power`
			`POWER_FILE:`
			`/home/kat/Dropbox/Kathleen_Thesis/Analysis/Models/PowerLogs`
			`/power_basic.log`
			`POWER_STORED: 8000`
			`POWER_RATES: 0.4 0.6`
			`LOW_POWER_MODE: 0.75`

		time left	
		power-management mode (2 – abort, 1 *power-saving* mode, 0 normal usage)	
		energy usage	
	Fig. 5-41	belief distribution of states	

Section 5.2.2 insufficient capacity with 5750 joules stored (approximate 71% of section 5.2.1).	Fig. 5-42	energy capacity	`#This is a simple AUV that only has a depth system`
			`AUV_NAME: AUV_lowcapacity`
			`MISSION_TIME: 10000`
			`NOISE: 0`
			
			`#subsystem power`
			`POWER_FILE:`
			`/home/kat/Dropbox/Kathleen_Thesis/Analysis/Models/PowerLogs`
			`/power_basic.log`
			`POWER_STORED: 5750`
			`POWER_RATES: 0.4 0.6`
			`LOW_POWER_MODE: 0.75`

		time left	
		power-management mode (2 – abort, 1- *power-saving* mode, 0 normal usage)	
		energy usage	
	Fig. 5-43	belief distribution of states	

<table>
<tr>
<td rowspan="5">Section 5.2.3
critical capacity with 5000 joules stored (approximate 62% of section 5.2.1).</td>
<td rowspan="4">Fig. 5-44</td>
<td>energy capacity</td>
<td rowspan="5">#This is a simple AUV that only has a depth system
AUV_NAME: AUV_Critical
MISSION_TIME: 10000
NOISE: 0

#subsystem power
POWER_FILE:
/home/kat/Dropbox/Kathleen_Thesis/Analysis/Models/PowerLogs
/power_basic.log
POWER_STORED: 5000
POWER_RATES: 0.4 0.6
LOW_POWER_MODE: 0.75</td>
</tr>
<tr>
<td>time left</td>
</tr>
<tr>
<td>power-management mode (2 – abort, 1- power-saving mode, 0 normal usage)</td>
</tr>
<tr>
<td>energy usage</td>
</tr>
<tr>
<td>Fig. 5-45</td>
<td>belief distribution of states</td>
</tr>
</table>

thesis section - power and depth sub-system simulations	figure	generated time histories	Initialization file for AUV simulator
5.3.1 independent model, sub-systems act independently using shallow-waters from 5.1.1.1 and power capacity 8000 (approximate 100% of 3.2.1)	Fig. 5-46	measured AUV depth, seabed depth	#This is a simple AUV that has a depth system and power system AUV_NAME: AUV_PowerDepth_Basic MISSION_TIME: 6000 NOISE: 0 #subsystem depth DEPTH_FILE: /home/kat/Dropbox/Kathleen_Thesis/Analysis/Models/DepthLogs/basicshallow.log ALTITUDE_MIN: 8 ALTITUDE_MAX: 12 DVL_RANGE: 20 MAX_DEPTH: 35 MIN_DEPTH: 5 START_DEPTH: 2 NOISE_DEPTH: 2 MOVEMENT_MAG: 2 HIGH_ANGLE: 15 LOW_ANGLE: 2 ANGLE_CHANGE: 3 #subsystem power POWER_FILE: /home/kat/Dropbox/Kathleen_Thesis/Analysis/Models/PowerLogs/power_basic63.log POWER_STORED: 8000 POWER_RATES: 0.4 0.6 LOW_POWER_MODE: 0.75
		altitude	
		pitch	
		fin position(-1 down, +1 up, 0 level)	
	Fig. 5-47	energy capacity	
		time left	
		power-management mode (2 – abort, 1- *power-saving* mode, 0 normal usage)	
		energy usage	

<table>
<tr>
<td rowspan="8">5.3.2
dependent model, depth sub-system is dependent on power-management. When power capacity is low vehicle will rise in water column, when very low and critical vehicle will surface. Uses shallow-waters from 5.1.1.1 and power capacity 5200 (approximate 65% of 3.2.1)</td>
<td rowspan="4">Fig. 5-48</td>
<td>measured AUV depth, seabed depth</td>
<td rowspan="8">#This is a simple AUV that has a depth system and power system
AUV_NAME: AUV_PowerDepth_Integrated
MISSION_TIME: 10000
NOISE: 0
ABORT_AT_COMMAND: 1
CASCADE_FAILURE: 0

#subsystem depth
DEPTH_FILE:
/home/kat/Dropbox/Kathleen_Thesis/Analysis/Models/DepthLogs/basicshallow.log
ALTITUDE_MIN: 8
ALTITUDE_MAX: 12
DVL_RANGE: 20
MAX_DEPTH: 35
MIN_DEPTH: 5
START_DEPTH: 2
NOISE_DEPTH: 2
MOVEMENT_MAG: 2
HIGH_ANGLE: 15
LOW_ANGLE: 2
ANGLE_CHANGE: 3

#subsystem power
POWER_FILE:
/home/kat/Dropbox/Kathleen_Thesis/Analysis/Models/PowerLogs/power_basic63.log
POWER_STORED: 5200
POWER_RATES: 0.4 0.8
LOW_POWER_MODE: 0.75</td>
</tr>
<tr><td>altitude</td></tr>
<tr><td>pitch</td></tr>
<tr><td>fin position(-1 down, +1 up, 0 level)</td></tr>
<tr>
<td rowspan="4">Fig. 5-49</td>
<td>energy capacity</td>
</tr>
<tr><td>time left</td></tr>
<tr><td>power-management mode (2 – abort, 1- power-saving mode, 0 normal usage)</td></tr>
<tr><td>energy usage</td></tr>
</table>

<table>
<tr>
<td rowspan="8">5.3.3
cascade failure model, depth sub-system is dependent on power-management. When power capacity is low vehicle will rise in water column, when very low and critical vehicle will surface; however, altitude and depth measurements will be loss. Uses shallow-waters from 5.1.1.1 and power capacity 5200 (approximate 65% of 3.2.1)</td>
<td rowspan="4">Fig. 5-50</td>
<td>measured AUV depth, seabed depth</td>
<td rowspan="8">
<pre>
#This is a simple AUV that has a depth system and power system
AUV_NAME: AUV_PowerDepth_Integrated
MISSION_TIME: 10000
NOISE: 0
ABORT_AT_COMMAND: 1
CASCADE_FAILURE: 1

#subsystem depth
DEPTH_FILE: /home/kat/Dropbox/Kathleen_Thesis/Analysis/Models/DepthLogs/basicshallow.log
ALTITUDE_MIN: 8
ALTITUDE_MAX: 12
DVL_RANGE: 20
MAX_DEPTH: 35
MIN_DEPTH: 5
START_DEPTH: 2
NOISE_DEPTH: 2
MOVEMENT_MAG: 2
HIGH_ANGLE: 15
LOW_ANGLE: 2
ANGLE_CHANGE: 3

#subsystem power
POWER_FILE: /home/kat/Dropbox/Kathleen_Thesis/Analysis/Models/PowerLogs/power_basic63.log
POWER_STORED: 5200
POWER_RATES: 0.4 0.8
LOW_POWER_MODE: 0.75
</pre>
</td>
</tr>
<tr><td>altitude</td></tr>
<tr><td>pitch</td></tr>
<tr><td>fin position(-1 down, +1 up, 0 level)</td></tr>
<tr><td rowspan="4">Fig. 5-51</td><td>energy capacity</td></tr>
<tr><td>time left</td></tr>
<tr><td>power-management mode (2 – abort, 1- power-saving mode, 0 normal usage)</td></tr>
<tr><td>energy usage</td></tr>
</table>

<table>
<tr>
<td rowspan="8">5.3.3
cascade failure model, depth sub-system is dependent on power-management. When power capacity is low vehicle will rise in water column, when very low and critical vehicle will surface; however, altitude and depth measurements will be loss. Uses a reversed shallow-waters from 5.1.1.1 to demonstrate the rise in the AUV even while the seabed descend and power capacity 5200 (approximate 65% of 3.2.1)</td>
<td rowspan="4">Fig. 5-52</td>
<td>measured AUV depth, seabed depth</td>
<td rowspan="8">#This is a simple AUV that has a depth system and power system
AUV_NAME: AUV_PowerDepth_Integrated
MISSION_TIME: 10000
NOISE: 0
ABORT_AT_COMMAND: 1
CASCADE_FAILURE: 1

#subsystem depth
DEPTH_FILE:
/home/kat/Dropbox/Kathleen_Thesis/Analysis/Models/DepthLogs/reversebasicshallow.log
ALTITUDE_MIN: 8
ALTITUDE_MAX: 12
DVL_RANGE: 20
MAX_DEPTH: 35
MIN_DEPTH: 5
START_DEPTH: 2
NOISE_DEPTH: 2
MOVEMENT_MAG: 2
HIGH_ANGLE: 15
LOW_ANGLE: 2
ANGLE_CHANGE: 5

#subsystem power
POWER_FILE:
/home/kat/Dropbox/Kathleen_Thesis/Analysis/Models/PowerLogs/power_basic63.log
POWER_STORED: 5200
POWER_RATES: 0.4 0.8
LOW_POWER_MODE: 0.75</td>
</tr>
<tr><td>altitude</td></tr>
<tr><td>pitch</td></tr>
<tr><td>fin position(-1 down, +1 up, 0 level)</td></tr>
<tr>
<td rowspan="4">Fig. 5-53</td>
<td>energy capacity</td>
</tr>
<tr><td>time left</td></tr>
<tr><td>power-management mode (2 – abort, 1- power-saving mode, 0 normal usage)</td></tr>
<tr><td>energy usage</td></tr>
</table>

APPENDIX D - ACRONYMS

- **AUV** autonomous underwater vehicle
- **ASV** autonomous surface vehicle
- **BN** Bayesian network
- **DAG** directed acyclic graph
- **DBN** dynamic Bayesian networks
- **DN** decision networks
- **DDN** dynamic decision networks
- **DVL** Doppler velocity log
- **FDIR** fault detection identification and recovery
- **GPS** global positioning system
- **MDP** Markov decision process
- **PGM** probabilistic graphical model
- **POMDP** partially observable Markov decision process
- **PBPI** point-based policy iteration
- **PBVI** point-based value iteration
- **ROS** robot operating system
- **ROV** remotely operated vehicle
- **UUV** unmanned underwater vehicle
- **UAV** unmanned aerial vehicle

APPENDIX E - NOMENCLATURE

- O: probability of making an observation function $O(o \mid s, a)$
- R: reward function $R(s, a)$
- T: probability of transitioning between states function $T(s' \mid s, a)$
- V: value function $V(s, a)$
- a: finite set of actions that the AUV can perform
- b: belief distribution of across all possible states for which state the vehicle is in
- h: horizon
- o: finite set of observations the AUV can make
- s: finite number of AUV and environment states
- η: normalizing constant of belief statement
- π: policy of POMDP
- Υ: discount
- $\perp$: Independence