Journal of Intelligent & Robotic Systems (2023) 109:31
https://doi.org/10.1007/s10846-023-01934-y

REGULAR PAPER

A Bayesian Approach to Risk-Based Autonomy, with Applications to Contact-Based Drone Inspections

Sverre Velten Rothmund[1] · Christoph Alexander Thieme[2] · Ingrid Bouwer Utne[2] · Tor Arne Johansen[1]

Received: 29 September 2022 / Accepted: 17 July 2023 / Published online: 26 September 2023

Abstract
Enabling higher levels of autonomy while ensuring safety requires an increased ability to identify and handle internal faults and unforeseen changes in the environment. This article presents an approach to improve this ability for a robotic system executing a series of independent tasks by using a dynamic decision network (DDN). A simulation case study of an industrial inspection drone performing contact-based inspection is used to demonstrate the capabilities of the resulting system. The case study demonstrates that the system is able to infer the presence of internal faults and the state of the environment by fusing information over time. This information is used to make risk-informed decisions enabling the system to proactively avoid failure and to minimize the consequence of faults. Lastly, the case study demonstrates that evaluating past states with new information enables the system to identify and counteract previous sub-optimal actions.

Keywords Autonomy · Bayesian methods · Decision-making · Dynamic Decision Network · Risk analysis · Safety

1 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, 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 be able to 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 suboptimal or erroneous decisions that it should detect and act to minimize the consequences of.

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. This article focuses on operational decision-making for a robotic system executing a series of independent tasks, such as inspection or intervention at multiple locations.

✉ Sverre Velten Rothmund
sverre.v.rothmund@ntnu.no

Christoph Alexander Thieme
christoph.thieme@ntnu.no

Ingrid Bouwer Utne
ingrid.b.utne@ntnu.no

Tor Arne Johansen
tor.arne.johansen@ntnu.no

[1] Department of Engineering Cybernetics, Norwegian University of Science and Technology, Trondheim, Norway

[2] Department of Marine Technology, Norwegian University of Science and Technology, Trondheim, Norway

Previous literature exists on making decisions based on uncertainty or risk. In [12] a Bayesian belief network (BBN) is used to evaluate the collision risk during an under-ice operation with an autonomous underwater vehicle. Safety critical parameters, such as distance to the ice sheet, are automatically changed by considering how they affect the risk evaluated with the BBN. In [13] an emergency landing location for an unmanned aerial vehicle is chosen by evaluating the risk of the different landing locations with a BBN. In [14] a BBN is used to evaluate the effect of different recovery and security strategies during a cyberattack against an industrial control system. Even though these works make risk-based decisions, they do not consider improving the system's ability to interpret its state and the state of the environment.

An ability to infer the health state of the system based on indirect observations is demonstrated in [15–17] by using a Dynamic Bayesian Network (DBN). This previous research only considers estimating the state of the system and does not consider using the results for automatic decision-making. Furthermore, they do not consider how the choice of actions affects how the system develops over time.

Considering the action made by the system and using the inferred state of hidden variables for automatic decision-making has been done in educational systems [18–20] and dialog systems [21, 22]. These systems use a dynamic decision network (DDN) to infer the state of the user based on their observed response to different actions made by the system. Even though these systems show some of the capabilities needed, they are made for a distinctly different type of problem, making them not directly applicable to automatic decision-making for robotic systems.

In [23] a system is presented that infers the state of the environment and the health state of a robot based on indirect measurements in a DBN and uses this information for automatic emergency fault handling. In contrast to [23], this present article considers operational decision making which makes it necessary to consider the risk and reward of executing different actions, how the choice of action affects how the system develops over time, and to re-evaluate past actions when new information has become available, neither of which is considered in [23].

This article combines the capabilities presented in the earlier literature to make a risk-based decision system for an autonomous robot executing a sequence of independent tasks. The capability of making risk-based decisions presented in [12–14] is combined with the capability of identifying hidden states based on indirect observations presented in [15–17] and with the capability of considering the actions taken by the system itself as presented in [18–22]. Furthermore, this work introduces a new capability of evaluating past states with new information to identify past mistakes or sub-optimal choices.

This article develops a DDN which combines measurements available before a task is attempted with which action that was chosen and what the outcomes of the action were. As the model is dynamic the result of multiple task execution attempts are considered in light of each other to reveal faults with the robotic system and adverse environmental conditions that can not be measured directly. A heuristic is proposed which used the DDN to evaluate if executing a task should be attempted or skipped, which execution action that should be used if more are available, and whether maintenance of the robot is needed. Additionally, the system updates its belief regarding past states when new information becomes available thereby identifying previously attempted tasks that were wrongly skipped and should therefore be re-attempted.

To demonstrate the proposed method a case study of an industrial inspection multi-rotor drone is considered. The drone is tasked with mapping the thickness of metal surfaces in an industrial facility to identify damages to the structure. The measurements are conducted by contacting the surface with an ultrasound probe [24–28]. The large number of measurements needed to get sufficient coverage makes the operation costly for a human operator to directly and continuously monitor, thereby warranting the need for autonomous execution.

The rest of the article is structured as follows: Section 2 states the problem formulation. Section 3 gives some background on Bayesian models. Section 4 presents the proposed method for developing and using the DDN. This method is applied to the case study in Section 5. Simulation results from the case study are presented in Section 6. Section 7 discusses the proposed method in light of the results from the case study. A conclusion is given in section 8.

2 Problem Statement

This article considers a robotic system that executes a sequence of independent tasks. Tasks are considered independent when "no task provides a necessary precondition for the fulfillment of another task" [29]. This article does not consider in which order the tasks should be executed. It is assumed that the tasks are given as an ordered list at the start of the operation. The tasks are assumed to be time-independent with no deadlines that have to be considered when planning.

This article considers a part of the autonomy layer that should decide if and how a task should be executed and whether maintenance is needed. Task executions can fail either due to problems related to the task making it harder for the robotic system to solve that task in particular, problems with the robotic system making it harder for the robot

to solve tasks in general, or due to random failure. If the task execution fails then the robotic system needs to decide whether it should attempt the task again, skip the current task as it seems impossible to complete, or request maintenance of the robot which can repair faults that are hindering the robot from successfully completing tasks.

Attempting to execute a task can cause a hazardous event. A hazardous event is one which can in the worst case lead to a loss. Crashing is an example of a hazardous event while damage to the robotic system is an example of a loss. There can be different ways of executing the task with different direct costs associated with the execution and they can affect the probability of achieving the goal of the task and causing hazardous events.

There can be different ways of maintaining the system that repairs or mitigates different types of faults with the robotic system. Maintenance 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.

To make decisions, measurements of relevant features of the current task together with information on how past task execution attempts went are available. Based on this information, the robotic system must infer its own state and the state of the environment to have a foundation for decision-making.

3 Background Theory

BBNs are directed acyclic graphs (DAG) used for probabilistic inference. The arcs in a BBN point from a parent node to a child node and represent dependencies. Nodes are often modeled as being able to be in a discrete set of states. Conditional probability tables (CPT) can then be used to define the probability that a child node is in a particular state for each possible combination of parent node states.

BBNs can be made dynamic, a DBN, by repeating the network for each time step and connecting the nodes based on how they depend on each other across time. Decisions can be included in the network, making it a DDN, by letting some of the nodes represent decision variables and including the decision in the list of evidence. An example of a DDN is shown in Fig. 1.

BBNs are typically used to evaluate the probability that a particular node is in a particular state, given some evidence. Each piece of evidence specifies which state a particular node is in. The probabilities of interest are evaluated using Bayesian probability laws while considering the dependencies and CPTs defined by the BBN [30, 31]. Multiple general solvers exist for evaluating Bayesian models [32]. As these solvers do all of the necessary computations the rest of the article will focus on the development of the model and how the results evaluated with the model can be used for decision making.

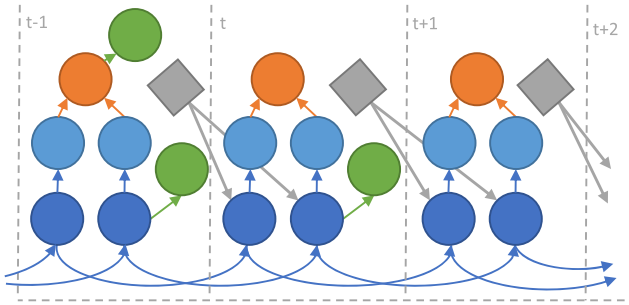


Fig. 1 View of a very simple DDN developed with the proposed method where the focus is placed on the time dynamics. Objective nodes are shown in orange, failure cause nodes in light blue, condition nodes in dark blue, measurement nodes in green, and action nodes in gray. The current time step (t) is shown together with one earlier time step ($t-1$) and one future time step ($t+1$)

4 Method

This section presents the proposed method for developing the DDN that will be used to infer the state of the robotic system and the environment, together with a strategy for using the DDN to choose what action the robotic system should take.

Figure 1 shows a simplified version of the network developed with the proposed method focusing on the time dynamics of the DDN. Figure 2 gives another example where more focus is placed on the nodes making up the network at a particular time step.

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.

4.1 Developing the DDN

This article proposes developing the DDN through a top-down approach. This approach ensures that only states that can be distinguished from each other are included. The following steps are used to develop the DDN:

1. Describe the operation and system.

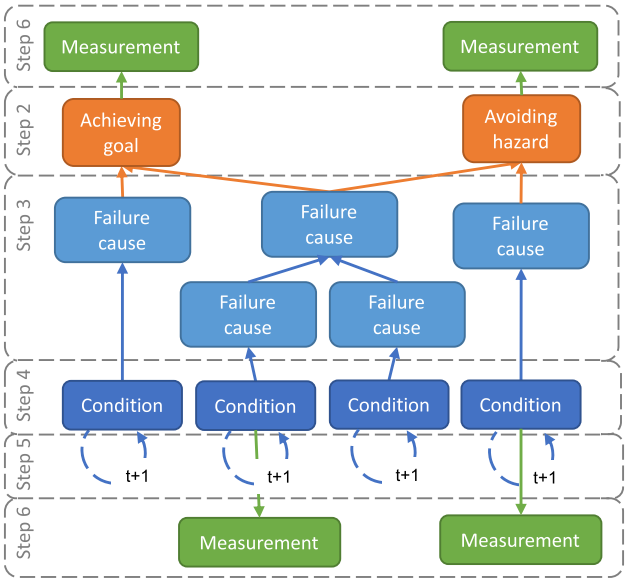


Fig. 2 Example network structure made using the proposed methods. Compared to Fig. 1 this figure considers a more complicated model. In this figure, the focus is placed on the nodes present at each time-step in the network in which of the steps presented in Section 4.1 they are introduced. The circular arrows represent connections across time steps

2. Model relevant objectives.
3. Model failure causes.
4. Model the condition of the failure causes.
5. Model dynamics.
6. Model measurements.
7. Quantification.

Step 1 - Describe the Operation and System

The operational description defines the tasks the system should execute and which actions the robotic system can choose between.

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 hazardous events. Not achieving the objectives is considered a failure, while the underlying cause of the failure is called the failure cause. Relevant hazards can be identified through different risk analysis methods, such as preliminary hazard analysis (PHA) [33] or system theoretic process analysis (STPA) [34]. A node is introduced in the DDN for every goal and hazard, as shown in Fig. 2. These nodes take on a binary state indicating whether the objective will be met or not 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 fulfilled. Not achieving an objective is considered a failure. The failure causes can be identified with a risk analysis; see [33, 34]. Nodes are introduced that represent groups of failure causes that cannot be distinguished from each other, shown as light blue in Fig. 2. All failure causes that affect different measurements or that are affected differently by the choice of action can potentially be distinguished from each other. The failure cause nodes take on a binary state indicating whether any failure cause in this group will cause a failure on this execution attempt.

Step 4 - Model the Condition of the Failure Causes

The failure cause nodes introduced in the last step consider the expected outcome of a single execution attempt. New nodes, called condition nodes, are introduced to model the general condition of the failure causes. These nodes could, for example, be defined as the amount of wear or the failure rate of a component. One condition node is introduced for each failure cause node as shown in dark blue in Fig. 2. These nodes can have multiple states to model varying ability to achieve the goal.

Step 5 - Model Dynamics

A new time step is introduced in the DDN for each decision that is made. The condition nodes introduced in step 4 are connected to themselves between time-step as shown with the dotted arrows in Fig. 2. This enables the DDN to combine information over time.

Some conditions can be independent for each task. These conditions can be modeled by having an instance of the node for each task in the operation. The nodes representing the current task are connected to the current time step. An example of this is given in Fig. 3.

Step 6 - Model Measurements

Separate measurement nodes are introduced to enable the modeling of measurement uncertainty. Measurements available before a task execution depend on the condition nodes, while measurements of how the execution went depend on the objective or failure cause nodes as shown in Fig. 2.

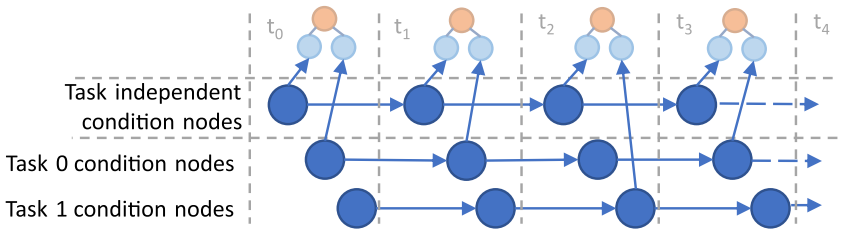


Fig. 3 Example of how different task-specific nodes can be connected to the rest of the network at different time steps. Task 0 is connected to the rest of the network at time steps 0, 1, and 3, while task 1 is connected at time step 2

Step 7 - Quantification

Bayesian models can be quantified based on expert judgment and 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 [30, 35]. This article simplifies the process by using Boolean operators to define which combination of failure causes that affect the different objective nodes. The CPT of the failure cause nodes that are children of condition nodes translates the condition into a probability of failure on this execution attempt. The CPTs of the condition nodes specify how the state can degrade or improve based on the choice of action. The CPTs of the measurement nodes quantify the measurement uncertainty.

4.2 Decision Policy

Finding the optimal decision policy requires solving a partially observable Markov decision problem (POMDP), which in the general case is intractable except for small problems [36]. To circumvent this problem, a heuristic policy is proposed. The policy considers the following three strategies consisting of one or multiple actions: 1) move on to another task, 2) attempt to execute the task once and then move on to another task, or 3) execute a maintenance action, attempt to execute the task execution, and then moving to another task. 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 another task. This ensures that the resulting closed-loop behavior can be closer to optimal behavior than any of the proposed strategies.

The cost of strategy 1, C_1, has only a cost if the goal of the current task is not achieved. This cost, C_G, is based on the consequence of not achieving the goal. This is shown in Eq. 1. More cases can be added if there can be a partial fulfillment of the goal.

The cost of strategy 2, $C_2(e)$, depends on the choice of execution action, e. There is a direct cost for executing action e, $C_E(e)$, and an indirect cost if a hazardous event occurs. There can be multiple different hazards, each associated with its own cost, which are given as elements in the vector $\mathbf{C}_H(e)$. This cost can depend on the choice of execution action. If the execution does not achieve the goal of this task, then there will be the additional cost of moving to another task, C_1. The probability of achieving the task's goal, P_G, and the probability of different hazardous events occurring, $\mathbf{P}_H$, when executing an action are evaluated using the DDN. These values are found by evaluating the probability that the objective nodes are in a failure state at the current time step. The resulting cost function is shown in equation Eq. 2. This cost is evaluated for all possible execution actions, e, applicable to the current task.

The cost of strategy 3, $C_3(m, e)$, depends on the choice of maintenance action, m, and execution action, e. The maintenance action can increase the probability of achieving the goal and reduce the probability of hazardous events occurring. The effect of the maintenance action is evaluated by inserting it as evidence in the action node at the current time step of the DDN and then simulating one step forward in time by temporarily adding a new time step to the DDN. The cost of execution (strategy 2) can then be evaluated at this time step, $C_{2,m}(e)$. The cost of the maintenance action must be included as well. This cost is often quite high but can improve the success rate of multiple future task execution attempts. The maintenance cost, $C_M(m)$, is divided by the expected number of executions until maintenance is needed again, $N(m)$. The resulting cost is shown in Eq. 3 and should be evaluated for all combinations of maintenance actions, m, and execution actions, e.

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

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

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

When moving to another task (strategy 1), the system can choose to revisit a previously attempted task. 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, $C_2(e)$, 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 Eq. 4. A task is reattempted if the visit is warranted for any of the available execution actions. If none of the previously attempted tasks are worth another attempt, then the system will move to the next task in the sequence that is not attempted.

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

Attempting a task before and after maintaining the system enables the system to identify if a maintenance action helped. This behavior is encouraged by always choosing an execution action if the current task has not been attempted and if strategy 2 is cheaper than strategy 1. If this is not the case, then the normal policy is followed.

5 Case Study

In this section, the proposed method is applied to a multirotor drone tasked with industrial inspection. The case study setup is developed in cooperation with the drone inspection technology company ScoutDI. Figure 4 shows the ScoutDI drone performing an ultrasound thickness measurement. The case study is based on simulation.

5.1 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. 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. A small amount of gel is dispensed from a tank mounted on the drone for each inspection. One maintenance action available to the drone is to refill this tank. Another is to request a full maintenance check by an operator. The drone can skip inspection points deemed too costly to inspect autonomously.

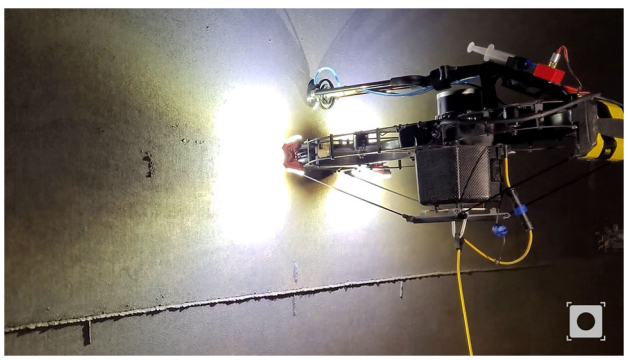


Fig. 4 A ScoutDI prototype drone during an ultrasound inspection of a storage tank. Courtesy ScoutDI

The drone is equipped with a lidar used to detect obstacles and navigate.

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. This makes damage to the drone the most relevant loss. A hazard that can cause this loss is uncontrolled contact with a surface or other object. Nodes representing the two objectives are shown on line L1 in Fig. 5.

Step 3 - Model Failure Causes

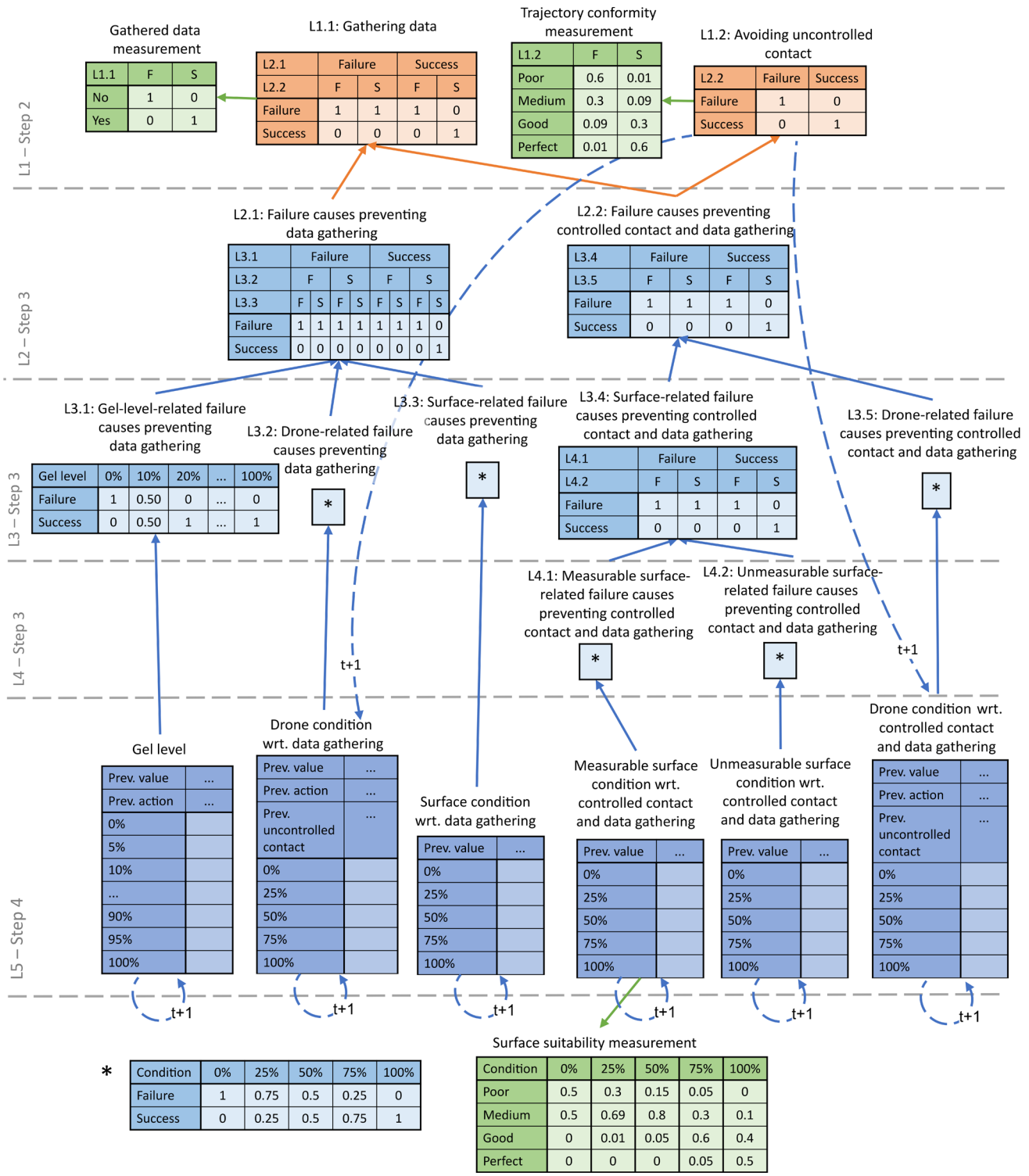
Through discussions with ScoutDI different failure causes were identified. Some of the failure causes, such as an empty gel tank, rust or dirt stuck on the ultrasound sensor, or inspection surfaces covered with rust or dirt can prevent data from being gathered. Other failure causes, such as a worn motor, poor navigation quality, or obstacles, can lead to uncontrolled contact in addition to preventing data from being gathered. To simplify modeling, two intermediate nodes are introduced: one for failure causes preventing data from being gathered, the other for failure causes preventing both controlled contact and data from being gathered. These are shown on line L2 in Fig. 5.

The drone and the surface of the inspection point are affected differently by choice of action. 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. These nodes are shown on line L3 in Fig. 5.

Before an inspection is executed, a lidar scan of the inspection surface can reveal protruding obstacles 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. A distinction between failure causes that are measurable and those that are not can therefore be made, as shown on line L4 in Fig. 5.

Step 4 - 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



L1.1	F	S
No	1	0
Yes	0	1

L2.1	Failure		Success	
L2.2	F	S	F	S
Failure	1	1	1	0
Success	0	0	0	1

L1.2	F	S
Poor	0.6	0.01
Medium	0.3	0.09
Good	0.09	0.3
Perfect	0.01	0.6

L2.2	Failure	Success
Failure	1	0
Success	0	1

L3.1	Failure				Success			
L3.2	F		S		F		S	
L3.3	F	S	F	S	F	S	F	S
Failure	1	1	1	1	1	1	1	0
Success	0	0	0	0	0	0	0	1

L3.4	Failure		Success	
L3.5	F	S	F	S
Failure	1	1	1	0
Success	0	0	0	1

Gel level	0%	10%	20%	...	100%
Failure	1	0.50	0	...	0
Success	0	0.50	1	...	1

L4.1	Failure		Success	
L4.2	F	S	F	S
Failure	1	1	1	0
Success	0	0	0	1

Prev. value	...
Prev. action	...
0%	
5%	
10%	
...	
90%	
95%	
100%	

Prev. value	...
Prev. action	...
Prev. uncontrolled contact	...
0%	
25%	
50%	
75%	
100%	

Prev. value	...
0%	
25%	
50%	
75%	
100%	

Prev. value	...
0%	
25%	
50%	
75%	
100%	

Prev. value	...
0%	
25%	
50%	
75%	
100%	

Prev. value	...
Prev. action	...
Prev. uncontrolled contact	...
0%	
25%	
50%	
75%	
100%	

*

Condition	0%	25%	50%	75%	100%
Failure	1	0.75	0.5	0.25	0
Success	0	0.25	0.5	0.75	1

Condition	0%	25%	50%	75%	100%
Poor	0.5	0.3	0.15	0.05	0
Medium	0.5	0.69	0.8	0.3	0.1
Good	0	0.01	0.05	0.6	0.4
Perfect	0	0	0	0.05	0.5

Fig. 5 The conditional probability tables and dependencies in the DDN for the inspection drone case study. Some tables are intentionally left blank and are instead given in Tables 1, 2, and 3. Nodes containing a * refer to the table marked with a * shown on the bottom left of the Fig

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 on line L5 in Fig. 5.

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 5 - Model Dynamics

Drone-related conditions have a probability of degrading with each inspection attempt. The probability and severity of the degradation depend on whether an uncontrolled contact occurred and whether a normal or safe inspection was performed. The gel level is gradually depleted with each inspection attempt. There can be some variation in the amount of gel dispensed making the number of inspection attempts before a refilled uncertain.

The surface-related conditions are assumed constant over time and independent at each inspection point. These are handled as discussed in Section 4.1 step 5 and illustrated in Fig. 3.

Step 6- Model Measurements

The “surface suitability measurement" is introduced as shown at the bottom of Fig. 5. This measurement is, as discussed in step 3, based on how flat the area around the inspection point seems based on the lidar scan.

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 measurement. 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.

Step 7 - Quantification

Contact-based inspection drones are in the early stage of development. It is, therefore, little or no operational data and experience to use as a basis for the quantification. The choice of hardware and software design 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 chosen in collaboration with ScoutDI. The following assumptions were considered during the quantification process:

- Some inspections require the operators to clean the inspection surface first [37]. As this is not possible for the drone, there is a chance 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. This 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 of damaging the drone is low as it is built to be robust to impacts.

The result of the quantification process is shown in Fig. 5 and Tables 1, 2, and 3. The initial probability distributions can be found in Table 1. Tables 2 and 3 show the probabilities of transitioning to a worse state for the different drone condition nodes when an inspection is attempted. 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 1 Initial probability distributions

	Gel level	Drone condition wrt. data gathering	Surface condition wrt. data gathering	Unmeasurable surface condition wrt. controlled contact and data gathering	Measurable surface condition wrt. controlled contact and data gathering	Drone condition wrt. controlled contact and data gathering
0%	0	0.03	0.3	0.02	0.01	0.005
25%	0	0.03	0.05	0.005	0.01	0.005
50%	0	0.04	0.05	0.005	0.01	0.005
75%	0	0.5	0.2	0.005	0.07	0.01
100%	1	0.4	0.4	0.965	0.9	0.975

Table 2 Probability of transitioning to worse states given the choice of action and whether an uncontrolled contact is avoided. The table gives the probability of degrading the state by different amounts. Transitions that lead to negative probabilities are omitted before the resulting distribution is normalized. The probability of transitioning to a better state is 0

Node name	Drone condition wrt. data gathering				Drone condition wrt. controlled contact and data gathering			
Action	Normal inspect		Safe inspect		Normal inspect		Safe inspect	
Avoiding uncontrolled contact	Failure	Success	Failure	Success	Failure	Success	Failure	Success
-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

5.2 Decision Policy

The decision policy presented in Section 4.2 is used with the parameters given in Table 4. These costs are based on the expected time use of the different actions. 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. 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. Evaluating an exact value for these costs can be difficult in practice. Some tuning of the values might therefore be necessary if the observed behavior is inadequate. Having values that can be interpreted still gives an advantage as it gives an intuition on what the values should be.

6 Results

This section presents four scenarios for the inspection drone case study. The scenarios represent different types of failures and events that are deemed likely to occur during the drone's mission. Scenario 1 considers a case where the ultrasound sensor is not working. In scenario 2 the drone is unable to have a controlled contact. Scenario 3 demonstrates the effect measurements have on the system's behavior. Lastly, scenario 4 considers a case where the gel is depleted.

Table 3 Probability of reducing the gel level by different amounts

	Gel level
-15%	0
-10%	0.1
-5%	0.8
-0%	0.1

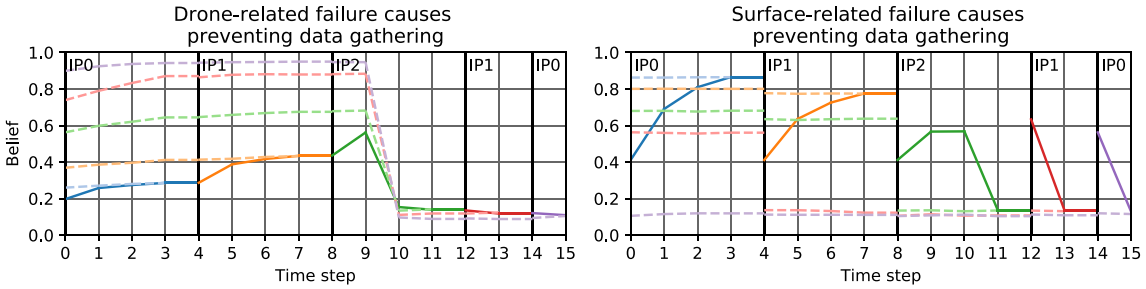
The simulations are done by having a model of the drones state and the sate of the different inspection surfaces as state machines. Their respective states define which measurements that are made and what the output will be of performing an action. The drone's state can either be working, with a defective ultrasound sensor, or in a state making it unable to make controlled contact. Each inspection surface can either be ideal, unable to be measured, with a measurable blocking obstacle, or with an immeasurable blocking obstacle. The DDN used to make decisions is evaluated using the SMILE [32] library for Python.

6.1 Scenario 1

In this scenario, the ultrasound sensor is not working. All inspections end with no data being gathered but perfect trajectory conformity. Figure 6 show how the belief of the system develops over time when new inspections are attempted. 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 scenario.

Table 4 The different costs used in the decision policy for the case study

Symbol	Cost
$C_E(e = \text{Normal inspect})$	0.5 min
$C_E(e = \text{Safe inspect})$	1 min
$C_H(e = \text{Normal inspect})$	20 min
$C_H(e = \text{Safe inspect})$	10 min
$C_M(m = \text{Refill})$	10 min
$C_M(m = \text{Full maintenance})$	60 min
$N(m = \text{Refill})$	20
$N(m = \text{Full maintenance})$	30
C_G	10 min
C_{ret}	0.5 min



Time step	Measured failure causes	Chosen action	Measured objective fulfillment
0	Surface suitability: Perfect	Normal inspect	Data: No, Trajectory conformity: Perfect
1	Surface suitability: Perfect	Normal inspect	Data: No, Trajectory conformity: Perfect
2	Surface suitability: Perfect	Normal inspect	Data: No, Trajectory conformity: Perfect
3	Surface suitability: Perfect	Move to IP1	
4	Surface suitability: Perfect	Normal inspect	Data: No, Trajectory conformity: Perfect
5	Surface suitability: Perfect	Normal inspect	Data: No, Trajectory conformity: Perfect
6	Surface suitability: Perfect	Normal inspect	Data: No, Trajectory conformity: Perfect
7	Surface suitability: Perfect	Move to IP2	
8	Surface suitability: Perfect	Normal inspect	Data: No, Trajectory conformity: Perfect
9	Surface suitability: Perfect	Full maintenance	
10	Surface suitability: Perfect	Normal inspect	Data: Yes, Trajectory conformity: Perfect
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. 6 Scenario 1. The table shows the measurements available before inspection, the choice of action, and the resulting measurements. The graph shows the state of failure causes relevant in this scenario. The solid line shows the belief of the drone that a failure cause is present at each time step. The dashed line shows the updated 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 dashed line indicates when the updated belief was evaluated

The four first time-steps of Fig. 6 shows the behavior and beliefs of the drone when it is at the first inspection point. As seen in the table in Fig. 6, 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, as shown by the solid blue line. 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 dashed blue line in Fig. 6 shows the system's belief about past states evaluated at time step 3. As the stat of the surface cannot change, the belief about the past states is equal to the newest belief. The state of the drone can, on the other hand, degrade, making the updated belief regarding the state of the drone at time step 0 slightly lower than at time step 3.

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 with the orange dashed line in Fig. 6, 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 surface-related failures caused the failed inspection at inspection point 0 decreases. This can be seen by the dashed orange line being lower than the dashed 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. As the inspection failed before the maintenance but succeeded after, it becomes more probable that there was a fault with the drone that was solved by the maintenance. Reasoning backward in time decreases the probability that surface-related failures caused the previ-

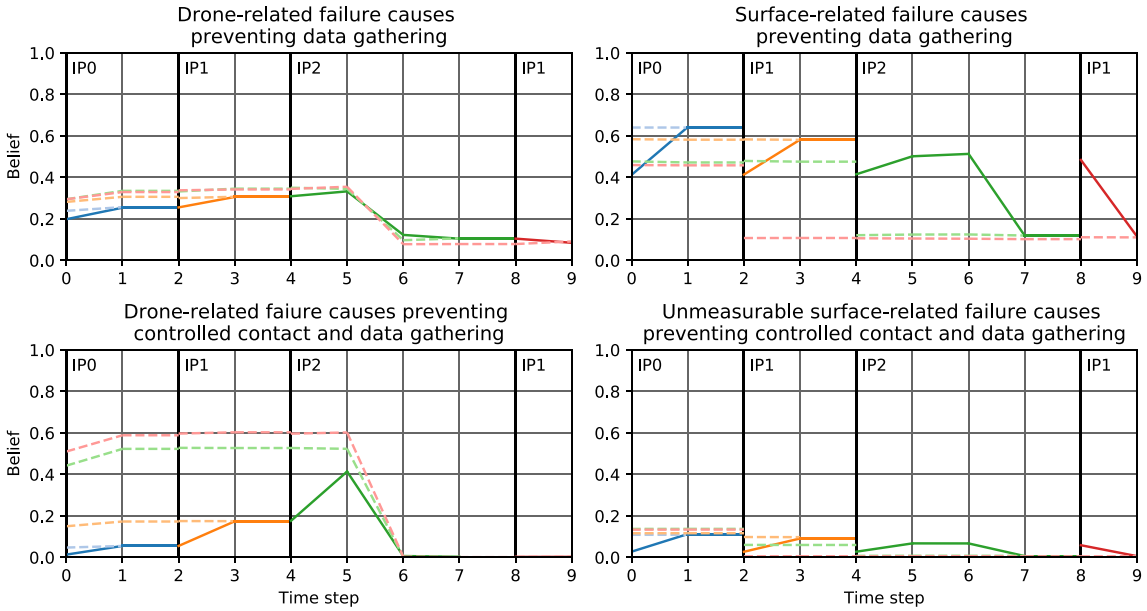
ously failed inspections, as shown with the dashed green line in Fig. 6

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 another attempt. 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.

6.2 Scenario 2

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 Fig. 7. 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 Fig. 7. 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 requesting 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



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. 7 Scenario 2. The table shows the measurements available before inspection, the choice of action, and the resulting measurements. The graph shows the state of failure causes relevant in this scenario. The solid line shows the belief of the drone that a failure cause is present at each time step. The dashed line shows the updated 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 dashed 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. 8 Scenario 3. The table shows the measurements available before inspection, the choice of action, and the resulting measurements. No graphs are shown as they give little additional information in this scenario

contact and data gathering at inspection point 2 decreased. When reasoning backward in time at time step 7, as shown by the dashed 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 another attempt 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.

6.3 Scenario 3

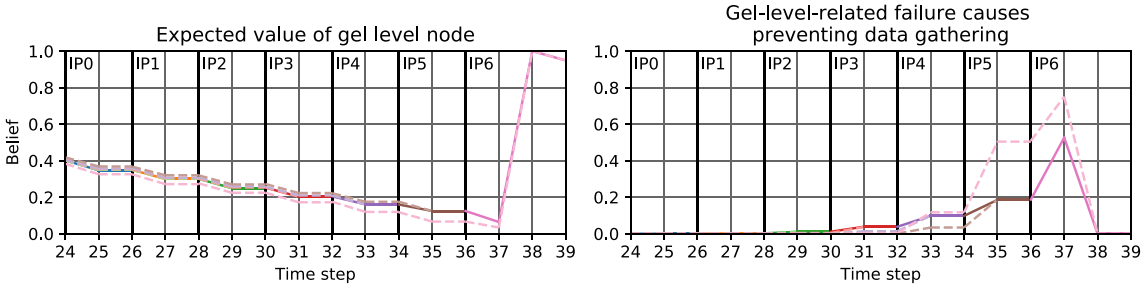
This scenario demonstrates how the surface suitability measurement affects the choice of actions. Figure 8 shows how the system decides not to attempt an inspection if the surface suitability measurement is poor. With a medium surface suitability measurement, a safe inspection is attempted, but the system only attempts one inspection. When the surface suitability measurement is good but not perfect, two inspection attempts are attempted before moving on.

6.4 Scenario 4

This scenario demonstrates the effects of the gel level node. Figure 9 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 expected gel level is close to depleted. 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.

7 Discussion

Scenario 1 shows that the system is able to distinguish between faults with the drone and adverse inspection surfaces by combining information over time. This enables the system



Time step	Measured failure causes	Chosen action	Measured objective fulfillment
...	...	...	...
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

Fig. 9 Scenario 4. The table shows the measurements available before inspection, the choice of action, and the resulting measurements. All inspections prior to time-step 33 resulted in data being gathered and perfect trajectory conformity. The graph shows the state of failure causes relevant in this scenario. The solid line shows the belief of the drone that a failure cause is present at each time step. The dashed line shows the updated 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 dashed line indicates when the updated belief was evaluated

to executing maintenance actions when needed. 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. This enables the system to return to previously failed tasks and reattempt the inspection.

Scenario 2 shows a case similar to scenario 1, with the difference that the drone experienced a worse trajectory conformity. This could be explained by an unmeasured obstacle in this current location, by a damage to the drone, or it could be a random failure. The possibility that there was an unmeasurable obstacle made the system not reattempt the failed task, as it did in scenario 1, but rather go directly to the next task. The possibility that there was a failure with the drone made the drone execute a safe inspection at the second location. This demonstrates how the system reasons with risk, and how it considers the underlying causes while doing so.

Scenario 3 demonstrates that the system considers the measurements available before the task execution to proactively manage risk. 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.

The proposed method for building the DDN ensures that the condition nodes, which are the possible explanations for the observed behavior, are quite general. Having general nodes ensures that the nodes actually represent features the system is able to distinguish based on the observations. 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 its motion. The sensor could be displaced, there might be 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. Constructing general condition nodes for all possible ways the system can be affected by actions and measurements ensures that the system has a possible explanation for all observations.

The DBN produced by the proposed method does not model the severity of the losses that can be caused by the occurrence of a hazard, such as having an uncontrolled contact. 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.

Based on the observed results, no obvious sub-optimal behavior with the proposed heuristic decision policy was observed. One drawback with the heuristic is the discount factor, $N(m)$. This factor could, in theory, be based on the probability of degrading the drone with each inspection attempt. This factor has a straightforward interpretation and effect on the resulting behavior, making the discount factor an acceptable trade-off between simplicity and quality of the heuristic.

The resulting decision policy needs to evaluate the network multiple times for each time step to simulate the effect of different maintenance actions and to evaluate whether the system should return to a previous point. This could potentially be alleviated by further simplifications, such as specifying thresholds for the different condition nodes. A predefined maintenance action can then be executed when the belief surpasses the threshold. The drawback of this approach is that it would lose information on the interaction between components. This is especially important for more complicated systems with more causal factors.

The point of the case study was to demonstrate capabilities that can be achieved with the proposed system. It was not to solve the case study in the most optimal or simplest manner. Similar behavior as the presented results could be achieved by, for example, defining an exhaustive set of conditional rules. These types of methods may work for very simple problems but do not scale well for more complex problems. Having a systematic approach, such as the one presented, that achieves the capabilities needed for an autonomous system to operate without human supervision can therefore be of great value.

Evaluating DDNs becomes computationally expensive when the number of time steps increases. The number of time steps can be limited by using a sliding window approach where only the n newest time steps are included in the DDN. The initial condition of the DDN must reflect the information that is no longer inside the sliding window. This can be achieved by setting the priors at the first time step inside the window equal to the posterior evaluated at the last step outside the window. A drawback with a sliding window approach is that only time steps inside the window will be considered when evaluating past states with new information. This constitutes a challenge for the proposed method as the number of time steps that are computationally feasible to consider might be too low to consider all previously attempted tasks that are interesting to reconsider. One possible way to alleviate this problem is to find a more compact way to represent information on previous tasks. Currently, previous tasks are represented by multiple time steps, one for each execution attempt and a time step for every move and repair action.

8 Conclusion

This article presents an approach for structuring a DDN and using it for operational decision-making. The article's goal is to contribute toward enabling autonomous systems to safely operate without direct human supervision. Through a case study of an industrial inspection drone it is demonstrated how the resulting system is able to increase the drones situation awareness about its own state and the state of the environment, and how the drone can use this information to make risk-based decisions. Additionally it is demonstrated how evaluating past states with new information can reveal tasks the drone wrongly skipped that it should return to for another inspection attempt.

Future work can consider how the severity of a hazard occurring can be modeled, how evaluating past states could be simplified such that a longer time horizon can be considered, and on experimental validation.

Acknowledgements We would like to thank ScoutDI for cooperation on the case study. Specifically we would like to thank Morten Fyhn Amundsen for discussion regarding the quantification and Kristian Klausen for general discussions on how the system works and potential failure causes.

Author Contributions S. Rothmund and C. Thieme developed the method and case-study with supervision from I. Utne and T. Johansen. Software and simulations were done by S. Rothmund. The first draft of the manuscript was written by S. Rothmund. C. Thieme, I. Utne, and T. Johansen revised the manuscript. Funding acquisition and project management was performed by I. Utne and T. Johansen.

Funding Open access funding provided by NTNU Norwegian University of Science and Technology (incl St. Olavs Hospital - Trondheim University Hospital). The work is sponsored by the Research Council of Norway through the UNLOCK project, project number 274441, and through the Centre of Excellence funding scheme, project number 223254, AMOS.

Data Availability The code used for simulation will made available upon request. Due to the use of a commercial third-party library restrictions apply preventing sharing of the complete code. No data set was produced in this study beyond what is shown in the figures.

Declarations

Ethics approval Not applicable as the research does not involve human or animal subjects.

Consent to participate Not applicable as the research does not involve human subjects.

Consent to publish Not applicable as the research does not involve human subjects.

Competing interests T. Johansen is a shareholder and board member in ScoutDI. S. Rothmund has had a part-time position at ScoutDI. C. Thieme and I. Utne have no financial or non-financal interests to disclose.

References

1. Schjølberg, I., Utne, I.B.: Towards autonomy in ROV operations. IFAC-PapersOnLine **48**(2), 183–188 (2015). https://doi.org/10.1016/j.ifacol.2015.06.030
2. Endsley, M.R.: Autonomous Horizons: System Autonomy in the Air Force - A Path to the Future. Technical Report 2015–0267, United States Air Force, Washington DC (2015). http://www.af.mil/Portals/1/documents/SECAF/AutonomousHorizons.pdf
3. Bruzzone, A.G., Massei, M., Di Matteo, R., Kutej, L.: Introducing Intelligence and Autonomy Into Industrial Robots to Address Operations Into Dangerous Area vol. 11472 LNCS, pp. 433–444. Springer, Cham (2013). https://doi.org/10.1007/978-3-030-14984-032
4. Seto, M.L.: Marine Robot Autonomy. Springer, New York, NY (2013). https://doi.org/10.1007/978-1-4614-5659-9. http://link.springer.com/10.1007/978-1-4614-5659-9
5. Wong, C., Yang, E., Yan, X.T., Gu, D.: Autonomous robots for harsh environments: a holistic overview of current solutions and ongoing challenges. Systems Science & Control Engineering **6**(1), 213–219 (2018). https://doi.org/10.1080/21642583.2018.1477634
6. Hagen, P.E., Hegrenæs, Ø., Jalving, B.,Midtgaard,Ø., Wiig, M., Hagen, O.K.: Making AUVs Truly Autonomous. In: Inzartsev, A.V. (ed.) Underwater Vehicles, pp. 129–152. I-Tech, Vienna, Austria (2009). Chap. 8. https://doi.org/10.5772/6700. http://www.intechopen.com/books/underwatervehicles/making_auvs_truly_autonomous
7. Jónsson, A., Morris, R.A., Pedersen, L.:Autonomy in space: Current capabilities and future challenges. AI Magazine **28**(4), 27–42 (2007)
8. German, C.R., Jakuba, M.V., Kinsey,J.C., Partan, J., Suman, S., Belani, A.,Yoerger, D.R.: 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), pp. 1–7. IEEE, Southampton, UK (2012). https://doi.org/10.1109/AUV.2012.6380753. http://ieeexplore.ieee.org/document/6380753/
9. Utne, I.B., Sørensen, A.J., Schjølberg, I.: Risk Management of Autonomous Marine Systems and Operations. In: Volume 3B: Structures, Safety and Reliability. American Society of Mechanical Engineers, Trondheim (2017). https://doi.org/10.1115/OMAE2017-61645. https://asmedigitalcollection.asme.org/OMAE/proceedings/OMAE2017/57663/Trondheim,Norway/280986
10. ISO: ISO 31000 Risk management -Principles and guidelines. International Organization for Standardization, Geneva, Switzerland (2018). https://www.iso.org/iso-31000-risk-management.html

11. Rajan, K., Saffiotti, A.: Towards a science of integrated AI and Robotics. Artificial Intelligence **247**, 1–9 (2017). https://doi.org/10.1016/j.artint.2017.03.003
12. Bremnes, J.E., Thieme, C.A., Sørensen, A.J., Utne, I.B., Norgren, P.: A Bayesian Approach to Supervisory Risk Control of AUVs Applied to Under-Ice Operations. Mar Technol Soc J **54**(4), 16–39 (2020). https://doi.org/10.4031/MTSJ.54.4.5
13. Coombes, M., Chen, W.-H., Render, P.: Site Selection During Unmanned Aerial System Forced Landings Using Decision-Making Bayesian Networks. Journal of Aerospace Information Systems **13**(12), 491–495 (2016). https://doi.org/10.2514/1.I010432
14. Qin, Y., Zhang, Q., Zhou, C., Xiong, N.: A Risk-Based Dynamic Decision-Making Approach for Cybersecurity Protection in Industrial Control Systems. IEEE Transactions on Systems, Man, and Cybernetics: Systems **50**(10), 3863–3870 (2020). https://doi.org/10.1109/TSMC.2018.2861715
15. Cai, B., Liu, Y., Xie, M.: A Dynamic- Bayesian-Network-Based Fault Diagnosis Methodology Considering Transient and Intermittent Faults. IEEE. Trans. Autom. Sci. Eng. **14**(1), 276–285 (2017). https://doi.org/10.1109/TASE.2016.2574875
16. Luque, J., Straub, D.: Reliability analysis and updating of deteriorating systems with dynamic Bayesian networks. Structural Safety **62**, 34–46 (2016). https://doi.org/10.1016/j.strusafe.2016.03.004
17. Gomes, I.P., Wolf, D.F.: Health Monitoring System for Autonomous Vehicles using Dynamic Bayesian Networks for Diagnosis and Prognosis. J Intell & Robot Syst. **101**(1), 19 (2021). https://doi.org/10.1007/s10846-020-01293-y
18. Hernandez, Y., Noguez, J., Sucar, E., Arroyo-Figueroa, G.: Incorporating an Affective Model to an Intelligent Tutor for Mobile Robotics. In: Proceedings. Frontiers in Education. 36th Annual Conference, pp. 22–27. IEEE, San Diego, CA (2006). https://doi.org/10.1109/FIE.2006.322407. http://ieeexplore.ieee.org/document/4116913/
19. Murray, R.C., Vanlehn, K., Mostow, J.: Looking Ahead to Select Tutorial Actions?: A Decision-Theoretic Approach. International Journal of Artificial Intelligence in Education (IJAIED) **14**, 235–278 (2004)
20. Conati, C.: Probabilistic assessment of user's emotions in educational games. Appl. Artif. Intell. **16**(7–8), 555–575 (2002). https://doi.org/10.1080/08839510290030390
21. Mott, B.W., Lester, J.C.: U-DIRECTOR: A decision-theoretic narrative planning architecture for storytelling environments. In: Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems - AAMAS '06, vol. 2006, p. 977. ACM Press, New York, New York, USA (2006). https://doi.org/10.1145/1160633.1160808. http://portal.acm.org/citation.cfm?doid=1160633.1160808
22. Bui, T.H., Poel, M., Nijholt, A., Zwiers, J.: A tractable hybrid DDN-POMDP approach to affective dialogue modeling for probabilistic frame-based dialogue systems. Natural Language Engineering **15**(2), 273–307 (2009). https://doi.org/10.1017/S1351324908005032
23. Codetta-Raiteri, D., Portinale, L.: Dynamic Bayesian Networks for Fault Detection, Identification, and Recovery in Autonomous Spacecraft. IEEE Transactions on Systems, Man, and Cybernetics: Systems **45**(1), 13–24 (2015). https://doi.org/10.1109/TSMC.2014.2323212
24. Trujillo, M., Martínez-de Dios, J., Martín, C., Viguria, A., Ollero, A.: Novel Aerial Manipulator for Accurate and Robust Industrial NDT Contact Inspection: A New Tool for the Oil and Gas Inspection Industry. Sensors **19**(6), 1305 (2019). https://doi.org/10.3390/s19061305
25. Mattar, R.A., Kalai, R.: Development of a Wall-Sticking Drone for Non-Destructive Ultrasonic and Corrosion Testing. Drones **2**(1), 8 (2018). https://doi.org/10.3390/drones2010008
26. Kocer, B.B., Tjahjowidodo, T., Pratama, M., Seet, G.G.L.: Inspection-while-flying: An autonomous contact-based nondestructive test using UAV-tools. Automation in Construction **106**(July), 102895 (2019). https://doi.org/10.1016/j.autcon.2019.102895
27. González-deSantos, L.M., Martínez-Sánchez, J., González-Jorge, H., Navarro-Medina, F., Arias, P.: UAV payload with collision mitigation for contact inspection. Automation in Construction **115**(March), 103200 (2020). https://doi.org/10.1016/j.autcon.2020.103200
28. Zhang, D., Watson, R., Dobie, G., MacLeod,C., Pierce, G.: Autonomous Ultrasonic Inspection Using Unmanned Aerial Vehicle. In: 2018 IEEE International Ultrasonics Symposium (IUS), vol. 2018-Octob, pp. 1–4. IEEE, Kobe, Japan (2018). https://doi.org/10.1109/ULTSYM.2018.8579727. https://ieeexplore.ieee.org/document/8579727/
29. Shani, G.: Task-Based Decomposition of Factored POMDPs. IEEE. Trans. Cybern. **44**(2), 208–216 (2014). https://doi.org/10.1109/TCYB.2013.2252009
30. Fenton, N., Neil, M.: Risk Assessment and Decision Analysis with Bayesian Networks, 2nd, editio Chapman & Hall/CRC, Boca Raton (2018)
31. Russell, S.J., Norvig, P.: Artificial Intelligence: A Modern Approach, 3rd edn. Pearson Education, Harlow (2014)
32. BAYESFUSION LLC: SMILE Engine. https://www.bayesfusion.com/smile/
33. Rausand, M., Haugen, S.: Risk Assessment, 1st edn. Wiley, New Jersey (2020). https://doi.org/10.1002/9781119377351. https://onlinelibrary.wiley.com/doi/book/10.1002/9781119377351
34. Leveson, N.G., Thomas, J.P.: STPA Handbook, MA, USA, p. 188 (2018). http://psas.scripts.mit.edu/home/get_file.php?name=STPA_handbook.pdf
35. Mkrtchyan, L., Podofillini, L., Dang, V.N.: Methods for building Conditional Probability Tables of Bayesian Belief Networks from limited judgment: An evaluation for Human Reliability Application. Reliability Engineering & System Safety **151**, 93–112 (2016). https://doi.org/10.1016/j.ress.2016.01.004
36. Ross, S., Pineau, J., Paquet, S., Chaibdraa, B.: Online Planning Algorithms for POMDPs. J. Artif. Intell. Res. **32**, 663–704 (2008). https://doi.org/10.1613/jair.2567
37. DNV GL AS: Non-destructive testing. Technical Report December (2015). https://rules.dnv.com/docs/pdf/DNV/CG/2015-12/DNVGL-CG-0051.pdf

Dr. Sverre Velten Rothmund received his MSc degree in 2018 and his PhD in 2023, both from the Department of Engineering Cybernetics at the Norwegian University of Science and Technology, Trondheim, Norway. His PhD was done in affiliation with the Center of Excellence on Autonomous Marine Operations and Systems (NTNU AMOS), where the focus of the PhD was on applying tools and concepts from the risk sciences in the control of autonomous systems. Sverre currently works as a senior engineer at the drone inspection technology company ScoutDI, where his work focuses on flight robustness and autonomy.

Dr. Christoph Alexander Thieme is currently a researcher at SINTEF Digital in Trondheim, Norway, where he applies his knowledge within the fields of risk, safety, human factors, and autonomous systems to different research projects related to technical safety and security of socio-technical systems. He holds a PhD in Marine Technology from NTNU, with specialization in safety, reliability, and risk assessment for autonomous systems. He is a co-organizer of the International Workshop on Autonomous System Safety and is Visiting Professor at the University of Toulon lecturing on Risk and Reliability engineering and potential application of AI methods.

Professor Ingrid Bouwer Utne is a professor at Department of Marine Technology, NTNU, where she performs research on risk assessment and modeling of marine and maritime systems. Utne is an affiliated Researcher in the Center of Excellence on Autonomous Marine Operations and Systems (NTNU AMOS). She is a principal investigator of the research projects UNLOCK and ORCAS. These projects focus on supervisory risk control and bridge the scientific disciplines of risk management and engineering cybernetics aiming to enhance safety and intelligence in autonomous systems. She is also responsible for the work package on Safety and Assurance in the Centre for Research based Innovation on Autonomous Ships (SFI Autoship).

Professor Tor Arne Johansen received the MSc degree in 1989 and the PhD degree in 1994, both in electrical and computer engineering, from the Nowegian University of Science and Technology, Trondheim, Norway. From 1995 to 1997, he worked at SINTEF as a researcher before he was appointed Associated Professor at the Norwegian University of Science and Technology in Trondheim in 1997 and Professor in 2001. He has published several hundred articles in the areas of control, estimation and optimization with applications in the marine, aerospace, automotive, biomedical and process industries. In 2002 Johansen co-founded the company Marine Cybernetics AS where he was Vice President until 2008. Prof. Johansen received the 2006 Arch T. Colwell Merit Award of the SAE, and is currently a principal researcher within the Center of Excellence on Autonomous Marine Operations and Systems (NTNU-AMOS) and director of the Unmanned Aerial Vehicle Laboratory at NTNU and the SmallSat Laboratory at NTNU. He recently co-founded the spin-off companies Scout Drone Inspection, UBIQ Aerospace, Zeabuz and SentiSystems.