A Bayesian Approach to Risk-Based Autonomy for a Robotic System Executing a Sequence of Independent Tasks

Sverre Velten Rothmund, Christoph Alexander Thieme, Ingrid Bouwer Utne, Tor Arne Johansen

Abstract—Enabling higher levels of autonomy requires an increased ability to identify and handle internal faults and unforeseen changes in the environment. This work presents an approach to improve this ability for a robotic system that is executing a series of independent tasks, such as inspection, sampling, or intervention, at different locations. A dynamic decision network (DDN) is used to infer the presence of internal faults and the state of the environment based on the available measurements. This knowledge is used to evaluate the risk of executing the current task, which is used to evaluate whether the task should be executed or skipped and whether maintenance actions are needed. Evaluating past states given new information is used to identify skipped tasks that should be revisited. The proposed approach is implemented for a drone tasked with contact-based ultrasound inspection of an industrial facility. The drone is able to successfully distinguish between different internal faults and adverse environmental states and act accordingly. The system makes risk-informed decisions based on uncertain knowledge, enabling it to minimize the time usage while minimizing the potential of harming the drone and maximizing mission completion.

I. INTRODUCTION

HIGHLY automatic or autonomous mission executions have advantages for reducing costs [1], improving performance [2], increasing safety [3], and enabling new types of operations [3], [4]. Examples of

This work has been submitted to the IEEE for possible publication. Copyright may be transferred without notice, after which this version may no longer be accessible.

This work was done as a part of the UNLOCK project at the Norwegian University of Science and Technology in collaboration with Scout Drone Inspection. The project is funded through the Research Council of Norway, project number 274441, and through the Research Council of Norway Center of Excellence, Center for Autonomous Marine Operations and Systems (AMOS), project number 223254.

S. V. Rothmund and T. A. Johansen are with the Department of Engineering Cybernetics, Norwegian University of Science and Technology, Trondheim, Norway (e-mail: sverre.v.rothmund@ntnu.no; tor.arne.johansen@ntnu.no).

C. A. Thieme and I. B. Utne are with the Department of Marine Technology, Norwegian University of Science and Technology, Trondheim, Norway (e-mail: christoph.thieme@ntnu.no; ingrid.b.utne@ntnu.no).

such systems include autonomous underwater vehicles, dynamic positioning systems for ships, and autopilots. Today's systems often rely on human operators to monitor them and to manually intervene if necessary [2], [5], [6]. Developing autonomous robotic systems that can operate without direct human supervision can enable a wider range of missions, such as underwater [4] and space [7] operations, where communication is limited, and long-term [8] and multiagent operations [2] that are economically infeasible to continuously and directly monitor.

For a system to operate without direct human supervision, it must evaluate the situation and handle deviations from normal operation [9]. Hagen et al. [6] argued that a system's "ability to sense, interpret and act upon unforeseen changes in the environment and the [system] itself" is vital for achieving a high level of autonomy. These articles [6], [9] have highlighted the need for an autonomous system to understand and act based on its own state and the state of the environment. Based on this need, four requirements are proposed. The first two consider situational evaluation, and the last two consider acting based on this evaluation:

- 1) The system must be able to identify the state of the environment during execution.
- 2) The system must be able to identify internal faults during execution.
- 3) The system must act proactively to avoid failures.
- 4) The system must act to minimize the consequences of identified faults.

The purpose of this article is to develop a risk-based decision system that fulfills these requirements for a robotic system that is executing a sequence of independent tasks. These tasks can, for example, be inspection, sampling, or intervention by a robot at different locations. This article focuses on high-level decision-making, considering how and if a task should be executed, or if maintenance actions, such as repairing or servicing the system, should be executed instead. It does not consider emergency fault handling.

The environment's effect on a system and the occurrence of a failure are often connected with uncertainty, making it necessary to consider the risk of a task or operation. Risk can be defined as the effect of uncertainty on objectives [10]. Bayesian belief networks (BBNs) model probabilistic relations, making them suitable to model risk. This capability has been demonstrated for evaluating the risk before [11]–[14] and during [15], [16] an operation. BBNs can be used to proactively avoid failure (requirement 3) as demonstrated in Bremnes et al. [17] for autonomous under-ice operations. And can be used reactively to minimize the consequence of failures (requirement 4) as demonstrated in Coombes et al. [18] where a BBN was used to evaluate the risk of emergency landing location for an unmanned aerial vehicle. These works [17], [18] demonstrated a fulfillment of requirements 3 and 4, but did not consider the identification of the state of the environment and internal faults during execution (requirements 1 and 2)

A robotic system acting in an uncertain environment must have sensors that enable it to measure the environmental state and the system's own state. Most real-world systems and environments are too complex to perfectly measure all relevant information. Instead, this information can be inferred by using a model. BBNs have been used to identify the presence of faults [19] and their root causes [20] based on available system information. The system dynamics can be modeled with a dynamic Bayesian network (DBN), which enables the model to incorporate information over time to infer the system state [19], [21], [22]. Furthermore, the decisions of the system can be included in the network, which is referred to as a dynamic decision network (DDN). DDNs are extensively used in educational systems [23]–[25] and dialog systems [26], [27] to better understand the state of the user. The user can in these works [23]–[27] be considered as the environmental state that must be inferred. The literature presented in this paragraph [19]–[27] have demonstrated an ability to identify internal faults and the state of the environment, thereby fulfilling requirements 1 and 2.

The previously presented literature demonstrates that Bayesian models can be used to proactively avoid failures, to reactively minimize consequences, and to identify internal faults and the state of the environment. They fulfill parts of the proposed requirements, but not all at the same time. Codetta-Raitari and Portinale [28] fulfilled all four requirements with a DBN model. Their work focused on monitoring the state of the system and environment and executing preventive and reactive recovery actions when needed. In contrast, the present work focuses on decision-making during a mission using

a DDN. Moreover, the present work does not model specific scenarios; instead, it models the possible ways task execution can be affected by the state of the environment and internal faults. This makes the modeling process simpler, as one does not need to explicitly model all possible faults, only how the faults affect the system.

In many applications, robotic systems are forced to make decisions based on incomplete and uncertain information and thus cannot avoid erroneous or suboptimal decisions. A system should identify these decisions and act to minimize their consequences. Through utilizing new information, past states can be evaluated with higher confidence, thereby revealing suboptimal past decisions. None of the surveyed works considered evaluating past states to counteract the consequences of erroneous or suboptimal past actions.

This article proposes a Bayesian approach for fulfilling the four defined requirements for a robotic system executing a sequence of independent tasks. A top-down approach for building a DDN is presented, which focuses on only modeling states that can be distinguished based on the observations and actions of the system. The DDN is used to evaluate the risk of executing the current task based on its inferred knowledge of internal faults and the state of the environment. A policy is proposed that evaluates if and how the task should be executed and whether maintenance actions should be executed. Past tasks are reevaluated when new information is gathered to consider if the system should reattempt previously failed or skipped tasks. To demonstrate the proposed method, a case study on a multirotor inspection drone tasked with contact-based ultrasound thickness measurements is presented.

The contribution of this paper is twofold. First, an approach for building and using a DDN that enables the system to fulfill the proposed requirements is presented and demonstrated with a case study. Second, the usefulness of evaluating previous tasks, thereby making the system able to correct mistakes, is demonstrated.

The rest of the paper is structured as follows: Section II presents a method for building the DDN before this method is applied to the case study in section III. Simulation results from the case study are presented in section IV. Section V discusses the proposed method in light of the results from the case study, and a conclusion is presented in section VI.

II. METHOD

This section presents the proposed method for developing and using a DDN to make high-level decisions for a robotic system executing a sequence of independent similar tasks. Tasks are considered independent when

Fig. 1. Example of a simple DDN with the color coding used in this article. The objective of the task is shown in orange, the state of the failure causes in blue, actions in gray, and measurements in green. Measurements are available for the current time step (t) and previous time steps but not future time steps.

”no task provides a necessary precondition for the fulfillment of another task” [29].

A DDN can be seen as a sequence of connected BBNs, one at each time step, as illustrated in Figure 1. BBNs are directed acyclic graphs where the arcs represent dependencies. Conditional probability tables (CPTs) are used in each node to quantify the probability that the nodes take on a particular state given the state of all nodes it depends on. Bayes theorem is used to evaluate the resulting belief that a node is in a particular state. The connections across time steps in the DDN are modeled in the same way as the connections within a time step. For more information on BBNs and DDNs, see [30], [31].

The method for building the DDN is presented in section II-A. The decision policy is presented in section II-B. It attempts to minimize the expected cost, considering the direct cost and the associated risk of executing different actions. Each decision that has to be made is modeled as a time step in the DDN. In each time step, the risk of executing the current task is evaluated.

A. Building the DDN

This article proposes building the DDN through a top-down approach. The approach aims at only modeling states that can be distinguished by the observations and actions of the robotic system. The following steps are used to build the DDN. Figure 2 shows an example of the resulting network.

- 1) Describe the operation and system.
- 2) Model relevant objectives.
- 3) Model failure causes.
- 4) Model measurable failure causes.
- 5) Model the condition of the failure causes.
- 6) Model dynamics.
- 7) Model measurements.
- 8) Quantification.

Step 1 - Describe the operation and system: To build the DDN, a description of the operation and the robotic system is needed. The operational description needs to define the tasks the system should execute and which actions the robotic system can perform. In this article, three types of actions are considered. First, execution actions that execute the tasks should be described. These are associated with a probability of achieving the goal of the task and may lead to losses, which are discussed further in step 2. The execution actions are associated with a direct cost and an indirect cost if a loss occurs. Second, maintenance actions are described that repair or mitigate internal faults in the robotic system. These actions are associated with a direct cost that the system must weigh against the advantage of maintaining the system. Third, the system must be able to skip the current task and continue on to the next task. By skipping the task, the goal of the task will be unfulfilled. The cost associated with not fulfilling the goal is weighed against the expected cost of attempting the task. How these trade-offs are made is discussed in detail in section II-B.

In the description of the robotic system, a description of the available sensors and information from different subsystems, such as a navigation system, is given.

Step 2 - Model relevant objectives: As risk is the ”effect of uncertainty on objectives” [10], the relevant objectives must be identified to make risk-based decisions. Two types of objectives are considered: achieving the task goal and avoiding hazards. A hazard is a set of adverse conditions that can lead to a loss [32]. Damages to the robotic system is one example of a loss. Relevant hazards can be identified through different risk analysis methods, such as preliminary hazard analysis (PHA) [33] or system theoretic process analysis (STPA) [32]. A node is introduced in the DDN for every goal and hazard. An example is shown on line L1 in Figure 2 for a case with one goal and one hazard node. These nodes have two states, that indicating whether the objective will be fulfilled on this execution attempt.

Step 3 - Model failure causes: Different failure causes, such as faults in the robotic system and adverse environmental states, can prevent the objectives from being achieved. Not achieving an objective is considered a failure. The failure causes can be identified with a risk analysis; see [32], [33]. Intermediate nodes are introduced to group failure causes that affect the same set of objectives. These are shown on line L2 in Figure 2. These nodes are only introduced if there exist failure causes that affect this combination of objectives.

All failure causes that affect the same objective nodes and that are affected by the same set of actions are grouped together in the network. This is achieved by

Fig. 2. Generic DDN network built using the proposed method. Objectives are shown in orange, intermediate nodes in light blue, condition nodes in dark blue, and measurements in green. The different lines, L1-L5, are introduced in section II-A steps 2 to 5. The dotted arrows indicate connections between time steps and are described in step 6. Measurements are described in step 7.

introducing an intermediate node for each group. An example is given on line L3 of Figure 2, where action A and action B affect different failure causes. In the example, all "failure causes causing hazard" are affected by the same actions, making the introduction of new nodes unnecessary.

Step 4 - Model measurable failure causes: Measurements may be available that provide information on some of the failure causes represented by the nodes defined in the previous steps. A new intermediate node representing the measurable failure cause is introduced. In addition, an intermediate node representing the failure causes that are not measurable is introduced. An example is given on line L4 in Figure 2, where a measurement exists for one node. Before the measurement itself can be included in the model, the condition of the failure causes must be modeled.

Step 5 - Model the condition of the failure causes

The nodes introduced in steps 2 to 4 have two states indicating whether a failure cause will prevent the fulfillment of the objectives. It can be useful to model the state of the failure causes with higher fidelity. This is accomplished by introducing the condition nodes shown on line L5 in Figure 2. The state of the condition nodes can, for example, reflect the probability that the failure cause will lead to a failure. Alternatively, a physical quantity can be used if its relationship to the failure cause is known. This can, for example, be the state of components that wear out or deplete over time and have a higher failure probability when they are significantly worn or depleted, such as batteries.

Step 6 - Model dynamics: The condition nodes introduced in step 5 are connected between time steps, as indicated with the dotted arrow in Figure 2. The CPT of the condition node specifies the probability that the condition transitions from one state to another given the

Fig. 3. Example of how different task-specific nodes can be connected to the rest of the network at different time steps. "Network L1-L4" refers to line L1 to L4 in Figure 2. Task 0 is connected to the rest of the network at time steps 0, 1, and 3, while task 1 is connected at time step 2.

choice of action. This probability can also depend on whether the objectives were fulfilled, as illustrated with the dotted arrow to the right in Figure 2. Attempting to execute a task may degrade the condition, while maintenance actions may improve it.

Some conditions might be different for each independent task. These conditions can be modeled by introducing an instance of the condition node for each independent task in the operation. The condition nodes are then connected to the DDN at the time steps where their corresponding task is considered. An example of this is given in Figure 3.

Step 7 - Model measurements: In step 4, nodes are introduced to model the failure causes that can be measured before a task is executed. In this step, the measurements themselves are introduced. These should be influenced by the condition of the measured failure causes, as shown in green at the bottom of Figure 2.

After a task execution is attempted, measurements of whether the objective is met may be available. This information enables the system to infer knowledge about failure causes that cannot be measured directly. These measurements are shown in green at the top of Figure 2.

Step 8 - Quantification: The CPTs for the nodes that are not influenced by condition nodes can be modeled with Boolean logic [30]. These define for which combination of "failure" inputs the state of the current node will be "failure". The CPTs of the nodes influenced by a condition node translate the condition into a probability that the underlying failure causes will prevent the fulfillment of the objectives on this execution attempt.

For the condition nodes, the probability of transitioning from one state to another must be specified for all actions and outcomes of relevant objectives. Different expert judgment and data-driven approaches exist for CPT quantification [30], [34]. The initial probability that the condition is in a particular state should reflect how often this state is expected in the operation.

The CPTs of the measurement nodes give the probability of measuring the different values given the state of

the node that is measured [30]. For the measurement of the condition, the probability that the different condition states result in the different measured values is needed. For the measurements of the objectives, the probability of obtaining the different measured values when the objective is met or not is needed.

B. Decision policy

This section presents a policy for using the DDN to choose an appropriate action. The policy will typically be developed in parallel with the DDN. The policy combines the probability of not achieving the different objectives with their respective consequences such that risk-based decisions can be made. Three strategies consisting of one or multiple actions are considered: 1) move on to the next task, 2) attempt to execute the task once and then move on to the next task, or 3) execute a maintenance action before attempting task execution once and then moving on. The expected cost of each strategy is evaluated, and the first action of the cheapest strategy is executed. After executing the first action of the strategy, the optimal strategy is re-evaluated. If strategy 2 is chosen multiple times in a row, then the system executes the current task multiple times without moving to the next task.

The cost of strategy 1, moving on to the next task, depends on whether the goal of the current task is achieved or not. If the goal is achieved, then there is no cost associated with moving to the next task. If the goal is not achieved, then there is a cost, C_G , based on the consequence of not achieving the goal. This is shown in equation (1). More cases can be added if there can be a partial fulfillment of the goal.

Strategy 2 attempts to execute the task, which has a direct cost associated with it, $C(e)$, and may cause hazards. The cost of the hazards, given by the vector $C_H(e)$, are based on their consequences, which can vary with the choice of execution action. The probability of the hazard occurring is evaluated with the DDN and denoted by the vector P_H . Attempting a task execution might achieve the goal of the task. The probability of achieving the goal is evaluated with the DDN and denoted as P_G . If the goal is not achieved, then strategy 2 will additionally contain the cost of moving to the next task. The cost of strategy 2 is shown in equation (2). This cost is evaluated for all possible execution actions, e , that are applicable for the current task. The different execution actions can have different direct costs related to them, can have different costs associated with their consequences, and might evaluate different probabilities in the DDN.

Strategy 3 executes a maintenance action before attempting a task execution. The maintenance action can increase the probability of achieving the goal and reduce the probability of hazards occurring. The effect of the maintenance action is evaluated by setting it as the current action in the DDN and then simulating one step forward in time. The evaluated probability of hazards and goal achievement at the next time step in the DDN is then used to evaluate the cost of execution, $C_{2,m}(e)$. Additionally, the cost of the maintenance action itself must be included. This cost is often quite high but can improve multiple future task execution attempts. The maintenance cost, $C(m)$, is divided by the expected number of executions until maintenance is needed again, $N(m)$. The resulting cost is shown in equation 3 and should be evaluated for all combinations of maintenance (m) and execution (e) actions.

$$C_1 = \begin{cases} 0 & \text{If goal of current task is achieved} \\ C_G & \text{Else} \end{cases} \quad (1)$$

$$C_2(e) = C(e) + \mathbf{C}_H(e)^\top \mathbf{P}_H(e) + (1 - P_G(e))C_1 \quad (2)$$

$$C_3(m, e) = C(m)/N(m) + C_{2,m}(e) \quad (3)$$

Updating past states given new information might reduce the expected cost of re-attempting previously failed tasks. The system returns to a previously attempted task if the expected cost of executing the task plus the cost of returning to the previous task, C_{Ret} , is lower than the cost of omitting the task, C_1 , as shown in equation (4). A task is reattempted if the visit is warranted for any of the available execution actions. The closest previously attempted task in the sequence that is worth reattempting is attempted first.

$$C_{Ret} + C_2(e) < C_1 \quad (4)$$

Attempting a task before and after maintaining the system will enable the system to identify if the maintenance helped. To encourage this behavior, the policy always chooses an execution action if the current task has not been attempted and if the execution action is cheaper than the move action. If this is not the case, then the cheapest action is chosen.

III. CASE STUDY

In this section, the proposed method is applied to a multirotor drone tasked with industrial inspection. The case study is based on simulation.

TABLE I
NOTATION FOR THE DECISION POLICY

C_i	Expected cost of strategy i
e	An action that executes the current task
m	An action that maintains the system
C_G	Cost of not fulfilling the goal
$C(e)$	Direct cost of executing the task with action e
$C(m)$	Direct cost of executing maintenance action m
$\mathbf{C}_H(e)$	Cost of the potential hazards when executing e
$\mathbf{P}_H(e)$	Probability of potential hazards when executing e
$P_G(e)$	Probability of achieving the goal when executing e
$C_{2,m}(e)$	Cost of strategy 2 after maintenance action m is executed
$N(m)$	Mean number of executions between failures for maintenance action m
C_{Ret}	Cost of returning to a previous task

A. Building the network

Step 1 - Describe the operation and system: The operation consists of measuring metal surface thickness with an ultrasound sensor mounted on a multirotor drone [35]–[38]. The ultrasound sensor needs a thin gel layer and must be in stable physical contact with the surface of the inspection point for data to be gathered. A large number of points are to be inspected. Every inspection point is considered a task in the proposed method. The system can choose between two different ways of inspecting the surface of the inspection point: a normal inspection and a slower but safer inspection. For each inspection, a small amount of gel is dispensed from a tank mounted on the drone. One maintenance action available to the drone is to refill this tank. Another is to request a full maintenance check where an operator identifies and repairs all faults of the drone. The drone can skip inspection points that are deemed too costly to inspect autonomously. The costs associated with the actions in the case study are based on time use, making the goal of the decision policy to minimize the operation time. The skipped inspection points will have to be inspected manually by an operator, which will extend the operation time. The different costs are discussed in more detail in section III-B.

The drone is equipped with a lidar that can be used for detecting obstacles and for navigation. The navigation estimate can be used to monitor the trajectory of the drone.

Step 2 - Model relevant objectives: The goal of each task is to measure the surface thickness of the inspection points. As the drone operates in controlled industrial facilities consisting of metal surfaces without any human beings present, the most relevant loss is damage to the drone. A hazard that can lead to this loss is uncontrolled contact with a surface or other object. The two objectives, "gathering data" and "avoiding uncontrolled contact", are introduced as shown on line L1 in Figure

Fig. 4. The conditional probability tables and dependencies in the dynamic decision network for the inspection drone case study. "Wrt." stands for "with respect to". Some tables are intentionally left blank and are instead given in Tables II, III, IV, and V. Nodes containing a * refer to the table marked with a * shown on the bottom of the figure.

4.

Step 3 - Model failure causes: Some failure causes, such as an empty gel tank, rust or dirt stuck on the ultrasound sensor, or inspection surfaces covered with rust or dirt, can prevent data from being gathered. Other failure causes, such as a worn motor, poor navigation quality, or obstacles, can lead to uncontrolled contact in addition to preventing data from being gathered. No failure causes are considered that lead to uncontrolled contact but do not prevent data from being gathered. Two intermediate nodes are introduced: one for failure causes preventing data from being gathered, and the other for failure causes preventing controlled contact and data from being gathered.

The drone and the surface of the inspection point are affected by different actions. Executing an inspection may damage the drone, while the surface will not be affected. Similarly, maintaining the drone does not affect the surface. Moving to a new inspection point will change the surface but not affect the drone. A distinction between drone-related and surface-related nodes is therefore needed. Furthermore, the refill-gel action will only affect the gel level. The gel level does not affect the ability to avoid uncontrolled contact. Lines L2 and L3 in Figure 4 show the resulting nodes and their connections.

Step 4 - Model measurable failure causes: Before an inspection is executed, part of the surface-related failure causes that prevent controlled contact and data gathering is measured with the lidar. The lidar scan of the surface can reveal the presence of protruding obstacles that will prevent controlled contact and data gathering. Separate nodes are introduced for the failure causes that are measurable and those that are not, as shown on line L4 in Figure 4. This makes it possible to distinguish between failure causes that are measurable with the lidar and those that are not. The unmeasurable failure causes can, for example, be minor surface irregularities, such as welding joints, that are typically missed by lidar beams. Other measurements, such as navigation quality, could be included if desirable.

Step 5 - Model the condition of the failure causes

A slightly dirty or uneven surface, or a small fault in the drone, can reduce the likelihood of an inspection succeeding without hindering it completely. The states of the condition nodes, for all nodes except the "gel level" node, reflect the average frequency at which the respective conditions will cause a failure. These frequencies are discretized into different states, as shown in Figure 4. The DDN is used to evaluate the probability distributions over these states.

The state of the gel level indicates the amount of gel left. When the gel level approaches zero, an insufficient amount of gel might be deployed. This will prevent data from being gathered.

Step 6 - Model dynamics: There is a probability of introducing drone-related failure causes with each inspection attempt. The probability and severity will depend on whether uncontrolled contact occurred and whether a normal or safe inspection is performed. The gel level decreases with each inspection and is unaffected by the occurrence of an uncontrolled contact and the choice of inspection action.

Surface-related states are independent for each inspection point and are handled as discussed in section II-A step 6 and illustrated in Figure 3. The condition of the surface is assumed to be constant for each inspection point.

Step 7- Model measurements: The "surface suitability measurement" is introduced as shown at the bottom of Figure 4. The measurement is based on how flat the area around the inspection point seems based on the lidar scan. The suitability is discretized into four distinct states ranging from poor to perfect.

After an inspection is executed, a measurement of how the execution went is needed. Whether data is successfully gathered is readily available from the ultrasound thickness sensor. Whether an uncontrolled contact occurred cannot be directly measured. Instead, this can be inferred based on the trajectory conformity measure. This measurement is made by comparing the observed trajectory of the drone with the intended trajectory and identifying any deviations in position, velocity, and heading. This measurement is also discretized into four states ranging from poor to perfect based on data from the drone's navigation system.

Step 8 - Quantification: The CPTs are shown in Figure 4 and Tables II, III, IV, and V. The initial probability distribution over the state for the different surface condition and drone condition nodes can be found in Tables II and III. Tables IV and V show the probabilities of transitioning to a worse state for the different drone condition nodes when an inspection is attempted. The probability of transitioning to a better state is zero, as it is assumed that the drone cannot accidentally repair itself. When evaluating the new probability distribution based on Tables IV and V, transitions that lead to negative states are omitted, and the resulting distribution is normalized such that it sums to 100%. The condition of the drone can only be improved by maintenance actions. The refill gel action will set the gel level to 100%. The full maintenance action sets all drone-related nodes, including the gel level, to their initial distribution.

TABLE II

INITIAL PROBABILITY DISTRIBUTION FOR THE DIFFERENT SURFACE CONDITION NODES. "WRT." IS SHORT FOR "WITH RESPECT TO".

State	Node name		
	Surface condition wrt. data gathering	Unmeasurable surface condition wrt. controlled contact and data gathering	Measurable surface condition wrt. controlled contact and data gathering
0%	0.3	0.02	0.01
25%	0.05	0.005	0.01
50%	0.05	0.005	0.01
75%	0.2	0.005	0.07
100%	0.4	0.965	0.9

TABLE III

INITIAL PROBABILITY DISTRIBUTION FOR THE DIFFERENT CONDITION NODES RELATED TO THE DRONE. "WRT." IS SHORT FOR "WITH RESPECT TO".

State	Node name		
	Drone condition wrt. data gathering	Drone condition wrt. controlled contact and data gathering	Gel level
0%	0.03	0.005	0
25%	0.03	0.005	0
50%	0.04	0.005	0
75%	0.5	0.01	0
100%	0.4	0.975	1

TABLE IV

PROBABILITY OF TRANSITIONING FROM ONE STATE TO ANOTHER GIVEN THE CHOICE OF ACTION AND WHETHER AN UNCONTROLLED CONTACT IS AVOIDED. THE PROBABILITY DEPENDS ON THE DIFFERENCE IN VALUE OF THE NEW STATE COMPARED TO THE PREVIOUS ONE. THE PROBABILITY OF TRANSITIONING TO A BETTER STATE IS 0. "WRT." IS SHORT FOR "WITH RESPECT TO".

Node name	Drone condition wrt. measurement				Drone condition wrt. controlled contact and data gathering			
	Normal inspect		Safe inspect		Normal inspect		Safe inspect	
Action	Failure	Success	Failure	Success	Failure	Success	Failure	Success
Avoiding uncontrolled contact	Failure	Success	Failure	Success	Failure	Success	Failure	Success
Change in state	Probability of transition							
-100%	0.025	0.005	0.005	0.0025	0.025	0	0.005	0
-75%	0.025	0.005	0.005	0.0025	0.025	0	0.005	0
-50%	0.025	0.005	0.005	0.0025	0.025	0	0.005	0
-25%	0.025	0.005	0.005	0.0025	0.025	0	0.005	0
-0%	0.9	0.98	0.98	0.99	0.9	1	0.98	1

TABLE V

PROBABILITY OF REDUCING THE GEL LEVEL BY DIFFERENT AMOUNTS.

Reduction in gel level	Probability
-15%	0
-10%	0.1
-5%	0.8
-0%	0.1

time use. The expected time use of an uncontrolled impact is based on the expected time needed to repair the different degrees of damages that can occur times the likelihood of them occurring from an uncontrolled impact. The cost of not achieving the goal is based on the additional time used for a manual inspection.

IV. RESULTS

The numerical values presented in this section are selected to showcase the algorithm. They are not based on operational data.

B. Decision policy

The decision policy presented in section II-B is used in this case study. The values of the different parameters are shown in Table VI. These costs are based on the expected

This section presents four scenarios for the inspection drone case study. These scenarios were chosen because they demonstrate different aspects of the method and represent failures that are deemed likely to occur. The scenarios are simulated in Python using the SMILE [39] library to evaluate the DDN. In each scenario, the measurements that the system will experience are specified for each inspection point. The actions that the drone chooses and the drone's belief regarding relevant

TABLE VI
THE DIFFERENT COSTS USED IN THE DECISION POLICY IN THE CASE STUDY.

Description	Symbol	Cost [min]
Normal inspection cost	$C_E(e = \text{Normal Inspect})$	0.5
Safe inspection cost	$C_E(e = \text{Safe inspect})$	1
Cost of uncontrolled contact with normal inspection	$C_H(e = \text{Inspect})$	20
Cost of uncontrolled contact with safe inspection	$C_H(e = \text{Safe inspect})$	10
Refill gel cost	$C_M(m = \text{Refill})$	10
Full maintenance cost	$C_M(m = \text{Full maintenance})$	60
Mean number of executions between empty gel	$N(m = \text{Refill})$	20
Mean number of executions between full maintenance	$N(m = \text{Full maintenance})$	30
Failure to achieve goal cost	C_G	10
Return cost	C_{ret}	0.5

nodes throughout the simulation are presented. To keep the plots simple, only the probabilities that the nodes influenced by the condition nodes are in the failure state is shown.

A. Scenario 1

The first scenario demonstrates the system's ability to identify the cause of a failure based on the observed behavior and the ability to reason over past states. In this scenario, the ultrasound thickness sensor is not working, making all inspections end with no data being gathered but perfect trajectory conformity. Figures 5, 6, and 7 show how the belief of the system develops over time when new inspections are attempted and new information is gathered. Only the belief that drone-related and surface-related failure causes will prevent data gathering is shown. The rest of the nodes have a belief close to 0 throughout this example.

Figure 5 shows the behavior and beliefs of the drone when it is at the first inspection point. As seen in the table in Figure 5, the drone attempts to execute an inspection, which results in no data but perfect trajectory conformity. After the first inspection fails, the belief that surface-related failure causes prevent data from being gathered increases. The belief that drone-related failure causes prevent data gathering also increases but much less. The reason is that it is more probable that a single failed inspection is caused by the surface than the drone. This trend continues for the subsequent inspection attempts. At time step 3, the belief that surface-related failure causes will prevent data gathering is high enough to make the drone skip the current inspection point and move on to inspection point 1.

The broken line in Figure 5 shows the system's belief about past states evaluated at time step 3. The surface is modeled as static, making the belief about its past state equal to the newest belief. The updated belief for the drone-related failure causes preventing data gathering at

time step 0 is higher than it was initially but not as high as in time step 3. This is because there is a probability, although low, that the drone was damaged in one of the previous inspection attempts.

Figure 6 extends Figure 5 by showing 11 time steps. At inspection point 1, the same behavior is observed as at inspection point 0. After three failed attempts, the system skips this inspection point and moves on to inspection point 2. When evaluating the past states at time step 7, shown as the orange broken line, the probability that the drone-related failure causes are preventing data gathering has increased. The probability that a drone-related failure cause was introduced in one of the last time steps is low, making it probable that if there is a drone-related failure cause at time step 7, then there was probably one in the earlier time steps as well. Since the belief that drone-related failure causes prevented data gathering in time steps 0-3 has increased, the belief that the surface-related failure causes of inspection point 0 prevent data gathering decreases. This can be seen by the orange broken line being lower than the blue broken line at time step 0-3 for the surface-related failure causes.

After failing an inspection at inspection point 2 as well, the belief that the drone-related failure causes are preventing data gathering is high enough, making full maintenance worth the cost. After maintenance, the next inspection at time step 10 is successful. This shows that inspection point 2 can be inspected, which decreases the likelihood that there are surface-related failure causes preventing data gathering. This in turn increases the likelihood that drone-related failure causes prevents data gathering at time step 8. Evaluating backwards in time, the probability that drone-related failures caused the previous failed inspection attempts increases. This in turn reduces the belief that there are surface-related failure causes that prevent data gathering at inspection points 0 and 1, as shown with the green broken line in Figure 6.

When considering where to go next, the system will evaluate whether revisiting a previously visited inspec-

Fig. 5. Scenario 1. The table shows the measurements available before inspection, the choice of action, and the resulting measurements. The full line shows the belief of the drone that a failure cause is present at each time step. The broken line shows the belief of past states evaluated every time the drone moves to a new inspection point (IP).

Fig. 6. Scenario 1, an extension of Figure 5. The table shows the measurements available before inspection, the choice of action, and the resulting measurements. The full line shows the belief of the drone that a failure cause is present at each time step. The broken line shows the belief of past states evaluated every time the drone moves to a new inspection point (IP), which is marked with a vertical line. The color of the broken line indicates when the updated belief was evaluated.

Time step	Measured failure causes	Chosen action	Measured objective fulfillment
11	Surface suitability: Perfect	Return to IP1	
12	Surface suitability: Perfect	Normal inspect	Data: Yes, Trajectory conformity: Perfect
13	Surface suitability: Perfect	Return to IP0	
14	Surface suitability: Perfect	Normal inspect	Data: Yes, Trajectory conformity: Perfect
15	Surface suitability: Perfect	Move to IP3	

Fig. 7. Scenario 1, an extension of Figures 5 and 6. The table shows the measurements available before inspection, the choice of action, and the resulting measurements. The full line shows the belief of the drone that a failure cause is present at each time step. The broken line shows the belief of past states evaluated every time the drone moves to a new inspection point (IP), which is marked with a vertical line. The color of the broken line indicates when the updated belief was evaluated.

tion point is worth another inspection attempt. Since the belief that the surfaces on these inspection points caused the failures decreased, the system concludes that they are worth revising. As shown in Figure 7, the system first visits inspection point 1 again, where data are gathered successfully. This further strengthens the belief that there are no surface-related failure causes preventing data gathering at inspection point 1. This increases the belief that the failed data gathering attempt was related to the drone the last time inspection point 1 was visited. This in turn increases the belief that the drone-related failure causes were present the last time inspection point 0 was visited as well. This in turn decreases the belief that there are failure causes related to the surface at inspection point 0, as shown by the pink broken line in Figure 7. The system then returns to inspection point 0 and has a successful inspection before moving on to a new inspection point.

B. Scenario 2

This scenario demonstrates how the risk of damaging the drone affects the behavior. The drone is in a condition such that it cannot establish good contact with the surface. All inspections result in data not being gathered and a medium path conformity. The belief that drone-related and surface-related failure causes prevent data gathering and that drone-related and unmeasurable surface-related failure causes prevent controlled contact and data gathering is shown in Figure 8. The rest of

the nodes have a belief close to zero throughout this scenario.

In this scenario, the drone does not attempt to inspect the inspection point again after the failed inspection attempt at time step 0, as shown in Figure 8. This is due to the large cost associated with the possibility of having uncontrolled contact if the inspection is reattempted. At inspection point 1, a safe inspection action is performed since there is a considerable probability that the failure at time step 0 was caused by the drone. The system attempts one last inspection at inspection point 2 before executing full maintenance. After maintenance, the belief that there are drone-related failure causes preventing controlled contact, and data gathering is close to zero. In the next time step, a safe inspection is executed since the failure might have been caused by the surface, which was unaffected by the maintenance action. Since the inspection was successful, the belief that there were surface-related failure causes preventing controlled contact, and data gathering at inspection point 2 decreased. When evaluating backward in time at time step 5, as shown by the green broken line, the belief that drone-related failure causes prevents controlled contact and data gathering at time steps 0 and 2 is significantly increased. This decreases the belief that the failed inspection was caused by surface-related failure causes, making a revisit worth its cost. A safe inspection is performed at inspection point 1, as there could still be surface-related failure causes at this inspection point.

Time step	Measured failure causes	Chosen action	Measured objective fulfillment
0	Surface suitability: Perfect	Normal inspect	Data: No, Trajectory conformity: Medium
1	Surface suitability: Perfect	Move to IP1	
2	Surface suitability: Perfect	Safe inspect	Data: No, Trajectory conformity: Medium
3	Surface suitability: Perfect	Move to IP2	
4	Surface suitability: Perfect	Safe inspect	Data: No, Trajectory conformity: Medium
5	Surface suitability: Perfect	Full maintenance	
6	Surface suitability: Perfect	Safe inspect	Data: Yes, Trajectory conformity: Perfect
7	Surface suitability: Perfect	Return to IP1	
8	Surface suitability: Perfect	Safe inspect	Data: Yes, Trajectory conformity: Perfect
9	Surface suitability: Perfect	Move to IP0	

Fig. 8. Scenario 2. The table shows the measurements available before inspection, the choice of action, and the resulting measurements. The full line shows the belief of the drone that a failure cause is present at each time step. The broken line shows the belief of past states evaluated every time the drone moves to a new inspection point (IP), which is marked with a vertical line. The color of the broken line indicates when the updated belief was evaluated.

C. Scenario 3

This scenario demonstrates how the surface suitability measurement affects the choice of actions. Figure 9 shows how the system decides to not attempt an inspection if the surface suitability measurement is poor. With a medium surface suitability measurement, a safe inspection is attempted, but the system only attempts one inspection. When the surface suitability measurement is good but not perfect, two inspection attempts are warranted before moving on.

D. Scenario 4

This scenario demonstrates the effects of the gel level node. Figure 10 shows the expected value of the gel-level node in addition to the belief that gel-level-related failure causes prevent data gathering to better show how the gel depletes over time. The figure starts after 12 successful inspections. With each inspection, the expected gel level decreases. The belief that the gel-level-related failure causes prevent data gathering first starts to increase when the belief that the gel level is close to depleted increases. When no data is gathered in the inspection attempt at time step 37, the drone assumes it is caused by a low gel level, and a refill is therefore executed.

Time step	Measured failure causes	Chosen action	Measured objective fulfillment
0	Surface suitability: Poor	Move to IP1	
1	Surface suitability: Medium	Safe inspect	Data: No, Trajectory conformity: Perfect
2	Surface suitability: Medium	Move to IP2	
3	Surface suitability: Good	Safe inspect	Data: No, Trajectory conformity: Perfect
4	Surface suitability: Good	Normal inspect	Data: No, Trajectory conformity: Perfect
5	Surface suitability: Good	Move to IP3	
6	Surface suitability: Perfect	Normal inspect	Data: Yes, Trajectory conformity: Perfect
7	Surface suitability: Perfect	Move to IP4	

Fig. 9. Scenario 3. The table shows the measurements available before inspection, the choice of action, and the resulting measurements. The belief of the system is omitted because it gives little information in this scenario.

Time step	Measured failure causes	Chosen action	Measured objective fulfillment
24	Surface suitability: Perfect	Normal inspect	Data: Yes, Trajectory conformity: Perfect
25	Surface suitability: Perfect	Move to IP1	
26	Surface suitability: Perfect	Normal inspect	Data: Yes, Trajectory conformity: Perfect
27	Surface suitability: Perfect	Move to IP2	
28	Surface suitability: Perfect	Normal inspect	Data: Yes, Trajectory conformity: Perfect
29	Surface suitability: Perfect	Move to IP3	
30	Surface suitability: Perfect	Normal inspect	Data: Yes, Trajectory conformity: Perfect
31	Surface suitability: Perfect	Move to IP4	
32	Surface suitability: Perfect	Normal inspect	Data: Yes, Trajectory conformity: Perfect
33	Surface suitability: Perfect	Move to IP5	
34	Surface suitability: Perfect	Normal inspect	Data: Yes, Trajectory conformity: Perfect
35	Surface suitability: Perfect	Move to IP6	
36	Surface suitability: Perfect	Normal inspect	Data: No, Trajectory conformity: Perfect
37	Surface suitability: Perfect	Refill gel	
38	Surface suitability: Perfect	Normal inspect	Data: Yes, Trajectory conformity: Perfect
39	Surface suitability: Perfect	Move to IP7	

Fig. 10. Scenario 4. The table shows the measurements available before inspection, the choice of action, and the resulting measurements. The full line shows the belief of the drone that a failure cause is present at each time step. The broken line shows the belief of past states evaluated every time the drone moves to a new inspection point (IP), which is marked with a vertical line. The color of the broken line indicates when the updated belief was evaluated.

V. DISCUSSION

Scenario 1 shows that the system is able to distinguish between faults with the drone and adverse inspection points by combining information over time. This enables the system to reactivity handle faults by executing maintenance actions when needed, even though the information available to the system from a single inspection attempt is insufficient to identify the cause of the failure. Furthermore, this scenario demonstrates that reasoning backward in time enables the system to realize that the previously visited inspection points were not the cause of the failure. This enables the system to reactivity handle failure in achieving the mission objective by returning the previously failed tasks.

Scenario 2 shows that the system is less willing to attempt multiple executions when there is a high probability of uncontrolled contact. After the first uncontrolled contact, the system executes a safe inspection action even though it has moved to a new inspection point. This shows that the system acts proactively to minimize the risk of damage, as there was a chance that the failure was caused by the drone.

Scenario 3 demonstrates that the system considers the measurements available before the task execution and acts proactively to avoid losses. Scenario 4 demonstrates how the gel-level node affects the behavior. In scenario 1, the system does not believe that the failed inspections are caused by the gel level, as the failure occurs immediately after take-off. In scenario 4, many inspections were successfully performed before the execution failed, making it probable that the gel was depleted.

These scenarios demonstrate that re-evaluating past states enables the system to maximize the mission objective while keeping the cost low. The system manages to fulfill the feasible tasks without having to unnecessarily maintain the system.

The resulting system is not able to directly identify the cause of the observed behavior. When there is a high belief that drone-related failure causes prevent data gathering, the system does not know what the failure cause is. It could be anything preventing the drone from gathering data at multiple inspection points that does not affect the motion of the drone. The sensor could be displaced, there might be some dirt on the sensor, or the sensor might be wrongly calibrated or not suitable for the current mission. Which of these scenarios is true is irrelevant, as they all prevent data from being gathered and have the same solution: requesting human assistance. The advantage of this approach is that the system does not need a model of all possible causes but rather a model of the different ways the task execution can be affected

by failure causes.

One of the advantages of Bayesian models is that they can be based on expert judgment in addition to experimental data. This enables the models to be used on novel systems where operational data is missing. Quantification of CPTs based on expert judgment is not a trivial task, and many different methods exist to simplify the process [30]. This article simplifies the process by using Boolean operators for most of the network. This reduces the number of nodes that require further quantification to the condition and measurement nodes. This has the advantage of reducing the number of model parameters that must be determined and enables the evaluation of each parameter in isolation. A challenge with this approach is that it is not clear from the network which failure causes that should be considered when quantifying the CPTs. A hazard analysis can be performed to identify these causes [32], [33], which can help with quantification.

The main drawback of this method is with regard to computational time. Evaluating large networks can in some cases be done very quickly ($<1s$) with the exact solution from the SMILE [39] library. However, for some models and some sequences of evidence, the computational time can suddenly become very large or the problem can become unsolvable for the exact solver. Slightly different models or different evidence can require substantially different computational times. Which models and scenarios that are problematic are not easy to identify in advance. The SMILE library also includes several approximate sampling-based algorithms that have more stable computational time and can handle all models and evidence. The execution time of the approximate algorithms are substantially longer than the exact solution for simple problems but does not increase significantly for complex problems. The precision of the approximate algorithms can be adjusted to reduce the computational time. As the network only needs to be evaluated every time a task is considered, a computational time of a few seconds can be acceptable. Using the proposed method online seems to be feasible, but more work is recommended on optimizing the algorithm for online use.

VI. CONCLUSION

This article presents an approach for building a DDN for decision-making for an autonomous robotic system executing a sequence of independent tasks. A case study on drone inspection demonstrates that the system is able to identify internal faults, such as sensor defects, and the state of the environment, such as unmeasured obstacles. The system manages to act proactively to avoid potential

losses and acts reactively by maintaining the system and revisiting failed inspection points when needed. The proposed method thereby demonstrates the fulfillment of the four requirements proposed in the introduction.

This work has demonstrated that evaluating past states is a useful tool when considering a sequence of tasks. Due to the limited information available at a single task, the system might make suboptimal decisions, such as skipping the task when a maintenance action is instead needed. When updating the belief of past states with new information, better estimates can be made. This enables the system to identify that it should reattempt a previously skipped task, thereby achieving a higher degree of mission goal fulfillment.

The goal of this article is to contribute to more autonomous robotic systems that do not require direct human supervision. The case study demonstrates an increased ability to identify and act upon internal faults and the state of the environment beyond direct sensor measurements. This indicates that the proposed method provides the system with part of the autonomy that is needed to operate without direct human supervision. The method still needs experimental validation.

In addition to experimental validation, further work could explicitly model the consequences of actions and the quality of the measurements. The use of continuous probability functions instead of discrete states is also considered for further work.

REFERENCES

- [1] I. Schjøberg and I. B. Utne, "Towards autonomy in ROV operations," *IFAC-PapersOnLine*, vol. 48, no. 2, pp. 183–188, 2015. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S2405896315002694>
- [2] M. Endsley, "Autonomous Horizons: System Autonomy in the Air Force - A Path to the Future," United States Air Force, Washington DC, Tech. Rep. 2015-0267, 2015. [Online]. Available: <http://www.af.mil/Portals/1/documents/SECAF/AutonomousHorizons.pdf>
- [3] A. G. Bruzzone, M. Massei, R. Di Matteo, and L. Kutej, *Introducing Intelligence and Autonomy into Industrial Robots to Address Operations into Dangerous Area*. Springer International Publishing, 2019, vol. 11472 LNCS. [Online]. Available: http://dx.doi.org/10.1007/978-3-030-14984-0_32
- [4] M. L. Seto, *Marine Robot Autonomy*, M. L. Seto, Ed. New York, NY: Springer New York, 2013. [Online]. Available: <http://link.springer.com/10.1007/978-1-4614-5659-9>
- [5] C. Wong, E. Yang, X.-T. Yan, and D. Gu, "Autonomous robots for harsh environments: a holistic overview of current solutions and ongoing challenges," *Systems Science & Control Engineering*, vol. 6, no. 1, pp. 213–219, 1 2018. [Online]. Available: <https://www.tandfonline.com/doi/full/10.1080/21642583.2018.1477634>
- [6] P. E. Hagen, Ø. Hegrenæs, B. Jalving, Ø. Midtgaard, M. Wiig, and O. K. Hagen, "Making AUVs Truly Autonomous," in *Underwater Vehicles*, A. V. Inzartsev, Ed. Vienna, Austria: I-Tech, 2009, ch. 8, pp. 129–152. [Online]. Available: http://www.intechopen.com/books/underwater_vehicles/making_aUVs_truly_autonomous
- [7] A. Jónsson, R. A. Morris, and L. Pedersen, "Autonomy in space: Current capabilities and future challenges," *AI Magazine*, vol. 28, no. 4, pp. 27–42, 2007.
- [8] C. R. German, M. V. Jakuba, J. C. Kinsey, J. Partan, S. Suman, A. Belani, and D. R. Yoerger, "A long term vision for long-range ship-free deep ocean operations: Persistent presence through coordination of Autonomous Surface Vehicles and Autonomous Underwater Vehicles," in *2012 IEEE/OES Autonomous Underwater Vehicles (AUV)*. IEEE, 9 2012, pp. 1–7. [Online]. Available: <http://ieeexplore.ieee.org/document/6380753/>
- [9] I. B. Utne, A. J. Sørensen, and I. Schjøberg, "Risk Management of Autonomous Marine Systems and Operations," in *Volume 3B: Structures, Safety and Reliability*. Trondheim: American Society of Mechanical Engineers, 6 2017. [Online]. Available: <https://asmedigitalcollection.asme.org/OMAE/proceedings/OMAE2017/57663/Trondheim,Norway/280986>
- [10] ISO, "ISO 31000 Risk management - Principles and guidelines," Geneva, Switzerland, p. 34, 2018.
- [11] M. Brito and G. Griffiths, "A Bayesian approach for predicting risk of autonomous underwater vehicle loss during their missions," *Reliability Engineering & System Safety*, vol. 146, pp. 55–67, 2 2016. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S0951832015002860>
- [12] C. A. Thieme and I. B. Utne, "A risk model for autonomous marine systems and operation focusing on human–autonomy collaboration," *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, vol. 231, no. 4, pp. 446–464, 8 2017. [Online]. Available: <http://journals.sagepub.com/doi/10.1177/1748006X17709377>
- [13] J. Hegde, I. B. Utne, I. Schjøberg, and B. Thorkildsen, "A Bayesian approach to risk modeling of autonomous subsea intervention operations," *Reliability Engineering & System Safety*, vol. 175, no. February, pp. 142–159, 7 2018. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S0951832017304222>
- [14] Z.-X. Yang, G.-G. Wu, L. Song, and L.-P. Zhang, "Control Change Cause Analysis-Based Bayesian Network Modeling for System Risk Assessment," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 50, no. 8, pp. 1–11, 2019. [Online]. Available: <https://ieeexplore.ieee.org/document/8370792/>
- [15] I. B. Utne, B. Rokseth, A. J. Sørensen, and J. E. Vinnem, "Towards supervisory risk control of autonomous ships," *Reliability Engineering & System Safety*, vol. 196, no. October 2019, p. 106757, 4 2020. [Online]. Available: <https://doi.org/10.1016/j.ress.2019.106757>
- [16] Y. Qin, Q. Zhang, C. Zhou, and N. Xiong, "A Risk-Based Dynamic Decision-Making Approach for Cybersecurity Protection in Industrial Control Systems," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 50, no. 10, pp. 3863–3870, 2020. [Online]. Available: <https://ieeexplore.ieee.org/document/8438948/>
- [17] J. E. Bremnes, C. A. Thieme, A. J. Sørensen, I. B. Utne, and P. Norgren, "A Bayesian Approach to Supervisory Risk Control of AUVs Applied to Under-Ice Operations," *Marine Technology Society Journal*, vol. 54, no. 4, pp. 16–39, 7 2020. [Online]. Available: <https://www.ingentaconnect.com/content/10.4031/MTSJ.54.4.5>
- [18] M. Coombes, W.-H. Chen, and P. Render, "Site Selection During Unmanned Aerial System Forced Landings Using Decision-Making Bayesian Networks," *Journal of Aerospace*

- Information Systems*, vol. 13, no. 12, pp. 491–495, 12 2016. [Online]. Available: <http://arc.aiaa.org/doi/10.2514/1.1010432>
- [19] B. Cai, Y. Liu, and M. Xie, “A Dynamic-Bayesian-Network-Based Fault Diagnosis Methodology Considering Transient and Intermittent Faults,” *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 1, pp. 276–285, 1 2017. [Online]. Available: <http://ieeexplore.ieee.org/document/7495018/>
- [20] J. Schumann, T. Mbaya, and O. J. Mengshoel, “Software and System Health Management for Autonomous Robotics Missions,” 2012. [Online]. Available: https://kilthub.cmu.edu/articles/journal_contribution/Software_and_System_Health_Management_for_Autonomous_Robotics_Missions/6710654
- [21] C. Iamsumang, A. Mosleh, and M. Modarres, “Monitoring and learning algorithms for dynamic hybrid Bayesian network in on-line system health management applications,” *Reliability Engineering & System Safety*, vol. 178, no. May, pp. 118–129, 10 2018. [Online]. Available: <https://doi.org/10.1016/j.ress.2018.05.016>
- [22] J. Luque and D. Straub, “Reliability analysis and updating of deteriorating systems with dynamic Bayesian networks,” *Structural Safety*, vol. 62, pp. 34–46, 9 2016. [Online]. Available: <http://dx.doi.org/10.1016/j.strusafe.2016.03.004>
- [23] Y. Hernandez, J. Noguez, E. Sucar, and G. Arroyo-Figueroa, “Incorporating an Affective Model to an Intelligent Tutor for Mobile Robotics,” in *Proceedings. Frontiers in Education. 36th Annual Conference*. IEEE, 2006, pp. 22–27. [Online]. Available: <http://ieeexplore.ieee.org/document/4116913/>
- [24] R. C. Murray, K. Vanlehn, and J. Mostow, “Looking Ahead to Select Tutorial Actions : A Decision-Theoretic Approach,” *International Journal of Artificial Intelligence in Education (IJAIED)*, vol. 14, pp. 235–278, 2004. [Online]. Available: <https://dl.acm.org/doi/10.5555/1434913.1434915>
- [25] C. Conati, “Probabilistic assessment of user’s emotions in educational games,” *Applied Artificial Intelligence*, vol. 16, no. 7-8, pp. 555–575, 8 2002. [Online]. Available: <http://www.tandfonline.com/doi/abs/10.1080/08839510290030390>
- [26] B. W. Mott and J. C. Lester, “U-director,” in *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems - AAMAS '06*, vol. 2006. New York, New York, USA: ACM Press, 2006, p. 977. [Online]. Available: <http://portal.acm.org/citation.cfm?doid=1160633.1160808>
- [27] T. H. Bui, M. Poel, A. Nijholt, and J. Zwiers, “A tractable hybrid DDN-POMDP approach to affective dialogue modeling for probabilistic frame-based dialogue systems,” *Natural Language Engineering*, vol. 15, no. 2, pp. 273–307, 4 2009. [Online]. Available: https://www.cambridge.org/core/product/identifier/S1351324908005032/type/journal_article
- [28] D. Codetta-Raiteri and L. Portinale, “Dynamic Bayesian Networks for Fault Detection, Identification, and Recovery in Autonomous Spacecraft,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 1, pp. 13–24, 1 2015. [Online]. Available: <https://ieeexplore.ieee.org/document/6823144>
- [29] G. Shani, “Task-Based Decomposition of Factored POMDPs,” *IEEE Transactions on Cybernetics*, vol. 44, no. 2, pp. 208–216, 2 2014. [Online]. Available: <http://ieeexplore.ieee.org/document/6494590/>
- [30] N. Fenton and M. Neil, *Risk Assessment and Decision Analysis with Bayesian Networks*, 2nd ed. Chapman & Hall/CRC, 2018.
- [31] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. Pearson Education, 2014.
- [32] N. G. Leveson and J. P. Thomas, *STPA Handbook*, MA, USA, 2018. [Online]. Available: http://psas.scripts.mit.edu/home/get_file.php?name=STPA_handbook.pdf
- [33] M. Rausand and S. Haugen, *Risk Assessment*. Wiley, 3 2020. [Online]. Available: <https://onlinelibrary.wiley.com/doi/book/10.1002/9781119377351>
- [34] L. Mkrtchyan, L. Podofilini, and V. Dang, “Methods for building Conditional Probability Tables of Bayesian Belief Networks from limited judgment: An evaluation for Human Reliability Application,” *Reliability Engineering & System Safety*, vol. 151, pp. 93–112, 7 2016. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S0951832016000132>
- [35] R. A. Mattar and R. Kalai, “Development of a Wall-Sticking Drone for Non-Destructive Ultrasonic and Corrosion Testing,” *Drones*, vol. 2, no. 1, p. 8, 2 2018. [Online]. Available: <http://www.mdpi.com/2504-446X/2/1/8>
- [36] B. B. Kocer, T. Tjahjowidodo, M. Pratama, and G. G. L. Seet, “Inspection-while-flying: An autonomous contact-based nondestructive test using UAV-tools,” *Automation in Construction*, vol. 106, no. July, p. 102895, 10 2019. [Online]. Available: <https://doi.org/10.1016/j.autcon.2019.102895>
- [37] L. González-deSantos, J. Martínez-Sánchez, H. González-Jorge, F. Navarro-Medina, and P. Arias, “UAV payload with collision mitigation for contact inspection,” *Automation in Construction*, vol. 115, no. March, p. 103200, 7 2020. [Online]. Available: <https://doi.org/10.1016/j.autcon.2020.103200>
- [38] D. Zhang, R. Watson, G. Dobie, C. MacLeod, and G. Pierce, “Autonomous Ultrasonic Inspection Using Unmanned Aerial Vehicle,” in *2018 IEEE International Ultrasonics Symposium (IUS)*, vol. 2018-Octob. IEEE, 10 2018, pp. 1–4. [Online]. Available: <https://ieeexplore.ieee.org/document/8579727/>
- [39] BAYESFUSION LLC, “SMILE Engine.” [Online]. Available: <https://www.bayesfusion.com/smile/>