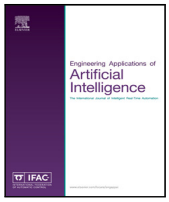
Engineering Applications of Artificial Intelligence 125 (2023) 106673

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai



A multidimensional Bayesian architecture for real-time anomaly detection and recovery in mobile robot sensory systems

Manuel Castellano-Quero [a,*], Manuel Castillo-López [b], Juan-Antonio Fernández-Madrigal [a], Vicente Arévalo-Espejo [a], Holger Voos [b], Alfonso García-Cerezo [a]

[a] Systems Engineering and Automation Department, University of Málaga, Campus de Teatinos, 29071 Málaga, Spain
[b] Automation and Robotics Research Group, Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg, 4369, Luxembourg

ARTICLE INFO

Keywords:
Mobile robots
Sensors
Bayesian networks
Fault diagnosis

ABSTRACT

For mobile robots to operate in an autonomous and safe manner they must be able to adequately perceive their environment despite challenging or unpredictable conditions in their sensory apparatus. Usually, this is addressed through ad-hoc, not easily generalizable Fault Detection and Diagnosis (FDD) approaches. In this work, we leverage Bayesian Networks (BNs) to propose a novel probabilistic inference architecture that provides generality, rigorous inferences and real-time performance for the detection, diagnosis and recovery of diverse and multiple sensory failures in robotic systems. Our proposal achieves all these goals by structuring a BN in a multidimensional setting that up to our knowledge deals coherently and rigorously for the first time with the following issues: modeling of complex interactions among the components of the system, including sensors, anomaly detection and recovery; representation of sensory information and other kinds of knowledge at different levels of cognitive abstraction; and management of the temporal evolution of sensory behavior. Real-time performance is achieved through the compilation of these BNs into feedforward neural networks. Our proposal has been implemented and tested for mobile robot navigation in environments with human presence, a complex task that involves diverse sensor anomalies. The results obtained from both simulated and real experiments prove that our architecture enhances the safety and robustness of robotic operation: among others, the minimum distance to pedestrians, the tracking time and the navigation time all improve statistically in the presence of anomalies, with a diversity of changes in medians ranging from $\simeq 20\%$ to $\simeq 500\%$.

1. Introduction

Mobile robots are increasingly present in a wide variety of contexts, including industrial (Galar et al., 2020), domestic (Luperto et al., 2019), healthcare (Kyrarini et al., 2021), commercial (Shakhatreh et al., 2019) and military (Kang et al., 2020), among many others. The majority of tasks for which mobile robots are deployed require platforms that operate with a certain degree of autonomy and safety; however, this is difficult to achieve in general due to the uncertain and complex scenarios where they operate. In this paper we are concerned with one of the most critical subsystems of a mobile robot in that sense: its sensory apparatus; in particular, the ability of the robot for diagnosing and overcoming sensory faults and misbehaviors, always taking uncertainty into account as well the real time requirements related to the particular task.

Faults can be classified according to the concrete aspect of the system they affect. In the general context of robotics, this may refer to problems related to either the robot hardware or software, or to other issues derived from the interaction of the robot with its environment (Khalastchi and Kalech, 2019a,c). This taxonomy of problematic situations represents a variety of undesirable conditions at different levels of abstraction. To illustrate this, consider, for instance, the problem of human motion detection and tracking (Beck and Bader, 2019; Brunetti et al., 2018), closely related to navigation safety in mobile robotics. From the perspective of the sensory apparatus, a faulty situation could provoke issues either at the low-level behavior of physical sensors (e.g., false detections of free space in *lidars* due to intense infrared lighting), at the level of more elaborate sensory information such as human presence detection (e.g., occlusion produced by static or dynamic obstacles) or at both. Abnormal situations like these should be addressed concurrently by the robotic agent for a correct operation, even though they are related to different ontologies and at the same time subjected to uncertainty and real-time constraints.

During the last decade, the research field of Fault Detection and Diagnosis (FDD) (Abid et al., 2021) has produced a broad variety of

* Corresponding author.
E-mail addresses: mcastellanoquero@gmail.com (M. Castellano-Quero), manucalop@gmail.com (M. Castillo-López), jafernandez@uma.es (J.-A. Fernández-Madrigal), varevalo@uma.es (V. Arévalo-Espejo), holger.voos@uni.lu (H. Voos), ajgarcia@uma.es (A. García-Cerezo).

https://doi.org/10.1016/j.engappai.2023.106673
Received 12 January 2023; Received in revised form 22 April 2023; Accepted 16 June 2023
Available online xxxx

methodologies aimed at addressing abnormal situations in the context of mobile robotics (Khalastchi and Kalech, 2019a,c). They can be classified either as *model-based*, *data-driven* or *knowledge-based* (Khalastchi and Kalech, 2019c; Cai et al., 2017a). Model-based approaches typically rely on a mathematical model of the expected behavior of a certain robotic component which is then compared to its actual, observed behavior in order to diagnose possible anomalies. Keliris et al. (2018), for instance, uses an estimation method for sensor fault detection and isolation based on the theory of non-linear systems, which is then tested in a simulated robotic manipulator. Data-driven methods, in contrast, do not employ models but instead use statistical information of expected robotic behavior for diagnosis. In Teh et al. (2020) a systematic review of common errors in physical sensor data, and techniques to detect and correct them can be found that deals particularly with principal component analysis (PCA) and artificial neural networks (ANN). Lastly, knowledge-based methodologies combine aspects of the previous approaches, usually with the aim of encoding human expert knowledge for the detection of abnormal conditions. This paradigm is applied, e.g., in Wang et al. (2021), by introducing novel fuzzy logic models to tackle sensory failures in the context of dynamical systems control.

Although the mentioned paradigms have demonstrated their utility for the diagnosis of abnormal conditions affecting sensors, they do not provide either a simultaneous treatment of such anomalies at different levels of abstraction or a rigorous mathematical representation of uncertainty, which affects the scope of their application, and they are not always focused on achieving real time performance. To solve the issue of dealing with uncertainty rigorously, there exist knowledge-based methodologies (Khalastchi and Kalech, 2019c) that rely on Bayesian networks (BNs) (Pearl, 1985). BN are grounded on a probabilistic framework that enables modeling complex systems and reasoning about them through the encoding of human expert knowledge (Russell and Norvig, 2020; Darwiche, 2009; Koller and Friedman, 2009). However, despite the successful application of Bayesian networks for FDD problems in general (Cai et al., 2017a), their use in the context of robotic sensory systems is scarce (Cai et al., 2020), and current implementations such as Lin et al. (2020) and Liu et al. (2020) do not leverage the possibility of dealing with very complex and general systems that BNs can provide, in part due to the lack of a suitable structure capable of integrating different ontologies of anomalies and diverse sources of information while providing real-time performance. In a previous work (Castellano-Quero et al., 2021) we contributed with a first proposal of a BN-based architecture aimed at the detection and recovery of a diversity of sensory anomalies, but it was restricted to low-level issues of physical devices, and the obtained performance was not suitable for real-time use in general.

In this paper we present an evolution of Castellano-Quero et al. (2021) aimed at overcoming the mentioned limitations and the ones of other existing works. In particular, we introduce a new architecture that can capture homogeneously a more generic variety of sensory abnormal conditions. This is based on a novel multidimensional paradigm that: (i) can encode causal relationships among the components of a robotic sensory system, including sensory data sources, anomalies detectors and recovery processes; (ii) enables the modeling of temporal evolution of sensory behavior, since it is grounded on *dynamic* Bayesian networks (DBNs) (Koller and Friedman, 2009), and (iii) provides a way of incorporating, in a systematic manner, knowledge related to different levels of cognitive abstraction, i.e., belonging to diverse ontologies that are, in principle, difficult to reconcile. Anomaly detection and recovery with our approach is possible thanks to the definition of a suitable inference algorithm for the new architecture that fuses all information in an efficient way by leveraging the multidimensional structure.

Last but not least, our architecture is complemented with a systematic procedure to compile the Bayesian network into feedforward neural networks (Aggarwal, 2018) which also leverages the particular topologies of the proposed BN-based model for training, thus our solution is a holistic BN + NN approach.

In order to assess both qualitatively and quantitatively the utility of our new proposal and its advantages in generality and performance, we have implemented it for a complex sensory system in the context of mobile robot navigation in environments with human presence. This represents a use case in which the incorporation of the new features of our proposal is essential (i.e., the concurrent handling of several levels of cognitive abstraction, the representation of sensory dynamics and the real-time performance). Statistical results obtained from both simulated and real experiments prove that our approach serves to enhance the safety and robustness of robotic operation and enables the design of robust and efficient FDD systems for complex tasks like that one. We have used well-known measures of safety to evaluate this; among others, the minimum distance to pedestrians, the tracking time and the navigation time get all better in the presence of anomalies, showing improvements with respect to the baseline ranging from $\simeq$ 20% to $\simeq$ 500% in medians. For this quantitative assessment, we have performed ablation studies both concerning the ability of parts of our system to deal with different anomalies and its real-time performance (with or without NN compilation), and also carried out comparative studies with variations of the proposal, using as baseline a modular, very general and publicly available pedestrian detection and tracking system for robots that also pursues robustness but is not based on Bayesian Networks (Linder and Arras, 2016).

The remainder of the paper is as follows. Section 2 presents a review on existing works devoted to fault diagnosis and recovery, particularly in the context of robotic systems, taking also into account related implementations of Bayesian networks. Section 3 provides a formal definition of our new multidimensional architecture for sensory diagnosis and recovery. Section 4 develops the theoretical aspects of the methodology introduced in this work for performing real-time inference in the previous model. Section 5 describes both the implementation of our proposal for a particular sensory system and its validation through a set of simulated and real experiments, including the quantitative analysis of the statistical results. Finally, Section 6 summarizes the most relevant contributions of this work, its current limitations, and possible future research directions.

2. Related works

The general aim of research in Fault Detection and Diagnosis (FDD) is to conceive novel methodologies that serve to identify or predict malfunctions as well as to tackle such problems (Abid et al., 2021). Successful implementations of FDD can be found in a wide variety of technologies, including a diversity of physical systems, e.g., motors (Abid et al., 2019; Cai et al., 2021, 2022), electric power devices (Vaish et al., 2021), discrete electronic components (Guo et al., 2022) or even in physical control systems (Kong et al., 2022, 2023) among many others. Also, FDD methodologies have been applied to software, e.g., for the detection of errors in run time (Zhou et al., 2018), for prediction of defects (Shao et al., 2018), or even for development tasks (Hirakawa et al., 2021). However, the application of FDD to robotics in general is relatively recent (Khalastchi and Kalech, 2019a), and traditional approaches have been found not to be always suitable for the particular constraints imposed by robotic systems (Khalastchi and Kalech, 2019c). In spite of that, research in this scope has been increasing in recent years (Long et al., 2021; Graham Miller and Gandhi, 2021; Azzalini et al., 2020; Das et al., 2021).

Research in FDD methods for dealing with hardware anomalies has an important focus on sensory devices, both exteroceptive (Bader et al., 2017) and proprioceptive (Liao et al., 2021). There also exist numerous contributions devoted to tackle actuation faults (Yu and Dong, 2019; Kadiyam et al., 2020) and even for addressing abnormal situations related to both sensors and actuators concurrently (Doran et al., 2020). From the point of view of robotic software, existing methodologies enable for the detection of issues provoked by faulty algorithmic implementations (He et al., 2019) as well as for the treatment of execution

errors (Katz et al., 2020). Also, there exist hybrid approaches that tackle both hardware and software abnormal behavior (Medina et al., 2022). Research in FDD related to interaction problems has also been intense in the last decades, especially in the case of coordination of multiple robotic agents (Rakesh and Shrivastava, 2022; Khalastchi and Kalech, 2019b). In contrast, FDD methods have not been traditionally applied for human–robot interaction (Khalastchi and Kalech, 2019c; Honig and Oron-Gilad, 2018), although research related to this topic has been gaining relevance recently (Tolmeijer et al., 2020; Kontogiorgos et al., 2020).

Mobile robotic platforms in particular are often used in highly uncertain settings while facing a variety of potential abnormal and challenging conditions (e.g, when deployed in industries, hospitals, homes, etc. Bonci et al., 2021; Alatise and Hancke, 2020). Although the proposed methods in this case have been proven to be effective, they either lack a mathematical, rigorous treatment of uncertainty (Doran et al., 2020; Tarapore et al., 2019) or they are restricted to only tackle problems related to particular systems or kinds of robotic platforms (Zahaf et al., 2022; Keipour et al., 2019). There exist recent techniques in machine learning research that aim to compactly model heterogeneous and complex sources of information from available data, e.g., learning algorithms for graph neural networks (GNNs) (Jiang et al., 2022) and novel deep learning models for robotic semantic perception (Singh et al., 2023). Although relevant for modeling complex systems, these approaches do not explicitly contemplate fault diagnosis.

To overcome these limitations, a number of works in the scope of general FDD ground their methods on the use of Bayesian networks (Cai et al., 2017a, 2020). Research in Bayesian network-based methods for FDD has been intense in the last decade for a wide diversity of applications (e.g., energy systems Mirnaghi and Haghighat, 2020, critical infrastructures Hossain et al., 2020, manufacturing Zhang et al., 2021, etc.); however, the restricted amount of current solutions for robotics do not leverage all the possibilities offered by BNs, and most of these proposals are still conceived to only address specific anomalies in particular tasks (Zhi and Yangi-Shang, 2020; Li and Yang, 2022; Matsuoka and Sawaragi, 2022).

In Castellano-Quero et al. (2021) we presented a Bayesian network-based methodology aimed at the modeling of any kind of abnormal condition affecting physical robotic sensors, which also enables for the recovery of sensory information under problematic situations. Although proven useful, this proposal still has some important limitations. First, it does not allows for the representation of temporal evolution of sensory information, i.e., sensory dynamics, and it can only handle discrete random variables. In this work we present a new definition of our modeling architecture based on the theories of *dynamic* Bayesian networks and hybrid Bayesian networks (Koller and Friedman, 2009), which enables the modeling with both continuous and discrete random variables. These probabilistic frameworks have been successfully applied in a diversity of other contexts related to FDD (Cai et al., 2017a). For instance, there exist applications of DBNs for the diagnosis of large-scale (Zhao et al., 2020) and small-scale (Cai et al., 2017b) physical systems, including terrestrial (Gomes and Wolf, 2021) and aerial (Dezan et al., 2020) autonomous vehicles, in which the use of hybrid BNs has also been explored (Gomes and Wolf, 2019).

Another relevant limitation of previous works is related to the modeling of sensory anomalies at different levels of cognitive abstraction, that is, dealing with different ontologies of abnormal circumstances. In the literature of general FDD, there exist some works that implement abstraction in the form of hierarchical models that have also been successfully applied for the diagnosis of a variety of physical systems (Li et al., 2022; Mondal et al., 2021; Gomes and Wolf, 2019). However, none of these works are conceived for the diagnosis of robotic sensory systems, which is our focus.

Finally, we also aim to perform anomaly detection and recovery in real time, which our model achieves through the hybridization with feedforward neural networks (Aggarwal, 2018) in a systematic manner, as explained in Section 4.3. In artificial intelligence jargon, the process of representing a probabilistic query as a function is called *compilation* of Bayesian networks (Darwiche, 2009), which has also been addressed by using arithmetic circuits (Darwiche, 2003). There exist a number of works in the literature that solve this problem by employing neural networks (Jia et al., 2017; Löwe et al., 2021), but they are mostly conceived for discrete BNs. Also, they are all devised for monolithic networks, and do not leverage the particular structure of the model for increasing the efficiency of training, as we propose in this work.

As a summary of the main features provided in our solution and their presence or absence in other works related to FDD in robots, please see Table 1. Also, in the case of applying FDD to the particular task of navigation in environments with human presence, Table 2 lists the main anomalies dealt with in literature, including this work.

3. A multidimensional representation of robotic sensory systems

In this section we cover the definition of the basic element that we need to represent the sensory system of a mobile robot, using for that Bayesian networks with a restricted structure according to the system constraints. This basic element and the most common queries for the sensory system are presented in Section 3.1, while the complete network architecture is described in Section 3.2. Here, we redefine the models of Castellano-Quero et al. (2021) to allow the incorporation of sensory dynamics, the concurrent treatment of different levels of sensory abstraction and the handling of both discrete and continuous random variables.

Formally, a Bayesian network (BN) for a set of n random variables $\mathbf{Z}$ is a pair (G, Θ) consisting of a directed acyclic graph G over variables $\mathbf{Z}$, called the *network structure*, and a set of Conditional Probability Distributions (CPDs) Θ for each variable in $\mathbf{Z}$, called the *network parametrization* (Darwiche, 2009; Koller and Friedman, 2009). This model represents the joint probability distribution over the variables in $\mathbf{Z}$. However, in this work we also aim to represent sensory dynamics. For that, we need to rely on an extended definition of Bayesian network for discrete-time stochastic dynamic processes, called Dynamic Bayesian Network (DBN). Defined on a set of state variables that evolve over time, i.e. $\mathbf{Z}^{(t)}$, a DBN is a pair ($\mathcal{B}_0$, $\mathcal{B}_{\rightarrow}$), where $\mathcal{B}_0$ is a Bayesian network defined only over the initial distribution of the state variables $\mathbf{Z}^{(0)}$, called *initial network*, and $\mathcal{B}_{\rightarrow}$ is a fragment of Bayesian network whose structure is defined over the union of state variables at adjacent finite time intervals, $\mathbf{Z}^{(t-1)} \cup \mathbf{Z}^{(t)}$, and only parameterized for variables $\mathbf{Z}^{(t)}$, called *transition network* (Koller and Friedman, 2009). In general, Bayesian networks defined over both continuous and discrete random variables are known as *hybrid* Bayesian networks (Koller and Friedman, 2009), which is the kind of model we will use in the present work.

As it will be detailed later on, the conditional probabilities that complete the Bayesian network definition can come from expert knowledge, environmental information, etc., or they can be learnt from experimental data. In this work some of these distributions are deduced automatically, e.g. in the case of *virtual evidences*, as will be described in Section 5.3; others are provided through expert knowledge, as we also explain in the experimental section. In general, automatic learning of these data is out of the scope of this work.

Once a Bayesian network is completely defined, it is possible to use it to obtain new information from available one, i.e., to perform inference, in particular in its forms of induction —generalizing the knowledge present in the net for coping with the input data— and abduction —hypothesizing the causes of those data, e.g., identifying anomalies—. In general, this process consists in obtaining the conditional distribution $P(\mathbf{Q}|\mathbf{E})$, where $\mathbf{Q}$ is the set of query variables (the ones of interest) and $\mathbf{E}$ the set of observed variables (also known as *evidence*, which represents the existing knowledge). In our context, $\mathbf{Z} = \mathbf{Q} \cup \mathbf{E}$ and $\mathbf{Q} \cap \mathbf{E} = \infty$. The notion of evidence mentioned here is usually called *certain* or *hard* evidence, since it is normally assumed that the knowledge it represents is obtained with no uncertainty. However,

Table 1
Relevant features offered by current state-of-the-art paradigms for FDD in robotics. Columns are: *Appli.:* Applicability to different platforms and tasks, *Heter.:* Integration of heterogeneous sources of information, *Dynam.:* Dealing with dynamic sensory behavior, *Ontol.:* Integration of different ontologies of sensory information, *RT:* Real-time performance.

Method	Feature				
	Appli.	Heter.	Dynam.	Ontol.	RT
Azzalini et al. (2020)	–	–	✓	–	–
Das et al. (2021)	✓	✓	–	–	–
Doran et al. (2020)	✓	✓	✓	–	–
Keipour et al. (2019)	–	–	✓	–	✓
Li et al. (2022)	–	✓	–	–	–
Long et al. (2021)	–	–	–	–	✓
Matsuoka and Sawaragi (2022)	–	✓	✓	–	–
Tarapore et al. (2019)	–	✓	–	–	–
Zahaf et al. (2022)	–	–	✓	–	–
This paper	✓	✓	✓	✓	✓

Table 2
Common anomalies addressed by current state-of-the-art methods for FDD in human motion detection and tracking. Columns are: *Sh.Oc.:* Short-term occlusions, *Ln.Oc.:* Long-term occlusions, *F.P.Det.:* False positive detections, *Mult.Id.:* Multiple identification issues.

Method	Anomaly			
	Sh.Oc.	Ln.Oc.	F.P.Det.	Mult.Id.
Abebe-Assefa et al. (2022)	✓	–	✓	–
Dimitrievski et al. (2019)	✓	–	✓	✓
Lei et al. (2022)	✓	–	✓	✓
Sumi and Santha (2019)	✓	–	✓	✓
Yang et al. (2023)	✓	✓	✓	–
Zou et al. (2020)	✓	–	✓	–
This paper	✓	✓	✓	✓

if the available information is not completely certain, the evidence is said to be *soft*, and it can be emulated by employing a method known as *virtual* evidence (Darwiche, 2009). For a more in-depth treatment of Bayesian networks and inference, please refer to Darwiche (2009) and Koller and Friedman (2009).

3.1. The Bayesian sensor

The basic unit of the proposed architecture is a so-called *Bayesian sensor*, which represents a single aspect of the robotic sensory system under study. Such aspect may be one of the quantities measured by the on-board sensors (e.g., distance, speed, temperature), or more elaborated sensory information built upon low-level data (e.g., the presence or category of a detected obstacle). In general, a Bayesian sensor is conceived to capture any sensory information about the state of the robot or its environment, regardless of the complexity or the level of abstraction.

Formally, a Bayesian sensor $\mathcal{B}$ can be defined either as a static or a dynamic Bayesian network over a set of variables $\mathbf{B_z}$, depending on whether it is necessary to encode the temporal dynamics. In the static case, $\mathcal{B} = (B_g, B_\theta)$, with B_g the graph and B_θ the set of CPDs, while, in the dynamic one, $\mathcal{B} = (\mathcal{B}_0, \mathcal{B}_{\rightarrow})$, with $\mathcal{B}_0$ the initial network and $\mathcal{B}_{\rightarrow}$ the transition network, each one with its corresponding structure and parametrization. Recall that a Bayesian sensor can be defined over both discrete and continuous random variables, i.e., it can be represented by a *hybrid* Bayesian network. In the context of this work, hybrid networks will always be *Conditional Linear-Gaussian* (CLG) Bayesian networks (Koller and Friedman, 2009). The general structure of this basic element is depicted in its two possible forms in Fig. 1; this network is organized as follows:

- **Ideal sensor node** (I): it encodes the true state of the sensory aspect being represented, i.e., in the absence of abnormal conditions. This variable will usually correspond to the query set in an inference task ($\mathbf{Q} = I$) made to recover from faulty or unreliable sensory information. The ideal node is also structurally employed for defining the relationships among the existing Bayesian sensors in the system, by using it as parent of other ideal nodes, and also for encoding the dynamics of the sensory information it represents, i.e., for encoding distributions of the form $\mathrm{P}(I^{(t)}|I^{(t-1)})$, if necessary. All of this is shown in Fig. 1.
- **Real sensor nodes** ($\mathbf{R}$): this is a set of variables representing the values actually measured —observed— by the related sensory sources available on-board the mobile robot (e.g., physical sensors, algorithms producing sensory knowledge, etc.). Here, $|\mathbf{R}| = n_s$, where n_s is the number of such sources. This set has the role of evidence in an inference task ($\mathbf{R} \subseteq \mathbf{E}$), as long as the measured, available information is absolutely certain, i.e., if it represents *hard* evidence. If the available information is uncertain, the set $\mathbf{R}$ is replaced by the set of child nodes $\mathbf{R}'$ (see Fig. 1) in order to allow the incorporation of *virtual* evidence. For that, the observed values are imposed as hard evidence on these nodes, and the uncertainty associated with them is encoded in their corresponding CPDs, which are of the form $\mathrm{P}(R'_i|R_i)$, with $R'_i \subset \mathbf{R}'$ (see Section 4.1 for more details).
- **Anomalies subnetwork** ($\mathcal{A}$): this is a Bayesian network, defined over variables $\mathbf{A_z}$, that serves to reason about the existence of abnormal conditions affecting a particular Bayesian sensor. For that, this network may incorporate information from external sources as well as from any other sensor in the system. Regardless of its particular form, it must contain a subset of discrete binary random variables $\mathbf{A} \subseteq \mathbf{A_z}$, called *anomalies nodes*, each one representing the existence of a particular adverse situation. Any of these variables may be part of the query set ($\mathbf{Q} = A_i \subseteq \mathbf{A}$) in order to identify the causes for abnormal sensory behavior.
- **Virtual subnetwork** ($\mathcal{V}$): this is also a Bayesian network, defined over variables $\mathbf{V_z}$, that is aimed at inferring supplementary information related to the sensory aspect being represented by the ideal node. Such knowledge would be essential to allow the recovery of the correct state of the sensor in case of abnormal situations. This subnetwork may also integrate knowledge from external sources or other sensors, and it must contain a variable $V \subset \mathbf{V_z}$, called *virtual node*, encoding such result.

The Bayesian sensor is defined over variables $\mathbf{B_z} = I \cup \mathbf{R} \cup \mathbf{R}' \cup \mathbf{A_z} \cup \mathbf{V_z}$. Note that the same set forms the definition of the dynamic version of the sensor, i.e., $\mathbf{B}_\mathbf{z}^{(t)}$, although, in that case, only the ideal variable I evolves over time (see Fig. 1).

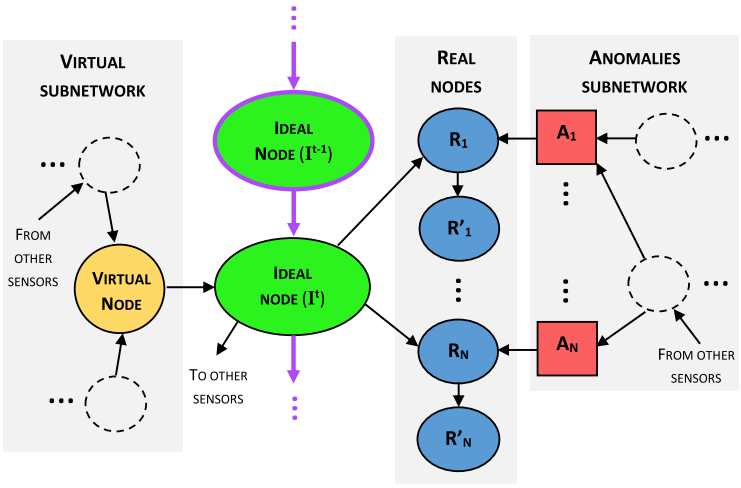


Fig. 1. Transition part $\mathcal{B}_{\rightarrow}$ of the dynamic Bayesian network representing the Bayesian sensor defined in this work. Both the initial network $\mathcal{B}_0$ of the dynamic Bayesian network and the static version of the sensor can be obtained by removing the node and the arcs highlighted in purple color. Squared nodes represent discrete random variables while the round ones may be either discrete or continuous.

To complete its definition, the network must also be parameterized by filling the corresponding CPDs. For that, we could use, for instance, expert knowledge, environmental information, etc., or we could even learn the parameters from experimental data (Koller and Friedman, 2009). The conditional probabilities assigned in the CPDs of the nodes (in our case, particularly the "Ideal nodes") allow the Bayesian network to do sensor fusion, since they take into account the probabilistic relations existing from other nodes (particularly, other sensors); when a query is solved, those conditional probabilities fuse information coming from different sources in a rigorous and coherent way thanks to its common probabilistic foundation.

All the existing Bayesian sensors in a sensory system and their relationships can be represented in a unique, *monolithic* Bayesian network that could be used either to recover from abnormal conditions or to identify the causes of those anomalies. These two tasks can be formally defined as queries over ideal and anomalies variables, respectively, and may be formulated in general for dynamic Bayesian networks as well. However, as we will explain later on, it is not always possible to get exact answers for such monolithic model.

Inference queries in the context of DBNs may adopt different forms. One of them is known as *filtering* (Murphy, 2002) and it is the one that will be used here. If the time slice representing the present in the DBN is denoted as t, the filtering query has the form $P(\mathbf{Q}^{(t)}|\mathbf{E}^{(0:t)})$. This notation aims to highlight the fact that the evidence variables range from the initial to the present time slice, i.e., $\mathbf{E}^{(0:t)} = \cup_{\tau=0}^{t}\mathbf{E}^{(\tau)}$. Here, we will only consider query sets with one variable, i.e., $|\mathbf{Q}| = 1$, since it is not necessary to compute the joint state of several sensors; instead, each one will be deduced sequentially, as explained in Section 4. Thus, a query variable $\mathbf{Q}^{(t)}$ could be either the ideal node of some Bayesian sensor $I^{(t)}$ or one of its anomaly nodes $A_i^{(t)} \subset \mathbf{A}^{(t)}$. Then, the evidence set for a given time slice t would be defined as:

$$\mathbf{E}^{(t)} = \bigcup_{i=1}^{n} \mathbf{R}_i^{(t)} \tag{1}$$

where each $\mathbf{R}_i^{(t)}$ represents the set of real sensor variables $\mathbf{R}$ for the ith Bayesian sensor $\mathcal{B}_i$ at time t in a sensory system with n sensors. Recall that the set $\mathbf{R}$ can be replaced by $\mathbf{R}'$ for some sensors in Eq. (1) depending on whether soft evidence is to be emulated or not, as explained before. In the particular case that all the Bayesian sensors in a system were static, queries must be re-formulated without considering the evolution of time, as described in Castellano-Quero et al. (2021).

3.2. A multidimensional Bayesian architecture for robotic sensory systems

The complete representation of the sensory apparatus of a mobile robot could be formalized as a monolithic Bayesian network containing all Bayesian sensors required in the system, with an arbitrarily complex structure, as explained in Section 3.1. Unfortunately, using that model for inference tasks poses several important drawbacks, starting with the computational cost of inference in a monolithic network (exponential in its *width* and linear in the number of nodes).

We address these issues in the following subsections. With the notions introduced there, it is possible to define a complete Bayesian sensory architecture for the representation of very complex robotic sensory systems, but take into account that this architecture is actually an approximate representation of the monolithic network of the whole sensory system, since it is based on different kinds of partitions of that model.

More concretely, the proposed sensory architecture is grounded on the propagation and fusion of information in three different ways (i.e., layered, cognitive and dynamic). Thus, the architecture could be described as a three-dimensional model whose axes are (as an illustration of this structure, please see Fig. 3):

- **Layered axis**: it represents an ordering of sensors according to their mutual dependencies, and thus, to their behavior in the system.
- **Temporal axis**: it describes the evolution of the information represented by a particular sensor over discrete time intervals.
- **Cognitive axis**: it represents different levels of cognitive abstraction (i.e., ontologies).

One of the core contributions of this paper is the demonstration that this multidimensional architecture not only preserves the natural sensor fusion capabilities of Bayesian networks (with some approximation error), but also enables the representation of very diverse, heterogeneous knowledge, that is, more complex networks can be devised within a rigorous and coherent framework.

3.2.1. Breaking exponential complexity: the layered axis

This was already addressed in Castellano-Quero et al. (2021), in which we introduced an approximate approach, based on a so-called *layered* network, that strategically splits the monolithic model in order to improve the efficiency of inference. In this work we re-use that

proposal to deal with different levels of cognitive abstraction and also adapt it to the incorporation of sensory dynamics through the use of Dynamic Bayesian Networks instead of the more conventional BNs.

The modeling of any sensory system with our architecture begins with the definition of the set of all the dependencies existing among its components. A dependency arises whenever the behavior of some sensory source can be explained or affected by that one of another. Unfortunately, we showed in Castellano-Quero et al. (2021) that it is also possible to get *cyclic* dependencies between pairs of sensors, i.e., to find that their behaviors mutually influence each other. For instance, this may happen for the case of two different sensors that capture similar physical magnitudes (e.g., a wheel encoder and a gyroscope). Cyclic Bayesian network models prevent inference in general. In that previous work, we solved this issue with an approximate approach that breaks each pair of cyclic dependencies into two non-cyclic ones in a systematic manner and that then compiles all those dependencies in a graph. This model, called dependency graph D_G, is annotated in this work to allow the incorporation of information about sensory dynamics and levels of abstraction. Please refer to Appendix A for its definition and a procedure for its construction.

In a nutshell, each layer is identified in D_G by a number $\alpha \in \mathbb{N}$, establishing a hierarchical ordering of sensors in which the lower ones explain or affect the behavior of the higher ones. Thus, a layer $\mathbf{L}_\alpha$ is a set of Bayesian sensors $\mathbf{L}_\alpha = \{\mathcal{B}_a, \mathcal{B}_b, \mathcal{B}_c, \dots\}$ each one verifying that the length of the longest directed path ending at the considered sensor in graph D_G is α.

In order to adapt this layered structure to different levels of abstraction as well as the temporal dynamics associated with the existing sensors, the complete sensory system of a mobile robot is represented as a pair $\mathcal{S} = (D_G, \mathcal{L})$, where D_G is the annotated dependency graph, which includes the information about sensory dynamics and levels of abstraction, and $\mathcal{L} = \{\mathbf{L}_0, \mathbf{L}_1, \dots, \mathbf{L}_n\}$ is a set of all the existing layers in the sensory system, which contains the definition of all the Bayesian sensors in it. In the proposed model, each layer or level $\mathcal{L}$ is an isolated Bayesian network. For this reason, ideal nodes are to be instantiated (i.e., replicated) for every Bayesian sensor that needs them. These replicated nodes serve as an *interface* to propagate information through the existing levels in the model, for the purposes of inference. The connection this interface represents is not always of the same nature, and it can be implemented in two different ways, as explained below.

3.2.2. Enabling dynamics: the temporal axis

The incorporation of sensory dynamics is simply done by using replicated ideal nodes representing ideal variables in the immediately previous time interval (e.g, $t-1$) as parents of the ones referring to the current interval t (see Fig. 1). More formal details have already been provided at the beginning of Section 3.

3.2.3. Improving generality and ordering complexity: the cognitive axis

Another one of the important issues mentioned above comes from the necessity of modeling sensory systems with heterogeneous components that produce information related to different *ontologies* or *levels of cognitive abstraction* at the same time (e.g., a sensory system simultaneously estimating the *pose* and the *identity* of a detected pedestrian); they are difficult to reconcile since they are subjected to different operations and relations, and this hinders the production of good network designs; more concretely, having sensors at different levels of abstraction in the same network might introduce, in general, undesired dependencies among them, potentially leading to inaccurate or unreliable inference results and increasing structural complexity and therefore the cost of inference.

The novel solution we propose in this work is to treat such sensors in separate networks, each one with a layered structure as explained before and based on DBNs. This has already been contemplated in the definition of the Bayesian sensor in Section 3.1, since it counts with a mechanism that allows decoupling from other sensors based on *virtual* evidence: in the case that a sensor needs to be connected to another one belonging to a higher level of cognitive abstraction, it would suffice to replace such connection by asserting the conclusions produced by the former sensor as virtual evidence in the latter.

The graph D_G encodes a dependency relationship between two sensors by representing one of them as a parent of the other. When the two sensors belong to the same level of cognitive abstraction, the dependency is said to be *purely layered*, since each sensor would be assigned to a different layer of the ones in $\mathcal{L}$. On the other hand, when the two sensors do not belong to the same level of abstraction, the dependency is said to be *cognitive*, since they are assigned to different layers and also to different levels of cognitive abstraction at the same time. In the first case, the connection is implemented by just replicating the ideal node corresponding to the parent sensor in the layer assigned to the child one. However, in the second case, two real nodes R and R' must be added as descendants of the replicated ideal one (see Fig. 1). This means that the connection between sensors belonging to different levels of cognitive abstraction is implemented by asserting *virtual* evidence in an auxiliary Bayesian sensor created in the higher level (see Section 4.1).

3.2.4. Discrete nodes with continuous parents

There is an extra constraint that has to be considered for the construction of our sensory architecture. The inference algorithm developed in this work, which will be introduced in Section 4, is conceived to only handle CLG Bayesian networks, i.e., hybrid networks in which discrete nodes are not allowed to have continuous parents (Koller and Friedman, 2009). For this reason, the architecture construction procedure is affected every time a sensor defined over continuous variables needs to convey information to another one based on discrete variables. If this happens, the connection between sensors is implemented as in the case of cognitive dependencies (i.e., through virtual evidence) with the exception that the support of the replicated ideal variable would also be discretized.

3.2.5. A simple example

A model of the complete sensory system can be obtained by implementing Algorithm 1. This is a contruction procedure for the sensory architecture given an updated list of Bayesian sensors and an annotated dependency graph for them (both obtained from the application of Algorithm A.1).

The construction procedure is illustrated more clearly with an example. Consider, for instance, a sensory system with three Bayesian sensors $\mathcal{B}_a$, $\mathcal{B}_b$ and $\mathcal{B}_c$ (for simplicity, they will be referred to as a, b and c). In this example, sensors b and c are defined over continuous random variables, and sensor a over discrete ones. Concerning cognitive levels, the ontologies related to sensor c belong to a low level of cognitive abstraction, while the ones related to sensors a and b belong to a higher level. Regarding temporal dynamics, only sensor c represents information that evolves over time. Also, there is a cyclic dependency between sensors a and b, while sensor c influences sensor a (see Fig. 2(a)).

The lists representing all the previous constraints (see Appendix A) are as follows: $\mathbf{B_{id}} = \{a, b, c\}$, $\mathbf{C_{id}} = \{1, 1, 0\}$, $\mathbf{T_{id}} = \{\bar{d}, \bar{d}, d\}$, $\mathbf{D_{nc}} = \{(c, a)\}$ and $\mathbf{D_c} = \{(a, b), (b, a)\}$. By applying Algorithm A.1, the set of dependencies is updated so that $\mathbf{D_{nc}} = \{(c, a), (a, b), (b, d), (a, d)\}$ (in this case, a new instance of sensor a named d is created). The resulting annotated dependency graph is shown in Fig. 2(b); the monolithic model of the sensory system is depicted in Fig. 3(a) and the resulting architecture is shown in Fig. 3(b).

This theoretical example aims to show all the capabilities of the proposed architecture at once. Please refer to Section 5.3 for an implementation in a real robotic context.

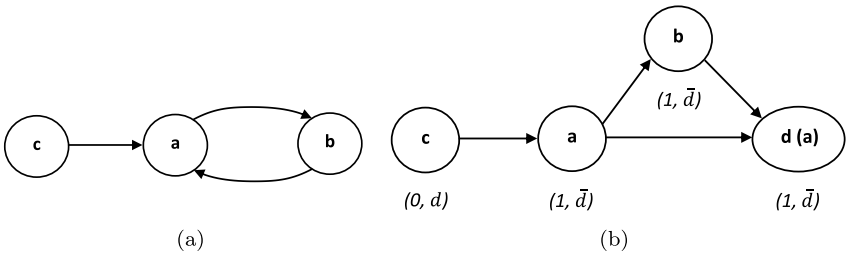


Fig. 2. Dependencies among Bayesian sensors for the example in the text. (a). Cyclic directed graph form. (b) Annotated dependency graph form. Recall that the sensor labeled as d is a new instance of sensor a.

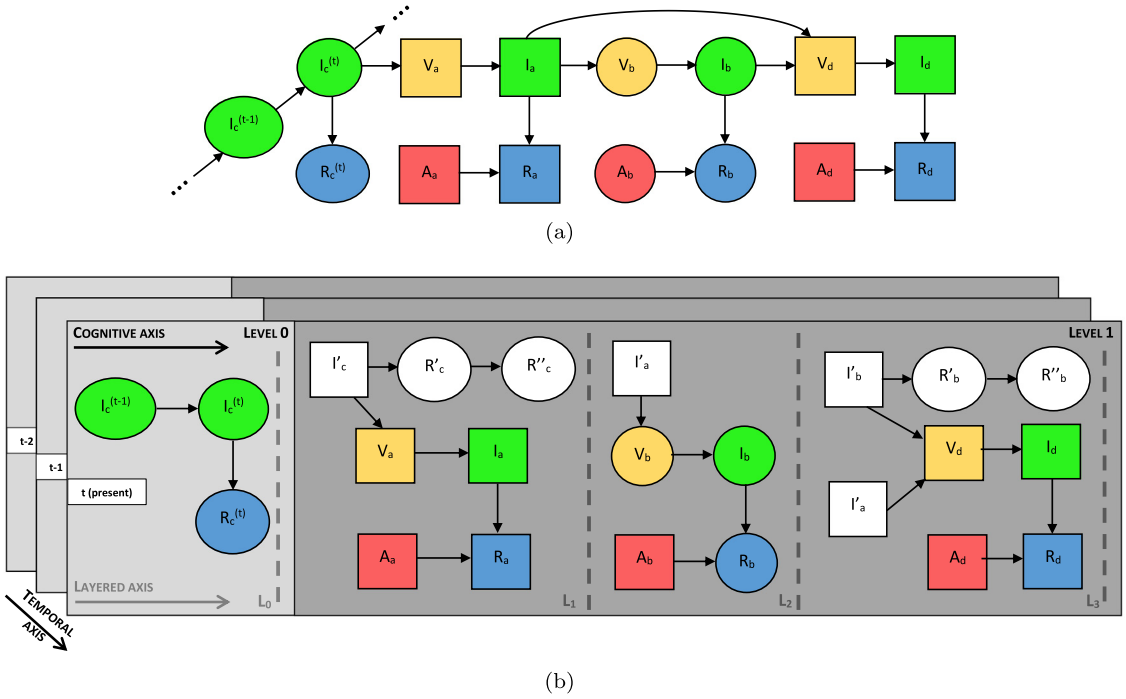


Fig. 3. Bayesian network representing the sensory system for the example in the text. Colors indicate the kind of node according to the definition of Bayesian sensor (see Fig. 1). Here, replicated ideal nodes and auxiliary Bayesian sensors are in white. Squared nodes represent discrete random variables and round ones represent continuous random variables. The nodes that are not marked with any time interval actually correspond to the current one t. (a) Transition network corresponding to the monolithic form of the sensory system. (b) Three-dimensional approximate model of the system obtained from Algorithm 1.

4. Real-time approximate reasoning in robotic sensory systems

This section presents the foundations of the algorithm we propose in this work for performing inference in our architecture. Such method, described in Section 4.1, is an extension of the one we developed in Castellano-Quero et al. (2021) that now handles multidimensional representations of sensory systems and allows for the treatment of both discrete and continuous random variables. In Section 4.2 we discuss the reasons why our inference method is approximate and its computational cost. After that, in Section 4.3, we introduce a methodology that leverages the proposed algorithm to compile probabilistic queries as feedforward neural networks, and that is aimed at enabling inference in real time.

4.1. Inference in multidimensional sensory systems

We propose a methodology conceived to perform efficient inference by leveraging the particular structure of the multidimensional model introduced in this work, which is an approximate, partitioned representation of the monolithic model of a sensory system, as explained in Section 3.2. Thus, the obtained results from the application of this method will also be approximate.

The proposed algorithm mainly consists in asserting posterior distributions related to ideal nodes of Bayesian sensors (see Section 3.1) as either prior or virtual evidences of their corresponding instances placed in other parts of the model, i.e., either in some higher layered level or in the next time slice. Our method begins by processing the layered axis, from the lowest level to the one containing the desired query variable, in sequential order. In the case that there is any dynamic sensor in the architecture, this whole process involves every time interval t until the desired one t^* is reached, taking into account that all the layers in the architecture must be processed for $t < t^*$. In this case, the obtained posterior distributions for dynamic sensors are stored for their use in future time intervals. Inference in a given layer is done by applying an adapted version of the exact *jointree* algorithm for CLG Bayesian networks (Koller and Friedman, 2009; Murphy, 2002).

As explained in Section 3.2, the propagation of information (represented in this case by posterior distributions) can be performed in three different ways, and this is done within each layer of the model as follows. To begin with, a posterior distribution corresponding to some ideal node is obtained by performing inference on its corresponding layer. Then, the distribution can be used in a different layer in one out of three different ways depending on the kind of connection existing between the sensors involved. First, if the posterior is obtained in a previous time interval $t-1$, the distribution is encoded as a prior of the

Algorithm 1 Construction of the multidimensional Bayesian architecture

input:
$\mathbf{B_{id}}$: updated list of identifiers of Bayesian sensors (algorithm A.1)
D_G: annotated dependency graph (algorithm A.1)

output:
S: sensory system

subroutines:
parents(G, n):
$P \leftarrow$ subset of vertices in graph G that are parents of node n
return P

cognitiveConnection(D_G, i, j):
$C \leftarrow$ boolean value indicating whether nodes i and j in D_G belong to a different cognitive level
return C

cdConnection(i, j):
$C \leftarrow$ boolean value indicating whether Bayesian sensors $\mathcal{B}_i$ and $\mathcal{B}_j$ are based on discrete and continuous variables respectively
return C

longestPath(G, n):
$\alpha \leftarrow$ length of the longest directed path in graph G ending at node n (see Cormen et al., 2022; Ahammad et al., 2020)
return α

main:
$\mathcal{L} \leftarrow \infty$ (empty set of layers)
for each identifier $i \in \mathbf{B_{id}}$ **do**
  $P \leftarrow$ *parents*(D_G, i)
  $\mathcal{B}_i \leftarrow$ design a complete Bayesian sensor (see Fig. 1) including new instances of the ideal nodes from the Bayesian sensors identified by P
  for each node $j \in P$ **do**
    $CC \leftarrow$ *cognitiveConnection*(D_G, i, j)
    $CD \leftarrow$ *cdConnection*(i, j)
    if $(CC \vee CD)$ is true **then**
      add auxiliary Bayesian sensor for virtual evidence in node j of $\mathcal{B}_i$
      if CD is true **then**
        discretize the support of variable associated with node j of $\mathcal{B}_i$
      end if
    end if
  end for
  $\alpha \leftarrow$ *longestPath*(D_G, i)
  if $\nexists \mathbf{L}_\alpha \in \mathcal{L}$ **then**
    $\mathbf{L}_\alpha \leftarrow \infty$ (empty set of Bayesian sensors)
    $\mathbf{L}_\alpha \leftarrow \mathbf{L}_\alpha \cup \mathcal{B}_i$
    $\mathcal{L} \leftarrow \mathcal{L} \cup \mathbf{L}_\alpha$
  else
    $\mathbf{L}_\alpha \leftarrow \mathbf{L}_\alpha \cup \mathcal{B}_i$
    replace old layer $\mathbf{L}_\alpha \in \mathcal{L}$ with the current one
  end if
end for
$S = (D_G, \mathcal{L})$
return S

corresponding ideal instance in the desired layer L for the current time interval t, i.e:

$$\mathrm{P}(I_L^{(t)}) = \mathrm{P}(I_L^{(t-1)} | \mathbf{E}^{(0:t-1)}), \tag{2}$$

where $I_L^{(t)}$ refers to some ideal node variable belonging to layer L. Second, if the dependency is purely *layered*, the posterior is encoded as a prior distribution of the corresponding replicated ideal node:

$$\mathrm{P}(I_{L_r}^{(t)}) = \mathrm{P}(I_L^{(t)} | \mathbf{E}^{(0:t)}), \tag{3}$$

where L_r denotes a higher layer containing the corresponding replicated instance of ideal node variable $I_L^{(t)}$. Lastly, if the dependency is *cognitive*, the posterior is encoded as an uncertain observation (i.e, as virtual evidence) of the auxiliary Bayesian sensor created in the target layer (see Algorithm 1). In this work, we implement virtual evidence as follows. Let R be one of the real sensor nodes in a certain Bayesian sensor and R' its auxiliary child (see Fig. 1). If R represents a continuous random variable, soft evidence consists in the observation of a certain value $\boldsymbol{\mu} \in \mathbb{R}^n$, with n the cardinality of R, along with an uncertainty defined by a positive semi-definite $n \times n$ covariance matrix Σ^2. Then, virtual evidence is implemented by asserting hard evidence $R' = \boldsymbol{\mu}$ and by parameterizing:

$$p(R' | R) = \mathcal{N}(R; \Sigma^2), \tag{4}$$

where p denotes a density and $\mathcal{N}$ represents a normal distribution whose mean is the same as the value of parent variable R and its variance, the observed uncertainty Σ^2. On the other hand, if R represents a discrete random variable, soft evidence consists in the observation of a mass distribution for variable R itself, given by a vector of real parameters $\boldsymbol{\Theta} = (\theta_1, \theta_2, \ldots, \theta_n)$ with $\sum_{i=1}^{n} \theta_i = 1$. If we denote the possible values for variable R as $r_1, \ldots, r_n$, virtual evidence is implemented by asserting hard evidence $R' = r_1$ and by parameterizing:

$$\mathrm{P}(R' = r_i | R = r_i) = \theta_1, \qquad \forall i \tag{5}$$

$$\mathrm{P}(R' = r_1 | R = r_i) = \theta_i, \qquad \forall i \tag{6}$$

and

$$\mathrm{P}(R' = r_i | R = r_j) = \theta_i, \qquad \forall (i > 1) \wedge (i \neq j). \tag{7}$$

The proposed inference methodology, which uses the subroutine described in algorithm 2, is presented more formally in algorithm 3 in its dynamic form, since that is the most general case. This proposal works by performing inference with the exact *jointree* on each one of the multiple CLG Bayesian networks in the multidimensional architecture, in the order indicated in the algorithms. The *jointree* inference method transforms an input Bayesian network into a secondary model, called the *jointree* $\mathcal{J}$, which is an undirected graph that is annotated with subsets of the joint distribution of the problem, called *factors*, $\mathcal{F} = \{\mathbf{F}_1, \mathbf{F}_2, \ldots\}$. The target query can always be obtained from at least one of this factors by marginalizing it. Please refer to Koller and Friedman (2009) for more details.

4.2. Approximate inference and computational complexity

In summary, the inference method proposed in this work is approximate due to two reasons. One the one hand, the defined multidimensional architecture is itself an approximate, partitioned representation of the whole monolithic model of a sensory system, as explained before. On the other hand, our proposal is also approximate because it might not take into account all the available pieces of evidence, due to the way the layered axis is defined, as we explain below.

For a given time slice, our algorithm propagates posterior distributions from layer zero to the desired one, regardless of the following higher layers. For this reason, it will not consider the evidences associated with those sensors from which there is no directed path to the sensor of interest. This leads to a more reduced set of evidences than the actual one, from the point of view of the layered axis. However, such axis is not the only one affected, since the format of query used here for temporal dynamics is the one of *filtering*, which accumulates evidences from the initial time interval to the present one. This also implies that, for a filtering query in a particular variable, the reduced set of evidences would lack the same subset of variables for all the considered time intervals. Our proposal provides an approximation given by:

$$\mathrm{P}(\mathbf{Q}^{(t)} | \mathbf{E}^{(0:t)}) \approx \mathrm{P}_a(\mathbf{Q}^{(t)} | \mathbf{E}_{\mathbf{r}}^{(0:t)}), \tag{8}$$

Algorithm 2 *processLayer*($\mathbf{L}_i$, $\mathbf{e}$, t, $\mathcal{T}^{(t-1)}$, $\mathcal{T}^{(t)}$)

input:
$\mathbf{L}_i$: i-th layer from $\mathcal{L}$ in sensory system $\mathcal{S}$
$\mathbf{e}$: set of evidences from sensory data
t: current time interval (if necessary)
$\mathcal{T}^{(t-1)}$: set of former posterior distributions for ideal nodes in dynamic sensors (if necessary)
$\mathcal{T}^{(t)}$: equivalent set of posteriors for the current time interval (if necessary)

output:
$\mathcal{P}$: set of posterior distributions of ideal and anomaly nodes of layer $\mathbf{L}_i$
$\mathcal{T}^{(t)}$: updated set of current dynamic posteriors

subroutines:
joinTree($\mathcal{N}$, $\mathbf{e}$):
  $\mathcal{D} \leftarrow \infty$
  $(\mathcal{J}, \mathcal{F}) \leftarrow$ apply the CLG version of the exact *jointree* algorithm (Koller and Friedman, 2009) to network $\mathcal{N}$ with evidences $\mathbf{e}$
  for each ideal node I_j and anomaly node A_k in network $\mathcal{N}$ **do**
    $\mathbf{P} \leftarrow$ get distribution $\mathrm{P}(I_j|\mathbf{e})$ from an appropriate $\mathbf{F}_i \in \mathcal{F}$
    $\mathcal{D} \leftarrow \mathcal{D} \cup (\mathbf{P}, I_j)$
    $\mathbf{P} \leftarrow$ get distribution $\mathrm{P}(A_k|\mathbf{e})$ from an appropriate $\mathbf{F}_i \in \mathcal{F}$
    $\mathcal{D} \leftarrow \mathcal{D} \cup (\mathbf{P}, A_k)$
  end for
return $\mathcal{D}$

updateDynamics($\mathbf{L}_i$, $\mathcal{P}$):
  $\mathbf{L}_i \leftarrow$ update priors of all the ideal nodes of dynamic sensors in layer $\mathbf{L}_i$ using posteriors from $\mathcal{P}$
return $\mathbf{L}_i$

storeDynamics($\mathcal{T}^{(t)}$, $\mathcal{P}$):
  $\mathcal{D} \leftarrow$ set of pairs $(\mathbf{P}, I_i^{(t)})$ representing posteriors of dynamic ideal variables from $\mathcal{P}$
  $\mathcal{T}^{(t)} \leftarrow \mathcal{T}^{(t)} \cup \mathcal{D}$
return $\mathcal{T}^{(t)}$

main:
1: $\mathbf{e}_i \leftarrow$ subset of evidences for layer $\mathbf{L}_i$ ($\mathbf{e}_i \subset \mathbf{e}$)
2: **if** $t > 0$ **then**
3: $\mathbf{L}_i \leftarrow updateDynamics(\mathbf{L}_i, \mathcal{T}^{(t-1)})$
4: **end if**
5: $\mathcal{P} \leftarrow joinTree(\mathbf{L}_i, \mathbf{e}_i)$
6: $\mathcal{T}^{(t)} \leftarrow storeDynamics(\mathcal{T}^{(t)}, \mathcal{P})$
7: **return** $(\mathcal{P}, \mathcal{T}^{(t)})$

Algorithm 3 Approximate inference in a Bayesian sensory architecture

input:
$\mathcal{S}$: sensory system (see algorithm 1)
$\mathbf{e}$: set of evidences from sensory data
Q: query variable (ideal or anomaly node)
L: layer associated with variable Q
t: current time interval (if necessary)
$\mathcal{T}^{(t-1)}$: set of former posterior distributions for ideal nodes in dynamic sensors (if necessary)

output:
$\mathrm{P}(Q^{(t)}|\mathbf{E_r}^{(0:t)} = \mathbf{e_r}^{(0:t)})$: filtered distribution for variable Q at time t given the *reduced* evidence (see text)
$\mathcal{T}^{(t)}$: set of current posterior distributions for ideal nodes in dynamic sensors (if necessary)

subroutines:
getDistribution($\mathcal{D}$, N):
return $\mathbf{P}$ as the probability distribution in $\mathcal{D}$ associated with node N

main:
1: $\mathcal{T}^{(t)} \leftarrow \infty$ (empty set of posteriors)
2: $\mathcal{L} \leftarrow$ initialize all the priors corresponding to replicated ideal nodes with uniform distributions
3: $(\mathcal{P}, \mathcal{T}^{(t)}) \leftarrow processLayer(\mathbf{L}_0, \mathbf{e}, t, \mathcal{T}^{(t-1)}, \mathcal{T}^{(t)})$
4: **for** $i = 1$ to L **do**
5: **for each** Bayesian sensor $\mathcal{B}_j \in \mathbf{L}_i$ **do**
6: $\mathbf{C} \leftarrow$ set of replicated ideal nodes in $\mathcal{B}_j$
7: **for each** node $c_k \in \mathbf{C}$ **do**
8: $\mathbf{P}_k \leftarrow getDistribution(\mathcal{P}, c_k)$
9: **if** c_k is part of an auxiliary Bayesian sensor (see algorithm 1) **then**
10: assert virtual evidence in that sensor by using $\mathbf{P}_k$
11: **else**
12: replace current CPD for node c_k by $\mathbf{P}_k$
13: **end if**
14: **end for**
15: **end for**
16: $(\mathcal{P}, \mathcal{T}^{(t)}) \leftarrow processLayer(\mathbf{L}_i, \mathbf{e}, t, \mathcal{T}^{(t-1)}, \mathcal{T}^{(t)})$
17: **end for**
18: $\mathrm{P}(Q^{(t)}|\mathbf{E_r}^{(0:t)} = \mathbf{e_r}^{(0:t)}) \leftarrow getDistribution(\mathcal{P}, Q)$
19: **return** $(\mathrm{P}(Q^{(t)}|\mathbf{E_r}^{(0:t)} = \mathbf{e_r}^{(0:t)}), \mathcal{T}^{(t)})$

where P_a denotes distributions over our approximate model and $\mathbf{E}_\mathbf{r}^{(0:t)} = \cup_{\tau=0}^{t} \mathbf{E}_\mathbf{r}^{(\tau)}$ represents a reduced set of evidences (i.e., $\mathbf{E}_\mathbf{r}^{(0:t)} \subseteq \mathbf{E}^{(0:t)}$). For a particular time slice t:

$$\mathbf{E}_\mathbf{r}^{(t)} = \{E | E \in \mathbf{E}^{(t)} \wedge directedPath(\mathbf{Q}^{(t)}, E) : E, \mathbf{Q}^{(t)} \in S_g\}, \tag{9}$$

where *directedPath* is true if there is at least one directed path from the variable in $\mathbf{Q}^{(t)}$ towards variable E in the graph of the monolithic DBN representing the sensory system, S_g.

There is also another issue that needs to be considered regarding the approximation in Eq. (8). In general, it is not always possible to compare such approximate inference to an exact distribution defined over a complete monolithic model, for two different reasons. One of them is that the monolithic model of a sensory system is not conceived to be a CLG Bayesian network in general, since it may contain discrete variables with continuous parents (this happens in the model of Fig. 3(a)). Exact inference is not possible in such a model, and the existing alternatives address the inference problem by using approximate numerical integration and particle-based methods (Koller and Friedman, 2009). The other reason refers to the treatment of different cognitive levels in the same network (this also happens in the model of Fig. 3(a)), which we avoid by splitting the model into separate parts through the mechanism of virtual evidence. For both situations, such an split monolithic model will be the one considered for the definition of the *exact* distributions referred to in Eq. (8).

The computational complexity of the inference method proposed in this work depends on the cost of the jointree algorithm, which can be expressed as $O(n \exp(w))$, with n the number of nodes in the jointree structure and w its *width* (please refer to Darwiche (2009) for further details). Note that this exponential character of the cost is mostly given by the presence of discrete random variables in the problem (Koller and Friedman, 2009). Since our inference algorithm applies this exact method once for each layer in the architecture, the complexity for one time interval can be expressed as:

$$O(n_1 \exp(w_1) + n_2 \exp(w_2) + \cdots + n_i \exp(w_i) + \cdots + n_L \exp(w_L)), \tag{10}$$

where L is the number of layers in the architecture. This expression should be multiplied by the total number of time intervals (from the initial to the present one), however, this is not necessary in our case, for the following reason. Many inference methods for DBNs perform *online* filtering by considering the complete set of evidences on every slice. The proposed algorithm, which can also be used for *online* inference, only incorporates evidences related to a particular time interval, thanks to the storage of certain posterior distributions across different slices. This makes unnecessary to accumulate all the existing evidences, as in the mentioned algorithms.

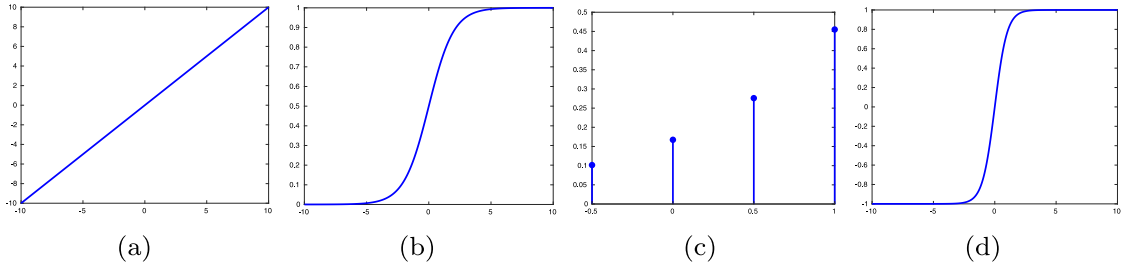


Fig. 4. Shapes of the activation functions used for the definition of the neural network models employed in this work. (a) Purely linear. (b) Logistic sigmoid. (c) Softmax (output corresponding to a four-dimensional input vector). (d) Hyperbolic tangent.

4.3. Enabling real-time inference with feedforward neural networks

The approximate inference algorithm introduced in Section 4.1 enables to reduce the computational cost w.r.t inference over monolithic models under certain conditions. Regarding Eq. (10), we can affirm that $w_i \leq w_m$, being w_m the treewidth of the complete monolithic network. This holds because our architecture is defined by deleting arcs among different layers, which also implies the elimination of any existing undirected loop in the original model, generally leading to a more reduced treewidth (Darwiche, 2009). However, this algorithm does not guarantee real-time performance in general, since the treewidth of each layer is not bounded and the computation time would still be exponential.

Fortunately, it is possible to overcome this efficiency limitation of our algorithm by leveraging certain aspects of its implementation. For that, it is important to note that a Bayesian network can be seen as a family of functions, each one corresponding to a particular query $\mathrm{P}(\mathbf{Q}|\mathbf{E})$; thus, a BN defined over variables $\mathbf{Z}$ represents a single function f for each possible partition of this set into $\mathbf{Q}$ and $\mathbf{E}$ such that $\mathbf{Z} = \mathbf{Q} \cup \mathbf{E}$ and $\mathbf{Q} \cap \mathbf{E} = \infty$. Each function f would then map an instantiation of the evidence to a set of parameters representing the target probability distribution. Formally, for the case of hybrid Bayesian networks:

$$f : \mathbb{N}^d \times \mathbb{R}^c \rightarrow \mathbb{R}^p, \tag{11}$$

where d is the cardinality of the subset of the evidence defined over discrete random variables $\mathbf{E}_d$, c the cardinality of the subset of the evidence defined over continuous random variables $\mathbf{E}_c$ and p the dimension of the space of parameters defining distribution $\mathrm{P}(\mathbf{Q}|\mathbf{E})$. Note, also, that $\mathbf{E} = \mathbf{E}_d \cup \mathbf{E}_c$.

The process of representing probability queries as functions is usually referred to as *compilation* of Bayesian networks (Darwiche, 2009). The proposal presented in this work relies on the use of feedforward neural networks (Aggarwal, 2018) for that. Our strategy consists in the design and training of a different neural network for approximating each possible query related to the sensory system. This would significantly reduce the on-line computational cost, since virtually all of the work performed by our algorithm would be transferred to the off-line training process and only function evaluation effort would be needed for on-line inference. This generally leads to real-time performance, as we show in our validation experiments in Section 5.6.

The training process could also be performed by using a monolithic model of the system. Unfortunately, inference in potentially large monolithic networks can be extremely inefficient in general, and possibly impractical, which would hinder or prevent the obtention of the necessary training data. The other problem, related to the training process itself, relies on the fact that the cardinality of the evidence set may be arbitrarily high. Such a large set would lead to a neural network with a large amount of inputs, which could be, in turn, really difficult to train.

The approach for neural network design and training we propose in this work is based on our Bayesian architecture and its inference algorithm, which allow to alleviate the mentioned computational requirements. In our case, a query is always related to only one layer of the architecture, thus, it is not necessary to perform inference in the whole monolithic network. In order to obtain a training dataset, the subset of evidences corresponding to the layer of interest would be considered as input data, although this is not enough, since each layer includes an interface formed by a set of replicated ideal nodes $\mathbf{I}_r$ whose CPDs for priors $\mathrm{P}(I_r \subset \mathbf{I}_r)$ have to be included as well. Note, however, that this incorporation makes inference in lower layers unnecessary, which represents an advantage w.r.t the monolithic approach. On the other hand, output data will correspond to parameters of queries $\mathrm{P}(\mathbf{Q}|\mathbf{E})$. The training datasets for each neural network are obtained by sampling the space of possible values for the inputs and then calculating the corresponding outputs. This process represents a simulation of the different Bayesian networks defined in the architecture. If the number of inputs is high enough, sampled data should only contain values representing common situations. The rest will be generalized by the neural network itself.

One aspect that has to be considered before training is the definition of each neural network in terms of number of hidden layers and neurons as well as the shape of the activation functions employed. There are no strict rules to choose an appropriate number of neurons for each hidden layer or the number of them, however, the output activation function must fit the shape of the output data, which represent parameters of some pdf or pmf, as mentioned before. Here, purely linear activation functions are used for outputs referring to queries involving continuous variables; in the discrete case, a logistic sigmoid function is employed if the query variable has a discrete binary support (since its image is defined between zero and one) and a softmax function is used if the support is non-binary (since its shape is the one that best fits the form of a pmf). Regarding hidden layers, the chosen functions are always hyperbolic tangents. Fig. 4 shows the shapes of the mentioned activation functions, and the neural network design is formalized in Algorithm 4.

The last aspect to be taken into account for the training process is the choice of an appropriate learning algorithm and a hardware platform to run it (here, this refers either to a CPU or to a GPU). Again, there is no general rule for such a decision, thus, it can be made based on the experience acquired throughout different trials. The most suitable algorithms for this case are the well-known *Levenberg–Marquardt* (LM) (Marquardt, 1963) and the *Scaled Conjugate Gradient* (SCG) (Møller, 1993). The method proposed in this work for obtaining a set of neural networks representing all possible queries in a sensory system is formalized in algorithm 5, which also relies on algorithm 4.

Finally, note that our hybridization of BNs with NNs should be considered as a whole system where both parts cannot be separated while maintaining the same performance: on the one hand, compilation leverages the structure of the Bayesian architecture in order to be feasible (compilation of an equivalent monolithic BN would not be practical for the cost of the inferences). On the other hand, NNs cannot be trained for the same problem alone while preserving the rigorous probabilistic foundations of Bayesian inference: NNs, per se, perform induction,

Algorithm 4 *designNeuralNetwork*(n_{in}, n_{out}, Q)

input:
n_{in}: number of inputs
n_{out}: number of outputs
Q: query variable

output:
N: untrained neural network for performing queries over variable Q

main:
N ← empty neural network with n_{in} inputs and n_{out} outputs
if Q is discrete and binary **then**
  N ← add a logistic sigmoid function as output activation for N
else if Q is discrete and non-binary **then**
  $\lambda \leftarrow |Q| - 1$ (cardinality of the support of Q minus one)
  N ← add a λ-dimensional softmax function as output activation for N
else if Q is continuous **then**
  N ← add a purely linear function as output activation for N
end if
N ← add an appropriate number of hidden layers and neurons for N
  (always use hyperbolic tangent functions for that)
return N

while BNs also do other kinds of inference, e.g., abduction (when, for instance, they estimate the most likely hypothesis that explains certain anomalies/evidences), which is not easy to be provided by NNs without the underlying support of the BN formalism. Also considering the explainability capabilities of BNs and the faster responsiveness of NNs to anomalies in real-time, the complete architecture (BNs + NNs) can be more powerful, safe and robust than addressing the same problem with NNs or BNs only.

5. Implementation and validation for robot navigation in environments with human presence

In this section we implement and validate our Bayesian sensory architecture and the corresponding real-time inference methodology for the problem of mobile robot navigation in environments with human presence. Section 5.1 provides a description of the key aspects of that problem regarding sensory anomaly detection and recovery. Then, Section 5.2 describes the experimental setup we have used for all our tests. Section 5.3 covers the instantiation of our architecture for this particular case and Section 5.4 its evaluation in simulated experiments. After that, Section 5.5 describes the experimental validation tests we have carried out in a real environment with a mobile robot. Finally, Section 5.6 covers the analysis of the performance of the proposed methodology, in terms of error and computation time.

5.1. Problem overview

Navigation in environments with dynamic obstacles constitutes a key part of countless applications related to service robotics, such as industrial, medical and domestic, among many others (IFR, 2022). Safety represents a major concern in these applications, since it is crucial to preserve the integrity of the agents (humans and robots) involved The challenges that represent a safe robotic operation have been extensively addressed from the perspective of navigation (Ferrera et al., 2017; Fan et al., 2020), however, the presence of human agents, usually referred to as pedestrians in this scope, poses some problems related to their detection and tracking that must be solved prior to the integration of mobile service robots in the mentioned applications.

One of the most common concerns is the problem of *occlusion* (Brunetti et al., 2018), which prevents or hinders pedestrian detection in presence of static and/or dynamic obstacles. This has been thoroughly studied for a wide variety of situations, e.g., for both indoor and outdoor settings, known and unknown scenarios, etc. The problem

Algorithm 5 Sensory architecture compilation into feedforward neural networks

input:
$\mathcal{S}$: sensory system (see algorithm 1)

output:
$\mathbb{I}_{LS}$: set of neural networks for ideal node queries, indexed by layer and sensor
$\mathbb{A}_{LSN}$: set of neural networks for anomaly node queries, indexed by layer, sensor and node

subroutines:
getTrainingDataset($\mathcal{B}, \mathbf{R}, \mathbf{I}_r$):
  $N \leftarrow$ choose sample size
  $\mathfrak{D}_{\text{IN}} \leftarrow \infty$, $\mathfrak{D}_{\text{OUT}} \leftarrow \infty$
  $\mathfrak{D}_{\text{IN}} \leftarrow$ generate N samples from the space of possible instantiations of the set $\mathbf{R} \cup \mathbf{I}_r$
  for each instantiation $\delta \in \mathfrak{D}_{\text{IN}}$ **do**
    ($\mathcal{B}$, **e**) ← re-parameterize sensor $\mathcal{B}$ and generate evidence **e** by using data from δ
    $\mathcal{P} \leftarrow$ *joinTree*($\mathcal{B}$, **e**) (see algorithm 2)
    $\mathfrak{D}_{\text{OUT}} \leftarrow \mathfrak{D}_{\text{OUT}} \cup \mathcal{P}$
  end for
  return $\mathfrak{D}_{\text{IN}}$ and $\mathfrak{D}_{\text{OUT}}$

trainNeuralNetwork(N, $\mathfrak{D}_{\text{IN}}$, $\mathfrak{D}_{\text{OUT}}$, Q):
  $\mathfrak{D}_{\text{OUT}} \leftarrow$ filter $\mathfrak{D}_{\text{OUT}}$ so that it only contains posteriors involving variable Q
  A ← choose a training algorithm, either LM or SCG (see text)
  P ← choose hardware platform for training, either CPU or GPU (see text)
  N ← train neural network N with algorithm A on platform P
    using $\mathfrak{D}_{\text{IN}}$ as input data and $\mathfrak{D}_{\text{OUT}}$ as output data
  return N

main:
$\mathbb{I}_{LS} \leftarrow \infty$, $\mathbb{A}_{LSN} \leftarrow \infty$
for each layer $\mathbf{L}_i$ in sensory system $\mathcal{S}$ **do**
  for each Bayesian sensor $\mathcal{B}_j \in \mathbf{L}_i$ **do**
    $r \leftarrow |\mathbf{R}|$ (cardinality of the set of real sensor variables)
    $c \leftarrow$ number of CPD parameters for the set of *replicated* ideal nodes $\mathbf{I}_r$
    $p \leftarrow$ number of CPD parameters for the ideal node variable I
    ($\mathfrak{D}_{\text{IN}}$, $\mathfrak{D}_{\text{OUT}}$) ← *getTrainingDataset*($\mathcal{B}_j$, $\mathbf{R}$, $\mathbf{I}_r$)
    $\mathbb{I}_{LS}(i,j) \leftarrow$ *designNeuralNetwork*($r+c$, p, I)
    $\mathbb{I}_{LS}(i,j) \leftarrow$ *trainNeuralNetwork*($\mathbb{I}_{LS}(i,j)$, $\mathfrak{D}_{\text{IN}}$, $\mathfrak{D}_{\text{OUT}}$, I)
    for each anomaly node A_k in Bayesian sensor $\mathcal{B}_j$ **do**
      $q \leftarrow$ number of CPD parameters for the anomaly node variable A_k
      $\mathbb{A}_{LSN}(i,j,k) \leftarrow$ *designNeuralNetwork*($r+c$, q, A_k)
      $\mathbb{A}_{LSN}(i,j,k) \leftarrow$ *trainNeuralNetwork*($\mathbb{A}_{LSN}(i,j,k)$, $\mathfrak{D}_{\text{IN}}$, $\mathfrak{D}_{\text{OUT}}$, A_k)
    end for
  end for
end for
return $\mathbb{I}_{LS}$ and $\mathbb{A}_{LSN}$

of occlusion is even more difficult to address when it persists over time, i.e., when it becomes a *long-term* occlusion problem (Zhang et al., 2020; Islam et al., 2018). Fortunately, the multidimensional Bayesian sensory architecture proposed in this work can be used to tackle this as well as other issues related to the identity of the pedestrians being tracked. For that, it can rely on fusing the sensory information obtained from a state-of-the-art pedestrian detection and tracking system with other sources of knowledge, as we will explain later on. Recall that all the features of the architecture need to be exploited in this case, since the mentioned sensory information is related to different levels of cognitive abstraction and also evolves over time.

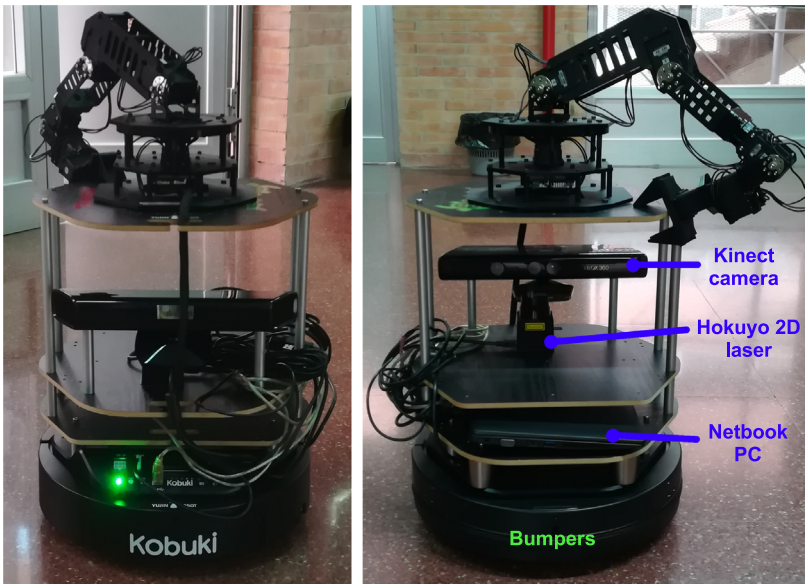


Fig. 5. The CRUMB mobile robot, used in this work, with its sensory apparatus. (a) Rear view. (b) Front view.

5.2. Experimental setup

All the validation tasks and experiments described in this work have been conceived for a concrete mobile platform, called CRUMB (Fernández-Madrigal and Cruz-Martín, 2020). This robot is based on a version of the *Turtlebot-2* that uses a two-wheeled *Kobuki* platform (Open Source Robotics Foundation, 2020), which is equipped with a bunch of basic sensors such as bumpers, encoders, cliff and wheel drop detectors and a tree-axis gyroscope. This sensory apparatus has been complemented with range sensors relying on infrared radiation, namely, a *Hokuyo URG-04-LX* 2-D laser (Hokuyo, 2020) and a *Kinect V1* RGB-D camera (Rodríguez, 2021; Adhikary et al., 2019). The robot is controlled via an on-board netbook PC with an Intel Celeron N2840 at 2.16 GHz and 2 GB DDR3 that runs Ubuntu 14.04 with ROS (Newman, 2017). The robot is shown in Fig. 5.

Concerning the use of this platform for the problem of navigation in human environments, the sensory system considered is more abstract. Instead of using low-level information directly, it is pre-processed in this case for people detection and tracking. More specifically, range data is employed by an external software, which is based on a ROS package publicly available, developed in the context of the EU FP7 research project known as *Spencer* (SPENCER, 2016) (from now on, this software will be referred to as the *Spencer system*). This package includes, among other features, a re-implementation of Arras et al. (2007) based on an Ada-Boost for people tracking with 2D laser range data and a Nearest-Neighbor Standard Filter (NNSF) for taking the data association decisions required for people tracking. These techniques exhibit a good performance, and do not require a significant computational cost. The Spencer system is suitable for our task due to its modularity, its efficiency and its flexibility; it has been tested previously in both real and simulated robots, it is multimodal, it is very complete, and it is publicly available in Github (SPENCER, 2023). For a comprehensive guide about this package, please refer to Linder and Arras (2016).

The real experiments for navigation in human environments have been carried out with the CRUMB robot in a scenario with a pedestrian (see Section 5.5), which is sufficient to prove the advantages of our solution, i.e., no crowded environment application is intended. For all these experiments, the proposed Bayesian sensory architecture has been implemented in MATLAB by using the *Bayes Net Toolbox* (BNT) (Afrassa et al., 2019). The robot navigates autonomously in the mentioned environment thanks to the use of the Model Predictive Control (MPC) approach from (Castillo-Lopez et al., 2020), which incorporates a unicycle model of the robot to generate collision-free trajectories with safety guarantees. In the experiments, the MPC controller, the Spencer system and the sensory architecture are coordinated within the ROS framework, which is run on board the robot and also on an external, remote PC with an Intel i7-9700K at 3.6 GHz and 32 GB DDR4. Such external hardware is employed to avoid the execution of several nodes with potentially high computational requirements (e.g., the ones related to the MPC controller or the sensory architecture) in a single machine, which could reduce the system performance in certain situations.

Regarding the simulations, a similar setup has been employed. In this case, the CRUMB robot is integrated as part of a simulated environment based on Gazebo (Aguilar-Moreno et al., 2018) (see Fig. 6(a)). The rest of the software used for the experiments is the same as in the real setup; however, an extra module is added for the simulation of human presence and motion. This is also a ROS package, known as *Pedestrian Simulator* or simply *Pedsim*, which enables 2D pedestrian simulation and visualization in real time (see Fig. 6). This package is based on the social model force of Helbing and Molnár (1995), and it was also developed during the Spencer project (SPENCER, 2016). Only the external PC mentioned before has been used as the hardware for the simulated experiments.

5.3. Definition of the Bayesian architecture

The instantiation of the Bayesian architecture for the problem of navigation in human environments is carried out by following the procedure detailed in Section 3.2. As explained before, all the features offered by the Bayesian sensory model will be necessary in this case, since there is sensory information evolving over time and also belonging to different levels of cognitive abstraction. A list with the necessary Bayesian sensors along with the knowledge they rely on is compiled in Table 3.

The Bayesian sensory architecture is built upon the nine Bayesian sensors listed in Table 3; in this particular case there are no cyclic dependencies among them. Taking into account the existing relations among these sensors, a dependency graph as the one shown in Fig. 7 can be defined. Note that the model is instantiated identically for

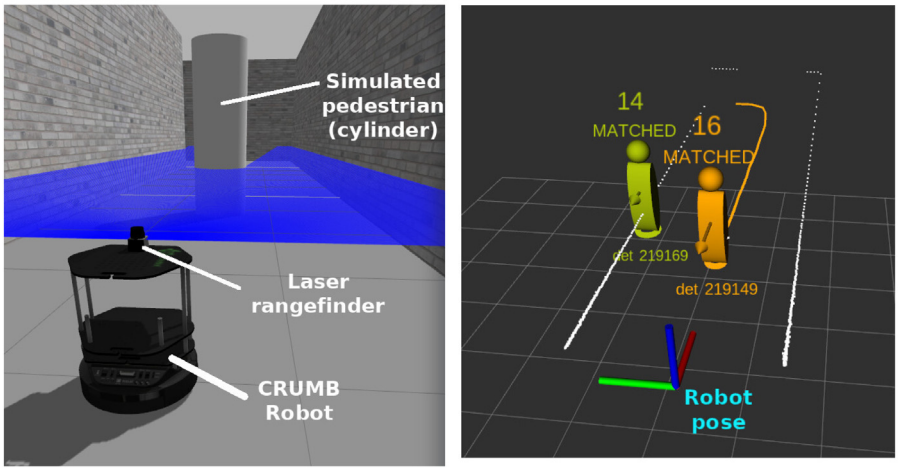


Fig. 6. Simulated environment for the problem of navigation in human environments. (a) View of the environment in Gazebo, with the CRUMB robot and a simulated pedestrian controlled by *Pedsim*. The blue area corresponds to the field of view of the laser rangefinder. (b) Three-dimensional view of the pedestrians detected by the *Spencer* system. The white lines correspond to the walls of the scene detected by the laser rangefinder. The pedestrian on the right corresponds to the one in Fig. 6(a), while the pedestrian on the left is fictitious (see Section 5.4).

Table 3
Bayesian sensors used in this model and their corresponding sources of data.

Bayesian sensor	Sensory information source(s)
Pose and velocity predictor	Pose and speed of pedestrians (Spencer system)
Age sensor	Age of the detected pedestrian (external computer vision system)
Ghost pedestrian sensor	Map of the scene and localization of the pedestrian
Situation sensor	Social knowledge, pose and age of the detected pedestrian
Pose and velocity estimator	Pose and speed of pedestrians (Spencer system)
Long-term occlusion sensor	Geometry of occlusion zones and their persistence over time
Distance sensor	Pose of several pedestrians of interest
Difference of orientation sensor	(idem)
Identity sensor	Identity of the detected pedestrian (Spencer system)

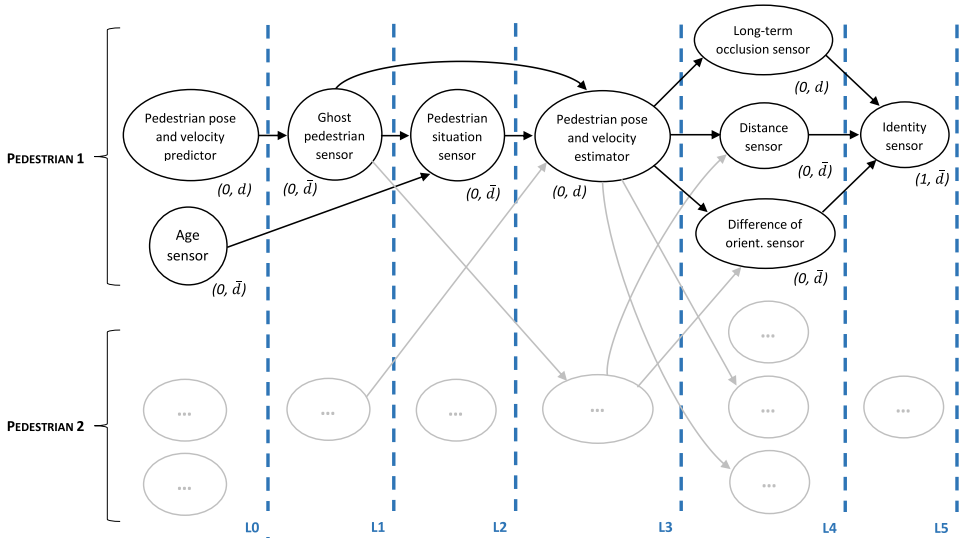


Fig. 7. Annotated dependency graph for the proposed Bayesian architecture for each detected pedestrian, with layer assignment. Recall that the annotations in the graph indicate the level of cognitive abstraction and the dynamic character of each sensor, respectively. The interactions existing between different pedestrians are also shown.

each detected pedestrian in the system, and that these instances exchange information with each other. Fig. 7 represents these connections only between two different pedestrians, since they can be applied analogously to every pair of detected pedestrians.

In the following, we only cover the definition of one of the Bayesian sensors in the architecture, for being the most relevant one for anomaly recovery in this context and for the sake of brevity. Please refer to Appendix B for a description of the remaining sensors.

In the proposed sensory architecture, layer three (Fig. 8) represents one of the most complex networks. It contains a sensor that serves to recover a useful estimation of the pose and velocity of a pedestrian in case of severe occlusion. Also, this network can be used to improve

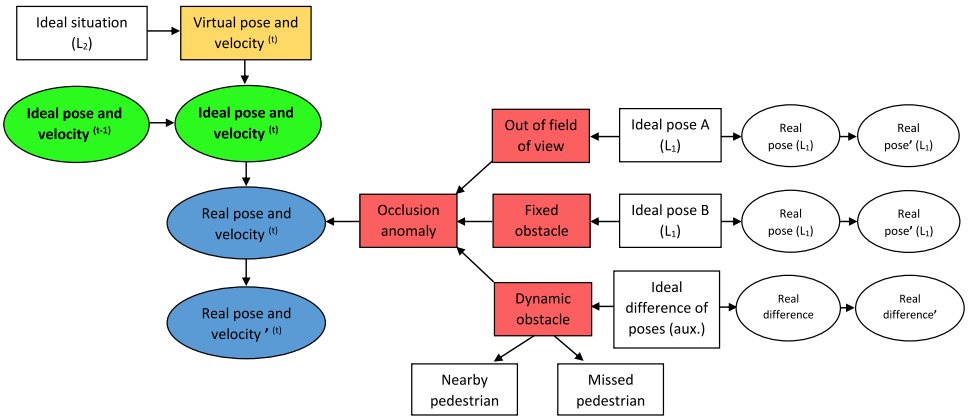


Fig. 8. Bayesian network corresponding to the pedestrian pose and velocity estimator (layer three). Here, replicated ideal nodes and auxiliary Bayesian sensors are represented in white.

the estimation of these state variables if there are no occlusions. In this context, the state of the pedestrian is represented by using six variables, namely, the three describing the two-dimensional pose of a pedestrian (i.e., x, y, θ) and their derivatives (i.e., v_x, v_y, ω, respectively), defined with respect to a global, fixed frame.

The anomaly subnetwork in this case is devoted to the detection of occlusion events, encoding three different causes for that: occlusion when the pedestrian gets out of the field of view of the range sensors, occlusion produced by a fixed obstacle (e.g., the scene itself) and occlusion provoked by other pedestrians (i.e., by dynamic obstacles). For the first two situations, the Bayesian sensor relies on two different discretizations of the pose of the considered pedestrian, obtained from layer one, along with environmental knowledge. Each one of these discretizations serves to determine whether the pedestrian is situated within one of these occlusion zones, defined either by the field of view of the range sensors or by the scene itself, respectively. For the case of occlusion provoked by dynamic obstacles, the sensor incorporates an auxiliary one representing the difference between the estimated pose of two different pedestrians, being one of them the pedestrian of interest, and the other one, a pedestrian chosen for being the most likely to be occluding the former. This difference is also discretized into different zones in order to define in which ones would be the pedestrian occluded. This last part of the anomaly subnetwork also includes two extra discrete variables representing flags provided by the Spencer system, which are useful to determine the occlusion situation.

The rest of the network is dedicated to the recovery of the pose and velocity of the considered pedestrian, as mentioned above. For that, it incorporates the information from the situation given by the corresponding sensor in layer two. This information serves to use an adequate motion model depending on the age of the pedestrian and the location in the scene. All these models are encoded in the CPD corresponding to the ideal node of the current time interval t. Each one corresponds to the mean of a multivariate Gaussian distribution, which is a constant velocity model of the form:

$$\begin{aligned} x_t &= x_{t-1} + v_{x_{t-1}}\,\Delta t \\ y_t &= y_{t-1} + v_{y_{t-1}}\,\Delta t \\ \theta_t &= \theta_{t-1} + \omega_{t-1}\,\Delta t \\ v_{x_t} &= V\cos(\theta_{t-1}) \\ v_{y_t} &= V\sin(\theta_{t-1}) \\ \omega_t &= \Omega, \end{aligned} \tag{12}$$

where V is a constant linear speed, Ω is a constant angular speed and Δt is the elapsed time between intervals t and $t-1$. Thus, each one of the mentioned models only differs from the others in its constants V and Ω, leading to a different motion depending on the situation of the pedestrian. The complete CPD is then a list of Gaussian distributions with different means depending on the value of the situation sensor (see Fig. 8). The covariance matrices for the Gaussian distributions in the CPD are all diagonal, only adding uncertainty depending on the amount of change of the corresponding variable over time. Recall, again, that all these state variables (i.e., pose and velocities) are defined over a global, fixed reference frame. Note, also, that the constant velocity model for pedestrians defined in Eq. (12) represents a nonlinear function of the previous state. In order to encode this in a CLG Bayesian network, the CPDs are transformed by using first-order Taylor series linearization, which involves the intermediate calculation of a Jacobian matrix of the model (please refer to Koller and Friedman, 2009 and Miklavcic, 2020 for further details).

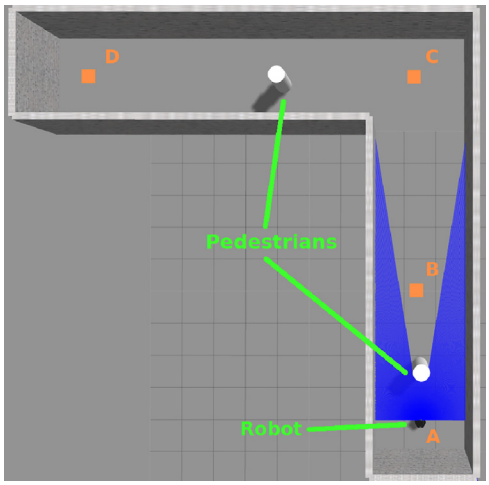


Fig. 9. Top view of the controlled scenario used for the simulated experiments, with some points highlighted. The blue area corresponds to the field of view of the laser rangefinder.

5.4. Simulated experiments

The proposed Bayesian architecture model for the problem of navigation in human environments has been tested in several simulations. As reported in Section 5.2, the experiments described in this subsection have been carried out in a simulated environment based on Gazebo. This environment includes an scenario where the CRUMB robot navigates surrounded by two pedestrians (see Fig. 9). For the simulated tests, three kinds of experiments have been carried out. In all of them, the pedestrians incessantly follow a cyclical path between points A and D (see Fig. 9), and they are not aware of the presence of the robot. The pedestrian simulator (i.e., *Pedsim*) allows the emulation of age

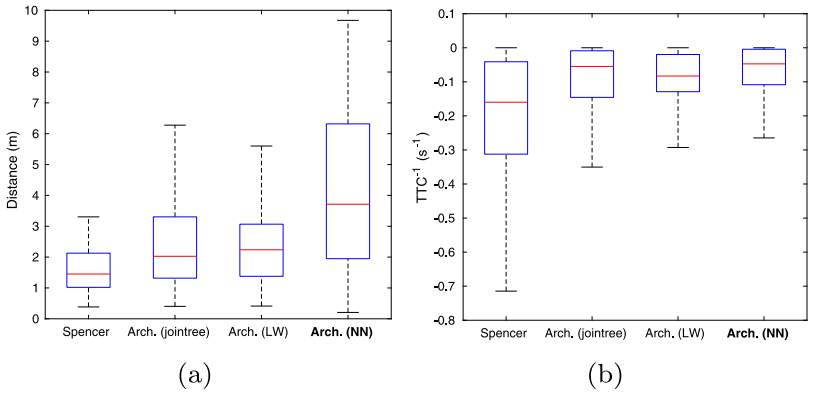


Fig. 10. Comparative boxplots for the measures of safety considered in the first simulated experiment, where the robot tries to maintain a given pose while avoiding nearby pedestrians. (a) Distance to the closest pedestrian. (b) Inverse time to collision.

by setting one of its parameters, which has been modified in order that each pedestrian moves at a different pace (two different ages are considered, namely, *young* and *elder*).

Each experiment or set of experiments is performed in the same conditions but with different approaches in order to obtain the most complete comparative that it is possible considering that no general benchmark exists in the literature for carrying out FDD in this particular robotic task —also notice that existing datasets are not suitable for testing closed-loop applications like ours—. The baseline in all experiments has been the previously referred method of Linder and Arras (2016) implemented in the SPENCER stack, which pursues robustness and efficiency in people detection and tracking with ad-hoc algorithms. In subsequent figures, we compare the results obtained by the baseline ('Spencer') and several variants/ablations of our solution: 'Arch (LW)' is our BN architecture with a Monte Carlo approximate inference method based on importance sampling, called *likelihood weighting* (Koller and Friedman, 2009), and no NN compilation; 'Arch (jointree)' refers to our BN architecture using the exact jointree inference algorithm described in Section 4.1 and also without the NN compilation; 'Arch (NN)' is the complete hybrid system —with the jointree algorithm— once compiled to NNs.

The first kind of experiment aims to prove that the use of the proposed Bayesian architecture serves to improve safety during navigation, even under adverse conditions. In the experiment, the robot tries to maintain its pose around point B while avoiding approaching pedestrians. The safety of navigation has been assessed in this case through two measures: the distance to the closest pedestrian d, and the inverse time to collision $TTC^{-1} = \dot{d}/d$, which are commonly used in the literature related to robotic navigation (Castillo-Lopez et al., 2020). In general, large and negative values of TTC^{-1} indicate high risk of collision, while values close to zero correspond to safe navigation.

The results of 5 min of simulation are shown in Fig. 10, where only the data concerning the moments when anomalies occur are compared. The most common anomaly in this case is the occlusion produced when the pedestrian leaves the horizontal field of view of the robot, which is of 180 degrees. The proposed Bayesian architecture and its variants manage to recover an estimated pose of the missed pedestrians better than Spencer, with particularly improved safety in the case of our complete BN+NN solution due to the fastest response times that compiled NNs provide (see Fig. 11). In the case of comparing the baseline with the complete BN+NN architecture, we find that the median distance to the closest pedestrian rises from 1.45 to 3.71 m (an improvement of 156%), and the median TTC^{-1} from −0.16 to −0.047 s^{-1} (71%). Also, the robot collides with the pedestrians a total of 7 times during the experiment carried out only with the Spencer system; when the proposed architecture is integrated, the test is collision free. Note that the other variants of our architecture (non-compiled and Monte Carlo inference algorithm) are not significantly different among them but also show improvements w.r.t. the baseline.

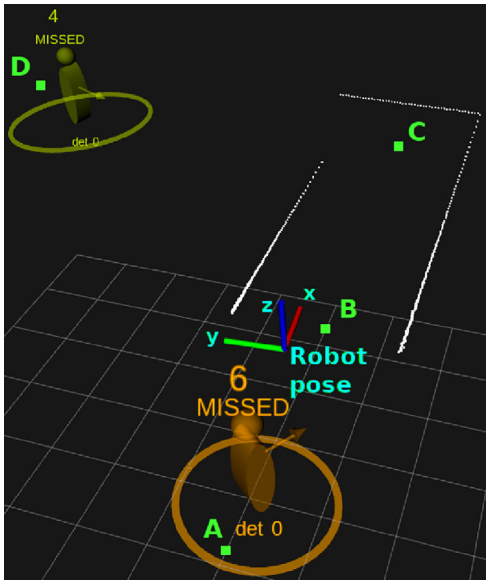


Fig. 11. Three dimensional view of the pedestrian tracking system using the Bayesian sensory architecture during the first experiment. The robot is pointing towards the positive sense of the X axis. Pedestrian 6 is recovered despite being placed behind the robot, and pedestrian 4 despite being occluded by the walls of the scene, detected by the 2D laser (white lines).

The second simulated experiment illustrates the effect of false positive detections, i.e., detections of nonexistent —*ghost*— pedestrians on the efficiency of navigation. The measure chosen to quantify such efficiency is the time that the robot takes to go from point A to C. The test has been prepared so that none of the pedestrians are nearby the robot along the trajectory, and the experiments have been repeated 20 times for each approach. When the baseline system is used, some nonexistent pedestrians appear, leading to avoidance maneuvering that is not actually needed (see Fig. 12). As a result, the total navigation time increases.

The impact of navigating around fictitious pedestrians has been also quantified through the linear correlation between the time that the robot is nearby them and the total navigation time. The same linear correlation has been calculated as well for the number of anomalous detections. The coefficient of determination R^2 is of 0.9234 for the former and of 0.5385 for the latter, thus, it is clear that the total time needed to complete the trajectory is strongly correlated with the amount of time the robot remains nearby ghost pedestrians, rather than with the number of them.

The results for this experiment are shown in Fig. 13 (the second linear fit is omitted for being of little relevance). In this case, the

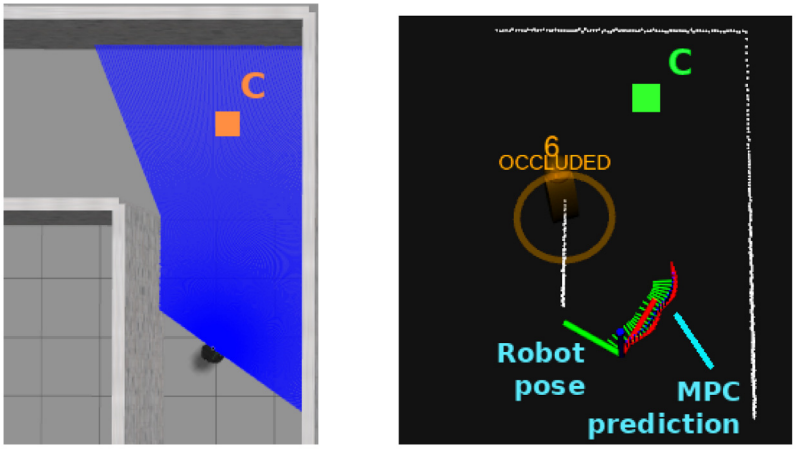


Fig. 12. Snapshot of the simulated environment during an avoidance maneuvering due to the presence of a so-called *ghost* pedestrian (second experiment). (a) Top view of the simulation in Gazebo. (b) Top view of the pedestrian tracking system when it is using the Spencer baseline approach. Here, the MPC controller predicts a maneuvering action to be performed in order to avoid the fictitious pedestrian.

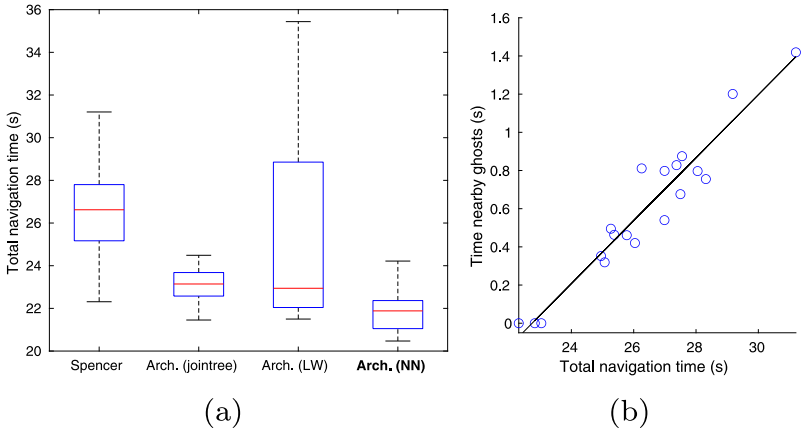


Fig. 13. Results for the second simulated experiment, when the robot has to navigate to a target even when false pedestrians are detected. (a) Comparative boxplots for the total navigation time. (b) Linear regression of the time nearby *ghost* pedestrians recorded by the Spencer baseline versus the total navigation time.

application of the full BN+NN architecture manages to reduce the median navigation time from 26.62 to 21.88 s (18%). Again, this is the best mark, mainly due to the shorter reaction times allowed by the use of the NNs, but the non-compiled and the LW alternative inference algorithm also improve the results of the baseline.

The third and last simulated experiments aim to assess the utility of the proposed Bayesian architecture in the case of identity anomalies. For that, the robot tracks the pose of one of the pedestrians throughout the scene until the identity is missed or confused with another one. In this case, the total tracking time is measured, and eight different configurations are used for the tests by combining the age of the tracked pedestrian (young or elder) with the approach implemented to navigate (baseline, LW, jointree, BN+NN). The experiment has been repeated a total of 10 times for each configuration. The tracking time results are depicted in Fig. 14. As it is shown, the full BN+NN approach increases the median tracking time from 20.68 to 125.14 s for the young pedestrian (505%) and from 42.46 to 136.19 s for the elder one (221%), which are greatly significant results w.r.t. the baseline, demonstrating that for tracking correctly a pedestrian with a Bayesian inference architecture the reaction time of the system is crucial; the non-compiled jointree and LW inference algorithms also improve tracking time significantly, but they are not able to reach the same marks due to their slower processes.

To reinforce this idea, it is also interesting to analyze the kind and amount of adverse events being overcome during these tests. Table 4 shows, for the cases of using our BN core architecture (jointree, non-compiled) and each kind of pedestrian, the mean percentage of time dedicated to recover from those situations, which are the ones behind the majority of identity problems. Recall that a pedestrian can be occluded when leaving the field of view of the robot or due to the presence of either an static or dynamic obstacle.

The obtained results show that the amount of abnormal situations recovered during the tests using the proposed Bayesian architecture is double in general. In other words, this means that the proposed approach is still robust despite a greater amount of anomalies occurring during a much longer period of time. In particular, this is true for the occlusion anomalies related to the field of view of the robot and the static obstacles, which tend to last longer in the case the architecture is employed. In contrast, the Spencer baseline recovers reasonably well

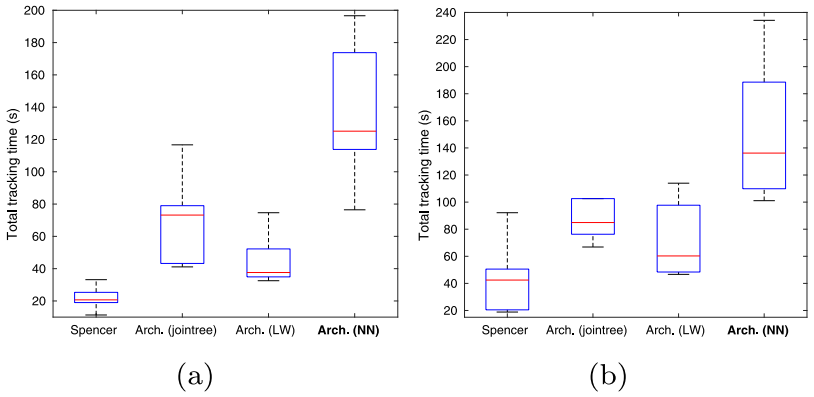


Fig. 14. Comparative boxplots for the total tracking time of pedestrians in the third simulated experiment, where the robot has to track a given pedestrian. (a) Results for a young pedestrian. (b) Results for an elder pedestrian.

Table 4
Average percentage of time for recovering occlusion anomalies.

Pedestrian	Configuration	Occlusion anomaly			Total
		FOV	Static	Dynamic	
Young	BN	10.98	11.50	6.65	**29.13**
	Baseline	2.42	5.38	6.63	14.43
Elder	BN	4.58	0.66	1.23	**6.47**
	Baseline	1.27	0.00	2.00	3.27

from the brief occlusions produced by dynamic obstacles (i.e., by other pedestrians).

5.5. Real experiments

The real experiments presented in this subsection have been conducted in situations similar to the ones analyzed in the simulations. They are intended both to demonstrate the strong correlation between simulated results and the possible real achievements of the proposed solution and to show its potential when implemented in a real robot. Recall that these experiments are not developed in crowded environments, since their aim is to properly assess the capabilities of the proposed approach.

In this case, two different experiments have been performed, both in the real scenario shown in Fig. 15. As commented in Section 5.2, the software and hardware employed is the same as the one used in the simulations, with the exception of the netbook available on board the CRUMB robot. Each test in these real experiments is repeated with and without incorporating the Bayesian architecture to the Spencer system, and the obtained results are compared, as in the simulated case.

The first experiment aims to validate the simulated results related to the efficiency of navigation in presence of ghost, non-existent pedestrians. For that, the robot is ordered to go from point A to C (see Fig. 15) in the absence of actual people. During this trajectory, some fictitious pedestrians are detected by the Spencer system, as in the simulated case. The impact of navigating around such false detections of human presence is assessed by measuring the time that the robot is placed nearby ghost pedestrians and the total navigation time. The obtained results are compiled in Table 5. As expected, they prove that the incorporation of the Bayesian architecture manages to reduce the navigation time, since it is conceived to notice false detections of pedestrians, allowing the robot to ignore them. The results also show the correlation between the total navigation time and the one that the robot is influenced by the presence of these pedestrians. In other words, they prove that, the longer the robot is situated around fictitious pedestrians, the more likely it is that the navigation time increases.

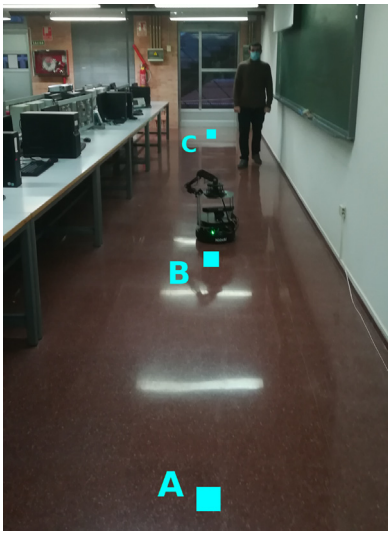


Fig. 15. Image of the setup used for the real experiments, with some points highlighted. The corridor shown is 13 m long. Recall that the pedestrian appearing here is only present during the second experiment.

Table 5
Comparative results for the real experiments (see the text).

Experiment	Measure	Baseline	BN Arch.	Improv.
First	Total navigation time	29.1 s	26 s	10.65%
	Time nearby ghosts	1.41 s	0.18 s	87.23%
Second	Number of collisions	4	0	–
	Time with occlusions	53.88 s	89.12 s	–

When this abnormal situation takes place in the real setting, it can be observed how the robot performs unnecesary avoidance maneuvering in the case that the Bayesian sensory architecture is not used.

The second and last experiment is intended to validate the simulated results concerning the safety of navigation. In this test, the robot tries to maintain its pose around point B while avoiding a pedestrian that follows a cyclical path between points A and C during 3 min. Here, safety is assessed by counting the times the robot collides with the pedestrian. Also, the total time of the test with occlusion anomalies is measured. The obtained results (Table 5) prove that the use of the Bayesian architecture increases safety, since the test is collision free when the proposed approach is incorporated. As in the simulated case, the period of time dedicated to recover from anomalous situations is much longer when the architecture is used, since it is prepared to

(a) (b)

Fig. 16. Snapshot of an instant recorded during the second experiment. Here, the pedestrian is walking from point A to C while being situated out of the field of view of the robot. (a) The robot manages to avoid the pedestrian thanks to the use of the Bayesian sensory architecture. (b) The pedestrian collides with the robot, which is only relying on the detections provided by the Spencer system.

handle long-term occlusions. This demonstrates, again, the robustness of the proposed approach, which enables to improve safety even under lasting abnormal conditions. In fact, it is observed in the real setting that only the use of the Bayesian architecture enables the robot not to collide with the pedestrian when he/she is out of its field of view (see Fig. 16).

5.6. Analysis of the performance of the proposed methodology

Due to the importance of compiling our BNs into NNs for the reaction times and therefore for the improvements achievable by the system, as it has been shown in the previous section, in this one we analyze in more detail the performance aspects of our architecture. We first describe the details of the process of compilation of our Bayesian sensory architecture as neural networks for the problem of navigation in environments with human presence. Then, we discuss the results these models achieve regarding both computation time and error w.r.t inference on Bayesian networks that have not been compiled.

The process of neural network design, training and evaluation has also been implemented in MATLAB, relying again on the BNT library for Bayesian networks. Regarding hardware, the platform employed for training is the same desktop PC used for the experiments carried out in this work, which has an Intel i7-9700K CPU. Also, those networks with higher computational requirements have been trained with an NVIDIA Tesla V-100 GPU, which is part of an NVIDIA DGX Station. The queries considered essential for the problem of navigation in human environments (and thus, the ones considered here) are listed in Table 6. The configuration of all the neural network models designed to approximate such queries is compiled in Table 7.

Note that some queries are represented by more than one neural network (see Table 7). This is the case for layer 0 (corresponding to the pose and velocity prediction), layer 2 (dedicated to the situation sensor) and layer 3e (conceived for the improved estimation of pose and velocity). The main reason for using several networks is to reduce the computational requirements for the training process. For instance, the two networks designed for layer 0 represent the mean and covariance of the prediction, respectively. In the case of layer 2, three equal networks have been defined, two of them for a different age of the detected pedestrian (i.e., *young* and *elder*), and the last one for the case in which the age is not known. Here, only these three possibilities are considered, for the sake of simplicity. Finally, layer 3e is represented by four different networks. The first one refers to the estimation mean, and the other three, to some elements of the covariance. Such matrix has not been fully represented, since there are numerous values in it that do not vary much across the different tests performed for both the simulated and real experiments. These values have been ignored in order to reduce the network output size, and thus, the complexity of its training.

Table 6
Query variables employed for the problem of navigation in human environments. Each variable corresponds to a layer of the proposed Bayesian sensory architecture defined in Section 5.3 and Appendix B. Layer numbering has been completed with letters to denote different queries in the same Bayesian network.

Layer	Query variable (Q)
0	Ideal pose and velocity (current time interval)
1	Ghost anomaly
2a	Ideal situation (*young* pedestrian)
2b	Ideal situation (*elder* pedestrian)
2c	Ideal situation (unknown age)
3a	Out of field of view (anomaly node)
3b	Fixed obstacle (anomaly node)
3c	Dynamic obstacle (anomaly node)
3d	Occlusion anomaly
3e	Ideal pose and velocity (current time interval)
4a	Ideal long-term occlusion
4b	Ideal distance
4c	Ideal difference of orientation
5	Identity anomaly

As explained in Section 4.3, for each neural network corresponding to a query, the subset of evidences of the layer of interest are considered as input data —plus the CPDs for the priors of the replicated ideal nodes that serve as interfaces with lower layers—; output data corresponds to the parameters that define the distribution of the query. The training dataset for each neural network is then obtained by sampling the space of possible values for the inputs and then calculating the corresponding outputs through the inference algorithm in the Bayesian network of the corresponding layer (see Algorithm 5). Fig. 17 depicts the results of the training of the neural networks corresponding to the queries of layer three (see Fig. 8 and Table 6). Note that the training dataset is randomly divided by MATLAB for the process into three disjoint subsets, i.e., one sub-dataset for the training itself and other two for validation and test, respectively. The results of other queries in other layers have been omitted for the sake of space —they are quite similar to the ones presented here.

Once all the necessary neural networks have been defined and trained, it is possible to assess the performance of our new inference approach. For that, new validation datasets have been used, all of them obtained the same way as the ones for training. Then, two measures of performance have been calculated, namely, the error in inference w.r.t. to the algorithm proposed in this work and the computation time. The former has been obtained by using the Hellinger's distance (Pacini, 2022). For the case of probability distributions over discrete random

Table 7
Configuration parameters for all the feedforward neural networks designed, each one representing a query in Table 6. Again, layer numbering has been completed with letters to denote different queries for the same layer in the Bayesian network. Here, the number of hidden layers and neurons corresponding to each neural network is expressed in the form *A* x *B*, which means that the network has *A* layers with *B* neurons each. The training algorithm employed for each network is specified along with the hardware platform used (see text).

Layer	Neural networks designed			
	# Inputs	*# Hidden layers and neurons*	*# Outputs*	*Training algorithm*
0	7	1 x 21	6	LM (CPU)
	16	1 x 48	21	LM (CPU)
1	2	2 x 25	1	SCG (CPU)
2a	5	2 x 25	20	SCG (CPU)
2b	5	2 x 25	20	SCG (CPU)
2c	5	2 x 25	20	SCG (CPU)
3a	2	2 x 25	1	SCG (CPU)
3b	12	2 x 25	1	SCG (CPU)
3c	13	4 x 15	1	SCG (CPU)
3d	3	2 x 25	1	SCG (CPU)
3e	35	1 x 595	6	SCG (GPU)
	30	1 x 180	1	SCG (GPU)
	30	1 x 180	1	SCG (GPU)
	30	1 x 180	1	SCG (GPU)
4a	3	2 x 25	1	SCG (CPU)
4b	2	2 x 25	1	SCG (CPU)
4c	2	2 x 25	1	SCG (CPU)
5	3	2 x 25	1	SCG (CPU)

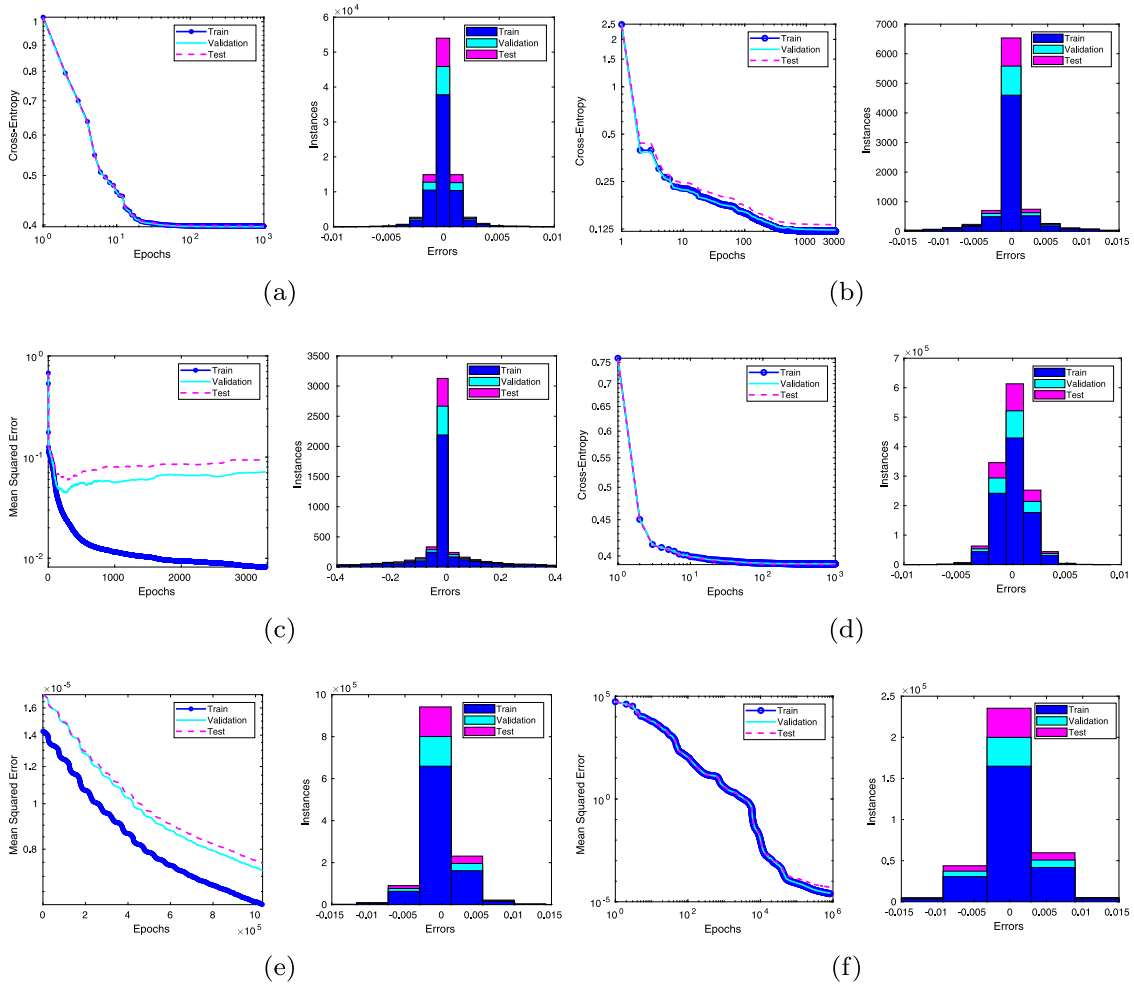


Fig. 17. Performance measures of the training of the neural networks associated to queries of layer three (see Fig. 8 and Table 6). The measured values are the evolution of either cross-entropy or mean squared error during training (left) and the errors w.r.t output targets after training (right), for all the datasets employed (train, validation and test). (a) Out of field of view anomaly, layer 3a. (b) Fixed obstacle anomaly, layer 3b. (c) Dynamic obstacle anomaly, layer 3c. (d) Occlusion anomaly, layer 3d. (e) Ideal pose and velocity, mean estimation (first network of layer 3e). (f) Ideal pose and velocity, variance estimation (second network of layer 3e).

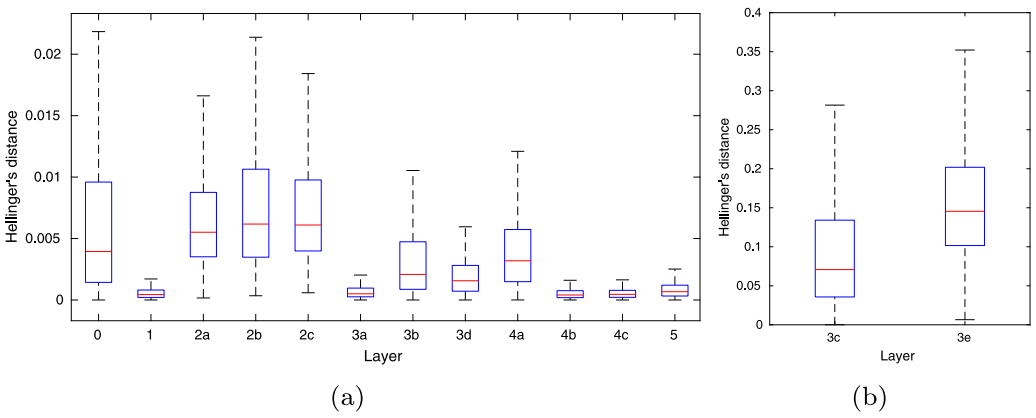


Fig. 18. Hellinger's distance for the distributions produced by the defined neural networks w.r.t the ones obtained by the approximate algorithm introduced in this work, calculated for each layer of the Bayesian architecture. These boxplots represent 9000 inferences per layer. (a) View of the lowest errors. (b) View of the highest errors.

Table 8
Single query time per layer achieved by inference with Bayesian networks and by inference with neural networks. The results are expressed in seconds, in the form $\mu \pm 2\sigma$, where μ and σ are the mean and standard deviation, respectively, of 9000 tests carried out per layer and inference method. The ratio is calculated for mean values only.

Layer	Single query time		
	BN inference	*NN inference*	*NN/BN ratio (means)*
0	$0.002 \pm 8.440 \cdot 10^{-4}$	$1.024 \cdot 10^{-4} \pm 1.930 \cdot 10^{-5}$	0.051
1	0.004 ± 0.001	$3.777 \cdot 10^{-5} \pm 2.421 \cdot 10^{-6}$	0.009
2a	0.030 ± 0.008	$3.549 \cdot 10^{-5} \pm 1.684 \cdot 10^{-6}$	0.001
2b	0.031 ± 0.009	$3.534 \cdot 10^{-5} \pm 1.628 \cdot 10^{-6}$	0.001
2c	0.030 ± 0.007	$3.580 \cdot 10^{-5} \pm 4.089 \cdot 10^{-6}$	0.001
3a	0.004 ± 0.002	$3.447 \cdot 10^{-5} \pm 1.831 \cdot 10^{-6}$	0.009
3b	$0.007 \pm 4.266 \cdot 10^{-4}$	$3.436 \cdot 10^{-5} \pm 1.447 \cdot 10^{-6}$	0.005
3c	0.015 ± 0.006	$3.548 \cdot 10^{-5} \pm 1.723 \cdot 10^{-6}$	0.002
3d	$9.762 \cdot 10^{-4} \pm 5.719 \cdot 10^{-4}$	$3.885 \cdot 10^{-5} \pm 3.506 \cdot 10^{-5}$	0.040
3e	0.013 ± 0.003	$5.379 \cdot 10^{-4} \pm 9.212 \cdot 10^{-4}$	0.041
4a	$9.252 \cdot 10^{-4} \pm 3.833 \cdot 10^{-4}$	$2.612 \cdot 10^{-5} \pm 1.902 \cdot 10^{-6}$	0.028
4b	$0.004 \pm 9.560 \cdot 10^{-4}$	$3.701 \cdot 10^{-5} \pm 1.006 \cdot 10^{-4}$	0.009
4c	0.004 ± 0.001	$3.856 \cdot 10^{-5} \pm 7.677 \cdot 10^{-6}$	0.009
5	$9.352 \cdot 10^{-4} \pm 3.447 \cdot 10^{-4}$	$3.466 \cdot 10^{-5} \pm 1.863 \cdot 10^{-6}$	0.037

variables, the Hellinger's distance H is defined as follows. Let $P_1(x_i)$ and $P_2(x_i)$ be two discrete probability distributions defined over the same support $x = \{x_1, x_2, \dots, x_n\}$; then,

$$H(P_1, P_2) = \sqrt{\frac{\sum_{i=1}^{n} \left(\sqrt{P_1(x_i)} - \sqrt{P_2(x_i)}\right)^2}{2}} \tag{13}$$

where $0 \leq H(P_1, P_2) \leq 1$. In our context, however, there are also some queries involving continuous variables, which will always be distributed normally. The Hellinger's distance between two multivariate normal distributions $P \sim \mathcal{N}(\mu_1, \Sigma_1)$ and $Q \sim \mathcal{N}(\mu_2, \Sigma_2)$ is (Pardo, 2006):

$$H(P, Q) = \sqrt{1 - \frac{\det(\Sigma_1)^{1/4} \det(\Sigma_2)^{1/4}}{\det\left(\frac{\Sigma_1 + \Sigma_2}{2}\right)^{1/2}} \exp\left(-\frac{1}{8}(\mu_1 - \mu_2)^T \left(\frac{\Sigma_1 + \Sigma_2}{2}\right)^{-1} (\mu_1 - \mu_2)\right)} \tag{14}$$

where, again, $0 \leq H(P, Q) \leq 1$.

The obtained results for the error performance are shown in Fig. 18. They prove that most of the compiled neural networks produce a reasonably good estimation of the desired queries. Nevertheless, some of them produce a significant but moderate error (see Fig. 18(b)) that can be improved by simply increasing the training time (i.e., none of the parameters established for these networks have to be modified). In general, the obtained error is not high enough to distort the original distributions queried, and thus, these approximations can be employed for the task of inference in the proposed navigation problem (as it has been shown in Section 5.4).

A comparative study of the computational efficiency has also been carried out. Table 8 shows the time needed by each approach (i.e., by the non-compiled inference done on the proposed BN architecture and by the neural networks) to obtain a single query. This experiment has been repeated 9000 times for each layer in the architecture and inference approach. The results show that, in general, the execution time achieved by using neural networks is several orders of magnitude lower compared to the one corresponding to performing inference with Bayesian networks.

Finally, another study regarding computational cost has also been carried out. As explained in Section 4, the proposed inference algorithm computes queries for each sensory sampling iteration taking all the existing layers into account. Also, in the current navigation problem, the Bayesian architecture is completely replicated for each detected pedestrian. Thus, it would be interesting to analyze the cost of inference per sampling iteration for both approaches, considering an increasing number of *actual* pedestrians. Recall that, in the proposed algorithm, each layer is inferred the same times as the number of detected pedestrians. Several sequences of 5000 iterations have been simulated in these conditions, each one for a different number of detected pedestrians, up to 20 of them. The obtained results are shown in Fig. 19. Both inference approaches achieve a computational cost that is approximately linear with the number of processed pedestrians. These results also prove that the use of feedforward neural networks

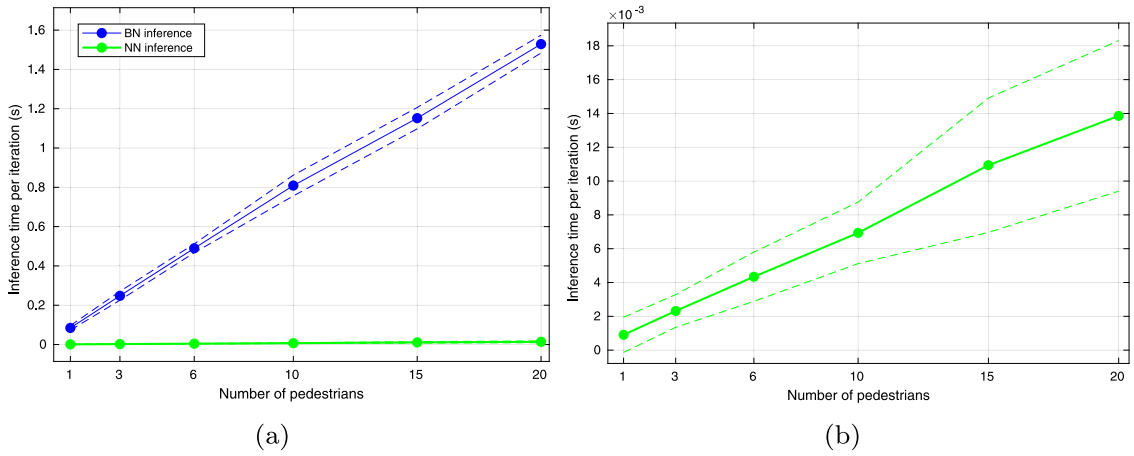


Fig. 19. Mean inference time per sensory sampling iteration achieved by inference with Bayesian networks and by inference with neural networks, as a function of the number of detected pedestrians. Dashed lines correspond to $\pm 2\sigma$, where σ is the standard deviation for each test, containing 5000 samples. (a) Comparative results for both inference approaches. (b) Vertically zoomed view of the results for inference with neural networks.

does not only increases the efficiency for a single layer, but that also enables for real time inference with a reasonable amount of pedestrians. In contrast, direct inference over Bayesian networks only achieves reasonable execution times for a reduced number of them. Recall that the time of inference in these experiments has been measured by using the same desktop PC as in the training process.

6. Conclusions and future work

In this work we have contributed with a new probabilistic model for the representation and management of abnormal sensory behavior in mobile robots along with an inference methodology that enables for diagnosis and recovery in real time. The most relevant novelty in our approach is its multi-dimensional architecture, that allows us to cope efficiently with dependencies among sensors, their temporal dynamics and the complexities of their diverse kinds of data and operations, all at once. Furthermore, this is based on the rigorous formalism of Bayesian networks, whose power for doing knowledge inference is leveraged and enhanced in order to target complex systems, augmenting the expressiveness of monolithic approaches in different aspects. This is especially noticeable for the concurrent treatment of several levels of cognitive abstraction, which is not found among the works in the literature of FDD related to robotics.

We have adapted existing inference methods to work in this multidimensional setting and also have presented a systematic methodology for its compilation as neural networks, aimed at providing real-time performance, that leverages the particular topologies of our BN-based models to alleviate the requirements of training. We have implemented all these proposals for mobile robot navigation in environments with human presence, a particular demanding task with a diversity of sensors and possible anomalies which could have been much more hard to model without the multidimensional features of the proposed architecture.

The obtained quantitative results in different sets of both simulated and real experiments show that our proposal serves to increase the safety of operation while performing anomaly detection and recovery in real time. For that purpose we have used well-known measures of safety for environments with vehicles and persons; all of them show significant statistical improvements in many aspects of the task under anomalies, e.g., minimum distance to pedestrians, tracking time and navigation time.

Being aimed this work at solving relevant limitations of previous ones, it still has its own. On the one hand, its intrinsic approximate nature may produce more or less important errors in inference depending on the way the different layerings of the structure are decided. In this paper we do not deal with the automatic learning of that structure, thus the expert is responsible for finding good trade-offs concerning where to cut dependency cycles or placing layers, which concentrates a big part of the engineering effort for this system; in the future, structural learning methods adapted to our particular architecture should be devised. On the other hand, the very cognitive axis should have a systematic procedure for definition, and/or an utility measure that can serve to optimize it with respect to the achievable performance and the ease of design simultaneously; expert knowledge has an important role here, but even that subjectivity could be provided with suitable guidelines.

Besides doing further research on these issues, there are different topics that can be addressed in the future. The proposed multidimensional Bayesian model and inference algorithm have been validated in a real mobile robot; however, they should be implemented for a wider variety of robotic platforms (aerial, terrestrial and submarine) and, above all, tasks that use more complex sensory devices and need complex sets of anomalies and recovery processes in order to complete their validation.

The modeling process should also be improved in order to be done more automatically and autonomously, but keeping its ability to reflect human knowledge as well as other heterogeneous sources of information. For that, it would be interesting to explore the existing algorithms for parameter and structure learning in the context of Bayesian networks (Koller and Friedman, 2009).

It would be also interesting to study up to what extent the integration of human knowledge at different levels of abstraction enhances the robustness of existing algorithms related to pedestrian detection and tracking.

CRediT authorship contribution statement

Manuel Castellano-Quero: Conceptualization, Methodology, Software, Validation, Data curation, Writing – original draft, Writing – review & editing. **Manuel Castillo-López:** Methodology, Software, Validation, Data curation. **Juan-Antonio Fernández-Madrigal:** Conceptualization, Methodology, Validation, Resources, Writing – review & editing, Supervision. **Vicente Arévalo-Espejo:** Software, Validation, Data curation. **Holger Voos:** Resources, Supervision, Project administration, Funding acquisition. **Alfonso García-Cerezo:** Resources, Supervision, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Manuel Castellano-Quero reports financial support was provided by Spain Ministry of Science and Innovation. Alfonso Garcia-Cerezo reports financial support was provided by Ministry of Economic Affairs and Digital Transformation.

Data availability

Data will be made available on request

Acknowledgments

This work has been supported by the Spanish government through the national grant FPU16/02243, by the University of Málaga, Spain through its local research program and the International Excellence Campus Andalucía Tech, Spain, and by the Spanish national research project RTI2018-093421-B-100. The authors would also like to thank the anonymous reviewers for their valuable comments and suggestions, which have greatly helped to improve the quality of the paper.

Appendix A. Dependency graphs for multidimensional sensory systems

Consider a list of identifiers $\mathbf{B_{id}} = \{a, b, c, \dots\}$, where each element is associated with a Bayesian sensor (e.g., $\mathcal{B}_a$, $\mathcal{B}_b$, $\mathcal{B}_c$, etc.). We denote a dependency involving two sensors a and b as a pair (a, b). Then, a homogeneous binary relation can be defined as a set of pairs $\mathbf{D} \subseteq \{(a,b)|(a,b) \in \mathbf{B_{id}^2} : a \neq b\}$. For non-cyclic dependencies, $\mathbf{D}$ is an asymmetric homogeneous relation $\mathbf{D_{nc}}$, where any two elements (a, b) and (c, d) verify $(a \neq d) \vee (b \neq c)$. For the cyclic case, $\mathbf{D}$ is a symmetric homogeneous relation $\mathbf{D_c}$, where for any element (a, b) there exist another and unique element (c, d) such that $(a = d) \wedge (b = c)$.

In order to capture the interactions among sensors in an abstract and compact manner while considering levels of abstraction and temporal dynamics, a *dependency graph* D_G is defined from $\mathbf{D_{nc}}$, which is a directed acyclic graph whose vertices are the set of identifiers $\mathbf{B_{id}}$, and its edges the set of dependencies in the sensory system, thus, $D_G = (\mathbf{B_{id}}, \mathbf{D_{nc}})$. Each node in D_G is also annotated with a pair of elements from two different sets. One of them is $\mathbf{C_{id}} = \{c_a, c_b, c_c, \dots\}$, which is formed by natural numbers, each one indicating the level of cognitive abstraction to which each sensor represented in $\mathbf{B_{id}}$ belongs, in the same order (e.g., c_a would be the level for sensor $\mathcal{B}_a$, etc.). These numbers range from the lowest possible level (i.e, zero) to the highest one (i.e., the total number of cognitive levels in the system c_l minus one). The other set is $\mathbf{T_{id}} = \{t_a, t_b, t_c, \dots\}$, which contains boolean values indicating whether temporal dynamics are encoded for the Bayesian sensors represented in $\mathbf{B_{id}}$, in the same order (e.g., $t_a = d$ indicates that Bayesian sensor $\mathcal{B}_a$ is dynamic, $t_b = \bar{d}$ indicates that sensor $\mathcal{B}_b$ is not, etc.). A procedure to build this graph given a list of Bayesian sensors, the interactions among them, the levels of abstraction these sensors belong to, and the indicators of their dynamic character is detailed in Algorithm A.1. For the sake of simplicity, it is assumed there that the assignment of cognitive levels in $\mathbf{C_{id}}$ is defined such that every node in D_G verifies that each one of its children belongs to an equal or higher level of abstraction.

Algorithm A.1 Creation of the annotated dependency graph

input:
$\mathbf{B_{id}}$: list of identifiers of Bayesian sensors
$\mathbf{C_{id}}$: list of levels of cognitive abstraction associated with Bayesian sensors
$\mathbf{T_{id}}$: list of temporal dynamics indicators for Bayesian sensors
$\mathbf{D_{nc}}$: list of non-cyclic dependencies
$\mathbf{D_c}$: list of cyclic dependencies

output:
D_G: annotated dependency graph
$\mathbf{B_{id}}$: updated list of identifiers of Bayesian sensors

main:
for each pair of elements of the form $\{(i,j),(j,i)\} \subset \mathbf{D_c}$ **do**
  $k \leftarrow$ choose one of the sensors, either i or j
  $\mathbf{B_{id}} \leftarrow \mathbf{B_{id}} \cup k$
  if k is a new instance of i **then**
    $\mathbf{C_{id}} \leftarrow \mathbf{C_{id}} \cup c_k$, with $c_k = c_i$
    $\mathbf{T_{id}} \leftarrow \mathbf{T_{id}} \cup t_k$, with $t_k = t_i$
    $\mathbf{D_{nc}} \leftarrow \mathbf{D_{nc}} \cup \{(i,j),(j,k),(i,k)\}$
  else if k is a new instance of j **then**
    $\mathbf{C_{id}} \leftarrow \mathbf{C_{id}} \cup c_k$, with $c_k = c_j$
    $\mathbf{T_{id}} \leftarrow \mathbf{T_{id}} \cup t_k$, with $t_k = t_j$
    $\mathbf{D_{nc}} \leftarrow \mathbf{D_{nc}} \cup \{(j,i),(j,k),(i,k)\}$
  end if
end for
$D_G = (\mathbf{B_{id}}, \mathbf{D_{nc}})$
Annonate each node i in D_G with a pair (c_i,t_i), where $c_i \in \mathbf{C_{id}}$ and $t_i \in \mathbf{T_{id}}$
return D_G and $\mathbf{B_{id}}$

Appendix B. Complete description of the proposed Bayesian sensory architecture

In this appendix, we provide a complete description of the remaining models in the proposed Bayesian sensory architecture. Some of the Bayesian sensors used rely, in turn, on auxiliary sensors that have not been included in the model definition for the sake of simplicity. All of them will be described as they appear within each layer of the architecture. Also, only some key CPDs will be shown, for the sake of brevity.

Layer *zero* (Fig. B.1) contains two different Bayesian sensors. One of them is a dynamic sensor (Fig. B.1(a)), defined over continuous random variables, that serves to predict the pose and velocity of a pedestrian that was detected by the Spencer system in a previous iteration but not in the current one for some reason. In the case that the pedestrian is not missing, this sensor can also be used as a filtered estimation of the state by incorporating the information from Spencer as evidence (thus, the evidence set would be empty in the previous case). The prediction is based on a simple constant velocity model over six variables, which is improved in subsequent layers, as explained in the main text (see Section 5.3).

The other sensor in this layer is a so-called *age* sensor (Fig. B.1(b)), which represents the age of a detected pedestrian using discrete binary variables for that (the possible values considered here are *young* and *elder*). This commonsense knowledge will be useful in subsequent layers in order to estimate the speed at which a detected pedestrian is most likely to move. This sensory information should come from an external source, for instance, from a computer vision system, with some uncertainty. Note that the age of a pedestrian could be considered at a higher level of cognitive abstraction; however, the sensor is assigned here to a low cognitive level since the information it represents only serves to modify the belief in the expected speed of a pedestrian, as mentioned before.

Layer one (Fig. B.2) contains a unique Bayesian sensor, dedicated to the detection of non-existent, "ghost" pedestrians that are sometimes produced by the Spencer tracker. These fictitious pedestrians, if any, usually appear nearby walls or other boundaries. In order to include such an environmental knowledge, the anomalies subnetwork of this sensor is based on an auxiliary one that serves to determine whether the distance of a given pedestrian to the closest wall is too short. To obtain that, the auxiliary sensor counts with the pose estimation for the considered pedestrian from layer zero and with a map of the scene. In case of anomaly, i.e., in case the distance is actually too short,

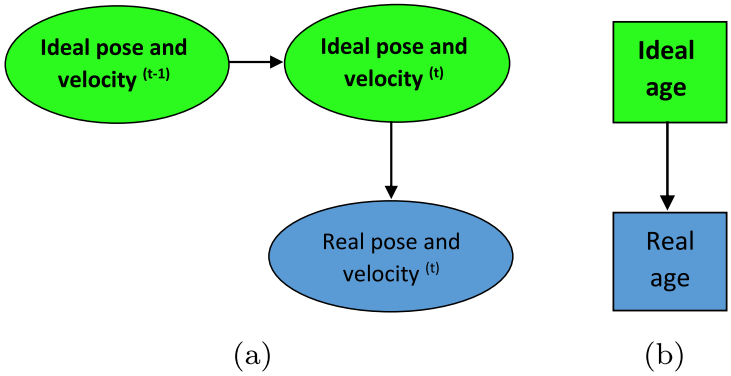
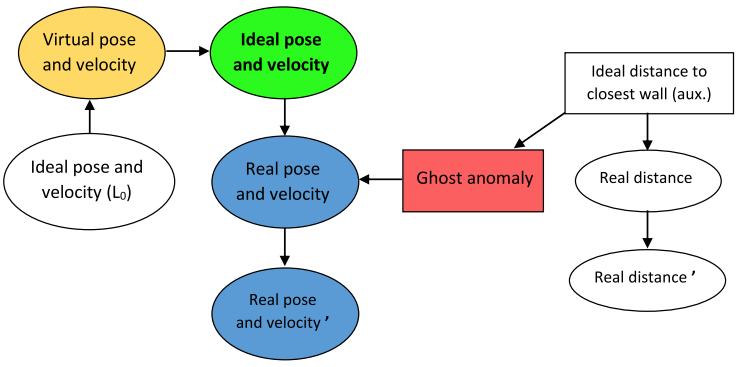


Fig. B.1. Bayesian sensors assigned to layer *zero* in the architecture. Recall that squared nodes represent discrete random variables and that round ones represent continuous variables. (a) Pedestrian pose and velocity predictor. (b) Age sensor.



Ideal distance	p(Real Distance|Ideal Distance)
Too short	$\mathcal{N}(0\,;\, d^2/4)$ (m)
Normal	$\mathcal{N}(d\,;\, d^2/4)$ (m)

(b)

Fig. B.2. (a) Bayesian network corresponding to the ghost pedestrian sensor (layer one). Replicated ideal nodes from lower layers are specified, as well as auxiliary sensors. Again, squared nodes represent discrete variables and round ones continuous variables. (b) CPD for the real distance node of the auxiliary Bayesian sensor, with physical units. Here, d refers to the typical distance that a pedestrian usually keeps from nearby walls.

the recovered pose from the ideal node will have a highly inflated uncertainty, thus severely deteriorating its value. This is done in order that such a fictitious pedestrian can be eliminated from the system, since the implemented version of the architecture ignores detected pedestrians if their uncertainty is too high.

Layer 2 (Fig. B.3) also contains only one sensor, which serves to reason about the situation of a given pedestrian within the scene. More specifically, this sensor aims to encode a more abstract version of the pose of a considered pedestrian, while simultaneously capturing the age. This information is represented as a discrete value indicating the *kind* of zone the pedestrian is situated along with the age, which will be useful to improve the pose and velocity estimation in subsequent layers. It is common that the map of a scene, specially in structured environments, is constituted by recognizable parts such as hallways, end of hallways, corners, etc. These ones are precisely the kind of zones considered in this sensor. Combining them with the age of a pedestrian, it would be possible to define a suitable motion prediction model according to this information, considered social knowledge. For instance, a pedestrian is more likely to move forward in a hallway, and also, is more likely to turn while being around a corner.

Taking all the above into account, there will be as many possible values for the ideal sensor variable as the number of possible combinations of kinds of zones and ages. To encode the information, the sensor uses a replicated version of the age sensor from layer zero, and also discretizes the support of the continuous pose obtained from layer one, by defining different zones. Recall, again, that the use of more abstract information does not imply that the sensor must correspond to a high cognitive level, as long as such information is only employed for reasonings involving low-level data.

Layer 4 (Fig. B.4) contains three different Bayesian sensors. One of them is the long-term occlusion sensor (Fig. B.4(a)), which is employed to determine whether any of the occlusions events defined before persist over time. For that, this sensor incorporates an auxiliary one that discretizes the recovered pose from layer three (see Section 5.3). This serves to determine whether a given pedestrian is situated within any of the mentioned occlusion regions. Note that the ideal node of the auxiliary sensor is defined over a discrete binary random variable, which takes a *true* value when the occlusion situation occurs. The more this situation persists over time, the greater will be the belief in a long-term occlusion. The other two sensors in this layer, i.e., the distance and

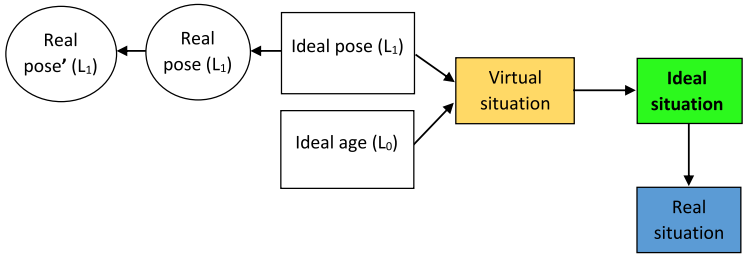


Fig. B.3. Bayesian network corresponding to the situation sensor (layer two). Again, replicated ideal nodes and auxiliary Bayesian sensors are specified.

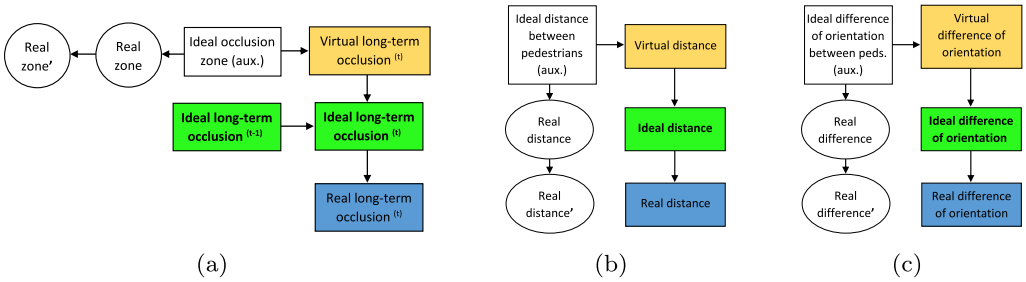
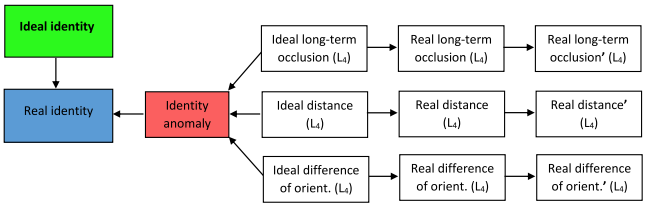


Fig. B.4. Bayesian sensors assigned to layer four in the architecture. (a) Long-term occlusion sensor. (b) Distance sensor. (c) Difference of orientation sensor.



(a)

Long-term occlusion	Distance	Difference of orient. \ Anomaly	True	False
True	Short	Low	0.99	0.01
True	Long	Low	0.05	0.95
True	Short	High	0.05	0.95
True	Long	High	0.01	0.99
False	Short	Low	0.05	0.95
False	Long	Low	0.001	0.999
False	Short	High	0.01	0.99
False	Long	High	0.001	0.999

(b)

Fig. B.5. (a) Bayesian network corresponding to the identity sensor (layer five). (b) Tabular CPD for the identity anomaly node.

difference of orientation sensors (Figs. B.4(b) and (c), respectively), rely on the recovered pose from the layer three of two different pedestrians, the one considered and the one having the most similar pose compared to the former. These sensors serve to provide a discrete distance and difference of orientation between these pedestrians, in order to determine whether their poses are actually similar. This information will be used by the identity sensor, as explained later on.

Finally, layer 5 (Fig. B.5) contains the identity sensor, which serves to recover the correct identity of a given pedestrian in case it has been confused with another one. This adverse situation occurs when a pedestrian re-appears after suffering a long-term occlusion event. In this case, the Spencer system would assign a new identity, leading to the presence of two pedestrians, the one being maintained by the sensory model during the occlusion period and the one just recovered by the tracker. The identity sensor allows to determine whether this is the case, relying on the information provided by the sensors in layer four. Thanks to the use of the identity sensor, the implemented version of the architecture is able to recover the correct identity of the affected pedestrian while ignoring the other one. Note that this sensor is considered here to belong to a higher level of cognitive abstraction, since its ideal node is defined over a discrete random variable encoding a reasoning about the identity of a pedestrian. In other words, low-level information of pose and distance is being used in this case to produce high-level information about identity, in contrast to the case of sensors in lower layers.

References

Abebe-Assefa, A., Tian, W., Nketia-Acheampong, K., Umar-Aftab, M., Ahmad, M., 2022. Small-scale and occluded pedestrian detection using multi mapping feature extraction function and modified soft-NMS. Comput. Intell. Neurosci. 2022, 11.

Abid, A., Khan, M.T., Iqbal, J., 2021. A review on fault detection and diagnosis techniques: Basics and beyond. Artif. Intell. Rev. 54 (5), 3639–3664. http://dx.doi.org/10.1007/s10462-020-09934-2.

Abid, A., Khan, M.T., Lang, H., de Silva, C.W., 2019. Adaptive system identification and severity index-based fault diagnosis in motors. IEEE/ASME Trans. Mechatronics 24 (4), 1628–1639. http://dx.doi.org/10.1109/TMECH.2019.2917749.

Adhikary, A., Vatsa, R., Burnwal, A., Samanta, J., 2019. Performance evaluation of low-cost RGB-depth camera and ultrasonic sensors.

Afrassa, K.W., Boz, A.Z., Amasyali, M.F., Tahar, S., 2019. Benchmarking BNT inference engines using an early warning system. In: 2019 Innovations in Intelligent Systems and Applications Conference. ASYU, pp. 1–5.

Aggarwal, C.C., 2018. Neural Networks and Deep Learning. A Textbook. Springer, p. 497.

Aguilar-Moreno, M., Cruz-Martín, A., Fernández-Madrigal, J.-A., 2018. Modelado cinematico y simulacion realista del manipulador movil Turtlebot-2 + Widow-X en ROS. In: XXXIX Jornadas de Automatica. Badajoz (Spain), pp. 262–269.

Alatise, M.B., Hancke, G.P., 2020. A review on challenges of autonomous mobile robot and sensor fusion methods. IEEE Access 8, 39830–39846. http://dx.doi.org/10.1109/ACCESS.2020.2975643.

Arras, K.O., Mozos, O.M., Burgard, W., 2007. Using boosted features for the detection of people in 2D range data. In: Proceedings 2007 IEEE International Conference on Robotics and Automation. IEEE, pp. 3402–3407.

Azzalini, D., Castellini, A., Luperto, M., Farinelli, A., Amigoni, F., 2020. HMMs for anomaly detection in autonomous robots. In: AAMAS '20: Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems. pp. 105–113.

Bader, K., Lussier, B., Schön, W., 2017. A fault tolerant architecture for data fusion: A real application of Kalman filters for mobile robot localization. Robot. Auton. Syst. 88, 11–23. http://dx.doi.org/10.1016/j.robot.2016.11.015.

Beck, F., Bader, M., 2019. Map based human motion prediction for people tracking. In: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS, IEEE, pp. 1–7.

Bonci, A., Cen Cheng, P.D., Indri, M., Nabissi, G., Sibona, F., 2021. Human-robot perception in industrial environments: A survey. Sensors 21 (5), 1571. http://dx.doi.org/10.3390/s21051571.

Brunetti, A., Buongiorno, D., Trotta, G.F., Bevilacqua, V., 2018. Computer vision and deep learning techniques for pedestrian detection and tracking: A survey. Neurocomputing 300, 17–33. http://dx.doi.org/10.1016/j.neucom.2018.01.092.

Cai, B., Hao, K., Wang, Z., Yang, C., Kong, X., Liu, Z., Ji, R., Liu, Y., 2021. Data-driven early fault diagnostic methodology of permanent magnet synchronous motor. Expert Syst. Appl. 177, 115000. http://dx.doi.org/10.1016/j.eswa.2021.115000.

Cai, B., Huang, L., Xie, M., 2017a. Bayesian networks in fault diagnosis. IEEE Trans. Ind. Inform. 13 (5), 2227–2240. http://dx.doi.org/10.1109/TII.2017.2695583.

Cai, B., Liu, Y., Liu, Z., Chang, Y., Jiang, L., 2020. Bayesian Networks for Reliability Engineering. Springer, Singapore.

Cai, B., Liu, Y., Xie, M., 2017b. A dynamic-Bayesian-network-based fault diagnosis methodology considering transient and intermittent faults. IEEE Trans. Autom. Sci. Eng. 14 (1), 276–285. http://dx.doi.org/10.1109/TASE.2016.2574875.

Cai, B., Wang, Z., Zhu, H., Liu, Y., Hao, K., Yang, Z., Ren, Y., Feng, Q., Liu, Z., 2022. Artificial intelligence enhanced two-stage hybrid fault prognosis methodology of PMSM. IEEE Trans. Ind. Inform. 18 (10), 7262–7273. http://dx.doi.org/10.1109/TII.2021.3128245.

Castellano-Quero, M., Fernández-Madrigal, J.-A., García-Cerezo, A., 2021. Improving Bayesian inference efficiency for sensory anomaly detection and recovery in mobile robots. Expert Syst. Appl. 163, 113755. http://dx.doi.org/10.1016/j.eswa.2020.113755.

Castillo-Lopez, M., Ludivig, P., Sajadi-Alamdari, S.A., Sanchez-Lopez, J.L., Olivares-Mendez, M.A., Voos, H., 2020. A real-time approach for chance-constrained motion planning with dynamic obstacles. IEEE Robot. Autom. Lett. 5 (2), 3620–3625. http://dx.doi.org/10.1109/LRA.2020.2975759.

Darwiche, A., 2003. A differential approach to inference in Bayesian networks. J. ACM 50 (3), 280–305. http://dx.doi.org/10.1145/765568.765570.

Darwiche, A., 2009. Modeling and Reasoning with Bayesian Networks, vol. 9780521884. Cambridge University Press, New York, pp. 1–548.

Das, D., Banerjee, S., Chernova, S., 2021. Explainable AI for robot failures. In: Proceedings of the 2021 ACM/IEEE International Conference on Human-Robot Interaction. ACM, New York, NY, USA, pp. 351–360.

Dezan, C., Zermani, S., Hireche, C., 2020. Embedded Bayesian network contribution for a safe mission planning of autonomous vehicles. Algorithms 13, 155.

Dimitrievski, M., Veelaert, P., Philips, W., 2019. Behavioral pedestrian tracking using a camera and LiDAR sensors on a moving vehicle. Sensors 19 (2), 391. http://dx.doi.org/10.3390/s19020391.

Doran, M., Sterritt, R., Wilkie, G., 2020. Autonomic architecture for fault handling in mobile robots. Innov. Syst. Software Eng. 16 (3–4), 263–288. http://dx.doi.org/10.1007/s11334-020-00361-8.

Fan, T., Long, P., Liu, W., Pan, J., 2020. Distributed multi-robot collision avoidance via deep reinforcement learning for navigation in complex scenarios. Int. J. Robot. Res. 39 (7), 856–892. http://dx.doi.org/10.1177/0278364920916531.

Fernández-Madrigal, J.-A., Cruz-Martín, A., 2020. The CRUMB Project website. URL https://babel.isa.uma.es/crumb/.

Ferrera, E., Capitán, J., Castaño, A.R., Marrón, P.J., 2017. Decentralized safe conflict resolution for multiple robots in dense scenarios. Robot. Auton. Syst. 91, 179–193. http://dx.doi.org/10.1016/j.robot.2017.01.008.

Galar, D., Kumar, U., Seneviratne, D., 2020. Robots, Drones, UAVs and UGVs for Operation and Maintenance. CRC Press.

Gomes, I.P., Wolf, D.F., 2019. A health monitoring system with hybrid Bayesian network for autonomous vehicle. In: 2019 19th International Conference on Advanced Robotics. ICAR, IEEE, pp. 260–265.

Gomes, I.P., Wolf, D.F., 2021. Health monitoring system for autonomous vehicles using dynamic Bayesian networks for diagnosis and prognosis. J. Intell. Robot. Syst. 101 (1), 19. http://dx.doi.org/10.1007/s10846-020-01293-y.

Graham Miller, O., Gandhi, V., 2021. A survey of modern exogenous fault detection and diagnosis methods for swarm robotics. J. King Saud Univ., Eng. Sci. 33 (1), 43–53. http://dx.doi.org/10.1016/j.jksues.2019.12.005.

Guo, Q., Zhang, X., Li, J., Li, G., 2022. Fault diagnosis of modular multilevel converter based on adaptive chirp mode decomposition and temporal convolutional network. Eng. Appl. Artif. Intell. 107, 104544. http://dx.doi.org/10.1016/j.engappai.2021.104544.

He, Y., Yu, Z., Li, J., Ma, G., Xu, Y., 2019. Fault correction of algorithm implementation for intelligentized robotic multipass welding process based on finite state machines. Robot. Comput.-Integr. Manuf. 59, 28–35. http://dx.doi.org/10.1016/j.rcim.2019.03.002.

Helbing, D., Molnár, P., 1995. Social force model for pedestrian dynamics. Phys. Rev. E 51 (5), 4282–4286. http://dx.doi.org/10.1103/PhysRevE.51.4282.

Hirakawa, R., Uchida, H., Nakano, A., Tominaga, K., Nakatoh, Y., 2021. Anomaly detection on software log based on temporal memory. Comput. Electr. Eng. 95, 107433. http://dx.doi.org/10.1016/j.compeleceng.2021.107433.

Hokuyo, 2020. Hokuyo Automatic Co. Ltd. corporate website. URL https://www.hokuyo-aut.jp/search/single.php?serial=166.

Honig, S., Oron-Gilad, T., 2018. Understanding and resolving failures in human-robot interaction: Literature review and model development. Front. Psychol. 9, http://dx.doi.org/10.3389/fpsyg.2018.00861.

Hossain, N.U.I., Amrani, S.E., Jaradat, R., Marufuzzaman, M., Buchanan, R., Rinaudo, C., Hamilton, M., 2020. Modeling and assessing interdependencies between critical infrastructures using Bayesian network: A case study of inland waterway port and surrounding supply chain network. Reliab. Eng. Syst. Saf. 198, 106898. http://dx.doi.org/10.1016/j.ress.2020.106898.

IFR, 2022. International Federation of Robotics. URL https://ifr.org/.

Islam, M.M., Hu, G., Liu, Q., Dan, W., Lyu, C., 2018. Correlation filter based moving object tracking with scale adaptation and online re-detection. IEEE Access 6, 75244–75258.

Jia, J., Zhou, H., Li, Y., 2017. Using deep neural network approximate Bayesian network. arXiv:1801.00282.

Jiang, B., Chen, S., Wang, B., Luo, B., 2022. MGLNN: Semi-supervised learning via multiple graph cooperative learning neural networks. Neural Netw. 153, 204–214. http://dx.doi.org/10.1016/j.neunet.2022.05.024.

Kadiyam, J., Parashar, A., Mohan, S., Deshmukh, D., 2020. Actuator fault-tolerant control study of an underwater robot with four rotatable thrusters. Ocean Eng. 197, 106929. http://dx.doi.org/10.1016/j.oceaneng.2020.106929.

Kang, H., Joung, J., Kim, J., Kang, J., Cho, Y.S., 2020. Protect your sky: A survey of counter unmanned aerial vehicle systems. IEEE Access 8, 168671–168710. http://dx.doi.org/10.1109/ACCESS.2020.3023473.

Katz, D.S., Hutchison, C., Zizyte, M., Goues, C.L., 2020. Detecting execution anomalies as an oracle for autonomy software robustness. In: 2020 IEEE International Conference on Robotics and Automation. ICRA, IEEE, pp. 9366–9373.

Keipour, A., Mousaei, M., Scherer, S., 2019. Automatic Real-time anomaly detection for autonomous aerial vehicles. In: 2019 International Conference on Robotics and Automation. ICRA, IEEE, pp. 5679–5685.

Keliris, C., Polycarpou, M., T., P., 2018. An adaptive approach to sensor bias fault diagnosis and accommodation for a class of input-output nonlinear systems. In: 2018 IEEE Conference on Decision and Control. CDC, pp. 6334–6339. http://dx.doi.org/10.1109/CDC.2018.8619307.

Khalastchi, E., Kalech, M., 2019a. Fault detection and diagnosis in multi-robot systems: A survey. Sensors 19 (18), 4019. http://dx.doi.org/10.3390/s19184019.

Khalastchi, E., Kalech, M., 2019b. Fault detection and diagnosis in multi-robot systems: A survey. Sensors (Basel, Switzerland) 19.

Khalastchi, E., Kalech, M., 2019c. On fault detection and diagnosis in robotic systems. ACM Comput. Surv. 51 (1), 1–24. http://dx.doi.org/10.1145/3146389.

Koller, D., Friedman, N., 2009. Probabilistic Graphical Models: Principles and Techniques. The MIT Press, Cambridge, Massachusetts, p. 1270.

Kong, X., Cai, B., Liu, Y., Zhu, H., Liu, Y., Shao, H., Yang, C., Li, H., Mo, T., 2022. Optimal sensor placement methodology of hydraulic control system for fault diagnosis. Mech. Syst. Signal Process. 174, 109069. http://dx.doi.org/10.1016/j.ymssp.2022.109069.

Kong, X., Cai, B., Liu, Y., Zhu, H., Yang, C., Gao, C., Liu, Y., Liu, Z., Ji, R., 2023. Fault diagnosis methodology of redundant closed-loop feedback control systems: Subsea blowout preventer system as a case study. IEEE Trans. Syst., Man, Cybern.: Syst. 53 (3), 1618–1629. http://dx.doi.org/10.1109/TSMC.2022.3204777.

Kontogiorgos, D., Pereira, A., Sahindal, B., van Waveren, S., Gustafson, J., 2020. Behavioural responses to robot conversational failures. In: Proceedings of the 2020 ACM/IEEE International Conference on Human-Robot Interaction. ACM, New York, NY, USA, pp. 53–62.

Kyrarini, M., Lygerakis, F., Rajavenkatanarayanan, A., Sevastopoulos, C., Nambiappan, H.R., Chaitanya, K.K., Babu, A.R., Mathew, J., Makedon, F., 2021. A survey of robots in healthcare. Technologies 9 (1), 8. http://dx.doi.org/10.3390/technologies9010008.

Lei, M., Song, Y., Zhao, J., Wang, X., Lyu, J., Xu, J., Yan, W., 2022. End-to-end network for pedestrian detection, tracking and re-identification in real-time surveillance system. Sensors 22 (22), http://dx.doi.org/10.3390/s22228693.

Li, Y., Yang, H., 2022. Fault localization of industrial robot system based on knowledge graph and Bayesian network. In: 2022 China Automation Congress. CAC, pp. 2902–2907.

Li, T., Zhou, Y., Zhao, Y., Zhang, C., Zhang, X., 2022. A hierarchical object oriented Bayesian network-based fault diagnosis method for building energy systems. Appl. Energy 306, 118088. http://dx.doi.org/10.1016/j.apenergy.2021.118088.

Liao, Y., Yeaser, A., Yang, B., Tung, J., Hashemi, E., 2021. Unsupervised fault detection and recovery for intelligent robotic rollators. Robot. Auton. Syst. 146, 103876. http://dx.doi.org/10.1016/j.robot.2021.103876.

Lin, Y., Zakwan, S., Jennions, I.K., 2020. A Bayesian approach to fault identification in the presence of multi-component degradation.

Linder, T., Arras, K.O., 2016. People detection, tracking and visualization using ROS on a mobile service robot. pp. 187–213.

Liu, C., Sun, J., Wang, F., Ning, S., Xu, G., 2020. Bayesian network method for fault diagnosis of civil aircraft environment control system. Proc. Inst. Mech. Eng., Part I: J. Syst. Control Eng. 234, 662–674.

Long, J., Mou, J., Zhang, L., Zhang, S., Li, C., 2021. Attitude data-based deep hybrid learning architecture for intelligent fault diagnosis of multi-joint industrial robots. J. Manuf. Syst. 61, 736–745. http://dx.doi.org/10.1016/j.jmsy.2020.08.010.

Löwe, M., Frellsen, J., Hansen, P.L., Risi, S., 2021. Rapid risk minimization with Bayesian models through deep learning approximation. In: 2021 International Joint Conference on Neural Networks. IJCNN, pp. 1–8.

Luperto, M., Monroy, J., Ruiz-Sarmiento, J.R., Moreno, F.-A., Basilico, N., Gonzalez-Jimenez, J., Borghese, N.A., 2019. Towards long-term deployment of a mobile robot for at-home ambient assisted living of the elderly. In: 2019 European Conference on Mobile Robots. ECMR, IEEE, pp. 1–6.

Marquardt, D.W., 1963. An algorithm for least-squares estimation of nonlinear parameters. J. Soc. Ind. Appl. Math. 11 (2), 431–441. http://dx.doi.org/10.1137/0111030.

Matsuoka, S., Sawaragi, T., 2022. Recovery planning of industrial robots based on semantic information of failures and time-dependent utility. Adv. Eng. Inform. 51, 101507. http://dx.doi.org/10.1016/j.aei.2021.101507.

Medina, G., Guiochet, J., Lesire, C., Manecy, A., 2022. A skill fault model for autonomous systems. In: 2022 IEEE/ACM 4th International Workshop on Robotics Software Engineering. RoSE, pp. 55–62.

Miklavcic, S.J., 2020. An Illustrative Guide To Multivariable and Vector Calculus. Springer, p. 325.

Mirnaghi, M.S., Haghighat, F., 2020. Fault detection and diagnosis of large-scale HVAC systems in buildings using data-driven methods: A comprehensive review. Energy Build. 229, 110492. http://dx.doi.org/10.1016/j.enbuild.2020.110492.

Møller, M.F., 1993. A scaled conjugate gradient algorithm for fast supervised learning. Neural Netw. 6 (4), 525–533. http://dx.doi.org/10.1016/S0893-6080(05)80056-5.

Mondal, P.P., Ferreira, P.M., Kapoor, S.G., Bless, P.N., 2021. Monitoring and diagnosis of multistage manufacturing processes using hierarchical Bayesian networks. Procedia Manuf. 53, 32–43. http://dx.doi.org/10.1016/j.promfg.2021.06.007.

Murphy, K.P., 2002. Dynamic Bayesian Networks: Representation, Inference and Learning (Ph.D. thesis). University of California, Berkeley, p. 225.

Newman, W., 2017. A Systematic Approach To Learning Robot Programming with ROS. Chapman and Hall/CRC, p. 502.

Open Source Robotics Foundation, 2020. Turtlebot website. URL https://www.turtlebot.com/turtlebot2/.

Pacini, D.H., 2022. Identification in parametric models: The minimum hellinger distance criterion. Econometrics.

Pardo, L., 2006. Statistical Inference Based on Divergence Measures. Chapman and Hall/CRC, New York, p. 512.

Pearl, J., 1985. Bayesian networks: A model of self-activated memory for evidential reasoning. In: Proceedings of the 7th Conference of the Cognitive Science Society. pp. 329–334.

Rakesh, S.K., Shrivastava, M., 2022. Performance analysis of fault tolerance algorithm for pattern formation of swarm agents. Knowl.-Based Syst. 240, 108020. http://dx.doi.org/10.1016/j.knosys.2021.108020.

Rodríguez, J.S., 2021. A comparison of an RGB-d cameras performance and a stereo camera in relation to object recognition and spatial position determination. Electron. Lett. Comput. Vis. Image Anal. 20, 16–27.

Russell, S., Norvig, P., 2020. Artificial Intelligence: A Modern Approach. Pearson, p. 1136.

Shakhatreh, H., Sawalmeh, A.H., Al-Fuqaha, A., Dou, Z., Almaita, E., Khalil, I., Othman, N.S., Khreishah, A., Guizani, M., 2019. Unmanned aerial vehicles (UAVs): A survey on civil applications and key research challenges. IEEE Access 7, 48572–48634. http://dx.doi.org/10.1109/ACCESS.2019.2909530.

Shao, Y., Liu, B., Wang, S., Li, G., 2018. A novel software defect prediction based on atomic class-association rule mining. Expert Syst. Appl. 114, 237–254. http://dx.doi.org/10.1016/j.eswa.2018.07.042.

Singh, A., Raj, K., Kumar, T., Verma, S., Roy, A.M., 2023. Deep learning-based cost-effective and responsive robot for autism treatment. Drones 7 (2), http://dx.doi.org/10.3390/drones7020081.

SPENCER, 2016. Official web page of the EU FP7 research project 2013–2016: Social situation-aware perception and action for cognitive robots (SPENCER). URL http://spencer.eu/.

SPENCER, 2023. GitHub page of the EU FP7 spencer research project. URL https://github.com/spencer-project/spencer_people_tracking.

Sumi, A., Santha, T., 2019. Frame level difference (FLD) features to detect partially occluded pedestrian for ADAS. J. Sci. Ind. Res. 78, 831–836.

Tarapore, D., Timmis, J., Christensen, A.L., 2019. Fault detection in a swarm of physical robots based on behavioral outlier detection. IEEE Trans. Robot. 35 (6), 1516–1522. http://dx.doi.org/10.1109/TRO.2019.2929015.

Teh, H.Y., Kempa-Liehr, A., Wang, K.I.-K., 2020. Sensor data quality: A systematic review. J. Big Data 7, 1–49.

Tolmeijer, S., Weiss, A., Hanheide, M., Lindner, F., Powers, T.M., Dixon, C., Tielman, M.L., 2020. Taxonomy of Trust-Relevant Failures and Mitigation Strategies. In: Proceedings of the 2020 ACM/IEEE International Conference on Human-Robot Interaction. ACM, New York, NY, USA, pp. 3–12.

Vaish, R., Dwivedi, U., Tewari, S., Tripathi, S., 2021. Machine learning applications in power system fault diagnosis: Research advancements and perspectives. Eng. Appl. Artif. Intell. 106, 104504. http://dx.doi.org/10.1016/j.engappai.2021.104504.

Wang, H., Kang, Y., Yao, L., Wang, H., Gao, Z., 2021. Fault diagnosis and fault tolerant control for T–S fuzzy stochastic distribution systems subject to sensor and actuator faults. IEEE Trans. Fuzzy Syst. 29 (11), 3561–3569. http://dx.doi.org/10.1109/TFUZZ.2020.3024659.

Yang, Z., Huang, Z., He, D., Zhang, T., Yang, F., 2023. Dynamic representation-based tracker for long-term pedestrian tracking with occlusion. J. Vis. Commun. Image Represent. 90, 103710. http://dx.doi.org/10.1016/j.jvcir.2022.103710.

Yu, Y., Dong, Y., 2019. Global fault-tolerant control of underactuated aerial vehicles with redundant actuators. Int. J. Aerospace Eng..

Zahaf, A., Bououden, S., Chadli, M., Chemachema, M., 2022. Robust fault tolerant optimal predictive control of hybrid actuators with time-varying delay for industrial robot arm. Asian J. Control 24 (1), 1–15. http://dx.doi.org/10.1002/asjc.2444.

Zhang, W., Kang, B., Zhang, S., 2020. Spatial-temporal aware long-term object tracking. IEEE Access 8, 71662–71684.

Zhang, D., Liu, Q., Yan, H., Xie, M., 2021. A matrix analytic approach for Bayesian network modeling and inference of a manufacturing system. J. Manuf. Syst. 60, 202–213. http://dx.doi.org/10.1016/j.jmsy.2021.05.016.

Zhao, Y., Tong, J., Zhang, L., Wu, G., 2020. Diagnosis of operational failures and on-demand failures in nuclear power plants: An approach based on dynamic Bayesian networks. Ann. Nucl. Energy 138, 107181. http://dx.doi.org/10.1016/j.anucene.2019.107181.

Zhi, H., Yangi-Shang, 2020. Remote performance evaluation, life prediction and fault diagnosis of RV reducer for industrial robot. J. Phys. Conf. Ser. 1676.

Zhou, Y., Burg, S., Bringmann, O., Rosenstiel, W., 2018. A software reconfigurable assertion checking unit for run-time error detection. In: 2018 IEEE 23rd European Test Symposium. ETS, pp. 1–6.

Zou, T., Yang, S., Zhang, Y., Ye, M., 2020. Attention guided neural network models for occluded pedestrian detection. Pattern Recognit. Lett. 131, 91–97. http://dx.doi.org/10.1016/j.patrec.2019.12.010.

Further reading

Ahammad, T., Hasan, M.Z., Hassan, M.Z., 2020. A new topological sorting algorithm with reduced time complexity. In: ICO.

Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C., 2022. Introduction to Algorithms. The MIT Press, pp. 1–1312.