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 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 by fusing information over time. This knowledge is used to make risk-informed decisions enabling the system to proactively avoid failure and to minimize the consequence of faults. Past states are evaluated with new information to identify and counteract previous sub-optimal actions. A case study on an inspection drone tasked with contact-based ultrasound inspection is presented. The case study successfully demonstrates the proposed capabilities while minimizing time use 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 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],

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).

[5], [6]. Developing autonomous robotic systems that can operate without direct human supervision can enable a wider range of missions. One example is missions where communication is limited, such as underwater [4] and space [7] operations. Another is long-term [8] and multi-agent operations [2] that would otherwise be 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]. These deviations 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]. 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. Information on the state of real-world systems and environments is often uncertain or incomplete [11]. When acting with uncertain and incomplete information, the system cannot avoid making sub-optimal or erroneous decisions.

This article aims at developing a risk-based decision system that improves the ability of an autonomous system to interpret and act upon deviations from normal operation and to counteract the consequences of past erroneous or sub-optimal choices. Based on the considerations presented above, the following five requirements are defined to achieve the goal of this article.

- R1) The system must be able to identify the state of the environment during execution.
- R2) The system must be able to identify internal faults during execution.
- R3) The system must be able to identify its own errors or sub-optimal choices.
- R4) The system must act proactively to avoid failures.
- R5) The system must act to minimize the consequences of identified faults, errors, and sub-optimal choices.

The ability to identify the state of the environment

beyond what is directly measurable (R1) is demonstrated in, for example, educational system [12]–[14] and dialog systems [15], [16]. In these systems, the user’s state is unknown and must be inferred based on indirect measurements. These works [12]–[16] utilize a dynamic decision network (DDN) to infer the state of the user based on their measured behavior in response to decisions made by the system.

The ability to identify the presence of internal faults (R2) has been demonstrated in [17]–[19]. These works model the degradation of different components over time with a dynamic Bayesian network (DBN). Different available measurements are inserted into the network and used to infer component health.

The ability to proactively avoid failures (R4) has been demonstrated by Bremnes et al. [20]. They use a Bayesian belief network (BBN) to evaluate the collision risk during an under-ice operation with an autonomous underwater vehicle. By changing the safety margin to the ice sheet based on the risk level, they proactively avoid collision.

Coombes et al. [21] and Qin et al. [22] both demonstrate abilities to minimize consequences (R5) when an unwanted event occurs. Coombes et al. [21] consider choosing an emergency landing location for an unmanned aerial vehicle. A BBN is used to evaluate the risk of different candidate locations based on the available information about each location. Qin et al. [22] consider a cyberattack against an industrial control system. They use a BBN to evaluate the effect of different recovery and security strategies.

Codetta-Raitari and Portinale’s [23] work has fulfilled all the requirements except for R3. The work considers a case study of the power supply system of a Mars rover. A DBN is used to infer the presence of internal faults and adverse environmental conditions (R1 and R2) based on available observations. The DBN is then used to evaluate if one of four pre-defined hazardous scenarios is likely to occur in the future or if one or several has already occurred. If one of the scenarios is likely to occur in the future, then a preventive action is proposed (R4). If one has already occurred, then a recovery action is proposed (R5).

None of the reviewed publications considers identifying erroneous or sub-optimal choices made by the system itself (R3). Nevertheless, the previously presented literature demonstrates that Bayesian methods are a promising tool for achieving this article’s goal. Using risk for making decisions during operation was shown in [20], [21] to be a feasible approach. This further strengthens the case for Bayesian models, such as BBN, DBN (the dynamic counterpart of BBNs), and DDN

(DBNs that include the decisions made by the system), as these model probabilistic relationships making them suitable to model risk [24].

This article proposes a Bayesian approach for fulfilling all five defined requirements. A DDN is used to identify and distinguish between internal faults (R2) and adverse external conditions (R1) by combining information over time while considering the actions made by the system. Past states of the DDN are updated when new information is gathered. This enables the autonomous system to identify and handle previous sub-optimal or erroneous decisions (R3). The DDN evaluates the risk of executing different actions enabling the system to proactively avoid failure (R4). The effect of different recovery actions is simulated, allowing the system to minimize the consequence of internal faults and erroneous or sub-optimal choices (R5).

This article considers a robotic system executing a sequence of independent tasks, such as inspection, sampling, or intervention. It considers high-level decision-making, such as if and how a task should be executed, whether a task should be re-attempted, or if maintenance actions are needed. A case study on a multirotor inspection drone tasked with contact-based ultrasound thickness measurements is presented to demonstrate the proposed method.

The novelty and main contribution of this article lie in the proposed method for developing the DDN and the proposed decision algorithm that utilizes it. The resulting method addresses all five of the proposed requirements for a robotic system executing a sequence of independent tasks. The scope and method of this present article are substantially different from the earlier work that considered multiple of the proposed requirements [23]. This present article considers decision-making during operation while Codetta-Raitari and Portinale [23] considered emergency fault handling. In their article [23] they evaluated whether any of the pre-defined scenarios had occurred. In contrast, this present article identifies different types of internal and external failures and models their effect on task execution.

The rest of the paper is structured as follows: Section II gives some background on Bayesian models. Section III presents the proposed method for structuring and using the DDN. This method is applied to the case study in section IV. Simulation results from the case study are presented in section V. Section VI discusses the proposed method in light of the results from the case study, and a conclusion is presented in section VII.

Fig. 1. Example BBN consisting of 4 nodes.

II. BACKGROUND

Bayesian belief networks (BBN) are directed acyclic graphs (DAG) used for probabilistic inference. An example is shown in figure 1. The arcs in a BBN point from a parent node to a child node and describe dependencies, two nodes that are not connected by an arc are conditionally independent of each other. Conditional probability tables (CPT) are often used to quantify these dependencies. A CPT defines the probability of a node being in a particular state for all possible combinations of states of its parent nodes.

BBNs are typically used to evaluate the probability that a particular node is in a particular state, given some evidence, $\mathcal{E}$. Evidence is a set of knowledge regarding the state of one more of the nodes in the BBN. For example, if it is known (or assumed) that a node B is in state b and that a node D is in state d , then the evidence can be written as follows: $\mathcal{E} = \{B = b, D = d\}$. The probability of a node A being in state a given evidence $\mathcal{E}$ can then be written as:

$$P(A = a|\mathcal{E}) = P(A = a|B = b, D = d) \quad (1)$$

This can be reformulated using the conditional probability formula:

$$P(A = a|\mathcal{E}) = \alpha P(A = a, \mathcal{E}) \quad (2)$$

Where α is a normalization factor, making the probability distribution of A sum to 1.

There might be other nodes in the BBN in addition to the ones we want to evaluate, A , and the nodes we have evidence on, $\mathcal{E}$. In figure 1 this would be node C . These are included by summing over all possible states they can be in.

$$P(A = a|\mathcal{E}) = \alpha \sum_c P(A = a, \mathcal{E}, C = c) \quad (3)$$

Including these nodes makes the expression dependent on the joint probability distribution over all the nodes in the network. This makes it possible to evaluate the

expression by using the chain rule while adhering to the dependencies specified by the arcs of the BBN. For the example given in figure 1, the following can be derived:

$$P(A = a, \mathcal{E}, C = c) = P(A = a, B = b, C = c, D = d) \quad (4)$$

$$= P(A = a|C = c)P(B = b|C = c)P(C = c|D = d)P(D = d) \quad (5)$$

The resulting conditional probabilities are given by the CPTs.

Bayesian belief networks can be made dynamic by repeating the network for each time step and connecting the nodes based on how they depend on each other across time. The resulting dynamic Bayesian network (DBN) can be evaluated as before, the only difference being the introduction of a new copy of all nodes for each time step. Decisions can be included in the network, making it a dynamic decision network (DDN). This is done by introducing nodes representing the decision variable and including the decision in the list of evidence.

As multiple general solvers for Bayesian models exists [25], the rest of the article will focus on how to develop and use the DDN to achieve the goal of this article. More information on BBNs and DDNs can be found in [24], [26].

III. 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 similar independent tasks. Tasks are considered independent when “no task provides a necessary precondition for the fulfillment of another task” [27].

The basic procedure for using the DDN is as follows:

- 1) If available, insert evidence based on measurements available before a task is attempted.
- 2) Evaluate the risk and gain of executing different actions.
- 3) Execute the optimal action.
- 4) If available, insert evidence based on the observed outcome of the action.
- 5) Make a new time-step in the DDN. Each time step represents a decision that is made.

Section III-A presents a step-by-step method for developing the DDN. How decisions are made based on the DDN is presented in section III-B

A. Developing the DDN

This article proposes developing the DDN through a top-down approach. This approach ensures that only

Fig. 2. Generic DDN network developed using the proposed method. Objectives are shown in orange, intermediate nodes in light blue, condition nodes in dark blue, measurements in green, and which action that is chosen in gray. The different lines, L1-L5, are introduced in section III-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. The action node is described in step 6.

states that can be distinguished based on the observations and actions of the robotic system are included. The following steps are used to develop the DDN:

- 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.

Figure 2 shows an example of the resulting network.

Step 1 - Describe the operation and system: A description of the operation and the robotic system is needed. The operational description defines the tasks the system should execute and which actions the robotic system can choose between. This article considers three types of actions: executive actions, maintenance actions,

and moving to a different task.

Executive actions are associated with a probability of achieving the goal of the task and may lead to losses, which are discussed further in step 2. These actions are associated with a direct cost and an indirect cost if a loss occurs.

Maintenance actions 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.

When changing task, the system can choose between going to the next task in the sequence or returning to a previous task. Leaving a task without fulfilling its goal is associated with a cost. This cost is weighed against the expected cost of attempting the task. How these trade-offs are made is discussed in detail in section III-B.

In the description of the robotic system, the available sensors and information from different subsystems, such

as a navigation system, are 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 [28]. Damage 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) [29] or system theoretic process analysis (STPA) [28]. 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.

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 fulfilled. Not achieving an objective is considered a failure. The failure causes can be identified with a risk analysis; see [28], [29]. Intermediate nodes are introduced to group all 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.

A new set of intermediate nodes are introduced on line L3 of Figure 2 that distinguish between failure causes that are affected by the choice of actions in different ways. All failure causes that affect the same objectives and that are affected by the choice of action in similar ways are grouped together. This enables the system to distinguish between different failure causes based on their behavior when different actions are executed.

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 step. For each measurement, separate intermediate nodes are introduced that represent all failure causes that this measurement gives information on. If there exist failure causes that cannot be directly measured, then an additional intermediate node is introduced to represent all failure causes that are not measured. 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

A new set of nodes, called condition nodes, are introduced on line L5 in Figure 2. These nodes model the general condition of the failure causes, such as the amount of wear or the failure rate of a component.

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.

In contrast, the nodes introduced in the previous steps model the expected outcome of a single execution attempt.

Step 6 - Model dynamics: A new time-step is introduced in the DDN for each decision that is made. The condition nodes introduced in step 5 are connected to themselves between time steps, as indicated with the dotted arrow in Figure 2. This enables the modeling of how the conditions develop over time when different actions are chosen and hazards occur. The dependencies of choices in actions and the occurrence of hazards are illustrated with dotted arrows in figure 2.

Some conditions might be different for each task. These conditions can be modeled by introducing an instance of the condition node for each 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 modeled. Having separate measurement nodes enables the modeling of measurement uncertainty. The condition of the measured failure causes should influence these measurement nodes, as shown in green at the bottom of Figure 2. The different states of the measurement nodes are the possible values that the measurement can give. When a measurement is made, its value is inserted as evidence on its corresponding measurement nodes.

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 nodes introduced in steps 2 to 4, lines L1 to L4 in figure 2, take on a binary state. The state of the objective nodes indicate whether the objective will be fulfilled or not on this execution attempt. The state of the intermediate nodes given on lines L2

to L4 indicate whether the failure causes represented by these nodes will prevent the fulfillment of one or more objectives on this execution attempt. The CPTs for the nodes affected by these intermediate nodes define which combination of failure inputs that lead to a failure state. These CPTs can be populated using Boolean logic [24].

The condition nodes can have more than two states. These states can, for example, be the failure rate of a component or the amount of wear or depletion a component has experienced. The CPTs of the nodes influenced by a condition node translate this condition into a probability of failure on this execution attempt.

The CPTs of the condition nodes specify how the states of the nodes at one time step depend on their state at the previous time step. These CPTs give the probability of degrading the condition by doing different executive actions and improving the condition with maintenance actions. Different expert judgment and data-driven approaches can be used to quantify these probabilities [24], [30]. The initial probability distribution of the condition node should reflect how probable the different states are at the start of the operation.

The CPT of the measurement node gives the probability of measuring different values as a function of the underlying condition or outcome of the task execution. These CPTs quantify the measurement uncertainty.

B. Decision policy

This section presents the proposed decision policy for using the DDN. This 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 the different strategies are given in equations (6)-(8). The notation is explained in table I.

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 (6). 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, $C(e)$, and may cause hazards. There can be multiple different hazards, each associated with its own cost. The cost of the different hazards are based on their potential consequences and are given as elements in the vector $\mathbf{C}_H(e)$. If the execution does not achieve the goal of this task, then there will be the additional cost of moving to the next task (strategy 1).

The probability of achieving the task's goal and of different hazards occurring when executing an action are evaluated with the DDN. This is done by inserting the action as evidence on the "Action" node and evaluating the goal and hazard nodes. The probability that the goal is fulfilled is denoted as P_G while the probabilities that the different hazards occur are given as elements in the vector $\mathbf{P}_H$.

The cost of strategy 2 is shown in equation (7). This cost is evaluated for all possible execution actions, e , applicable to the current task. The different execution actions can have different direct costs related to them, 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 inserting it as evidence in the "Action" node of the DDN and then simulating one step forward in time. The simulating is done by temporarily adding a new time-step to the DDN. The evaluated probability of hazards and goal achievement at this new time step in the DDN is then used to evaluate the cost of execution (strategy 2). Additionally, the cost of the maintenance action itself must be included. This cost is often quite high but can improve the success rate of 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 8 and should be evaluated for all combinations of maintenance (m) and execution (e) actions.

$$C_1 = \begin{cases} 0 & \text{If goal of current task is achieved} \\ C_G & \text{Otherwise} \end{cases} \quad (6)$$

$$C_2(e) = C(e) + \mathbf{C}_H(e)^\top \mathbf{P}_H(e) + (1 - P_G(e))C_1 \quad (7)$$

$$C_3(m, e) = C(m)/N(m) + C_{2,m}(e) \quad (8)$$

Updating past states given new information might reduce the expected cost of reattempting previously failed tasks. The expected cost of executing a previously attempted task is evaluated by simulating that the system moves to this task. 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 (9). A task is reattempted if the visit is warranted for any of the available execution actions.

$$C_{Ret} + C_2(e) < C_1 \quad (9)$$

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
$C_H(e)$	Cost of the potential hazards when executing e
$P_H(e)$	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

Attempting a task before and after maintaining the system will enable the system to identify if the maintenance helped. This behavior is encouraged by always choosing an execution action if the current task has not been attempted and if the execution action is cheaper than the move action (strategy 1). If this is not the case, then the cheapest action is chosen.

IV. 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.

A. Developing the DDN

Step 1 - Describe the operation and system: The operation consists of measuring metal surface thickness with an ultrasound sensor mounted on a multirotor drone [31]–[34]. 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 typically 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 are based on expected time use and discussed in more detail in section IV-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. The drone is assumed to operate in controlled industrial facilities consisting of metal surfaces without any humans 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 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 only affects 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, a lidar scan of the inspection surface can reveal the presence of protruding obstacles

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.

that will prevent controlled contact and data gathering. The limited resolution of the lidar can cause it to systematically miss thin obstacles, such as welding joints or minor surface irregularities. Separate nodes are introduced for the failure causes that are measurable and those that are not, as shown on line L4 in Figure 4.

Step 5 - Model the condition of the failure causes:

A slightly dirty or uneven surface, or a minor fault in the drone, can reduce the likelihood of an inspection succeeding without hindering it completely. For all nodes except the “gel level” node, the states of the condition nodes 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 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: In this step, the dynamics of the different condition nodes are modeled. For drone-related conditions, this dynamic is the possibility of damaging the drone, or in other ways introducing drone related failures, and the usage of gel with each inspection attempt. The probability and severity of introducing drone-related failures depend on whether an uncontrolled contact occurred and whether a normal or safe inspection was performed.

The surface-related conditions are assumed constant over time and independent at each inspection point. These are handled as discussed in section III-A step 6 and illustrated in Figure 3.

Step 7- Model measurements: The “surface suitability measurement” is introduced as shown at the bottom of Figure 4. This measurement is, as discussed in step 4, based on how flat the area around the inspection point seems based on the lidar scan. This node 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: Contact-based inspection drones are in early development stage. It is, therefore,

little or no operational data and experience to base the quantification on. Many different hardware and software designs can be used for these operations. Which design that is chosen will significantly affect the quantification process. The quantification will be sensitive to factors such as how robust the ultrasound sensor is, how robust the drone is to impact, and how well the drone manages to navigate. To demonstrate the proposed algorithm, some example values are evaluated in collaboration with a drone inspection company. New values must be quantified before using the resulting algorithm on a specific platform and environment.

The following assumptions were used during the quantification:

- Some inspections require the operators to clean the inspection surface first [35]. As this is not possible for the drone, it is assumed that there will be surfaces where the drone cannot gather data.
- The drone must be in stable contact to get a measurement. Touching the wall correctly with the sensor is difficult, making it likely that the drone will fail at some inspection attempts.
- The sensor can become defect due to dirt or rust sticking to it, which can happen even without an uncontrolled contact occurring
- An uncontrolled contact can displace the sensor or damage the drone’s integrity, making it unable to continue. The likelihood that an uncontrolled contact will damage the drone’s ability to operate is low as the drone is built to be robust to impacts.
- It is assumed that an attempt is made to choose inspection points without obstacles. There can still be obstacles at the inspection points as the prior knowledge of the facility might be imperfect and due to navigational uncertainty making the drone miss the intended target.

The result of the quantification process is shown in Figure 4 and Tables II, III, IV, and V. The initial probability distributions 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 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. data gathering				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

uncontrolled impact. It is assumed that an uncontrolled contact will seldom damage the drone, making the cost relatively low. Even if the drone is not damaged, it might require human assistance if it falls to the ground. The cost of not achieving the goal is based on the additional time used for a manual inspection.

V. RESULTS

B. Decision policy

The decision policy presented in section III-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 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

This section presents four scenarios for the inspection drone case study. These scenarios are chosen to demonstrate how the proposed approach fulfills the five previously defined requirements. The scenarios represent failures and events that are deemed likely to occur during the drone’s mission. Scenario 1 demonstrates the ability to identify and distinguish between internal faults and adverse environments (Requirements R1 and R2) and correct internal faults by requesting maintenance

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

Description	Symbol	Cost
Normal inspection cost	$C_E(e = \text{Normal Inspect})$	0.5 min
Safe inspection cost	$C_E(e = \text{Safe inspect})$	1 min
Cost of uncontrolled contact with normal inspection	$C_H(e = \text{Inspect})$	20 min
Cost of uncontrolled contact with safe inspection	$C_H(e = \text{Safe inspect})$	10 min
Refill gel cost	$C_M(m = \text{Refill})$	10 min
Full maintenance cost	$C_M(m = \text{Full maintenance})$	60 min
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 min
Return cost	C_{ret}	0.5 min

(R5). Furthermore, it demonstrates that the drone reasons backward in time to identify sub-optimal decisions (R3) and corrects them by returning to past tasks (R5). Scenario 2 demonstrates that the system distinguishes between faults that may cause damage and those that only affect mission completion. This enables the drone to act proactively to avoid damages (R4). Scenario 3 demonstrates that the measurements available before a task is attempted are used to proactively avoid damage (R4). Lastly, scenario 4 shows that the drone distinguishes between internal faults that develop differently over time and have different recovery solutions (R2).

The scenarios are simulated in Python using the SMILE [25] library to evaluate the DDN. In each scenario, which measurements that the system experiences are specified ahead of time. The actions that the drone chooses and the drone's belief regarding relevant nodes throughout the simulation are presented. To keep the plots simple, only the probabilities that the nodes are in the failure state are shown.

A. Scenario 1

The first scenario demonstrates the system's ability to identify the cause of 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 failure causes 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. This is due to it being more probable that a single failed inspection is caused by the surface than by 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, making 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 due to there being 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. 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 broken orange line being lower than the broken blue 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 prevent data gathering is high enough, making a full maintenance worth the cost. After maintenance, the following inspection at time step 10 is successful. This indicates

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.

that the repair might have solved the problem, making it more probable that there was a fault with the drone in the earlier attempts. Reasoning backward in time decreases the likelihood that surface-related failure causes were the cause of the previously failed inspections, as shown with the broken green line in Figure 6

When considering where to go next, the system evaluates whether a previously visited inspection point is worth another inspection attempt. Since the belief that the surfaces on these inspection points caused the failures has decreased, the system concludes that they are worth revising. As shown in Figure 7, the system first visits inspection point 1 again, where data is gathered successfully. This further strengthens the belief that the drone caused the previously failed inspections. 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 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

failure causes 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, a safe inspection is executed since the failure might have been caused by the surface, which was unaffected by the maintenance action. As the inspection was successful, the belief that surface-related failure causes prevented controlled contact and data gathering at inspection point 2 decreased. When reasoning backward in time at time step 7, as shown by the broken green 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.

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.

C. Scenario 3

This scenario demonstrates how the surface suitability measurement affects the choice of actions. Figure 9 shows how the system decides not to 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 preventing data gathering” first increases 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 a low gel level caused it, making it execute a refill.

VI. DISCUSSION

The case study results demonstrate the capabilities of the presented method with respect to the five previously defined requirements. Scenario 1 shows that the system is able to distinguish between faults with the drone (R2) and adverse inspection points (R1) by combining information over time. This enables the system to reactivity handle faults by executing maintenance actions when needed (R5). 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 as it previously assumed (R3). This enables the system to minimize the consequences of the previous sub-optimal actions by returning the previously failed tasks and reattempting the inspection (R5).

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 even though it has moved to a new inspection point. This shows that the system acts proactively to minimize the risk of damage (R4), as there was a substantial chance that the drone’s condition caused the failure.

Scenario 3 demonstrates that the system considers the measurements available before the task execution

and acts proactively to avoid losses (R4). Scenario 4 demonstrates how the gel-level node affects the behavior. In scenario 1, the system does not believe that the gel level caused the failed inspections, 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. This shows that the system manages to distinguish between different types of internal faults when it affects the system differently.

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

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 do not affect the drone’s motion. The sensor could be displaced, there might be some dirt on the sensor, or the sensor might be wrongly calibrated or unsuitable 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 maintenance. 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 operational 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 [24]. This article simplifies the process by using Boolean operators for most of the network. 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 [28], [29], which can help with quantification.

The DBN produced by the proposed method does not model the losses related to the occurrence of a hazard. The different losses that can occur may have different severity and probabilities associated with them. Modeling the losses could enable the system to distinguish between different levels of severity, enabling the system to change its behavior accordingly.

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.

The main challenge with the proposed method concerns computational time. Evaluating large networks can, in some cases, be done very quickly ($<1s$) with the exact solution from the SMILE [25] library. However, the computational time may become very large, or the problem can become unsolvable for some models and sequences of evidence. Slightly different models or evidence can require substantially different computational times. Which models and scenarios that may become problematic with respect to computational time are not easily identified in advance. The SMILE library also includes several approximate sampling-based algorithms with more stable computational time and that can handle most models and evidence. The computational time of the approximate algorithms are substantially longer than the exact solution for simple problems. But it 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 may be acceptable. Using the proposed method during operation seems feasible, but more work is recommended on optimizing the algorithm for operational use.

VII. CONCLUSION

This article presents an approach for structuring a DDN and a decision-making algorithm for an autonomous robotic system executing a sequence of independent tasks. A case study on an inspection drone demonstrates that the system can identify internal faults, such as sensor defects, and the state of the environment, such as unmeasured obstacles. By evaluating past states with new information, the system identifies past sub-optimal actions, such as tasks that were skipped when a maintenance action was required. The system manages

to act proactively to avoid potential losses and acts reactively by requesting maintenance of the system and revisiting failed inspection points when needed. The proposed method thereby demonstrates the fulfillment of the five requirements defined in the introduction.

This article aims 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 needed to operate without direct human supervision. The method still needs experimental validation. In addition to experimental validation, further work could explicitly model the losses by the different hazards occurring.

REFERENCES

- [1] I. Schjølberg 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. Schjølberg, "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] K. Rajan and A. Saffiotti, "Towards a science of integrated AI and Robotics," *Artificial Intelligence*, vol. 247, pp. 1–9, jun 2017. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S0004370217300310>
- [12] 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/>
- [13] 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>
- [14] 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>
- [15] 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>
- [16] T. H. Bui, M. Poel, A. Nijholt, and J. Zwieters, "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
- [17] 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/>
- [18] C. Iamsuamang, 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.res.2018.05.016>
- [19] 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>
- [20] 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>
- [21] 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>
- [22] 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/>
- [23] 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>
 - [24] N. Fenton and M. Neil, *Risk Assessment and Decision Analysis with Bayesian Networks*, 2nd ed. Chapman & Hall/CRC, 2018.
 - [25] BAYESFUSION LLC, “SMILE Engine.” [Online]. Available: <https://www.bayesfusion.com/smile/>
 - [26] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. Pearson Education, 2014.
 - [27] 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/>
 - [28] 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
 - [29] M. Rausand and S. Haugen, *Risk Assessment*. Wiley, 3 2020. [Online]. Available: <https://onlinelibrary.wiley.com/doi/book/10.1002/9781119377351>
 - [30] 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>
 - [31] 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>
 - [32] 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>
 - [33] 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>
 - [34] 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/>
 - [35] DNV GL AS, “Non-destructive testing,” Tech. Rep. December, 2015. [Online]. Available: <https://rules.dnv.com/docs/pdf/DNV/CG/2015-12/DNVGL-CG-0051.pdf>