













Article

An Overview of Verification and Validation Challenges for Inspection Robots

Michael Fisher ^{1,*}, Rafael C. Cardoso ¹, Emily C. Collins ¹, Christopher Dadswell ², Louise A. Dennis ¹, Clare Dixon ¹, Marie Farrell ³, Angelo Ferrando ¹, Xiaowei Huang ⁴, Mike Jump ², Georgios Kourtis ¹, Alexei Lisitsa ⁴, Matt Luckcuck ³, Shan Luo ⁴, Vincent Page ², Fabio Papacchini ⁴, and Matt Webster ⁵

¹ Department of Computer Science, University of Manchester, Manchester M13 9PL, UK; rafael.cardoso@manchester.ac.uk (R.C.C.); e.c.collins@manchester.ac.uk (E.C.C.); louise.dennis@manchester.ac.uk (L.A.D.); clare.dixon@manchester.ac.uk (C.D.); angelo.ferrando@manchester.ac.uk (A.F.); georgios.kourtis@manchester.ac.uk (G.K.)

² School of Engineering, University of Liverpool, Liverpool L69 3GH, UK;

C.M.Dadswell@liverpool.ac.uk (C.D.); Mjump1@liverpool.ac.uk (M.J.); V.Page@liverpool.ac.uk (V.P.)

³ Department of Computer Science, Maynooth University, Maynooth, Co., W23 N7F6 Kildare, Ireland; marie.farrell@mu.ie (M.F.); matt.luckcuck@mu.ie (M.L.)

⁴ Department of Computer Science, University of Liverpool, Liverpool L69 3BX, UK;

xiaowei.huang@liverpool.ac.uk (X.H.); A.Lisitsa@liverpool.ac.uk (A.L.); shan.luo@liverpool.ac.uk (S.L.); Fabio.Papacchini@liverpool.ac.uk (F.P.)

⁵ School of Computer Science and Mathematics, Liverpool John Moores University, Liverpool L3 3AF, UK; M.P.Webster@ljmu.ac.uk (M.W.)

* Correspondence: michael.fisher@manchester.ac.uk



Citation: Fisher, M.; Cardoso, R.C.; Collins, E.C.; Dadswell, C.; Dennis, L.A.; Dixon, C.; Farrell, M.; Ferrando, A.; Huang, X.; Jump, M.; et al. An Overview of Verification and Validation Challenges for Inspection Robots. *Robotics* **2021**, *10*, 67. <https://doi.org/10.3390/robotics10020067>

Academic Editor: Marco Ceccarelli

Received: 10 March 2021

Accepted: 26 April 2021

Published: 29 April 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

Abstract: The advent of sophisticated robotics and AI technology makes sending humans into hazardous and distant environments to carry out inspections increasingly avoidable. Being able to send a robot, rather than a human, into a nuclear facility or deep space is very appealing. However, building these robotic systems is just the start and we still need to carry out a range of verification and validation tasks to ensure that the systems to be deployed are as safe and reliable as possible. Based on our experience across three research and innovation hubs within the UK's "Robots for a Safer World" programme, we present an overview of the relevant techniques and challenges in this area. As the hubs are active across nuclear, offshore, and space environments, this gives a breadth of issues common to many inspection robots.

Keywords: robotic inspection; software architectures; formal verification; testing

1. Introduction: Sending Robots to Look at Things

In this paper, we present an overview of current techniques for overcoming the verification and validation challenges of robotic systems used for remote inspection. This is the collected experience of working across three (out of four in total) research and innovation hubs that are part of the UK's "Robots for a Safer World" programme (<https://www.ukri.org/our-work/our-main-funds/industrial-strategy-challenge-fund/future-of-mobility/robots-for-a-safer-world-challenge/> (assessed on 28 April 2021)). These three hubs cover the nuclear (RAIN hub), offshore (ORCA hub), and space (FAIR-SPACE hub) sectors. Each of these hubs provides examples of inspection robots that pose a variety of challenges to verification and validation activities. Some challenges stem from the deployment environment, others from the robot's implementation.

In this section, we introduce: in Section 1.1, why it is relevant to use robots for inspection tasks; in Section 1.2, what kind of robotic systems we are interested in; in Section 1.3, how remote inspection tasks can be performed (for example using teleoperation or autonomy); and finally, in Section 1.4, where these inspection robots can be deployed (that is, their deployment environment). We finish the section, in Section 1.5, with the research questions that this paper addresses together with an outline of the rest of the paper.

1.1. Why?

It is clear that sending humans into hazardous, or distant, environments is dangerous; sending robots instead can significantly reduce human risk. It is much less clear that sending robots rather than humans will result in increased effectiveness or efficiency concerning the inspection tasks.

There are a wide range of environments for which we might consider robotic deployment, corresponding to a plethora of hazards. Purely for safety reasons, we might wish to send robots into situations with any of the following hazards:

- Chemicals and Radiation—both in space and on earth, e.g., nuclear fission/fusion.
- Explosions and Fire—exploring burning buildings, forest fires, rescue scenarios, etc.
- Liquids and Flows—underwater, in sewers, handling strong currents, external pipeline inspection, inspection inside pipelines, etc.
- Wind and Weather—inspecting structures such as buildings, turbines, or oil-rigs, in bad weather, or inspecting meteorological phenomena themselves.
- Pressure and Heat—e.g., from water (deep sea), from earth (deep excavations), from sunlight, as well as from fire.
- Lack of atmosphere—e.g., space exploration, deep sea environments.
- Dangerous flora or fauna—e.g., inspecting hornets' nests (Inspecting Hornet Nests—<https://www.bbc.co.uk/news/world-europe-jersey-41141330> (assessed on 28 April 2021)), or areas infected with dangerous microorganisms.

Yet, beyond these there are a variety of much less hazardous, but often important, inspection tasks that humans currently undertake but that robots could be employed to handle:

- Inspecting structures for leaks, for corrosion, or for general wear-and-tear;
- Security inspection within restricted areas;
- Monitoring human safety, e.g., an isolated person;
- Conservation tasks and wildlife monitoring (Spotting sharks—https://www.youtube.com/watch?v=z007p-8e_QA (assessed on 28 April 2021)).

While all of these tasks might well be very effectively carried out by humans there may be reasons, such as cost, duration or unwillingness, that may mean that a robotic solution is more appropriate.

As such, the idea of sending robots to inspect areas or assets can be quite appealing. However, all such systems not only need design and engineering, but effective verification and validation. Without the latter, engineers, regulators, and users will have little confidence that these robotic solutions are indeed safe, reliable and effective.

In this article we will look at common verification and validation aspects across all the above type of systems, which we characterise as inspection tasks. We will not be getting the robot to directly impact anything, for example through manipulation, and will especially avoid human contact. While manipulation tasks will clearly be important in the future, deliberate, physical interaction between robots and humans or artefacts is both complex and contentious. Moreover, the class of inspection robots that we address here is already quite broad.

1.2. What?

Once we decide we need a robotic device for use in an inspection task, there are a wide range of options for its construction and programming. These can range from simple sensors all the way through to large, multi-functional robots and vehicles. The system may comprise (depending on the environment):

- a variety of construction materials—metal, soft materials, nano-technology, etc.;
- a variety of sensors for inspection—infrared, pressure, lidar, radioactivity, heat, non-destructive testing (NDT) components, etc.;
- a variety of propulsion mechanisms—wings, wheels, propellers, buoyancy, etc.;
- a variety of communication mechanisms—radio, WiFi, speech, etc.; and

- a variety of software components—for deep learning, explanation, planning, decision-making, navigation, etc.

As our main goal is inspection, we do not expect to be manipulating assets. Nevertheless, our robotic device might have a variety of tools—drills, grippers, lasers, etc.—to be used either in an emergency or in carrying out its inspection. The construction materials and resilience aspects are also important, especially when the robot will be deployed in hazardous environments. Clearly, in extreme environments such as space, nuclear, or deep sea, these resilience aspects will be vital.

As we build more sophisticated robotic systems and as we expect them to carry out more complex tasks, software becomes increasingly crucial to many aspects of the system. Key software sub-components might address sensor filtering and data fusion, simultaneous localisation and mapping (SLAM) algorithms, planning, fault detection and diagnosis, prediction and decision making, control of mobility, communication and interaction, etc.

1.3. How?

Once we have a robotic device designed to achieve some inspection task, and some environment to deploy it in, we need to decide how to go about the task. In traditional work, we would have the human operator close to (perhaps even carrying) the inspection device and directly controlling its activity. For example, a human operator might have a camera or NDT component on a pole and may move it around to the desired positions. However, as we move to more advanced robotic inspection then further possibilities emerge.

Rather than having the operator directly (physically) controlling the device, we might have a remote controlled/piloted version. Here, the operator controls the device at a distance via some electronic communications link. This remote pilot/control option is very popular as the first step towards autonomy since (a) the human operator is removed from the dangerous environment while (b) the human is still in control of the device. However, as we will see in Section 2, these types of systems may well be error prone and inefficient.

As we increase the level of autonomy used, the operator can increasingly delegate greater responsibilities to the device itself. In many cases this improves efficiency, since the local decision-making of the autonomous device can be a step up from the distant decision-making of the operator, but now the decision-making is (to some extent) taken away from the operator. As the delegation of significant tasks to the device increases then eventually the device will be fully autonomous, being responsible for all of its key actions and decisions. This high level of autonomy clearly causes a problem for many; not only engineers, but users and regulators. Again, as we will see in Section 2, there are also verification techniques that can be used to assess these, more autonomous, systems and to gain confidence in their activities.

1.4. Where?

The environments within which these robots might be deployed are very varied, ranging from extremely hazardous (nuclear, space, etc.) to the more mundane (though still not pleasant for humans) such as sewers, precarious structures, etc. We will not consider examples of every one of these, but some typical scenarios include:

- land (underground)—e.g., tunnel inspections;
- land (surface)—e.g., looking for pot-holes on a road surface;
- land (extra-terrestrial)—e.g., looking for specific minerals on a planet;
- water (surface)—e.g., assessing wind turbine support structures;
- water (sub-surface)—e.g., assessing sea-bed pipelines;
- air—e.g., assessing high oil-rig structures;
- space—e.g., assessing satellite structures;
- in confined/constricted spaces—e.g., assessing internals of pipes;
- in adverse weather conditions—e.g., carrying out inspections of structures in high winds;
- ... and so on ...

In all these environments, there will be:

- a range of different dangers that the human, and now the robot, might be exposed to,
- a range of different inspection tasks to be carried out, and
- a range of different requirements we place on our robots.

While most requirements need several verification (and validation) techniques to be used, different scenarios and deployments will often require different combinations of techniques. Nevertheless, as we will see through this article, there is a (finite) set of verification and validation techniques that can be brought together to tackle the analysis of many, if not all, robotic inspections.

1.5. Research Questions and Paper Outline

This paper seeks to address the following three research questions for robotic systems used to perform remote inspection:

RQ1: What are the verification and validation challenges for robotic remote inspection systems?

RQ2: What existing verification and validation techniques can we use to tackle the challenges found in answering RQ1?

RQ3: What are some of the remaining challenges not tackled by the existing techniques found in answering RQ2?

As previously described, to address these issues we present our combined experience from working on verification and validation across three research and innovation hubs within the UK's "Robots for a Safer World" programme, between 2017–2021. We also make use of our (joint) previous experience working within the research communities of formal methods, software engineering, autonomous systems, and robotics.

The remainder of this paper is structured around four main issues, often encountered during the verification and validation of a (autonomous) robotic system. In Section 2 we discuss responsibility and how, depending on the level of operation control (human, autonomy, etc.) in the system, different verification and validation techniques may be more or less appropriate. Section 3 concerns the reliability of individual components in the system, such as controllers, neural networks for processing images, or symbolic autonomous agents, while in Section 4 we look into system reliability and how to provide assurances about the system as a whole. In Section 5 we highlight some of the classes of properties and requirements commonly found in autonomous robotics and, finally, we provide a summary and future research directions in Section 6.

2. Verifying and Validating Responsibility: Who or What Is "In Charge"?

The core aspect of autonomy is that a system, rather than a human operator, driver, or pilot, will make significant decisions and possibly take critical actions. The levels of responsibility are captured by a number of different 'levels of autonomy' scales, such as the PACT characterisation [1]. All such scales range from full human control to fully autonomous systems, often by delegating increasing responsibility from the human to the system. While levels of autonomy scales can be helpful for discussions of how much decision-making responsibility the system has, it is important to be clear about which scale is being used.

Throughout many industries, particularly those that involve hazardous environments, the increased use of robotics is natural. This moves humans from direct danger and so quickly increases safety. However, the first step along the route from direct human control is to move to remote human control. Very often this has an adverse effect on efficiency, since fast and effective remote control is extremely difficult. This may also be exacerbated by, for example, communications delays (e.g., during planetary exploration) and poor situational awareness (e.g., from weak sensing/vision). In this sense, using robots to replace humans in various tasks will lead to increased safety but decreased efficiency [2]. We can capture these views (often anecdotal) in the diagram in Figure 1.

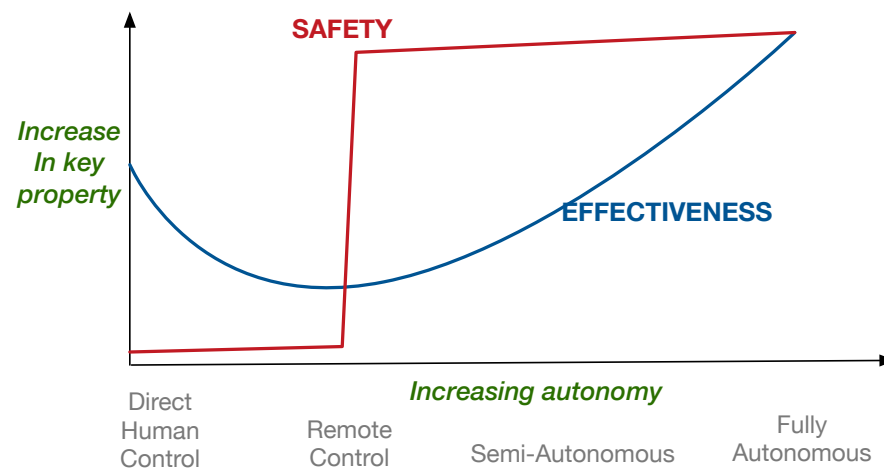


Figure 1. Representation of the effect of increasing autonomy on both safety and effectiveness.

Clearly, relative safety increases significantly once we move the human away from danger. It is only once we move away from remote control and delegate much more responsibility to the system that efficiency can potentially be regained. However, once we take this step towards greater autonomy we will need much stronger verification since there is no human to ‘fall back’ on when difficult decisions are needed.

In the sections below, we consider the verification and validation challenges of a selection of different responsibility levels, from direct human control through to full autonomy.

2.1. Direct Human Control

Here, we are very much in a ‘normal’ situation with the human pilot, driver, or operator being in complete control of key decisions and actions. Systems are verified for reliability but there is nothing here that is different to V&V in standard cyber-physical systems. However, since the human must be near the system (in order to be in direct physical control) they are exposed to the potentially dangerous environments.

2.2. Remote Human Control

Here, there is still a human operator in ‘charge’ but the robot will likely be out of sight and, sometimes, very distant. The human operator has delegated very specific responsibilities to the system, not in terms of decision-making but in requiring that the system provide effective images/sensing of the actual situation and that the operator’s commands are transmitted effectively and accurately to the remote device.

EXAMPLE: The Joint European Torus (JET) fusion reactor, and its In Vessel Training Facility (IVTF) are remotely controlled by a team of five operators with the use of a two-armed, master–slave tele-manipulator named MASCOT. Each of MASCOT’s arms has seven degrees of freedom and a force-reflecting servo-system that provides haptic feedback to the human operator. It is used for training, inspection, maintenance and reconfiguration.

V&V TECHNIQUES: In order to assess the operation of remotely human-controlled systems a number of V&V techniques may be employed:

- physical testing;
- user validation—asking operators how they find use of the system;
- simulation-based testing (and training);
- formal verification, for example as part of corroborative verification [3].

ISSUES: Remotely controlling a robot that is inaccessible to the human operator poses a number of complex challenges. The lack of access to the local system (which is operated at a distance by the master control system) leaves the human controller reliant on sensors and feedback to navigate, hampering situational awareness. The precision required to

operate a robot removed from the human also leads to increases in stress responses, as mistakes made and unable to be rectified at the local-end, could lead to system shutdown. Furthermore lack of transparency in system commands contributes to confidence and trust issues between the human controller and the robot, especially in the case of unforeseen actions by the local-end.

2.3. Supervised, or Semi-Autonomous, Behaviour

The direct remote control of robots can be inaccurate and error-prone [2] and so an appealing option is to delegate some of the control to the robot itself, with the (distant) human operator taking a more supervisory role. For example, rather than having to remotely pilot a Unmanned Aerial Vehicle (UAV) inspecting an offshore oil platform, the operator might just highlight a target area to inspect and then allow the UAV to fly to that area itself. This has the advantage that autopilot software can deal with local adjustments that distant human operators find difficult. The human operator still makes all of the key decisions, but low-level interactions are delegated.

EXAMPLE: Offshore energy assets, such as wind farms and oil platforms, are dangerous and difficult to access for human workers. Consequently, such activities are also time-consuming and expensive. Unmanned aircraft are increasingly used to perform remote inspection tasks in which different parts of the offshore structure are examined using cameras or touch sensors (see Figure 2).



Figure 2. Unmanned aircraft operation near offshore structures. Source: ORCA—<https://orcahub.org/> (assessed on 28 April 2021).

An increasingly common approach allows operators to use an advanced ground control station to determine mission goals, select key waypoints, and flight characteristics such as risk tolerance and timing constraints. This is in contrast to traditional remote operation in which the human operator, for example, remotely manipulates control surfaces on the unmanned aircraft to effect manoeuvres. In the ‘ground control station’ mode of operation the operator controls the mission at a higher level of abstraction, providing goals for autonomous systems on-board the unmanned aircraft to work towards. For example, in remote inspection of an oil platform, the operator may direct the unmanned aircraft to particular points of interest. An autonomous system takes these points of interest and generates a route plan which is then enacted using complex flight control systems that do not require direct operator input. Sensors and cameras on-board the unmanned aircraft provide the operator with real-time information on the vehicle systems and progress of the mission (see Figure 3), therefore allowing human oversight (and intervention if necessary).

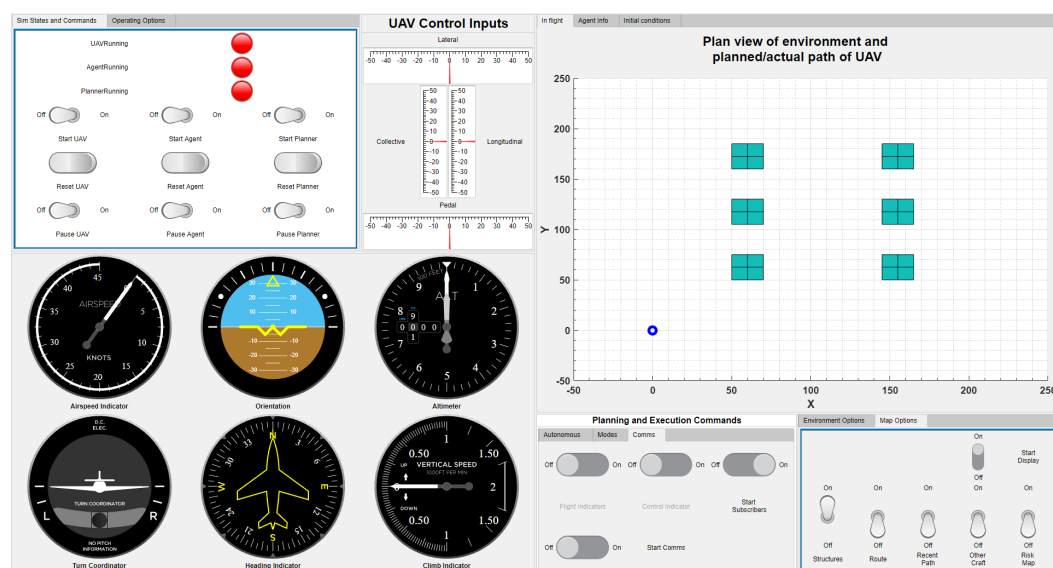


Figure 3. Unmanned aircraft ground control station software.

V&V TECHNIQUES:

- Traditionally, analytic stability/avoidance proofs for control algorithms based on control theory are used [4].
- More recently, simulation-based testing [5] has been used to develop a virtual prototype, or digital twin, of the unmanned vehicle.
- Formal methods can be used [6,7] but can quickly become intractable (if used throughout) without significant abstractions or approximations of the environment.
- At later stages in the product development lifecycle, hardware-based techniques such as hardware-in-loop and real-world testing can be used [3]. The latter can be costly, however, and is generally left until the prototype has reached a mature point in its development.

ISSUES: There are several issues that remain to be addressed, as follows.

- A central concern when using ground control stations with semi-autonomous systems, such as unmanned aircraft, is what happens if the communication link with the unmanned aircraft is disrupted. In the worst case, the unmanned aircraft could become an ‘uncontrolled missile’. Consequently, it is necessary to have some fallback systems on-board the aircraft to take over and maintain safe operations [8].
- Since safety parameters need to be assigned for the robot or vehicle, there will be a considerable amount of calibration needed to ensure the appropriate levels that retain both effectiveness and safety. This calibration might potentially have to be carried out separately for every new mission.

2.4. Autonomy

As we delegate more control to the system we might choose to delegate responsibilities that have greater importance, for example in terms of business-, safety-, or mission-criticality. While in semi-autonomous behaviour above, the human operator/pilot/driver still made the key decisions, we might now let the autonomous system (typically, its software) make these decisions. Part of the justification for doing this is as in the above, semi-autonomous, situation: the local decisions made by the system in the specific environment are likely to be more effective, and certainly more timely, than decisions made by a distant, and possibly partially informed, operator. Furthermore, if we choose to move in this direction we can also prescribe restrictions and bounds on any decisions made (and prove that decision-making will abide by these bounds, see below). However, given the nature of these cyber-physical systems, acting in partially known environments, then

all possible decisions cannot be prescribed. So, we will come to a point where either we prescribe all of the system's available decisions beforehand; or we ensure that the decision-making process is flexible and trustworthy enough to make the 'right' decisions, even in novel or unexpected situations.

Clearly, allowing such levels of autonomy is a significant step and we will require substantially more verification and validation to allow us to be confident in this step.

EXAMPLE: An autonomous robot is inspecting barrels containing nuclear waste to categorise them for decommissioning. The robot detects what appears to be a leak from one of the barrels with liquid slowly making its way towards its position. Does the robot have the necessary sensors to differentiate between a harmful spill and just spilled water? Should the robot analyse the spill or should it just move out of the way and carry on? If the robot detects another person in the building, should it assume that they work there or that there is an intruder? Who should the robot inform about the spillage/person, if anyone?

While all of these decisions and actions might normally be taken by a competent human inspector, we have now delegated these to our robotic inspector. In most non-trivial environments, the answers to such questions cannot all be encoded beforehand and so the autonomous system must make the most appropriate decision, given the information that it has at the time.

V&V TECHNIQUES:

- If we have decisional bounds, or a set of prescribed decisions, then we can:
 1. formally verify that the system always works within these restrictions;
 2. test a range of scenarios to ensure that the system always works within these restrictions—this might be real physical testing or testing in simulation;
 3. use runtime verification to monitor whether the system follows these restrictions in practice.
- If we have fully autonomous decision-making, then we can formally verify this process to ensure that the system always tries to make the right decisions and takes the important aspects into account [9]. Note that we can never guarantee what the external effects of any decision will be.

ISSUES: In this area there are a range of outstanding issues:

- Once we delegate some (limited) autonomy to the system then new issues come into play. For example, are correct decisions always made, given the information available? Moreover, are these decisions made for the right reasons, i.e., the robot has 'the best of intentions'? Related to this, what guarantees can we give? Can we formally verify this?
- Particularly concerning the aspects delegated, can the system effectively explain its decisions to humans? This, and the previous issue, rely on whether the decision-making is transparent, which is a basis both for explainability and verifiability.
- How effective and reliable are self-awareness and fault-recognition capabilities? For example, can a robot detect when one of its sensors is working badly? Similarly, given health management procedures, is the robot still able to function? In particular, in spite of failures, can the robot still achieve its aims?
- How is privacy dealt with, for example ensuring that personal data is not revealed to the wrong people? What guarantees (verification) can we provide for this aspect [10]?
- Finally, is the locus of responsibility clear? Who is in charge, the robot or the person, and who or what is to blame if things go wrong? Again, verification (and validation) is needed for these crucial issues.

2.5. Summary

As we have seen, there are certain verification and validation techniques that may be more appropriate depending on how much responsibility has been delegated to the robot. Systems under close human control (whether direct- or remote-control) usually rely

on physical testing and user validation because of the vital presence of a human in the loop. However, as the level of autonomy in the system increases, it is necessary to employ techniques that are more automated and formal to provide greater assurances about the safety and effectiveness of the software systems. Focusing these techniques on the software and eliminating as many bugs as possible before the robotic system is ever physically tested can also help reduce the likelihood of early-stage physical robots tests being dangerous.

3. Component Reliability: What, When, and Why it Works?

Individual components within our robotic system can be constructed using a range of different techniques. These components may be programmed in a wide variety of ways, for example:

- a single control or state estimation system,
- a neural network, or
- a symbolic AI system.

These components may also be linked by a software framework using a general-purpose programming language, such as C++ or Python. Note that we are not considering mechanical, material, or other hardware aspects as the same testing that is used as standard [11] will be used in inspection robots. We are primarily considering the additional software aspects, each of which leads to a range of possibilities for verification as follows.

3.1. Controller or State Estimation System

The field of Control Systems has a range of well-established techniques for developing controllers. Guarantees about behaviour are usually couched in terms of Applied Mathematics with analytical proofs being used to establish these. As many of these control systems involve feedback mechanisms, wherein the behaviour of the system is modified by input from its environment then, any mathematical estimates will be based on inexact models of the environment. Consequently, such systems are usually verified through testing rather than any specifically formal verification (since the differential equations describing the control theory ensure that formal approaches are problematic).

In addition to the control, another key component for robotics is to understand the external environment, by e.g., simultaneous localization and mapping, or SLAM. A SLAM system uses solutions, such as Kalman filters and covariance intersection, to estimate and continuously update the map of the environment and simultaneously keep track of the location of the robot.

EXAMPLE: Motion control refers to the low-level control behaviour of a robot. For example, after motion planning, an inspection robot decides to travel from one waypoint *A* to another waypoint *B*. This is generally implemented by a motion control module, which determines the robot's motion according to physical models by continuously monitoring and updating various parameters such as longitudinal velocity, yaw rate, longitude force, etc. A state estimation module, for example SLAM, is generally used to model the robot's environment and localise it within that model.

V&V TECHNIQUES:

- Traditionally, analytic stability/avoidance proofs for control algorithms [12,13].
- Formal verification techniques for controller [14,15] and state estimation systems [16,17] are available by generating formal models out of the underlying physical or software models.
- Simulation-based testing and user validation are also frequently used in this aspect.

ISSUES:

- Analytic proofs regarding control are dependent on mathematical models of (probably unknown) environments—in the case of distant (and even hazardous) environments, we are not at all sure exactly how accurate these models are. Consequently, while we know that 'all mathematical models are wrong', we are not sure exactly how wrong!

- Formal methods can quickly become intractable without severe abstractions/approximations concerning the practical environment. Further, there can be a significant number of false positives or false negatives, when considering such abstraction techniques.
- Simulation-based testing and user validation may not be able to identify all ‘corner’ cases.

3.2. Neural Networks and Sub-Symbolic AI

For some tasks for which symbolic programming (see Section 3.3) is hard to achieve a good level of performance, machine-learning techniques can be applied. For example, deep learning, based on neural networks, has been widely applied on perceptual tasks to process sensory inputs. Deep reinforcement learning is also gaining momentum on motion control of robotics [18]. Deep learning requires large amounts of labelled data for training, which is becoming possible with the advancement of technologies such as IoT, 5G, etc.

Deep Learning and Deep Reinforcement Learning have achieved significant breakthroughs in various perceptual and control tasks in the past decades. However, most of these algorithms encounter problems when generalizing the trained models to novel environments with significant differences from the training environment. Furthermore, the representations and policies learned by the deep neural networks are usually hard to interpret. To address these challenges, there have been attempts to represent the states by first-order logic that are fed into neural networks, so that the learned representations can be interpreted and generalised to novel environments [19,20].

EXAMPLE: An underwater inspection robot may use one or multiple sensors—such as compasses, visual cameras, depth sensors, sidescan and other sonars, magnetometers, thermistors and conductivity probes—to receive environmental information. For perception tasks, some state information may be determined from sensory inputs, e.g., whether there is an obstacle and what is the relative location of the obstacle. These can be implemented with convolutional neural networks. For motion control, a deep reinforcement learning model can be applied to guide the vehicle to move from one location to another location by determining, e.g., the rudder angle position and thrust.

V&V TECHNIQUES: The development of V&V techniques for deep learning models has been very active in the past few years (see [21] for a recent survey). They can be roughly categorised as formal verification methods [22–26], software testing methods [27–29], and statistical methods [30,31]. In general, formal verification methods are used to determine if a property—for example, the robustness of a given input—holds or not on a given trained model. Software testing methods consider developing coverage metrics [27] and test case generation methods, and statistical methods are to provide statistical evaluation of some property by either variants of Monte Carlo sampling [30] or studying the weights of the model [31].

ISSUES: Verification has proven hard for deep learning, with the following major issues for different categories of verification methods:

- Formal verification methods usually involve not only high computational complexity but also large state space due to the size of neural networks.
- Software testing methods require extra effort to make sense of the testing results. For example, it needs to clarify the relation between coverage rate and the satisfiability of a property.
- Statistical methods can only achieve statistical results and may be subject to the risks of ignoring corner cases.

Beyond their individual issues, the V&V techniques for deep learning are focusing on specific low-level requirements. It has become a serious question on how to utilise the evidence collected by these V&V methods to support the safety argument of a robotics system [32,33].

3.3. Symbolic AI and Logical Reasoning

Symbolic Artificial Intelligence [34] encompasses techniques that are concerned with the representation of AI problems at a high-level through the use of symbols, simulating the human reasoning process. Rational Cognitive agents [35,36] are a technique from symbolic AI that reason about incoming perceptions from the environment and the internal state of the agent and then decide on a course of action. Traditionally, agent programming languages have adopted the well known Belief-Desire-Intention (BDI) model [37,38] to reason about the following mental attitudes: beliefs that describe the agent's knowledge about the world; desires that correspond to the goals that the agent wants to achieve; and intentions that represent the commitment and know-how of the agent to achieve a particular goal. BDI agents have been successfully applied to perform high-level decision making in robots, as evidenced recently by [39–43]. In addition to BDI agents, other intensively-studied symbolic reasoning approaches include temporal reasoning [44], strategic reasoning [45–49], epistemic reasoning [50–55], and trust reasoning [43].

EXAMPLE: Robotic systems are often complex and modular. For example, an autonomous rover deployed to perform remote inspection tasks can have nodes for: low-level control of its actuators, neural networks for image classification using data from camera sensors, path and task planning algorithms, among others. If implemented ad-hoc, decision making will often be spread out among this large number of nodes. Embedding a rational cognitive agent in a robot allows us to effectively concentrate the high-level decision-making into a single node, making it easier to iterate the code and verify the decisions that can be made by the agent.

V&V TECHNIQUES: The most prominent V&V techniques for agent systems in the literature can be broadly categorised into model checking, theorem proving, runtime verification, and testing. A recent survey of the application of these approaches in agent systems can be found in [56]. Agents have also received specific attention in the formal specification and verification literature [57]. Some examples of techniques in each of these categories include:

- Model checking: the Model Checking Agent Programming Languages (MCAPL) project [58] includes the GWENDOLEN [59] programming language for programming BDI-based agents, and the Agent JavaPathFinder (AJPF) [60] program model checker for verifying GWENDOLEN agent programs; MCMAS [61] and MCK [46] are two symbolic model checkers for multi-agent systems based on ordered binary decision diagrams.
- Theorem proving: particularly for symbolic agents, there are a range of formalisms and theorem-proving approaches, such as combined temporal and modal logics [62,63]; additionally a logic for automatic verification of simple agent programs is presented in [64].
- Runtime verification: in [65] runtime verification is used to validate the abstract environment model of agent systems at runtime, that were previously verified offline by a program model checker; DecAMon [66] is an algorithm for decentralised monitoring of communication in multi-agent systems via agent interaction protocols.
- Testing: in [67], it was shown that BDI agent programs are more difficult to test than procedural programs of equivalent size, and conclude that their findings motivate the use of formal methods for the verification of agent systems.

ISSUES:

- Symbolic AI methods can be inflexible and require significant knowledge engineering effort to produce, when compared to learning based approaches, although this same effort enables tracability from implemented program features back to user requirements.
- Model-based verification techniques have significant state space explosion problems, which are exacerbated in the verification of agents because of the increased complexity required in their reasoning cycle.

3.4. Summary

In Table 1, we summarise the verification techniques for each type of approach to autonomy (controllers, sub-symbolic AI, and symbolic AI) found in the inspection robots that were discussed in this section.

Table 1. Examples of verification and validation techniques for component reliability.

	Model Checking	Theorem Proving	Runtime Verification	Automated Verification	Software Testing	Other
Controller	—	—	(1) [17]	—	(1) [16]	(5) [12–15,17]
Sub-Symbolic AI	—	(1) [22]	—	(5) [23–26,31]	(4) [27–29]	(1) [30]
Symbolic AI	(4) [46,58–61]	(2) [62–64]	(2) [65,66]	—	(1) [67]	—

This provides an indication of our collected experience during the three research and innovation hubs, not an exhaustive review of all the techniques that could be applied across these three areas. Nevertheless, we can see that model checking is predominant in Symbolic AI, mostly due to our use of rational agents (for example BDI agents) being amenable to formalisation and program model checking. On the other hand, sub-symbolic AI techniques (such as neural networks) are difficult to model and verify because of their complexity; consequently automated verification (such as constraint satisfaction) and non-formal techniques (such as software testing) provide more tractable approaches in this area. Finally, controllers can often benefit from runtime verification due to the difficulty in representing the dynamic environments that these controllers are often exposed to (the dynamic deployment environment is often a driver for using robots for remote inspection tasks) as well as other verification and validation techniques such as mathematical models and analytic proofs.

4. System Reliability: Putting It All Together

Autonomous systems typically comprise software components for decision-making and communication, with embodied autonomous systems (e.g. ‘robots’), requiring additional (physical) components, such as sensors and actuators (e.g. arms, wheels, propellers). Each physical component will also have some form of software control component that manages its behaviour. Once we have such a range of software components, they can be organised into numerous different architectures [68] to achieve the tasks associated with the overall system.

4.1. Architectures

Individual components within the system can be organised in a number of architectures. In an extreme case, the system might comprise just one (large, complex) component. More likely there will be some hierarchical or modular organisation. In each case, verification is required.

Monolithic. It is, of course, possible that the software for our system is encapsulated within a single, large, monolithic entity. For example, a neural network controlling all of the hardware (sensors, motion, communication, etc.). In such a case, we treat the whole (monolithic) system as one component and apply individual component V&V as above. Given that such a component will likely be large and complex, then the amount of deep V&V that we can carry out will also likely be limited.

Modular. Monolithic systems are increasingly rare and, in particular, the concept of modularity has now been widely adopted, both because of new international standards (such as ISO/DIS 22166) and through the widespread use of de facto standards such as the Robot Operating System (ROS) [69,70]. Such modularity separates key architectural elements and permits improved design, analysis, and maintenance. In addition to permitting and supporting inter-operability amongst sub-components, modularity will aid us in providing:

(1) a natural basis for compositional verification, so improving the viability and efficiency of formal approaches [71]; and (2) a basis for the heterogeneous verification techniques required across autonomous systems [72].

Once we take a modular approach, there are issues to be addressed, such as:

- specification of the behaviour of an individual module;
- specification of the linkage between, and communications to, other modules;
- flexibility in exchanging modules (reconfigurability, etc.);
- security, reliability, and real-time efficiency of modular interactions;
- etc.

Hierarchical. If we have a collection of controllers managing the system's physical interactions we can, in turn, generate a control system that manages a set of such sub-components. We can think of basic controllers as sitting at the bottom of this hierarchical control system [73,74]. Above these is a further layer of controllers, each of which manages some subset of the systems at the bottom layer. Further control elements, in the next layer up, handle sets of manager controllers, and so on. This idea of an abstraction 'hierarchy' also influences subsumption architectures that have been pivotal in robotics [75]. As in hierarchical structures, the higher layers represent more abstract behaviours than the lower layers. Individual layers in a subsumption architecture take input directly from the sensors, and data is transmitted between the nodes of each layer in order to form a 'behaviour'. A higher layer can subsume a lower layer by inserting new information into the links between its nodes [75].

Hybrid. As well as control-system modules and neural network modules, we will likely have symbolic modules, for example relating to planning [76,77]. In these, a plan is constructed given the current world model and the desired goals. Symbolic components have many benefits and while we might construct a full system from symbolic components, it is more likely that we will utilise a hybrid approach, combining symbolic/sub-symbolic modules as necessary. For example, hybrid agent architectures [78] have symbolic agents dealing with the high-level decision-making, while other components deal with (symbolic) planning and (sub-symbolic) feedback control. V&V for such hybrid architectures present a novel set of challenges (see below).

Patterns. While there can be a vast range of different architectures for robotics and autonomous systems, it is clear that there are many common elements. Patterns describe generic, recurring situations, and are widely used in areas as diverse as Software Engineering [79] and Architecture in order to explore wide-ranging solutions. Unsurprisingly, patterns concerning robotic missions, and likely organisations, can be specified [80]. This can reduce the scope of V&V issues to a slightly more manageable range.

Given all the above, we will now look at issues concerning the verification and validation of robotic systems.

4.2. Physical Testing

Testing the robot in the practical environment that it will be used in is, of course, important. However, for many of the environments that we are concerned with there are two issues. The first is the general problem with physical testing that it is impossible to replicate tests. Once we have tested our robot, something will have changed when we come to test it again or use it. This change might be in the robot itself (energy, degradation, learning) or might be in the environment (temperature, air/water flow, radiation). The second issue is that for robotic deployments in space, in nuclear environments, and even offshore, it is often infeasible to carry out many tests before deployment. An obvious example is a Mars rover—we clearly cannot test this on Mars before sending it off for its mission.

4.3. Laboratory Testing

The traditional answer for both of the above problems is to try to recreate the environmental conditions in a laboratory setting. For example,

- Nuclear—we might set up irradiation and obstacles, and expose our robot to both,
- Water—we might set up a laboratory tank with (erratic) water flows and test our robot within this, and
- Space—we might utilise a desert landscape as an approximation for a lunar or Martian surface.

These are clearly very important but, again, we are only able to test a few dimensions and only for a few times. We do not have enough evidence to assert the likelihood of success. Not to be under-estimated is the problem that such tests are expensive.

EXAMPLE: A mobile inspection robot in a nuclear environment needs to identify obstacles and potential hazards with sensors such as visual cameras and laser scanners, while being exposed to extreme levels of radiation exposure. One can set up a laboratory environment without the nuclear radiation to simulate the real nuclear plant to test the performance of the developed robot in localisation, mapping and obstacle detection. However, the visual cameras and laser scanners may fail to function in the nuclear environments. Nuclear reactors can be simulated in a laboratory environment to test the performance of these sensors under radiation, but it is always expensive to generate radiations and it will have risks of radiation leaks, posing a threat to the health of the researchers. Furthermore, it is challenging to simulate the real nuclear plant due to the fact that light conditions, terrains, layout of the plants may be hard to be duplicated in a laboratory.

V&V TECHNIQUES: Crucial to laboratory-based testing are:

- the design of experiments,
- the fidelity of the lab model of the real environment, and
- the ways in which additional failures/constraints can be represent, e.g., vision system failure.

ISSUES: Most of the current robot systems are tested and verified in the laboratory settings, which will have a significant gap to the real environments that they will be deployed to. The gap may arise from different aspects, e.g., different light conditions, weather, radiation levels, terrains, water flow conditions, which may be hard to simulate in laboratory testing.

- We must be clear about the details of this gap. What aspects do we simulate well; what elements are less realistic; which ones do we not even represent?
- We must also be precise about failures that might occur in the real environment and whether these can occur in the lab-based system. And, if not, are these to be modelled/simulated? For example, radiation might affect a particular sensor in a strange way, gradually degrading its capabilities. We can introduce software to simulate this aspect in a laboratory setting.
- Given the above explorations relating to the gap between real and laboratory environments, we can have an estimate of all of the elements of the environment to be covered. Then, experiments can be designed that address many, though probably not all, issues. This aspect of experimental design and coverage of the issues, is both complex and important.

4.4. Simulation-Based Testing

High fidelity computer simulations can be used to build virtual prototypes (sometimes termed as “digital twins”) of the system under examination [81]. Such simulations generally use detailed physical models of the system and its environment. Simulation-based testing is particularly useful for examination of critical systems and/or advanced systems that use autonomy as traditional V&V techniques for such systems are often expensive and dangerous [82]. Using simulations for such tasks allows the use of targeted physical testing,

where the benign situations can be largely eliminated, leaving only the edge cases for physical testing.

EXAMPLE: Flight simulation software such as FLIGHTLAB or X-Plane can be used to develop detailed physical models of aircraft airframes, control systems, control surfaces, and environmental conditions. The outputs of these physical models can be used to drive detailed 3D graphical representations of the environment that allows engineers to intuitively generate and interpret interesting scenarios during the development of such systems. These physics-based simulations can then be coupled with autonomous software to test a wide range of autonomous operational scenarios, such as remote inspection. This then allows a relatively cheap and quick analysis that can be integrated into a development cycle (e.g., a system's engineering spiral model). The earlier stages can rely more on the simulations for development/testing and further design iterations and then for the later stages allow targeted physical testing.

V&V TECHNIQUES:

- V&V techniques such as simulation-based testing are non-exhaustive, meaning that it is not possible to examine every permutation of a scenario. Therefore a key issue is ensuring that the simulations that have been performed are likely to have captured erroneous behaviour. This can be done by examining cases in which the system is operated at the limits of its capabilities, for example, the ambient environmental conditions, e.g., wind speed, air density, temperature. These conditions can then be tested physically to validate the simulation results.
- No simulation, no matter how complex, is a perfect representation of reality. This means that all of the models used need to be validated using physical data to mitigate the differences. However, some degree of inaccuracy needs to be accepted.

ISSUES:

- Many of the issues relevant here also appear in the section on laboratory-based testing. In particular, assessing and understanding the gap between the simulation and the real world is again crucial, as is the development of experiments [83].
- However, the computational framework that these simulations sit in allows us to verify in a more extensive way. While replication of individual experiments in labs, or in the real-world, can be difficult this replication is straightforward in software based systems. Similarly, running many experiments can be time-consuming in labs and real test environments. But, with simulations, we can utilise high-performance computing, often large parallel compute servers, to run huge numbers of tests and experiments. And, while the coverage aspects and experimental design both remain important, the use of HPC allows us to employ brute-force techniques, for example assessing billions of randomly varied scenarios.

4.5. Heterogeneous Verification

Robotic systems combine heterogeneous hardware and software components, so each component may require different verification methods. Formal verification methods play a crucial role in providing robust evidence of a system obeying its safety requirements [57], but it is likely that some of a system's components will not be amenable to formal methods. Overcoming this challenge requires a heterogeneous approach to verification, where formal and non-formal verification techniques are integrated [72]. This approach also pairs well with Corroborative V&V [3], described in Section 4.6.

EXAMPLE: Consider the simplified planetary rover case study in [84], where the rover autonomously inspects points of interest in its map. This inspection is the rover's main task, but it also has to adapt its plan to accommodate adverse environmental conditions. The modules of this example system required different verification methods: the agent controlling the system is verified by program model checking, the system's interface with

its environmental sensors is verified using an automated theorem prover, and the communication between the system and its actuators is verified using standard model checking.

V&V TECHNIQUES: Heterogeneous verification can incorporate many distinct verification techniques, such as software testing, simulation, or formal methods. The main benefit of this approach is that it reduces the verification burden by decomposing the system into smaller modules that can each be tackled using an appropriate verification technique.

ISSUES: A key challenge of heterogeneous verification is ensuring that the module verification ‘fits’ together. For example, checking that the verification of one component does not make unverified assumptions about another component. A variety of techniques are useful here, such as Hoare logic [85] or, more broadly, Assume-Guarantee reasoning [86]. In an extension of the work described above, [87] presents a framework for linking heterogeneous verification approaches by first describing each module’s contract (its assumptions and guarantees) in First-Order Temporal Logic.

It is also difficult to ensure consistency between different models of a system that have been devised using disparate formalisms. In particular, it is not necessarily straightforward to argue that two models of a system are faithful representations of the same system specification. This is especially difficult if these models capture system functionality at different levels of abstraction.

Finally, heterogeneous verification is more difficult if the system is not sufficiently modular. Helpfully, robotic software is often implemented as a collection of communicating nodes (for example, ROS systems are popular in the literature) which improves modularisation. However, it is not clear that one node will always be a self-contained module that is easily tackled by a single verification technique. So the architectural issues discussed in Section 4.1 are important for the verifiability of the system.

4.6. Corroborative V&V

As well as breaking down the system into nodes or modules and applying different types of verification to different aspects as described in Section 4.5 we may want to apply different types of verification to the same module. There are a number of reasons why this may be beneficial. Firstly we might want to use a mix of formal verification and non-formal verification. The former is a mathematical analysis of systems using techniques such as model checkers and theorem provers. An advantage of such techniques is that they are exhaustive in that they can capture all runs through the system. However this comes at a cost and the computational complexity of the underlying problem is high. This means that we often run into the state explosion problem (that is the state space becomes very large) and the tool requires too much time or space to return an answer. To combat this, abstractions to the model may have to be made. Non-formal techniques such as simulation based testing, see for example [88], and end user experiments, see for example [89], can also be applied. However these are not exhaustive as they just examine a subset of runs through the system.

With simulation based testing, many test cases can be considered automatically but the simulator is necessarily an abstraction of the real world. For end user testing, it may be difficult or time consuming to do a large amount of tests. Whilst this is carried out in the real world so is more realistic than the other techniques, physical tests may be carried out in a laboratory setting or an environment as close as possible to where the system will be deployed. The outcome of one verification technique can be used to improve the models of another, or for example be used to identify a test to be carried out or be the focus of a specific end user experiment. In [3] three verification techniques are adopted, namely formal verification using probabilistic model checking, simulation based testing and end user experiments relating to a robot to human handover task. Simulation based testing revealed that the robot sometimes dropped the item due to gripper failure. End user experiments identified false negative results for some sensor readings. These were then used for further analysis in the other verification techniques, for example to refine the formal verification model and improve the simulator.

EXAMPLE: Consider further the simplified planetary rover case study described in Section 4.5 from [84] where a rover autonomously visits particular waypoints to make inspections. The decision making aspects could be verified using different verification techniques such as model checking the autonomous agent, simulation based testing and physical experiments. For the latter, tests would need to be carried out in an environment as similar to its intended use as possible.

V&V TECHNIQUES: Corroborative verification involves using several verification techniques, such as formal verification, simulation based testing and end user experiments on the same system component. Within these styles of verification different techniques may also be used. The advantage of this approach is to improve verification results by adopting a range of techniques which then can inform the others.

ISSUES: Whilst this approach may provide greater confidence in systems, it places a greater burden on the developer in terms of time and cost to apply the different techniques to key aspects. The approach is iterative and issues found during the application of one verification technique can be used to inform the others. Issues include when to stop the verification effort and to what extent do the verification results need to align, since they will never exactly match?

4.7. Scalable Verification

Recent years have seen an increased need for the verification of systems comprising large numbers of identical entities. Traditional verification techniques force us to consider each entity in such systems separately. As a result we face higher complexity, both at a modelling level (since we must manually keep track of all entities in a system and their interactions) and at a computational level (since by separately representing each entity the size of our representation grows considerably). In addition, we can only verify a given system for a fixed number of entities, whereas it would be ideal to consider arbitrary numbers of entities. To address such limitations, various approaches that scale better to systems with large numbers of identical entities have been proposed in the literature.

EXAMPLE: Consider a swarm comprising a large number of identical UAVs tasked to perform inspection of an oil rig. We may wish to verify various different types of protocols that are vital to the swarm's safe and effective operation, such as consensus protocols (for instance, verifying that when the UAVs receive a message to update an operational parameter, either they all update that parameter or none does) or broadcast protocols (for instance, verifying that if a UAV transmits a message to all of its neighbours, and the neighbours retransmit that message to all of their neighbours, and so on, so that the message will eventually reach the whole swarm).

V&V TECHNIQUES: Direct modelling of individual robots or sensors, for example using model checking, see for example [90,91] can provide useful results for small, fixed numbers of individuals. In [92,93] a bound is calculated such that if the emergent swarm property holds for that swarm size then it holds for all larger swarm sizes. Parameterised model checking with induction is also considered in [94]. Induction is used to show that any property from a class of properties that holds for the base topology will hold for a topology of any size and configuration.

Some proposed approaches providing better abstractions and more scalable techniques for the verification of systems comprising large numbers of identical entities are the use of symmetry reductions for example [95], model checking for parameterised and infinite state-systems [96], constraint-based verification using counting abstractions [97–99], parameterised population models via probabilistic model checking [100,101] and parameterised verification via monodic first-order temporal logic [102,103].

ISSUES: The direct modelling of individual sensors or robots suffers from the state explosion problem meaning that only a small number of individuals can be considered. Further, the verification results only hold for the cases checked and some other argument is needed

to show that the results hold in general. The latter is addressed in [92,93] however if the size of the bound on which to check the property is large the verification may not succeed again due to the state space explosion. In certain cases one may still face high computational complexity (even non-termination) when applying techniques for abstraction and scalability such as the ones mentioned above. Thus, those techniques are more effective when complementing traditional verification techniques. For example [101] models synchronisation protocols as both individual sensors and via a population model providing the formal link between the two approaches.

4.8. Runtime Verification

Runtime verification (RV) [104] is a dynamic verification method, that consumes events from a running system and checks if the events satisfy a formal property. When monitoring robotic systems, the events that a runtime monitor can intercept might be the robot's actions, sensor readings, or even internal messages. The property being verified could be the robot keeping to a safe speed or a set distance from an obstacle, for example.

It can be difficult to make these checks at design-time, because they depend on (unpredictable) environmental conditions. Abstracting away from the environment can reduce the complexity of the verification required, but causes a reality gap (the gap between design-time assumptions about a system's environment and the runtime reality). Whereas, making the formal model more faithful to the environment can lead to the model becoming intractable [105]. RV finds a midpoint by assessing the system's reactions to the real world against formal properties.

EXAMPLE: Consider a robot performing a remote inspection of a nuclear facility. If the robot is equipped with a radiation sensor, then it can patrol the area to check everything works properly, and there is no radiation leakage. The robot might have a set of safety properties, such as keeping a safe distance from high radiation and quickly moving out of sudden high radiation zones. RV can be used here to check that these safety properties are not violated at runtime.

V&V TECHNIQUES: In the literature, ROS is a popular middleware for robotic systems, where the system is composed of communicating nodes. ROSMonitoring (<https://github.com/autonomy-and-verification-uol/ROSMonitoring>, accessed on 28/04/21) [106] is an RV framework for ROS where the monitors intercept messages exchanged among the nodes. The messages are checked against a formal property and are only forwarded to their destination node if they satisfy the property. This shows how RV can also be used for runtime enforcement.

Providing traceability of safety requirements through to the final system is of key importance for systems that need regulatory approval before being deployed. The workflow described in [107] extracts the safety subsystem and requirements described in a system's safety document to use as an RV monitor. This helps to trace the requirements to an artefact of the final system, and has the side-benefit of checking the design of the safety subsystem.

ISSUES: By taking a more lightweight approach to verification, RV is not an exhaustive check of the system's state space. A monitor only checks the sequence of events that it observes, so there may be unchecked combinations of events that could cause failures.

Furthermore, RV can usually only report the violation of a property after it has occurred, because the system's events are being checked as they happen. However, predictive RV [108] uses a model of the system to try to foresee the satisfaction (or not) of a property, given the observed events. Approaches such as ROSMonitoring [106] also avoid this problem, by intercepting the system's internal communications and dropping messages that would violate a property.

4.9. Summary

System-level verification and validation can be performed using a wide array of techniques. Systems with a monolithic, hybrid, or pattern-based architecture are often

verified with traditional techniques such as testing, model checking, theorem proving, or runtime verification. Approaches that combine different verification and validation techniques such as heterogeneous verification and corroborative V&V are better suited to tackle systems with modular or hierarchical architectures, since different parts of the system may require a different technique (however, it is still important to make sure that these different techniques can be combined to provide assurances about the system as a whole).

5. Properties and Requirements

Given a robotic system (or design) to assess, and a particular target scenario for the system to be assessed in, what shall we actually verify? There are a wide range of issues that we can potentially use V&V for [109]; below we highlight a few common high-level properties for autonomous robotics.

5.1. Safety

There is a wide variety of safety standards and regulations aimed at safety-critical software-based systems and robotic systems, for example produced by BSI, ISO, IEEE, IEC, etc. While most of these remain relevant, adding autonomy can bring new challenges that are not yet tackled by these standards. The development of sector-specific standards (for aerospace, for manufacturing, for space, etc.) is likely to duplicate effort, so a general approach that focuses on safely developing autonomous systems would be preferable [110].

EXAMPLE: As previously mentioned, remote inspection robots can be usefully deployed in situations that are dangerous to humans (e.g., in nuclear decommissioning) or prohibitively distant (e.g., in outer space). It is crucial to demonstrate that new safety hazards that are caused by the introduction and use of an autonomous robotic system are sufficiently mitigated.

- In nuclear decommissioning, for example, a new hazard could involve the robot damaging vital infrastructure because of an incorrect decision or incorrect classification of sensor input, where a trained human is less likely to make these mistakes. Damaging this infrastructure could pose a hazard to other humans in the facility.
- Once we consider the example of a rover exploring a distant planet, then the risk is less to other humans but to the rover itself. Damage to the rover could mean mission failure, with damaged hardware remaining on the planet potentially making future missions more difficult. Proposals for the next steps in space exploration involve collaborations between astronauts and robots (e.g., ESA's 'moon village' vision (https://www.esa.int/About_Us/Ministerial_Council_2016/Moon_Village (accessed on 28 April 2021))) so an autonomous rover could conceivably be patrolling on another planet where humans are living. In this case, a collision between the rover and the habitat could have safety-critical implications.

In these examples, the simple safety property that the robot shall not collide with any obstacle is an obvious property to verify. While each deployment context is likely to bring specific safety properties, it is important to note that they are likely to be similar, and the necessity to verify these kinds of safety properties should be considered when designing and developing autonomous systems.

V&V TECHNIQUES: In the above examples a collision could potentially be caused by:

- sensor failure (e.g., environment not correctly detected),
- software failure (e.g., learning algorithm recognises environment incorrectly), or
- hardware failure (e.g., slippage due to tyres not being replaced).

Formal methods provide a useful tool in the verification toolbox, alongside high-fidelity simulation and careful software testing. For more complex missions, we will not be able to anticipate all potential hazards, so verification moves from assessing how well a system avoids or mitigates known hazards to recognising hazardous situations and dealing with them appropriately. In this case, verification concerns the hazard identification and

decision-making processes, and particularly making its decisions for the right (safety) reasons.

ISSUES: As previously mentioned, if the current patchwork of sector specific standards for safety-critical systems are each individually extended to include some attention to autonomous systems, this will not only duplicate effort but may also prevent portability of autonomous systems (and their verification) between sectors. Taking the opposite approach, and trying to apply one sector-specific standard (that has been updated to include autonomous systems) is likely to be unsatisfactory to regulators of other sectors. Both of these approaches are likely to miss issues that are specific to autonomous systems, such as ethical impacts (Section 5.3) and system trustworthiness (Section 5.4). Autonomous systems standards should be approached as a stand-alone issue, at a high-level, with consultation to then realise the requirements in each sector [110]. This is being attempted, to large extent, through the IEEE's P70xx suite of standards.

5.2. Security

The previous section considered safety; we now move onto security. Safety is about ensuring that the system does not affect the environment (no unacceptable harm is caused), while security is about keeping the environment from affecting the system (no deliberate harm is caused). In many cases, the security issues in inspection robots are similar to security issues relevant to any cyber-physical system, though some attacks (such as involving attacking machine-learning components) may be unusual.

Security issues are usually identified following a threat modelling phase where potential vulnerabilities and attackers are assessed. It is difficult to identify all possible attacks and attackers for a particular system but the usual approaches involve analysing potential aspects. For example, the STRIDE threat modelling paradigm involves analysing Spoofing (attacker pretends to be another system entity), Tampering (attacker manipulates data maliciously), Repudiation (attacker denies sending a message that it has sent), Information disclosure (attacker causes the system to reveal information to those it is not intended for), Denial of service (attacker can prevent the system from functioning correctly) and Escalation of privilege (attacker can perform more actions than allowed) [111]. Threat modelling not only considers potential attacks on systems but also their likelihood so the focus is on the most likely issues. The latter may change over time as new technologies are developed, or in relation to social and economic aspects for example the opening up of space to commercial organisations not just governments.

EXAMPLE: Consider a team of two autonomous rovers that are inspecting a large piece of infrastructure. The rovers follow each other around the area to be patrolled, each potentially is equipped with different sensors. For example, the first measures the radiation in the area and the second uses a camera to detect damage to the infrastructure. Since the second follows the first, it is important that the leader sends messages about its status to the second (e.g., location, speed, etc.). If an 'attacker' vehicle is present, it could spoof these messages and tell the second rover that the first has continued to move when in fact it has stopped. This could cause a collision and potentially result in mission-failure and/or significant damage to either vehicle or infrastructure.

V&V TECHNIQUES: Analysing and protecting a system against cyber threats is usually a process which is completely separate to V&V. For example, for the space domain, a series of white/green papers have been produced by CCSDS which provide some guidance about space mission security [112] while the FAIR-SPACE hub has produced a report which discusses security issues [113]. Similarly, across sectors, security guidelines have been produced.

While such threat analysis has often been distinct, recent work in this vein seeks to link threat analysis with formal verification by defining a methodology whereby security properties are formalised so that they can be adequately verified [111,114].

ISSUES:

- The need for fast message passing amongst autonomous robots in certain scenarios tends to result in a lack of message encryption which leaves these systems vulnerable to attackers.
- Machine learning components can be particularly vulnerable to attackers. For example, in a deep learning vision system, an attacker may flash it with images leading it to incorrectly identify an obstacle or another robot, potentially causing a collision. (See Section 3.2.)

5.3. Ethics and Transparency

Powerful autonomous systems now share in our physical and e-space. Consider for example, industrial robots that have been in operation at least since the 1980s [115], and automated subway systems, which have been in operation for the past forty years (<http://www.railjournal.com/index.php/metros/uitp-forecasts-2200km-of-automated-metros-by-2025.html>, accessed on 28/04/21). Both of these types of machines have the capacity to seriously harm people and property, however they operate in a work envelope, a segregated space which only trained personnel are allowed to enter.

In contrast, machines such as “driverless” cars and assisted living devices have the ability to do significant harm since they are not operating in a segregated environment. When we add in the complexity of reasoning in under-specified highly dynamic environments then it can become important that a system be able to (truthfully, verifiably) explain its actions. Further, since all possibilities cannot be pre-coded into the system, the system is increasingly likely to have to reason ethically, i.e., with notions of ‘right’ and ‘wrong’, and potentially even be forced to resolve an ethical dilemma. (Note, however, that systems that make such decisions are nowhere near being approved for deployment.)

Standards are beginning to emerge that define the requirements for ethical behaviour for autonomous systems, such as BSI 8611 [116]. Many of these consider not only the ethics of the system’s own reasoning, but also ethical issues around its design, development and deployment. Crucial not only to ethics, but to strong notions of verification considered earlier, is the requirement for system transparency, in particular transparency concerning decisions made and actions taken. Correspondingly, key standards such as IEEE P7001, on Transparency in Autonomous Systems [117], will be a cornerstone of future autonomous systems development.

EXAMPLE: Consider an assistive robot deployed to aid a human in the inspection of a nuclear facility. This robot may be responsible for warning the human of any potential radiation exposure based on their proximity to, and strength of, the radiation. However, to perform this task the robot has to constantly follow the human around, which may eventually annoy or irritate them. If the human asks the robot to stop following it, should the robot comply with the request? In making such contentious and critical decision, how should the robot explain itself if requested? And, in all these cases, what verification will be required?

V&V TECHNIQUES: In general ethical reasoning is implemented by adopting an ethical theory from philosophy and encoding that together with relevant information about actions and outcomes. Transparency is delivered in a variety of ways such as log files, ethical black boxes [118], omniscient debugging techniques [119], and dialogue based explanation mechanisms [120].

- We can use formal verification to establish that an ethical theory has been implemented correctly (for instance to establish that the system will always choose the correct action according to that theory) [9,121].
- We can also use either formal verification or testing techniques to examine specific scenarios to check that the system makes the correct decisions in those specific scenarios [121].
- We can potentially use formal techniques to ensure that transparency mechanisms store and report accurate information. The appropriate use of architectures (Section 4),

ensuring that key decisions are made by symbolic AI components and so are much more amenable to explanations, are crucial.

ISSUES:

- All ethical theories have ‘pathological’ edge cases where the decisions are considered unethical by most people. Furthermore, when encoding the information to be used in ethical reasoning it is important to consider carefully whose opinion about this information is sought and avoid bias in the implemented ethics.
- Transparency mechanisms can prove opaque to users and, conversely, can also instill a false sense of confidence in the system [122].

5.4. Responsibility and Trust

Responsibility and trust are complex issues that come to the fore particularly in semi-autonomous or autonomous systems. At several points in this article we have mentioned the idea that a human operator might, at some point, delegate some responsibility to an autonomous device. By this we mean that some aspect the human was originally in-charge of is now left to the robot or vehicle to handle. These aspects range from simple functions, such as monitoring a location, to multi-faceted responsibilities such as driving a vehicle safely and quickly through traffic. This leads to many questions about how to represent these responsibilities and how they are actually transferred. However, a particular focus in inspection robots appears to be whether the human operator trusts the robotic device enough to even delegate any responsibilities to it. There are a number of issues to consider here. First, there is the overarching issue of what trustworthiness involves, especially in complex systems.

Trust is, in practice, engendered by a robotic system in its human operator and is essential to deployment. No matter how reliable or effective the robotic system is demonstrated to be via, for example, verification techniques, the control of the system is fundamentally dependent on how human operators use them; how much confidence and trust operators have in the systems. However, trust is itself a complex and multi-faceted concept [123,124], which depends of a multitude of factors including whether the operator and robot share mutual beliefs, intentions and goals [125], and how those mutual ideas are restored if they fail to align. A clear understanding of where responsibility lies upon the failure of outcomes, is one approach to maintaining trust in Human-Robot Interaction (HRI).

EXAMPLE: Consider a remote inspection robot designed to enter a hazardous area to make survey of assets. The robot is receiving high-level commands from its human operator who is working remotely and is unable to intervene if the robot fails. The robot is about to enter an area it will not be able to be extracted from due to floor debris. However, the operator recalls that on several previous occasions the robot has either misinterpreted instructions or failed to carry out commands. Does the operator now trust the robot enough to send it into a dangerous environment? Especially if the operator will be held responsible for the lost equipment? Clearly, trust here involves reliability. However, it also involves the confidence the operator has that the robot will be able to cope with unexpected or unanticipated issues [126]. Will it do the “right thing” when it encounters a problem?

V&V TECHNIQUES:

- As one aspect of trustworthiness concerns reliability, then the range of verification techniques described earlier are important in enhancing this. However, so are techniques for measuring trust [127] and for recognising trust failures and recovering this [128].
- Socio-cognitive research lends validation techniques to exploring trust in HRI by considering how humans will react in given scenarios. For example, humans utilise a Theory of Mind (ToM) to understand the intentions and goals of others, with trust associated with a person’s ability to make mental state inferences [129]. By extraction, an exploration of how humans develop a suitable mental model of robot capability

can be used to approximate how much information a robot should be relaying to an operator to maintain their trust [130].

ISSUES:

- While we know that a basis of trust is essential to the adoption of robotic systems, we do not yet know the answers to fundamental questions which would ensure this, including what is the best way to interact with robotics in specific situations?
- Trust research requires interdisciplinary effort, to take ideas from fundamental human science over to engineering design to produce optimal HRI systems. Further, the different dimensions of trust (e.g. dispositional and situational, etc) are currently under-researched within human-non-human agent interactions. Combined, this makes the property and requirement of a trustworthy system a huge endeavour that will not be resolved by a single team.

5.5. Energy and Sustainability

Two final areas from which we might generate properties are: modelling and predicting energy usage and sustainability. A large proportion of robotic systems are dependent on batteries for their energy, and so the careful prediction and modelling of energy usage is crucial. This aspect also links to sustainability, since over-use of energy or communication can have considerable environmental problems [131]. (Note that there are clearly many sustainability issues beyond energy and communication, but we will not consider these within this article.)

EXAMPLE: We send our robot to carry out its schedule of inspections, for example examining offshore wind turbines. The robot will likely have power through batteries and so the careful mission planning and energy assessment is important. However, if we are to deploy our robot in poor weather then wind shifts and increases in rain can seriously accelerate battery usage. In many cases, losing power is a severe and dangerous outcome.

V&V TECHNIQUES: The analysis of energy storage systems has been studied for a long time in practical robotic systems [132]. It is also well understood how to carry out various forms of verification, even formal verification, taking into account battery levels [133]. Any technique able to represent numerical values might be used to represent energy levels.

ISSUES: The particular issues concern mixing battery-awareness and (formal) verification with other activities, such as planning, scheduling [134], prognostics and health management [135]. In terms of verification, the representation of energy use in combination with a wide variety of other dimensions, still remains problematic.

5.6. Summary

In contrast to Sections 2–4, a system's high-level properties and requirements are usually not dependent on specific V&V techniques. Some approaches or tools can be seen as more appropriate to a particular type of property, because it enables the user to capture temporal or probabilistic properties but, for the most part, the decision of what kind of property to verify does not restrict the options available for V&V, whereas the level of autonomy, type of component, or system architecture (as discussed in previous sections) can have a significant impact on the viable V&V techniques.

6. Conclusions

It is clear that the use of robots in the inspection of assets and facilities can have significant benefits, particularly in hazardous or distant environments. Remote inspection robots are being used in increasingly diverse environments: from UAV-based inspection of offshore structures and underground inspection [136], through to remote inspection of a fusion reactor or the autonomous exploration of the surface of Mars. In all of these environments, using remote inspection robots can bring clear safety benefits, since humans are removed from direct danger. However, we must ensure that the robotic systems are

effective, reliable, and safe, particularly in the environment in which they are deployed. It is here that a range of V&V techniques are needed in order to assess these robotic systems, not only at design time but during and after inspection activities.

In this article we have highlighted a range of topics and developments related to V&V (RQ1) that we have utilised (RQ2) in our work on autonomous and semi-autonomous robotics in offshore, space, and nuclear environments. Many techniques developed for traditional cyber-physical systems remain important, while the step towards these new systems brings additional issues that we have described here (RQ3).

It is clear that standard software engineering approaches around modular and compositional systems are particularly important; the fact that many robotic systems are modular assists this. These architectural issues appear particularly important for supporting effective V&V in robotic systems and so we suggest that verification efforts should follow corroborative (Section 4.6) and heterogeneous (Section 4.5) approaches. Corroborative verification involves using several verification techniques, such as formal verification, simulation based testing and end user experiments on the same system component [3] while heterogeneous verification involves potentially using a different technique on each component and integrating these together [72] to provide a holistic view.

Finally, a new aspect that is often ignored in the V&V of robotic systems is the issue of ‘autonomy’. In the near future most of these robotic inspection devices will involve user interaction, usually at a distance. The effectiveness of these types of semi-autonomous systems is heavily dependent on the robot-operator link and issues across mental, physical, environmental, and emotional will all come into play. Concepts such as trust, responsibility, and explainability, as well as a range of HRI studies, will all be of importance here. Then, as we move towards fully autonomous systems, the need to strongly verify the autonomous decision-making within a robotic system becomes acute. In such a system we are potentially delegating responsibility for safety, reliability, and even security, to the software that will now make autonomous decisions. If we are to deploy these systems in non-trivial environments then the mere cataloguing and mitigation of all predictable hazards will surely be insufficient. While the deployment of these types of systems is quite distant, the V&V techniques needed for assessing autonomous decision-making, safety decisions, and even ethical decisions, are beginning to become available.

Author Contributions: Conceptualization, M.F. (Michael Fisher), R.C.C. E.C.C., L.A.D., C.D. (Christopher Dadswell), A.F., G.K., C.D. (Clare Dixon), M.J., V.P., M.F. (Marie Farrell), M.L., X.H., A.L., S.L., F.P. and M.W.; Writing—original draft preparation, M.F. (Michael Fisher); writing—review and editing, M.F. (Michael Fisher); All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the UK Industrial Strategy Challenge Fund (ISCF) delivered by UKRI and managed by EPSRC under the Robotics and AI for Extreme Environments programme with grants Robotics and AI in Nuclear (RAIN) Hub (EP/R026084/1), Future AI and Robotics for Space (FAIR-SPACE) Hub (EP/R026092/1), and Offshore Robotics for Certification of Assets (ORCA) Hub (EP/R026173/1) and the EPSRC under the Science of Sensor System Software programme grant (EP/N007565/1). Fisher was supported by the Royal Academy of Engineering via a Chair in Emerging Technologies.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Bonner, M.C.; Taylor, R.M.; Miller, C.A. Tasking Interface Manager: Affording Pilot Control of Adaptive Automation and Aiding. In *Contemporary Ergonomics 2000*; CRC Press, Boca Raton, FL, USA, 2004; pp. 70–74.
2. Hobbs, A.; Lyall, B. *Human Factors Guidelines for Remotely Piloted Aircraft System (RPAS) Remote Pilot Stations (RPS)*; Technical report; NASA, Washington, DC, USA, 2016. Available online: <https://ntrs.nasa.gov/citations/20190000211> (accessed on 28 April 2021).
3. Webster, M.; Western, D.; Araiza-Illan, D.; Dixon, C.; Eder, K.; Fisher, M.; Pipe, A.G. A Corroborative Approach to Verification and Validation of Human–Robot Teams. *Int. J. Robot. Res.* **2020**, *39*, 73–79.
4. Simrock, S. *Control Theory*; CAS—CERN Accelerator School: Digital Signal Processing; Geneva, Switzerland, 2008; pp. 73–130. doi:10.5170/CERN-2008-003.73.
5. Page, V.; Webster, M.; Fisher, M.; Jump, M. Towards a Methodology to Test UAVs in Hazardous Environments. In Proceedings of the 15th International Conference on Autonomic and Autonomous Systems (ICAS), Athens, Greece, 2019.
6. Fisher, M. *An Introduction to Practical Formal Methods Using Temporal Logic*; John Wiley & Sons: Hoboken, NJ, USA, 2011.
7. Garoche, P.L. *Formal Verification of Control System Software*; Princeton University Press: Princeton, NJ, USA, 2019.
8. Institute of Electrical and Electronics Engineers. P7009—Standard for Fail-Safe Design of Autonomous and Semi-Autonomous Systems; 2017. Available online: <https://standards.ieee.org/project/7009.html> (accessed on 28 April 2021).
9. Bremner, P.; Dennis, L.A.; Fisher, M.; Winfield, A.F.T. On Proactive, Transparent, and Verifiable Ethical Reasoning for Robots. *Proc. IEEE* **2019**, *107*, 541–561.
10. Arapinis, M.; Cheval, V.; Delaune, S. Verifying Privacy-Type Properties in a Modular Way. In Proceedings of the 25th IEEE Computer Security Foundations Symposium (CSF). IEEE Computer Society; Cambridge, USA, 2012; pp. 95–109.
11. Asadollah, S.A.; Inam, R.; Hansson, H. A Survey on Testing for Cyber Physical System. In Proceedings of the International Conference on Testing Software and Systems (ICTSS); Springer, Germany; 2015; Volume 9447, pp. 194–207.
12. Zabczyk, J. *Mathematical Control Theory: An Introduction*; Birkhauser, Switzerland; 2008.
13. Tabuada, P. *Verification and Control of Hybrid Systems—A Symbolic Approach*; Springer: New York City, NY, USA; 2009; doi: 10.1007/978-1-4419-0224-5.
14. Gerber, C.; Preuß, S.; Hanisch, H. A Complete Framework for Controller Verification in Manufacturing. In Proceedings of the 15th IEEE Conference on Emerging Technologies Factory Automation (ETFA); Bilbao, Spain, 2010; pp. 1–9.
15. Park, J.; Pajic, M.; Sokolsky, O.; Lee, I. LCV: A Verification Tool for Linear Controller Software. In *Proceedings of the Tools and Algorithms for the Construction and Analysis of Systems*; Springer: New York City, NY, USA; 2019; pp. 213–225; doi:10.1007/978-3-030-17462-0_12.
16. Sun, Y.; Zhou, Y.; Maskell, S.; Sharp, J.; Huang, X. Reliability Validation of Learning Enabled Vehicle Tracking. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA); Paris, France, 2020; pp. 9390–9396.
17. Huang, W.; Zhou, Y.; Sun, Y.; Sharp, J.; Maskell, S.; Huang, X. Practical Verification of Neural Network Enabled State Estimation System for Robotics. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, (IROS), Las Vegas, NV, USA, 25–29 October 2020.
18. Zhang, F.; Leitner, J.; Milford, M.; Upcroft, B.; Corke, P.I. Towards Vision-Based Deep Reinforcement Learning for Robotic Motion Control. *arXiv* **2015**, arXiv:1511.03791.
19. Jiang, Z.; Luo, S. Neural Logic Reinforcement Learning. In Proceedings of the International Conference on Machine Learning (ICML); California, USA; 2019; pp. 3110–3119.
20. Huang, W.; Zhao, X.; Huang, X. Embedding and Synthesis of Knowledge in Tree Ensemble Classifiers. *arXiv* **2020**, arXiv:2010.08281.
21. Huang, X.; Kroening, D.; Ruan, W.; Sharp, J.; Sun, Y.; Thamo, E.; Wu, M.; Yi, X. A Survey of Safety and Trustworthiness of Deep Neural Networks: Verification, Testing, Adversarial Attack and Defence, and Interpretability. *Comput. Sci. Rev.* **2020**, *37*, 100270.
22. Huang, X.; Kwiatkowska, M.; Wang, S.; Wu, M. Safety Verification of Deep Neural Networks. In Proceedings of the Computer Aided Verification (CAV); Springer, Germany; 2017; pp. 3–29.
23. Wicker, M.; Huang, X.; Kwiatkowska, M. Feature-guided Black-box Safety Testing of Deep Neural Networks. In Proceedings of TACAS; Thessaloniki, Greece; 2018; pp. 408–426.
24. Ruan, W.; Huang, X.; Kwiatkowska, M. Reachability Analysis of Deep Neural Networks with Provable Guarantees. In Proceedings of IJCAI; Stockholm, Sweden; 2018; pp. 2651–2659.
25. Wu, M.; Wicker, M.; Ruan, W.; Huang, X.; Kwiatkowska, M. A Game-based Approximate Verification of Deep Neural Networks with Provable Guarantees. *Theor. Comput. Sci.* **2020**, *807*, 298–329.
26. Ruan, W.; Wu, M.; Sun, Y.; Huang, X.; Kroening, D.; Kwiatkowska, M. Global Robustness Evaluation of Deep Neural Networks with Provable Guarantees for the Hamming Distance. In Proceedings of IJCAI; Macao, China; 2019; pp. 5944–5952.
27. Sun, Y.; Huang, X.; Kroening, D.; Sharp, J.; Hill, M.; Ashmore, R. Structural Test Coverage Criteria for Deep Neural Networks. *ACM Trans. Embed. Comput. Syst.* **2019**, *18*. doi:10.1145/3358233.
28. Sun, Y.; Wu, M.; Ruan, W.; Huang, X.; Kwiatkowska, M.; Kroening, D. Concolic Testing for Deep Neural Networks. In Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering (ASE); Montpellier, France; 2018; pp. 109–119.

29. Huang, W.; Sun, Y.; Sharp, J.; Ruan, W.; Meng, J.; Huang, X. Coverage Guided Testing for Recurrent Neural Networks. *arXiv* **2019**, arXiv:1911.01952.
30. Webb, S.; Rainforth, T.; Teh, Y.W.; Kumar, M.P. Statistical Verification of Neural Networks. In Proceedings of the International Conference on Learning Representations; New Orleans, LA, USA, 6–9 May 2019.
31. Jin, G.; Yi, X.; Zhang, L.; Zhang, L.; Schewe, S.; Huang, X. How does Weight Correlation Affect the Generalisation Ability of Deep Neural Networks. In Proceedings of NeurIPS; Vancouver, Canada, 6–12 December 2020.
32. Zhao, X.; Banks, A.; Sharp, J.; Robu, V.; Flynn, D.; Fisher, M.; Huang, X. A Safety Framework for Critical Systems Utilising Deep Neural Networks. In Proceedings of the 39th International Conf. Computer Safety, Reliability, and Security (SAFECOMP); Springer, Germany; 2020; Volume 12234, pp. 244–259.
33. Zhao, X.; Huang, W.; Banks, A.; Cox, V.; Flynn, D.; Schewe, S.; Huang, X. Assessing Reliability of Deep Learning Through Robustness Evaluation and Operational Testing. In Proceedings of SafeComp; York, UK; 2020.
34. Smolensky, P. Connectionist AI, Symbolic AI, and the brain. *Artificial Intelligence Review* **1987**, *1*, 95–109.
35. Wooldridge, M.; Rao, A., Eds. *Foundations of Rational Agency*; Kluwer Academic Publishers, Netherlands; 1999.
36. Wooldridge, M. *An Introduction to Multiagent Systems*; John Wiley & Sons, UK; 2002.
37. Bratman, M.E. *Intentions, Plans, and Practical Reason*; Harvard University Press, USA; 1987.
38. Rao, A.S.; Georgeff, M. BDI Agents: From Theory to Practice. In Proceedings of the First International Conference on Multi-Agent Systems (ICMAS), San Francisco, CA, USA, 12–14 June 1995; pp. 312–319.
39. Cardoso, R.C.; Dennis, L.A.; Fisher, M. Plan Library Reconfigurability in BDI Agents. *Engineering Multi-Agent Systems*. Springer, 2020, pp. 195–212.
40. Cardoso, R.C.; Ferrando, A.; Dennis, L.A.; Fisher, M. An Interface for Programming Verifiable Autonomous Agents in ROS. In *Multi-Agent Systems and Agreement Technologies*; Springer, Germany; 2020; pp. 191–205.
41. Stringer, P.; Cardoso, R.C.; Huang, X.; Dennis, L.A. Adaptable and Verifiable BDI Reasoning. *arXiv* **2020**, arXiv:2007.11743. doi:10.4204/EPTCS.319.9.
42. Onyedima, C.; Gavigan, P.; Esfandiari, B. Toward Campus Mail Delivery Using BDI. *J. Sens. Actuator Netw.* **2020**, *9*, 56.
43. Huang, X.; Kwiatkowska, M.; Olejnik, M. Reasoning about Cognitive Trust in Stochastic Multiagent Systems. *ACM Trans. Comput. Log.* **2019**, *20*, 21:1–21:64. doi:10.1145/3329123.
44. Manna, Z.; Pnueli, A. *The Temporal Logic of Reactive and Concurrent Systems*; Springer: Berlin, Germany, 1992.
45. Parikh, R. Propositional Game Logic. In Proceedings of the 24th Annual Symposium on Foundations of Computer Science (FOCS); Arizona USA; 1983; pp. 195–200.
46. Huang, X.; van der Meyden, R. Symbolic Model Checking Epistemic Strategy Logic. In Proceedings of the 28th AAAI Conference on Artificial Intelligence; AAAI Press, USA; 2014; pp. 1426–1432.
47. Huang, X. Bounded Model Checking of Strategy Ability with Perfect Recall. *Artif. Intell.* **2015**, *222*, 182–200.
48. Huang, X.; van der Meyden, R. An Epistemic Strategy Logic. *ACM Trans. Comput. Log.* **2018**, *19*, 26:1–26:45.
49. Huang, X.; Luo, C. A Logic of Probabilistic Knowledge and Strategy. In Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS); St. Paul, MN, USA, 6–10 May 2013; pp. 845–852.
50. Fagin, R.; Halpern, J.; Moses, Y.; Vardi, M. *Reasoning About Knowledge*; MIT Press, Cambridge, MA, USA; 1996.
51. Huang, X.; Ruan, J. ATL Strategic Reasoning Meets Correlated Equilibrium. In Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI); Melbourne, Australia; 19–25 August 2017; pp. 1102–1108.
52. Huang, X.; Kwiatkowska, M. Model Checking Probabilistic Knowledge: A PSPACE Case. In Proceedings of the 13th AAAI Conference on Artificial Intelligence; AAAI Press, USA; 2016; pp. 2516–2522.
53. Fu, C.; Turrini, A.; Huang, X.; Song, L.; Feng, Y.; Zhang, L. Model Checking Probabilistic Epistemic Logic for Probabilistic Multiagent Systems. In Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI); Stockholm, Sweden; 2018; pp. 4757–4763.
54. Huang, X.; van der Meyden, R. Synthesizing Strategies for Epistemic Goals by Epistemic Model Checking: An Application to Pursuit Evasion Games. In Proceedings of the 26th AAAI Conference on Artificial Intelligence; AAAI Press, USA; 2012.
55. Huang, X.; Maupin, P.; van der Meyden, R. Model Checking Knowledge in Pursuit Evasion Games. In Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI); Catalonia, Spain; 2011; pp. 240–245.
56. Bakar, N.A.; Selamat, A. Agent Systems Verification: Systematic Literature Review and Mapping. *Appl. Intell.* **2018**, *48*, 1251–1274.
57. Luckcuck, M.; Farrell, M.; Dennis, L.A.; Dixon, C.; Fisher, M. Formal Specification and Verification of Autonomous Robotic Systems: A Survey. *ACM Comput. Surv.* **2019**, *52*, 1–41. doi:10.1145/3342355.
58. Dennis, L.A. The MCAPL Framework including the Agent Infrastructure Layer and Agent Java Pathfinder. *J. Open Source Softw.* **2018**, *3*, doi:10.21105/joss.00617.
59. Dennis, L.A.; Farwer, B. Gwendolen: A BDI Language for Verifiable Agents. In Proceedings of the Workshop on Logic and the Simulation of Interaction and Reasoning, AISB, Aberdeen, UK, 3–4 April 2008.
60. Dennis, L.A.; Fisher, M.; Webster, M.; Bordini, R.H. Model Checking Agent Programming Languages. *Autom. Softw. Eng.* **2012**, *19*, 5–63.
61. Lomuscio, A.; Raimondi, F. MCMAS: A Model Checker for Multi-agent Systems. In Proceedings of the 12th Int. Conf. Tools and Algorithms for the Construction and Analysis of Systems (TACAS); Springer, Germany; 2006; Volume 3920, pp. 450–454.
62. Dixon, C.; Fisher, M.; Bolotov, A. Resolution in a Logic of Rational Agency. *Artif. Intell.* **2002**, *139*, 47–89.

63. Wooldridge, M.; Dixon, C.; Fisher, M. A Tableau-Based Proof Method for Temporal Logics of Knowledge and Belief. *J. Appl. Non Class. Logics* **1998**, *8*, 225–258.
64. Alechina, N.; Dastani, M.; Khan, F.; Logan, B.; Meyer, J.J., Using Theorem Proving to Verify Properties of Agent Programs. In *Specification and Verification of Multi-agent Systems*; Springer, Germany; 2010; pp. 1–33.
65. Ferrando, A.; Dennis, L.A.; Ancona, D.; Fisher, M.; Mascardi, V. Verifying and Validating Autonomous Systems: Towards an Integrated Approach. In *Proceedings of the 18th International Conference on Runtime Verification (RV)*; Springer, Germany; 2018; Volume 11237, pp. 263–281.
66. Ferrando, A.; Ancona, D.; Mascardi, V. Decentralizing MAS Monitoring with DecAMon. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*; São Paulo, Brazil; 8–12 May 2017; pp. 239–248.
67. Winikoff, M. BDI agent testability revisited. *Auton. Agents Multiagent Syst.* **2017**, *31*, 1094–1132.
68. Alami, R.; Chatila, R.; Fleury, S.; Ghallab, M.; Ingrand, F. An Architecture for Autonomy. *Int. J. Robot. Res.* **1998**, *17*, 315–337.
69. Quigley, M.; Conley, K.; Gerkey, B.P.; Faust, J.; Foote, T.; Leibs, J.; Wheeler, R.; Ng, A.Y. ROS: An Open-source Robot Operating System. In *Proceedings of the ICRA Workshop on Open Source Software*, Kobe, Japan, 12–17 May 2009.
70. Elkady, A.; Sobh, T. Robotics Middleware: A Comprehensive Literature Survey and Attribute-Based Bibliography. *J. Robot.* **2012**, *2012*.
71. Garavel, H.; Lang, F.; Mounier, L. Compositional Verification in Action. In *Formal Methods for Industrial Critical Systems*; Springer, Germany; 2018; pp. 189–210.
72. Farrell, M.; Luckcuck, M.; Fisher, M. Robotics and Integrated Formal Methods: Necessity Meets Opportunity. In *Proceedings of the 14th Int. Conf. Integrated Formal Methods (iFM)* Maynooth, Ireland, 5–7 September 2018, pp. 161–171.
73. Albus, J.S.; Barbera, A.J.; Nagel, R. Theory and Practice of Hierarchical Control. In *Proceedings of the 23rd IEEE Computer Society International Conference*. Washington, DC, USA, 15–17 September 1981; 1981.
74. Buxbaum, H.J.; Maiwald, W. Utilization of a Hierarchical Control System in a Robot-Based Flexible Assembly Cell. *IFAC Proc. Vol.* **1995**, *28*, 399–404.
75. Brooks, R. A Robust Layered Control System for a Mobile Robot. *J. Robot. Autom.* **1986**, *2*.
76. Cimatti, A.; Pistore, M.; Traverso, P. Automated Planning. In *Handbook of Knowledge Representation*; Elsevier, Netherlands; 2008; pp. 841–867.
77. Ghallab, M.; Nau, D.S.; Traverso, P. *Automated Planning and Acting*; Cambridge Univ. Press, UK; 2016.
78. Dennis, L.A.; Fisher, M.; Lincoln, N.K.; Lisitsa, A.; Veres, S.M. Practical Verification of Decision-Making in Agent-Based Autonomous Systems. *Autom. Softw. Eng.* **2016**, *23*, 305–359.
79. Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J., Eds. *Design Patterns*; Addison-Wesley Publishing Co., USA; 1995.
80. Menghi, C.; Tsigkanos, C.; Pelliccione, P.; Ghezzi, C.; Berger, T. Specification Patterns for Robotic Missions. *IEEE Trans. Softw. Eng.* **2019**, doi:10.1109/TSE.2019.2945329.
81. Xiao, A.; Bryden, K.M. Virtual Engineering: A Vision of the Next-Generation Product Realization Using Virtual Reality Technologies. In *Proceedings of the ASME 2004 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 2004.
82. Robert, C.; Guiochet, J.; Waeselynck, H. Testing a non-deterministic robot in simulation - How many repeated runs? In *Proceedings of the Fourth IEEE International Conference on Robotic Computing (IRC)*; Taiwan; 2020; p. 8p.
83. Cavalcanti, A.; Sampaio, A.; Miyazawa, A.; Ribeiro, P.; Filho, M.C.; Didier, A.; Li, W.; Timmis, J. Verified Simulation for Robotics. *Sci. Comput. Program.* **2019**, *174*, 1–37.
84. Cardoso, R.C.; Farrell, M.; Luckcuck, M.; Ferrando, A.; Fisher, M. Heterogeneous Verification of an Autonomous Curiosity Rover. In *Proceedings of the 12th International NASA Formal Methods Symposium (NFM)*; Springer, Germany; 2020; Volume 12229, pp. 353–360.
85. Hoare, C.A.R. An Axiomatic Basis for Computer Programming. *Comm. ACM* **1969**, *12*, 576–580.
86. Jones, C.B. Tentative Steps Toward a Development Method for Interfering Programs. *ACM Trans. Program. Lang. Syst.* **1983**, *5*, 596–619.
87. Cardoso, R.C.; Dennis, L.A.; Farrell, M.; Fisher, M.; Luckcuck, M. Towards Compositional Verification for Modular Robotic Systems. In *Proceedings of the 2nd Workshop on Formal Methods for Autonomous Systems (FMAS) [Virtual]*, 7 December 2020; pp. 15–22. doi:10.4204/EPTCS.329.2.
88. Araiza-Illan, D.; Western, D.; Pipe, A.G.; Eder, K. Coverage-Driven Verification—An Approach to Verify Code for Robots that Directly Interact with Humans. In *Proceedings of the 11th International Haifa Verification Conference (HVC)*, Haifa, Israel, 17–19 November 2015; pp. 69–84.
89. Salem, M.; Lakatos, G.; Amirabdollahian, F.; Dautenhahn, K. Would You Trust a (Faulty) Robot?: Effects of Error, Task Type and Personality on Human-Robot Cooperation and Trust. In *Proceedings of the 10th ACM/IEEE Int. Conf. Human-Robot Interaction (HRI)*. ACM, 2015; pp. 141–148.
90. Dixon, C.; Winfield, A.; Fisher, M.; Zeng, C. Towards Temporal Verification of Swarm Robotic Systems. *Robot. Auton. Syst.* **2012**, *60*, 1429–1441.
91. Webster, M.; Breza, M.; Dixon, C.; Fisher, M.; McCann, J. Exploring the Effects of Environmental Conditions and Design Choices on IoT Systems Using Formal Methods. *J. Comput. Sci.* **2020**, *45*.

92. Kouvaros, P.; Lomuscio, A. Verifying Emergent Properties of Swarms. In Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI); Buenos Aires, Argentina, 25–31 July 2015, pp. 1083–1089.
93. Lomuscio, A.; Pirovano, E. Verifying Emergence of Bounded Time Properties in Probabilistic Swarm Systems. In Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI); Stockholm, Sweden, 13–19 July 2018, pp. 403–409.
94. Graham, D.; Calder, M.; Miller, A. An Inductive Technique for Parameterised Model Checking of Degenerative Distributed Randomised Protocols. *Electron. Notes Theor. Comput. Sci.* **2009**, *250*, 87–103.
95. Donaldson, A.F.; Miller, A.; Parker, D. Language-Level Symmetry Reduction for Probabilistic Model Checking. In Proceedings of the 6th International Conference on the Quantitative Evaluation of Systems (QEST); Budapest, Hungary, 13–16 September 2009, pp. 289–298.
96. Abdulla, P.A.; Jonsson, B.; Nilsson, M.; Saksena, M. A Survey of Regular Model Checking. In Proceedings of the International Conference on Concurrency Theory, London, UK, 31 August - 3 September, 2004; pp. 35–48.
97. Delzanno, G. Automatic Verification of Parameterized Cache Coherence Protocols. In Proceedings of the International Conference on Computer Aided Verification, Chicago, IL, USA, 15–19 July 2000; pp. 53–68.
98. Delzanno, G. Constraint-based verification of parameterized cache coherence protocols. *Form. Methods Syst. Des.* **2003**, *23*, 257–301.
99. Esparza, J.; Finkel, A.; Mayr, R. On the Verification of Broadcast Protocols. In Proceedings of the 14th Symposium on Logic in Computer Science (LICS), Trento, Italy, 2–5 July 1999; pp. 352–359.
100. Konur, S.; Dixon, C.; Fisher, M. Analysing Robot Swarm Behaviour via Probabilistic Model Checking. *Robot. Auton. Syst.* **2012**, *60*, 199–213.
101. Gainer, P.; Linker, S.; Dixon, C.; Hustadt, U.; Fisher, M. Multi-Scale Verification of Distributed Synchronisation. *Form. Methods Syst. Des.* **2020**. doi:10.1007/s10703-020-00347-z.
102. Fisher, M.; Konev, B.; Lisitsa, A. Practical infinite-state verification with temporal reasoning. *NATO Secur. Through Sci. Ser. D Inf. Commun. Secur.* **2006**, *1*, 91.
103. Dixon, C.; Fisher, M.; Konev, B.; Lisitsa, A. Practical First-order Temporal Reasoning. In Proceedings of the 15th International Symposium on Temporal Representation and Reasoning (TIME), 2008; pp. 156–163.
104. Leucker, M.; Schallhart, C. A Brief Account of Runtime Verification. *J. Log. Algebr. Methods Program.* **2009**, *78*, 293–303. doi:10.1016/j.jlap.2008.08.004.
105. Desai, A.; Dreossi, T.; Seshia, S.A. Combining Model Checking and Runtime Verification for Safe Robotics. In Proceedings of the International Conference on Runtime Verification, Seattle, WA, US, 13–16 September 2017; Volume 10548, pp. 172–189.
106. Ferrando, A.; Cardoso, R.C.; Fisher, M.; Ancona, D.; Franceschini, L.; Mascardi, V. ROSMonitoring: A Runtime Verification Framework for ROS. In Proceedings of the towards Autonomous Robotic Systems (TAROS). Nottingham, UK, 16 September 2020; pp. 387–399. doi:10.1007/978-3-030-63486-5_40.
107. Luckcuck, M. Monitoring Robotic Systems using CSP: From Safety Designs to Safety Monitors. *arXiv* **2020**, arXiv:2007.03522.
108. Zhang, X.; Leucker, M.; Dong, W. Runtime Verification with Predictive Semantics. In *Proceedings of the 4th International Symposium NASA Formal Methods (NFM)*; Springer, Germany; 2012; Volume 7226, pp. 418–432.
109. Miyazawa, A.; Ribeiro, P.; Li, W.; Cavalcanti, A.; Timmis, J.; Woodcock, J. RoboChart: modelling and verification of the functional behaviour of robotic applications. *Softw. Syst. Model.* **2019**, *18*, 3097–3149.
110. Fisher, M.; Mascardi, V.; Rozier, K.Y.; Schlingloff, B.; Winikoff, M.; Yorke-Smith, N. Towards a Framework for Certification of Reliable Autonomous Systems. *Auton. Agents Multiagent Syst.* **2021**, *35*, 8.
111. Farrell, M.; Bradbury, M.; Fisher, M.; Dennis, L.A.; Dixon, C.; Yuan, H.; Maple, C. Using Threat Analysis Techniques to Guide Formal Verification: A Case Study of Cooperative Awareness Messages. In Proceedings of the International Conference on Software Engineering and Formal Methods, Oslo, Norway, 16–20 September 2019; pp. 471–490.
112. Book, G. *Security Threats against Space Missions*. CCSDS Secretariat: Washington, DC, USA, 2006.
113. Farrell, M.; Bradbury, M.; Fisher, M.; Dixon, C. *Workshop Report: Space Security Scoping*; Technical report, FAIR-SPACE Hub: Guildford, England, 2019; <https://www.fairspacehub.org> (accessed on 28 April 2021).
114. Maple, C.; Bradbury, M.; Yuan, H.; Farrell, M.; Dixon, C.; Fisher, M.; Atmaca, U.I. Security-Minded Verification of Space Systems. In Proceedings of the 2020 IEEE Aerospace Conference, Big Sky, MT, USA, 7–14 March 2020; pp. 1–13.
115. Nof, S. *Handbook of Industrial Robotics*; Number v. 1 in Electrical and electronic engineering; Wiley: New York City, NY, USA, 1999.
116. British Standards Institution (BSI). BS 8611 Robots and Robotic Devices—Guide to the Ethical Design and Application. 2016. Available online: <http://www.bsigroup.com> (accessed on 28 April 2021).
117. Institute of Electrical and Electronics Engineers. *P7001—Transparency of Autonomous Systems*; IEEE, New York City, NY, USA; 2016.
118. Winfield, A.F.T.; Jirotko, M. The Case for an Ethical Black Box. In Proceedings of the 18th Conference on Towards Autonomous Robotic Systems (TAROS), Springer, Germany; 2017; Volume 10454, pp. 262–273.
119. Koeman, V.; Dennis, L.A.; Webster, M.; Fisher, M.; Hindriks, K. The “Why did you do that?” Button: Answering Why-questions for end users of Robotic Systems. In Proceedings of EMAS, 2019.
120. Dennis, L.A.; Oren, N. Explaining BDI agent behaviour through dialogue. In Proceedings of the 20th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS), London, UK, 3–7 May 2021.

121. Dennis, L.A.; Fisher, M.; Slavkovik, M.; Webster, M.P. Formal Verification of Ethical Choices in Autonomous Systems. *Robot. Auton. Syst.* **2016**, *77*, 1–14.
122. Kaur, H.; Nori, H.; Jenkins, S.; Caruana, R.; Wallach, H.M.; Vaughan, J.W. Interpreting Interpretability: Understanding Data Scientists' Use of Interpretability Tools for Machine Learning. In Proceedings of the International Conference on Human Factors in Computing Systems (CHI), 2020; pp. 1–14.
123. Falcone, R.; Castelfranchi, C. Socio-cognitive model of trust. In *Human Computer Interaction: Concepts, Methodologies, Tools, and Applications*; IGI Global: Hershey, PA, USA, 2009; pp. 2316–2323.
124. Fisher, M.; List, C.; Slavkovik, M.; Weiss, A. Ethics and Trust: Principles, Verification and Validation (Dagstuhl Seminar 19171). *Dagstuhl Rep.* **2019**, *9*, 59–86. doi:10.4230/DagRep.9.4.59.
125. Sebo, S.S.; Krishnamurthi, P.; Scassellati, B. "I Don't Believe You": Investigating the Effects of Robot Trust Violation and Repair. In Proceedings of the 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI), Daegu, Korea 11–14 March 2019, pp. 57–65.
126. Chatila, R.; Dignum, V.; Fisher, M.; Giannotti, F.; Morik, K.; Russell, S.; Yeung, K., Trustworthy AI. In *Reflections on Artificial Intelligence for Humanity*; Springer: New York City, NY, USA; 2021; pp. 13–39.
127. Ullman, D.; Malle, B.F. What Does it Mean to Trust a Robot?: Steps Toward a Multidimensional Measure of Trust. In Proceedings of the Companion of the 2018 ACM/IEEE International Conference on Human-Robot Interaction, (HRI), Chicago, IL, USA, 5–8 March 2018; pp. 263–264.
128. Tolmeijer, S.; Weiss, A.; Hanheide, M.; Lindner, F.; Powers, T.M.; Dixon, C.; Tielman, M.L. Taxonomy of Trust-Relevant Failures and Mitigation Strategies. In Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction (CHI), Cambridge, UK, 23–26 March 2020; pp. 3–12.
129. Vanderbilt, K.E.; Liu, D.; Heyman, G.D. The development of distrust. *Child Dev.* **2011**, *82*, 1372–1380.
130. Malle, B.F.; Fischer, K.; Young, J.E.; Moon, A.; Collins, E. Trust and the Discrepancy between Expectations and Actual Capabilities of Social Robots. In *Human-Robot Interaction: Control, Analysis, and Design*; Zhang, D., Wei, B., Eds.; Cambridge Scholars Publishing: New York, NY, USA, 2020.
131. Schwartz, R.; Dodge, J.; Smith, N.A.; Etzioni, O. Green AI. *Commun. ACM* **2020**, *63*, 54–63.
132. Andoni, M.; Tang, W.; Robu, V.; Flynn, D. Data analysis of battery storage systems. *CIR—Open Access Proc. J.* **2017**, *2017*, 96–99(3).
133. Boker, U.; Henzinger, T.A.; Radhakrishna, A. Battery Transition Systems. In Proceedings of the 41st ACM Symposium on Principles of Programming Languages, (POPL); San Diego, CA, USA, 22–24 January 2014; pp. 595–606.
134. Wognsen, E.R.; Hansen, R.R.; Larsen, K.G. Battery-Aware Scheduling of Mixed Criticality Systems. In *Leveraging Applications of Formal Methods, Verification and Validation. Specialized Techniques and Applications*; Margaria, T., Steffen, B., Eds.; Springer: Berlin/Heidelberg, Germany, 2014; pp. 208–222.
135. Zhao, X.; Osborne, M.; Lantair, J.; Robu, V.; Flynn, D.; Huang, X.; Fisher, M.; Papacchini, F.; Ferrando, A. Towards Integrating Formal Verification of Autonomous Robots with Battery Prognostics and Health Management. In *Software Engineering and Formal Methods*; Springer, Germany; 2019; pp. 105–124.
136. Bekhit, A.; Dehghani, A.; Richardson, R. Kinematic Analysis and Locomotion Strategy of active Pipe Inspection Robot Concept for Operation in Active Pipelines. *Int. J. Mech. Eng. Mechatronics* **2012**, *1*, 15–27.