Information Fusion 122 (2025) 103136

Contents lists available at ScienceDirect

Information Fusion

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

Full length article

Vehicle localization in an explainable dynamic Bayesian network framework for self-aware agents

Giulia Slavic [a,b], Pamela Zontone [a,*], Lucio Marcenaro [a], David Martín Gómez [b], Carlo Regazzoni [a]

[a] Department of Electrical, Electronic, Telecommunications Engineering and Naval Architecture, University of Genoa, Via all'Opera Pia 11, 16145, Genoa, Italy
[b] Department of Systems Engineering and Automation, University Charles III of Madrid, Avda de la Universidad 30, Leganés, 28911, Madrid, Spain

ARTICLE INFO

Keywords:
Dynamic Bayesian networks
Visual-based localization
Explainable machine learning
Anomaly detection

ABSTRACT

This paper proposes a method to perform Visual-Based Localization within an explainable self-awareness framework, by combining deep learning with traditional signal processing methods. Localization, along with anomaly detection, is an important challenge in video surveillance and fault detection. Let us consider, for example, the case of a vehicle patrolling a train station: it must continuously know its location to effectively monitor the surroundings and respond to potential threats. In the proposed method, a Dynamic Bayesian Network model is learned. A vocabulary of clusters is obtained using the odometry and video data, and is employed to guide the training of the video model. As the video model, a combination of a Variational Autoencoder and a Kalman Filter is adopted. In the online phase, a Coupled Markov Jump Particle Filter is proposed for Visual-Based Localization. This filter combines a set of Kalman Filters with a Particle Filter, allowing us to extract possible anomalies in the test scenario as well. The proposed method is integrated into a framework based on awareness theories, and is data-driven, hierarchical, probabilistic, and explainable. The method is evaluated on trajectories from four real-world datasets, i.e., two terrestrial and two aerial. The localization accuracy and explainability of the method are analyzed in detail. We achieve a mean localization accuracy in meters of 1.65, 0.98, 0.23, and 0.87, on the four datasets.

1. Introduction

The ability of individuals to localize themselves in a familiar environment, and to build a mental map of it, is inherent to human beings. This is also true for the capability to detect unexpected events compared to previous experiences [1,2].

Similarly to how humans use their different senses to localize and detect changes, surveillance vehicles are equipped with diverse sensors, which belong to two classes: proprioceptive and exteroceptive. Proprioceptive sensors allow gaining awareness of one's own state (self-awareness); exteroceptive sensors allow gaining awareness of the surrounding environment (situational awareness). Examples of these two classes are odometric sensors (e.g., compasses, inertial measurement units, global positioning systems) and cameras, respectively. In a related manner, interoceptive sensors, by acquiring physiological signals from drivers [3–5], can also provide valuable information about the body's state and responses in various applications. It is crucial for an agent to be able to fuse the information coming from these different sensors (see also [6]).

When localization is performed using only visual data in the testing procedure, we talk about Visual-Based Localization (VBL) [7,8]. The review paper [7] mainly describes two different VBL categories: indirect and direct methods. The goal of the first ones is to offer rough pose details regarding the query's location, while the goal of the second ones is to immediately determine the pose of the query. While SLAM techniques allow one to find a relative pose of a data sequence and build a point-cloud map of the environment, VBL addresses the challenge of determining the absolute pose (see also Section 2.3). There are other recent papers in literature [9,10] that present various works carried out on VBL. Deep Learning (DL) techniques, using Convolutional Neural Networks (CNNs) or Long Short-Term Memory (LSTM) networks could also be used [8,11,12]. It is worth noting that additional subcategories of VBL methods fall under these two groups. For example, direct approaches also include feature to points matching methods, which find the correspondence between camera image features and points in a 3D point cloud model of the environment [13,14].

In VBL, during training, synchronous camera and odometry data are combined to learn a model for localizing the vehicle. During testing,

* Corresponding author.
E-mail address: pamela.zontone@unige.it (P. Zontone).

https://doi.org/10.1016/j.inffus.2025.103136
Received 18 September 2024; Received in revised form 22 January 2025; Accepted 10 March 2025
Available online 2 April 2025

the agent's positions are not provided anymore, and the learned model (e.g., a map) is employed to perform localization based on the recorded images. It can indeed happen that a system using video and odometry data during training does not have the possibility to use the odometry data also in testing. This might be due to several factors, including the absence or malfunctioning of some sensors during testing, possible environmental variability or sensor failures affecting the reliable use of odometry data, or the complexity and resource demand of integrating various sensors during testing (for running in real-time), making it unfeasible in some situations.

Simultaneously localizing oneself and detecting anomalies are two important tasks, related to each other, particularly in some scenarios such as video surveillance and fault detection with mobile robots. Let us consider a robot patrolling a pre-defined route. To perform the correct actions to follow the route and navigate the area securely, it constantly needs to know its location. Moreover, it must detect mobile or stationary anomalies, such as intruders or suspicious objects [15]. Another relevant example is the one related to drones (or Unmanned Aerial Vehicles, UAVs) supervising ships to detect faults, such as broken cables or leakages. It is indeed crucial to know where the drone is located in 3D space, as well as to recognize possible anomalies at certain positions to ensure safety and proper maintenance. Here, with the term anomaly we refer to possible abnormal data instances or unusual events that differ from a known set of rules or models (see also [16,17]).

This paper tackles the problem of VBL for surveillance vehicles. The following hypotheses are assumed to hold: *(a)* training and testing are performed in the same environment; this setting is coherent with a video surveillance application in which a robot always moves within the same building, or a drone operates on the same ship; *(b)* an estimation of the agent's position is available during the training phase to learn the correlations between video and odometry data.

So, we propose a novel method based on the awareness theories presented by neuroscientists such as Friston [18], Haykin [19], and Damasio [20]. Our objective is not to perform vehicle localization only, but to do it coherently within an explainable self-awareness framework for autonomous systems [21]. We combine Deep Learning (DL) based methods, such as Variational Autoencoders (VAEs) [22] with traditional signal processing methods, such as filtering based on Dynamic Bayesian Networks (DBNs) [23]. This way, on the one hand, we leverage the performance of Convolutional Neural Networks (CNNs) in learning a suitable low-dimensional latent space of the video data. On the other hand, we avoid the creation of a black-box approach. The proposed method is hierarchical, probabilistic, and explainable. DBNs also allow learning correlations between multiple sensor modalities, i.e., the odometry and video data, making the approach multi-sensorial. Additionally, the method is data-driven: only the video and odometry data of the sensors are provided, and the models are built using them without any prior information, model, or hand-crafted features. Other data-driven fusion methods proposed in the literature, in fact, could need some prior geometrical knowledge (to be learned in advance) that allows one to map the sensor information coming from different modalities, thus requiring an appropriate representation of heterogeneous multi-sensor information. In our case, a camera calibration parameter phase is also not necessary.

In [24], Gibson introduced the concept of "active perception", which not only entails the stimulation from the environment, but also from the attentive actions and reactions of the observer. Therefore, the observation of the environment of the agent cannot be separated from the actions that the agent executes during perception. As a consequence, we decided to combine the visual and odometry information in a single model. This was already introduced in our previous work [25], where we chose to learn the video model through clustering of the odometry information. In this work, instead, we employ both video and odometry data to guide the learning of the video model parameters. In both cases, therefore, the two sources of information are combined. However, the fusion is now performed at a lower level compared to our previous work [25], and this allows us to achieve better localization results.

More specifically, in the proposed method, during training, we filter the odometry data and learn a Variational Autoencoder (VAE) [22] on the video data. Two states are obtained from these two modalities, that are then clustered together, so we learn a DBN model for predicting the evolution of the odometry data, as in [26]. However, a DBN cannot be directly employed on high-dimensional data such as images. Therefore, a method for dimensionality reduction must first be adopted. We choose to use a VAE because it can perform dimensionality reduction on the images and, at the same time, enables us to maintain a probabilistic data representation. The clustering information is passed to the video model learning block, i.e., a CG-KVAE (Cluster-Guided Kalman Variational Autoencoder) [27], an evolution of the KVAE [28]. A Coupled DBN (CDBN) is then obtained from these two models. During testing, only video data is provided, and filtering is performed using a Coupled Markov Jump Particle Filter (CMJPF). The CMJPF is a modification of the MJPF proposed in [26], that is, a Markov Jump Linear System (MJLS) [29,30] composed of a set of Kalman Filters (KFs) [31] and a Particle Filter (PF) [32]. The CMJPF allows us to perform localization, while simultaneously enabling anomaly detection.

Furthermore, we examine how to choose an appropriate clustering, to avoid the problems of over-clustering and under-clustering. We propose an approach based on the covariance of the video and odometry states in each cluster. In other words, we propose a new approach to join video and odometry data through clustering, and we introduce a novel clustering optimization method. These two aspects are fundamental in an explainable self-aware method, since the clustering results constitutes the vocabulary of the model. Better clusters correspond to better letters (or symbols) for representing the video and odometry information, and thus better explainability. We analyze our modifications from the point of view of localization accuracy and vocabulary explainability. We will show that these changes allow us to improve the localization performance of our system while maintaining a similar computational complexity.

To sum up, our main contributions are: (i) the introduction of an improved VBL method that is data-driven, hierarchical, probabilistic, explainable, and does not require prior models (e.g., camera calibration parameters); (ii) the proposal of an approach to choose an approximate clustering level; (iii) the detailed analysis of the method, focusing on both localization and vocabulary explainability.

The rest of the paper is structured as follows: Section 2 provides an analysis of related work. In Section 3, the proposed method is described. Section 4 outlines the experimental setup, detailing the datasets used and the other methods employed for comparison. Section 5 shows and discusses the results obtained on four real-world datasets. Finally, Section 6 concludes the paper and provides insights into possible future work.

2. Background and related work

2.1. Dynamic Bayesian networks

DBNs [23,33] are a type of Probabilistic Graphical Model (PGM). PGMs combine probability theory and graph theory to model systems in which random variables interact with each other [34]. The nodes in a Bayesian Network (BN) represent random variables, and the links represent conditional dependences between the connected variables (in BNs there is directionality in the connection between variables). DBNs additionally describe how random variables change across time.

Fig. 1 shows an example of a DBN on 3 levels. As described above for BNs, the nodes represent variables (e.g., x_t, $\tilde{z}_t$, $\tilde{S}_t$), and the links conditional dependences between them (e.g., $P(x_t|\tilde{z}_t)$, $P(\tilde{z}_t|\tilde{S}_t)$, $P(\tilde{z}_{t+1}|\tilde{z}_t)$). The low-level variable (x_t) corresponds to sensor observation. Higher-level variables increasingly capture more conceptual information.

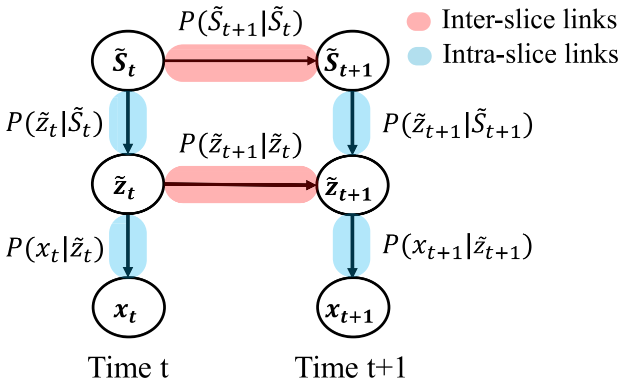


Fig. 1. A 3-level DBN, as the one describing a MJPF [26].

DBNs can be used to represent the filtering process of the state of an agent. The state is a vector that could, for example, contain odometry information of the agent such as its position and speed, or a compressed representation of what the agent is seeing in the environment. It is important to note that the hidden state is not directly observable but is obtained from observations. When dealing with odometry data from a vehicle, the observations could contain position, velocity, acceleration, and higher-level derivatives. Tracking the state of an object means predicting how the state evolves and correcting this prediction through the newly obtained observations (see also [35]). This process can be performed through two steps: *prediction* and *update*. In the prediction phase, the state $\tilde{z}_t$ of an object at time t is estimated based on the transition probability $P(\tilde{z}_t|\tilde{z}_{t-1})$ and given all the observations until step $t-1$ ($X_{1\ldots t-1}$) [36]:

$$P(\tilde{z}_t|X_{1\ldots t-1}) = \int P(\tilde{z}_t|\tilde{z}_{t-1})P(\tilde{z}_{t-1}|X_{1\ldots t-1})dx_{t-1} \tag{1}$$

As it can be observed, a marginal is calculated over all the possible values of the state at time $t-1$.

When the new observation x_t is obtained, the update phase is performed, i.e., the prediction is corrected based on the observation:

$$\begin{aligned} P(\tilde{z}_t|X_{1\ldots t}) &= P(\tilde{z}_t|X_{1\ldots t-1}, x_t) && (2)\\ &= \alpha P(x_t|\tilde{z}_t, X_{1\ldots t-1})P(\tilde{z}_t|X_{1\ldots t-1}) && (3)\\ &= \alpha P(x_t|\tilde{z}_t)P(\tilde{z}_t|X_{1\ldots t-1}) && (4)\end{aligned}$$

In the first step, the evidence $X_{1\ldots t}$ is split into two parts. In the second step, Bayes law is applied: $P(x|y) = \frac{P(y|x)P(x)}{P(y)}$. In the last step, the Markov property is used [36]. The parameter α represents a normalizing constant introduced such that the probabilities sum to 1. At a time instant t, $P(\tilde{z}_t|X_{1\ldots t-1})$ is called the *prior*, $P(x_t|\tilde{z}_t)$ is the *likelihood*, and $P(\tilde{z}_t|X_{1\ldots t})$ the *posterior* (or *posterior distribution*). The objective of Bayesian filtering is to find the posterior. At the next time instant $t+1$, the posterior in t serves as the new prior.

Different types of filters can be employed for performing state estimation. For example, a filter such as a MJPF [26] can be represented as a DBN (see Fig. 1). In this case, x_t corresponds to a sensor information, $\tilde{z}_t$ to a hidden state, and $\tilde{S}_t$ to a discrete semantic representation. We can distinguish two types of links in the figure: inter-slice links and intra-slice links. Inter-slice links define relationships between variables at distinct time instants, whereas intra-slice links define relationships between variables at the same time instant. DBNs of distinct sensors can also be connected to obtain a Coupled DBN (CDBN). This paper proposes a CDBN coupling an odometry and a visual sensor.

2.2. Self-aware autonomous systems

The proposed work is integrated into a framework [21] for self-aware autonomous systems. It is well known that the concept of awareness relates to the knowledge that an agent, whether biological or artificial, has of the surrounding environment. In contrast, self-awareness is associated with the consciousness that the agent has of its own state. Self-awareness is a broad concept derived from neuroscience and several interpretations of it exist [37]. Lewis et al. [38] state that "self-awareness is concerned with the availability, collection, and representation of knowledge about a system, by that system", whereas Morin et al. [39] define self-awareness as the "capacity of becoming the object of one's own attention". In their book on self-aware computing systems [40], the authors define self-aware computing systems as systems with two main characteristics. Firstly, they "learn models capturing knowledge about themselves and their environment (such as their structure, design, state, possible actions, and runtime behavior) on an ongoing basis". Secondly, they "reason using the models (e.g., predict, analyze, consider, and plan) enabling them to act based on their knowledge and reasoning (e.g., explore, explain, report, suggest, self-adapt, or impact their environment) in accordance with higher-level goals, which may also be subject to change". Building on these definitions, we can assert that a self-aware agent focuses not only on its external environment but also on the evolution of its own state. It has the capability to perceive and adapt to its own state as well as its surroundings. It recognizes its role within a system, is capable of adjusting its actions and decisions, and continuously refines its behavior to better deal with dynamic and changing scenarios. In this work, we mainly focus on the perception aspect of a self-awareness agent (as opposed to the decision-making aspect), with the goal of using the data collected from various sensors to create an internal representation of both the environment and the agent's own state. In this context, Hierarchical Dynamic Bayesian Networks allow us to learn appropriate models from these proprioceptive and exteroceptive signals acquired by our agent (see also [21]).

The aforementioned works [18–20] highlight the importance for a framework to have three characteristics. Firstly, a hierarchical structure, which mirrors the human brain in combining low-level sensor information into increasingly abstract concepts. Secondly, a probabilistic nature, to consider the uncertainty present in the world. Thirdly, the correlation of information from different sensor modalities, such as proprioceptive sensors, which allow the agent to be aware of its state, and exteroceptive sensors, which lead to situational awareness. The framework in [21] adopts DBNs to achieve these objectives. DBNs, indeed, allow us to obtain the three requirements listed above: they structure *stochastic* variables in a *hierarchy* and model correlations between variables related to *different types of sensor data*.

Note that [21] considers several steps for the development of incremental self-aware systems, that we can summarize as: the *initialization* of the system with a first model (1); the *memorization* of the learned information (2); the *prediction* of the future states based on the learned model (3); the *detection of anomalies* (4); the *creation of a new model* leveraging the found abnormalities (5); the *decision making* process of selecting the actions to perform, using the observations and the available models (6). Anomaly detection is, therefore, necessary to detect previously unseen objects or unknown motions and incrementally incorporate them into the model. This would allow us to perform continual learning. In this paper, which considers the case of a fixed environment for surveillance applications, we perform steps up to point (4). The considered vehicle is still not autonomous, as no actions are taken.

2.3. Visual-based localization (VBL)

The main objective of this work is to localize a moving agent in a *known* environment, only using camera data. By defining the environment as "known", we hypothesize to be provided with sequential synchronous visual and odometric data during training. Our work is, therefore, related to the fields denoted as *Visual-based Localization (VBL)*, *Visual Odometry (VO)*, and *Visual Simultaneous Localization and Mapping (V-SLAM)*.

VBL represents the task of retrieving a camera's pose given an image or a sequence of images from the camera itself [7]. The survey in [7] proposes a division of VBL methods into two main groups: *indirect/retrieval-based methods* and *direct methods*. *Indirect methods* map images of the training set in a chosen feature space, creating a database of descriptors for already seen images. When a query image is provided during the online execution of the method, the algorithm looks for the best match in the dataset. *Direct methods* directly recover the pose of the query according to a known reference. Pose regression approaches are one sub-type of this second category. CNNs can be used to obtain the pose from images: the first example of a CNN-based regression architecture for VBL is PoseNet [8], where camera and odometry data are fused during the training phase allowing the CNN parameters to be learned. This architecture is derived by modyfing a previously proposed deep neural network architecture called GoogLeNet [41]. In [42], several retrieval-based and regression-based architectures are tested on a real dataset of a supermarket, some of them using the PoseNet architecture. The authors in [43] also enhance the performance results of PoseNet by considering various loss functions that incorporate scene geometry to learn the camera pose. Other DL architectures are also exploited in VBL in combination with CNNs, such as spatial LSTM network to capture contextual information [11]. A multi-task CNN has also proposed in [44] to address orientation and translation interactions. Retrieval-based methods are typically more accurate than regression-based methods [45]. However, they are computationally less efficient, and their memory requirement increases linearly with the number of training images. Most VBL methods rely on a single image to estimate position and orientation. Whereas image sequences are not always available, they can increase accuracy when they are accessible. In our method, similarly to [46–49], we assume the presence of sequential data, which enables us to track the moving agent. In the literature, three additional VBL methods have been proposed [42], which we will also consider for performance comparison: an Image Retrieval (IR) method (denoted as IR-IV3), an IR method that takes into account the sequence of images (IR-TC-IV3), and one REG method (REG-PNET-RGB-POS-IV3). IR-IV3 executes IR based on features from Inception-v3 [50] pre-trained on ImageNet. IR-TC-IV3 adds a temporal constraint to the previous one. REG-PNET-RGB-POS-IV3 performs regression using PoseNet trained starting from an Inception-v3 backbone pre-trained on ImageNet. Other possible configurations, such as IR-PNET-VGG16 and REG-SVR-PNET-RGB-VGG16, have also been proposed, which use a VGG16 network as backbone (see also Section 5).

In contrast, the objective of VO is to estimate the *relative motion* of a camera using a sequence of images [51]. Consequently, VO is susceptible to estimation drift over time. It is important to note that VO is typically employed in scenarios where the environment map is not available, differently from VBL and our proposed objective.

Finally, in V-SLAM, an agent localizes itself in an *unknown* environment and simultaneously builds a map of it [51].

This paper proposes a method that combines characteristics of the three general methods discussed above in the specific case of video surveillance localization in a known environment. Our method incorporates the following features:

- Known Environment. Similar to what happens in retrieval-based and regression-based VBL methods, the environment is known. This means that both video and positional data are available during training. It is a limit compared to V-SLAM, which builds a map online.
- Absolute Positioning. Our method, like VBL, aims to estimate the absolute positioning.
- Sequential Visual Data. As in VO and V-SLAM, sequential visual data is leveraged, achieving a more robust localization than the one obtained by single-image VBL.
- Generative Map. During training, we build a generative map of the environment, which is not continuous but sparse, as it is composed of a set of clusters. Clustering enables a hierarchical representation and higher explainability. Each cluster allows for predicting positions and their changes during a navigation task.

Comparisons with several VBL methods (also tested in [42]) are provided in Section 5.

3. Proposed approach

The proposed method is divided into two parts: an offline training phase and an online testing phase. During the training phase, the DBN models for odometry and video data (First Person Viewpoint) are extracted. So, as input we consider both odometry and video data coming from a set of training trajectories, while as output of our learning model we obtain the KVAE vocabulary and odometry vocabulary. With "vocabulary" we henceforth define the final parameters computed as a result of the training phase of the KVAE and the odometry model. In the testing phase, the video data are provided, and the learned model parameters (i.e., the learned vocabularies) are used to predict the values of the odometry data (extracting anomalies as well). Fig. 2 summarizes both the training and the testing phases. Fig. 3 compares, on a higher level, the proposed approach with [25]. In the first version (see Fig. 3a), the odometry state is extracted by filtering the training positions, and is then used to perform clustering. For each cluster, we can thus compute a distance (i.e., Mahalanobis distance) between each datapoint and the cluster. All these distances are then given as input to the video training block to guide its learning phase. In fact, the matrices that allow us to correlate the video latent states are combined using a vector of probabilities, which is derived from the odometry distances. This way, the odometry vocabulary guides the learning of the KVAE, which leads to the extraction of the video vocabulary (images are not direcly used during the clustering phase). In the second version (see Fig. 3b), the clustering is performed using both the odometry states and the video latent states obtained from the VAE bottleneck. In Section 4.2 we expand on the differences between these two approaches.

3.1. Training phase: Learning the odometry and video models

Firstly, the models are learned through several consecutive steps (see Figs. 2 and 3).

STEP 1. Null Force Filter. A set of odometry observations $\{x_t^o\}_{t=1\ldots\tau}$ is provided, being τ the number of time instants considered in the training set. We define "odometry observations" as the positional measurements of the agent along the x, y, and z axes. From each odometry sensor observation x_t^o, a Generalized State (GS) [18] is then computed. The GSs contain the state and its first-order time derivative:

$$\tilde{z}_t^o = [z_t^o, \dot{z}_t^o] \tag{5}$$

So, we obtain $\tilde{z}_t^o$ as the output of x_t^o through a KF predicting that no motion is present. Such a KF is called a Null-Force Filter (NFF) [52]. The states are correlated to the observations supposing a linear relationship of the following type:

$$\dot{z}_t^o = \frac{x_t^o - H z_{t-1}^o}{\Delta t} + v_t \tag{6}$$

where H is an identity matrix and v_t a zero-mean Gaussian distribution with a certain covariance R^o.

STEP 2. Extraction of the VAE's latent states (this step can be performed in parallel with STEP 1). We first train the VAE by using the set of training images $\{x_t\}_{t=1\ldots\tau}$. The images are given as input to a VAE through the encoder $q_\phi(a_t|x_t)$. The VAE is trained to reconstruct the images through the decoder $p_\theta(x_t|a_t)$. Then, we input again the training images to the VAE and obtain a set of latent features described by $\{a_t\}_{t=1\ldots\tau}$.

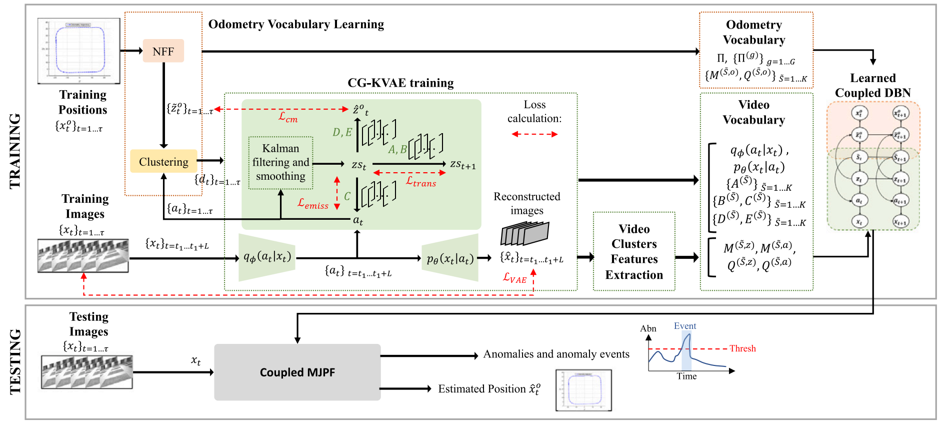
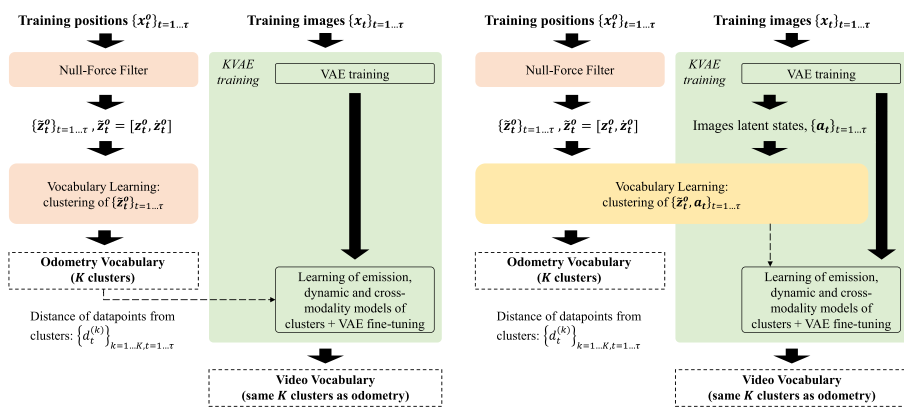


Fig. 2. Training and testing phase of the method. During the training phase, the odometry is first filtered and a VAE is built on the video data. Then, clustering is performed using both odometry and video states. A modified KVAE is learned, leveraging the information from the joint odometry-video clustering. The latent states obtained from the training images are used to extract the features of the video clusters. A CDBN is built from the odometry and video vocabulary. During the testing phase, a CMJPF, constructed from the CDBN, extracts anomalies and performs localization.



(a) First version. (b) Second Version.

Fig. 3. The proposed method (b) compared to [25] (a). In (a), the odometry state is extracted by filtering the training positions, and is then used to perform clustering. The odometry vocabulary guides the learning of the KVAE, which leads to the extraction of the video vocabulary. In (b), the clustering is performed using both the odometry states and the video latent states obtained from the VAE bottleneck.

STEP 3. Extraction of the vocabulary of clusters. Clustering is performed on the joint odometry-video state $j_t = [\tilde{z}^o_t, a_t]$. It is worth noting, however, that $\tilde{z}^o_t$ and a_t have different dimensionalities: $\tilde{z}^o_t$ can be either 4-dimensional or 6-dimensional, whereas a_t can have much higher dimensionality (as reported in the results section, we chose to vary it in the range $[16, 128]$, depending on the complexity of the dataset). We apply the Growing Neural Gas (GNG) algorithm [53], where a distance is calculated between each j_t and the temporary clusters proposed by the method. The obtained cluster centers are moved closer to the input data (and the nearest cluster to each input data point is the one that is moved with the greatest step). The used distance is a weighted MSE that is applied after performing standardization on the state. The weights of the different components are set so that equal importance is given to both the odometry and video parts. For example, if the odometry GS is 4-dimensional, and the video latent state is 16-dimensional, each odometry component is given a weight of $\gamma = \frac{1}{4}$, and each video component is given a weight of $\delta = \frac{1}{16}$.

After the clustering step, the odometry vocabulary components are extracted. More specifically, for each cluster $\tilde{S}$, with $\tilde{S} = 1 \dots K$, we obtain: the cluster mean $M^{(\tilde{S},o)}$, the cluster covariance $Q^{(\tilde{S},o)}$, a transition matrix Π defining the overall probabilities of moving from one cluster to the others, and a set of Temporal Transition Matrices (TTMs) $\Pi^{(g)}$ with the probabilities of moving from one cluster to the others, given that g time instants have been spent in the current cluster.

We also extract part of the video model vocabulary: the cluster means $M^{(\tilde{S},a)}$ and covariances $Q^{(\tilde{S},a)}$, related to the a_t state.

STEP 4. Cluster distances calculation. The distance $d_t^{(\tilde{S})}$ between each state j_t and the different clusters is calculated. It is the sum of two distances:

$$d_t^{(\tilde{S})} = \gamma d_t^{(\tilde{S},o)} + \delta d_t^{(\tilde{S},a)}, \tag{7}$$

where $d_t^{(\tilde{S},o)}$ is a probabilistic distance (e.g., Bhattacharyya D_B) between each odometry GS $\tilde{z}^o_t$ and each odometry cluster:

$$d_t^{(\tilde{S},o)} = D_B((\tilde{z}^o_t, \Sigma^o_t), (M^{(\tilde{S},o)}, Q^{(\tilde{S},o)})) \tag{8}$$

where Σ^o_t represents the covariance of $\tilde{z}^o_t$.

In contrast, $d_t^{\tilde{S},a}$ is the probabilistic distance between each video state $\tilde{a}_t$ and each video cluster:

$$d_t^{(\tilde{S},a)} = D_B((a_t, \Sigma^a_t), (M^{(\tilde{S},a)}, Q^{(\tilde{S},a)})) \tag{9}$$

where Σ^a_t is the covariance associated with a_t.

A set of $\tau * K$ distances is thus obtained. We define as $\boldsymbol{d}_t$ the set of K distances at each time step t, i.e., a vector containing the distances between the data point at time t and each cluster. These distances are given as input to the video training block to guide its learning phase. From these distances, it is possible to estimate the probability $\alpha_t^{(\tilde{S})}$ that the point at time t belongs to cluster $\tilde{S}$:

$$\alpha_t^{(\tilde{S})} = softmax\left(\frac{1}{d_t^{(\tilde{S})} + \epsilon} * \frac{1}{m}\right), \tag{10}$$

where m is a temperature, and ϵ is a small positive value added to avoid the denominator being zero. A low value of m leads to a multinomial probability distribution peaked around the nearest cluster, whereas a high value of m leads to a flatter multinomial probability distribution. This expression corresponds, therefore, to a softmax with temperature [54]. Across video training epochs, the value of m can be annealed.

We compute the probability vector $\boldsymbol{\alpha}_t$, which includes the value of $\alpha_t^{(\tilde{S})}$ for each cluster $\tilde{S}$.

STEP 5. Learning of the CG-KVAE. For each time instant t, an image x_t is passed through the encoder q_ϕ of a VAE, to obtain a latent state a_t. A second latent state, z_t, is also found through filtering, with a smaller dimension compared to a_t [28]. Whereas a_t captures the content information of the images, z_t focuses on the dynamics between images. The two latent states are correlated through a pseudo-observation model defined as:

$$a_t = \sum_{i=1}^{K} \alpha_t^{(i)} C^{(i)} z_t + v_t \tag{11}$$

The dynamical model connects z_t values at consequent times:

$$z_{t+1} = \sum_{i=1}^{K} \alpha_t^{(i)} A^{(i)} z_t + \sum_{i=1}^{K} \alpha_t^{(i)} B^{(i)} + \omega_t \tag{12}$$

where v_t and ω_t represent noises with zero-mean Gaussian distribution. The matrices $\{C^{(i)}\}_{i=1...K}$ and $\{A^{(i)}\}_{i=1...K}$ represent a set of pseudo-observation models and transition models, respectively. They are combined through the vector of probabilities $\boldsymbol{\alpha}_t$. The original KVAE method [28] also considered a set of matrices $\{B^{(i)}\}_{i=1...K}$, which multiply an optional control vector. Instead, in this paper, we consider a set of matrices B which provide an addition term. Through these models, we can loosely associate with a cluster a different way of selecting features from the VAE bottleneck (through the $C^{(i)}$ matrices) and different image dynamics across time ($A^{(i)}$ matrices).

To train the KVAE, at each loop of each epoch, we consider a batch of image sequences of length L, i.e., $\{x_t\}_{t_1...t_1+L}$. This batch is passed through the encoder $q_\phi(a_t|x_t)$ of the KVAE to obtain the latent states $\{a_t\}_{t_1...t_1+L}$. Kalman filtering and smoothing are then performed on these states, using the observation model in Eq. (11) and the prediction model in Eq. (12). From this point onwards, we define z_t as the low-dimensional latent state extracted only using filtering, and with zs_t the one after applying the smoothing process too.

In [28], after performing Kalman smoothing, four main losses are used to optimize the model: (i) the traditional reconstruction loss between x_t and its VAE reconstruction $\hat{x}_t$; (ii) the Kullback Leibler Divergence term of the VAE; (iii) a transition loss $\mathcal{L}_{trans}$ enforcing the learning of the transition models A and B; (iv) an emission loss $\mathcal{L}_{emiss}$ enforcing the learning of the emission models C. The first two losses compose the VAE loss $\mathcal{L}_{VAE}$.

Here, adding the localization, we aim to correlate a value of the latent state z_t (or better, the smoothing version zs_t of it) within a cluster to its corresponding odometry state $\tilde{z}_t^o$. Thus, we also train a set of matrices $\{D^{(i)}\}_{i=1...K}$ and $\{E^{(i)}\}_{i=1...K}$, such that:

$$\hat{z}_t^o = D^{(\tilde{S}_t)} zs_t + E^{(\tilde{S}_t)} + M^{(\tilde{S}_t,o)} + \omega_t, \tag{13}$$

where $\hat{z}_t^o$ is the estimated odometry state at time instant t and ω_t is the residual error.

To ease the training, we make the estimation of $\tilde{z}_t^o$ start from the odometry cluster centers $M^{(\tilde{S}_t,o)}$. Consequently, we add a *cross-modality loss* $\mathcal{L}_{cm}$:

$$\mathcal{L}_{cm} = \sum_{\tilde{S}=1}^{K} \alpha_t^{(i)} \left\| \hat{z}_t^o - \tilde{z}_t^o \right\|^2, \tag{14}$$

using five losses in total. Fig. 2 displays these five loss calculations with red dotted lines.

After the KVAE training, for each cluster $\tilde{S}$, we extract the final cluster centroids ($M^{(\tilde{S},a)}$, $M^{(\tilde{S},z)}$) and the cluster covariances ($Q^{(\tilde{S},a)}$, $Q^{(\tilde{S},z)}$) of both latent states a_t and z_t.

3.2. Training phase: Learning the coupled dynamic Bayesian network

The parameters of the two DBNs are learned (see Fig. 4). Together they form the CDBN. The features representing the links are shown in red.

In the *odometry DBN*, the observation matrix H links the observation x_t^o and the GS $\tilde{z}_t^o$. The probability of $\tilde{z}_t^o$ belonging to a cluster $\tilde{S}_t$ is described by the cluster's mean value $M^{(\tilde{S}_t,o)}$ and covariance $Q^{(\tilde{S}_t,o)}$. The evolution of $\tilde{z}_t^o$ can be predicted through the derivative part of $M^{(\tilde{S}_t,o)}$. In the *video DBN*, the VAE works as an observation model that connects x_t and a_t, whereas the matrices C describe a pseudo-observation model (Eq. (11)). The matrices A define the prediction model over z_t (Eq. (12)). As in the odometry DBN, the probability of z_t belonging to a cluster $\tilde{S}_t$ depends on the mean value $M^{(\tilde{S},z)}$ and covariance $Q^{(\tilde{S},z)}$ of the cluster. A direct link between a_t and $\tilde{S}_t$ is similarly defined through $M^{(\tilde{S},a)}$ and $Q^{(\tilde{S},a)}$.

The odometry and video DBNs are combined at the cluster level, as the same clustering phase is shared. A link is added between the video state z_t and the odometry GS $\tilde{z}_t^o$, represented by Eq. (13). The prediction of the next cluster (i.e., the prediction step at the S level) is carried out through the transition matrix Π and the temporal transition matrices $\Pi^{(g)}$.

3.3. Training phase: Clustering optimization

A major problem when performing GNG consists in deciding when to stop the clustering process. GNG is a clustering algorithm where the number of clusters does not need to be defined a priori. Clusters are added as the algorithm progresses, so a condition to stop the clustering needs to be defined [55]. In this section, we propose to analyze the characteristics of the obtained clustering graphs as the algorithm progresses, deciding which graph to choose at the end. $\mathcal{L}$ levels of clustering are obtained (or $\mathcal{L}$ graphs). In general, we would like to obtain clusters with the following main characteristics:

- the velocity prediction inside the cluster should be precise. In this way, the odometry prediction performed by the CDBN link $P(\tilde{z}_{t+1}^o|\tilde{z}_t^o)$ will tend to be more accurate;
- the image content in the cluster should be homogeneous. This choice allows for a better first estimate of the cluster probabilities from the video latent state;
- fewer clusters are desirable, so to have a simpler model which requires less memory to be stored.

Therefore, clusters should extend on an area as big as possible, while guaranteeing an accurate velocity prediction and image content homogeneity. The same consideration was made in [56,57] (except for the second point, as video data were absent). In this paper, we propose a similar but more general method. First, $\mathcal{L}$ clustering levels are computed. In each clustering level, the covariance over the positional, velocity, and video latent state information is calculated. We express these three covariances at clustering level l with the following symbols: $Q^{(\tilde{S},p,l)}$, $Q^{(\tilde{S},v,l)}$, and $Q^{(\tilde{S},a,l)}$. The eigenvalues of each of the three covariances are calculated, i.e., $eig(Q^{(\tilde{S},p,l)})$, $eig(Q^{(\tilde{S},v,l)})$, and $eig(Q^{(\tilde{S},a,l)})$. This operation is repeated for each cluster in the graph. In [57], quality metrics were calculated on the position and velocity covariances by averaging their values. In this paper, instead, we calculate the eigenvalues of the covariances, as they are more meaningful. The mean value of the eigenvalues, over the covariance dimension, and over the different clusters, is found for all three cases. We perform this

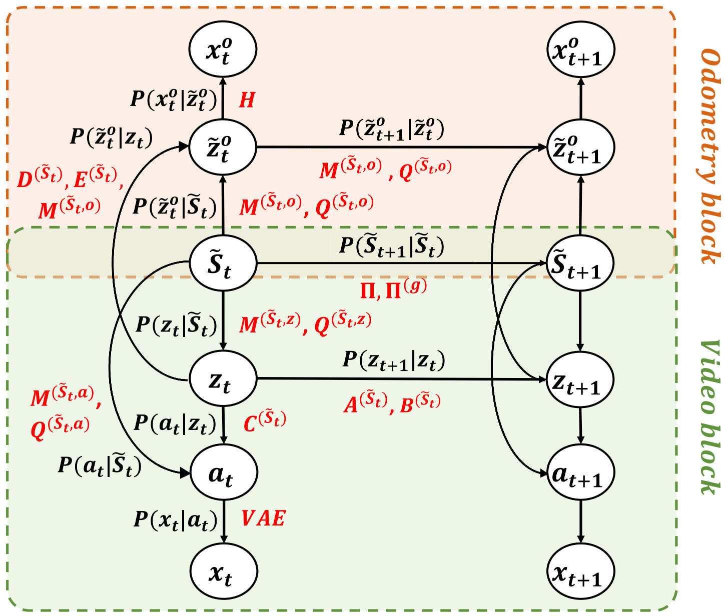


Fig. 4. Learned Coupled Dynamic Bayesian Network (CDBN).

calculation for each of the $\mathcal{L}$ graphs. Therefore, in the end, three quality metrics are obtained for each of the $\mathcal{L}$ levels, i.e., one related to the positional covariance (q^p), one to the velocity covariance (q^v), and one to the image content covariance (q^a). The definition of q^p is summarized by the following formula (the formulas for q^v and q^a are analogous):

$$q_l^p = \frac{\sum_{k=1}^{K} \sum_{d=1}^{D^p} eig^{(d)}(Q^{(k,p,l)})}{K * D^p} \tag{15}$$

In this equation, $eig^{(d)}$ refers to the dth eigenvalue, and D^p to the dimensionality of the positional covariance.

To obtain the three desired characteristics listed above, q_l^v and q_l^a should be low, and q_l^p should be high.

In our architecture, we suppose that the $\mathcal{L}$ clustering levels are such that, at the $\mathcal{L}$-th level, the q^p, q^a, and q^v metrics have reached the regime and do not significantly decrease any further. We normalize q^p, q^a, and q^v, and set a threshold from the regime for each of them. For example, the same threshold Th_r can be set for all three cases. The clustering level where each of the three quality metrics reaches the threshold is found. The number of clusters corresponding to these three levels is averaged. We select the closest one to this average as the optimum level, among the $\mathcal{L}$ available levels.

By using thresholds, instead of the intersection between quality metrics, more generalizable and meaningful results can be obtained. A remaining problem is choosing the threshold Th_r. However, the method that we propose aims to obtain an approximate clustering level, one that does not lead to under-clustering or excessive over-clustering. A range of threshold values should, therefore, lead to acceptable results. To explain further, let us suppose to set the threshold to 0.25. This means that the distance between the regime and the threshold is 25% of the distance between the highest value of the quality metric (typically the value for the first clustering level). Therefore, the threshold is set like this to favor a high velocity and video latent state covariance, as desired.

3.4. Testing phase: Coupled Markov jump particle filter

During the online (or testing) phase, at each time instant, only the video data x_t is given, and we derive the vehicle's localization using the information of the CDBN (see Fig. 4). The traditional MJPF in [26] was built on a single sensor model. A CMJPF is created by coupling the models of two sensors, where just one is observed (see also Fig. 5). In other words, because of the hierarchical structure of the CMJPF, we can compute the anomalies at each level of this architecture (by comparing, at each level, the prediction with the message coming from the observation). In addition, due to connections built during training between the odometry DBN and video DBN, we are able to derive an estimate of the position of the agent during testing (only using video data).

The MJPF uses a set of KFs at the state level and a PF at the cluster level. A PF maintains a set of N particles during filtering, with each particle representing a hypothesis and being associated with a mean, a covariance, and a cluster at each time step. At each instant, two phases (prediction and update) are executed. These phases, at the state level, use the state prediction and update models related to the cluster of that particle. The cluster-level prediction employs the transition matrices Π and $\Pi^{(g)}$. During the update phase, particles undergo resampling to eliminate the least probable hypotheses.

The proposed CMJPF is based on the original MJPF but includes significant changes. Firstly, each particle is characterized by two means (one for the odometry GS and one for the video state), two covariances, and a cluster. Secondly, the prediction and the update at the state level are split into two steps, as they are performed both for the odometry GS and for the video state. Unlike a classic odometry MJPF, in the proposed CMJPF, we do not have the odometry observation. Consequently, we note that the odometry "update" is not an actual update, as the positions are not observed. This update is in fact a prediction performed through the vocabulary of the video model, and using both D and E matrices.

In the following paragraphs, we thoroughly describe the steps executed by the algorithm at each time instant t (see also Alg.).

STEP 1. Particle-independent calculations. At each instant t, we perform a set of operations that do not depend on the N particles. The image x_t is passed through the VAE's encoder q_ϕ to extract the latent state a_t. The Bhattacharyya distance D_B is calculated between each video state and each video cluster: $d_{v,t}^{(\tilde{S})} = D_B((a_t, \Sigma_t^a), (M^{(\tilde{S},a)}, Q^{(\tilde{S},a)}))$. Finally, a vector $\boldsymbol{\alpha}_{v,t}$ of probabilities is obtained from the set of K values $d_{v,t}^{(\tilde{S})}$ using Eq. (10). It defines how probable it is to be in each cluster, based on the video observation. **STEP 2. Initialization of the particles (executed only once).** The particles of the CMJPF undergo initialization at the first instant. We define, for each particle, the corresponding cluster, the mean, and the covariance of both the odometric and video states. The cluster $\tilde{S}_{t=1,n}$, of particle n, is sampled using the

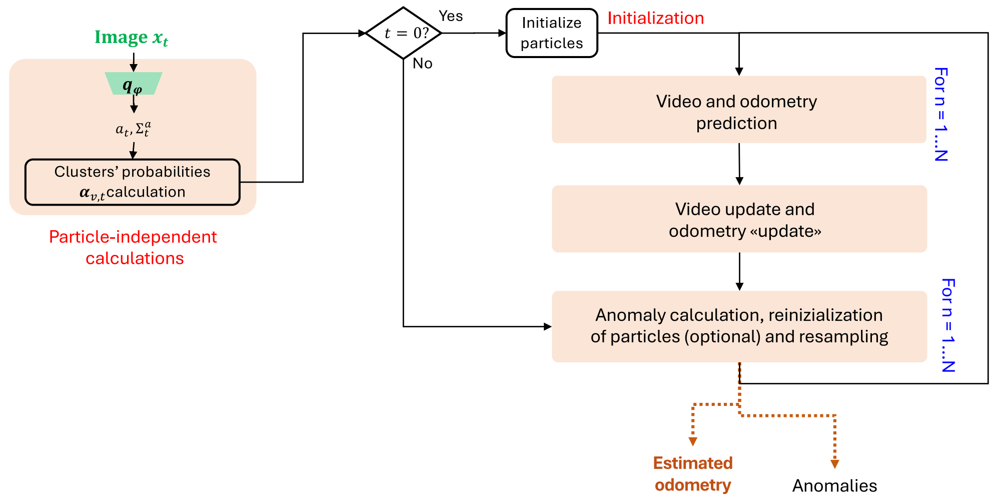


Fig. 5. General block scheme of the testing phase.

Algorithm 1 Method for obtaining odometry from video.

Input: Learned DBN vocabulary
$first\_resampling\_done = False$
for $t = 1$ **to** τ ← *Time evolution* **do**
  PARTICLE-INDEPENDENT CALCULATIONS:
   x_t ← Image at t
   Get the latent state of images using VAE's encoder:
   $a_t = q_\theta(x_t)$ ← Image latent state
   Find distances of video encodings from video clusters:
   $d_{v,t}^{(\tilde{S})} = D_B((a_t, \Sigma_t^a), (M^{(\tilde{S},a)}, Q^{(\tilde{S},a)}))$, for $\tilde{S} = 1 \dots K$
   Find video cluster probabilities $\alpha_{v,t}^{(\tilde{S})}$ from the distances $d_{v,t}^{(\tilde{S})}$, for $\tilde{S} = 1 \dots K$, using Eq. (10).
   if $t == 1$ ← *Initial iteration* **then**
    INITIALIZATION OF THE PARTICLES:
    for $n = 1$ **to** N ← *Particles* **do**
     Initialize cluster assignment $\tilde{S}_{t=1,n}$ of particle n using cluster probabilities $\alpha_{v,t}^{\tilde{S}}$, with $\tilde{S} = 1 \dots K$.
     Initialize n-th particle video state as:
     $z_{t=1,n} \sim \mathcal{N}(M^{(\tilde{S}_{1,n},z)}, Q^{(\tilde{S}_{1,n},z)})$.
     Initialize n-th particle odom. state as:
     $\tilde{z}^o_{t=1,n} \sim \mathcal{N}(M^{(\tilde{S}_{1,n},o)}, Q^{(\tilde{S}_{1,n},o)})$.
  else
    UPDATE:
    for $n = 1$ **to** N ← *Particles* **do**
     Perform the video update: $z_{t|t,n}, \Sigma_{t|t,n} = KF\_update(z_{t|t-1,n}, \Sigma_{t|t-1,n}, C^{(\tilde{S}_{t,n})}, a_t, R^v)$
     Get the odometry estimation from video: $\hat{z}^o_{t,n} = D^{(\tilde{S}_{t,n})} z_{t|t,n} + E^{(\tilde{S}_{t,n})} + M^{(\tilde{S}_{t,n},o)}$
     Perform the odometry 'update': $\tilde{z}^o_{t|t,n}, \Sigma^o_{t|t,n} = KF\_update(\tilde{z}^o_{t|t-1,n}, \Sigma^o_{t|t-1,n}, \hat{z}^o_{t,n}, R^o)$
    Calculate the anomalies. Based on them, assess if restarting particles is necessary.
    Reweight the particles.
    if *Particles Restarting necessary* **then**
     Resample, restart a subset of particles and set $first\_resampling\_done$ == True.
    else if $first\_resampling\_done == False$ *and* $N_{eff} < N_{th,l}$ **then**
     Resample particles and set $first\_resampling\_done$ == True.
    else if $N_{eff} < N_{th,h}$ **then**
     Resample particles.
  PREDICTION:
  for $n = 1$ **to** N ← *Particles* **do**
   Predict the next cluster $\tilde{S}_{t+1,n}$ from $\tilde{S}_{t,n}$ using Π, $\Pi^{(g_{t,n})}$.
   Predict the video state: $z_{t+1|t,n}, \Sigma_{t+1|t,n} = KF\_v\_pred(z_{t|t,n}, \Sigma_{t|t,n}, A^{(\tilde{S}_{t,n})}, Q^v)$.
   Predict the odometry state: $\tilde{z}^o_{t+1|t,n}, \Sigma^o_{t+1|t,n} = KF\_pred(z^o_{t|t,n}, \Sigma^o_{t|t,n}, M^{(\tilde{S}_{t,n},o)}, Q^{(\tilde{S}_{t,n},o)})$.
Output: Anomalies and $x^o_{t|t,n}$ with $n = 1 \dots N$

multinomial distribution described by $\alpha_{v,t=1}$. The mean and covariance of the video and odometry states of the particle are then defined by sampling from the Gaussian distributions $\mathcal{N}(M^{(\tilde{S}_{1,n},z)}, Q^{(\tilde{S}_{1,n},z)})$ and $\mathcal{N}(M^{(\tilde{S}_{1,n},o)}, Q^{(\tilde{S}_{1,n},o)})$, respectively.

STEP 3. Prediction phase. For every time step, the subsequent cluster, odometry state, and video state are predicted for each particle using the learned prediction models. The prediction is performed at the cluster level by combining the transition matrix Π and the TTM $\Pi^{(g)}$, where g is the time spent in the considered cluster. For the video state prediction, for each particle, we use the A matrix of the associated cluster, i.e., $A^{(\tilde{S}_{t,n})}$, in a standard KF prediction step. We denote this step (in Algorithm) as $KF\_v\_pred(z_{t|t,n}, \Sigma_{t|t,n}, A^{(\tilde{S}_{t,n})}, Q^v)$. This because the standard KF prediction equation is used on the video state, starting from the updated mean $z_{t|t,n}$ and covariance $\Sigma_{t|t,n}$ at time instant t, and using the prediction matrix $A^{(\tilde{S}_{t,n})}$ with the prediction uncertainty Q_v. For the odometry phase, a KF predicts that the object will move with the mean velocity of the cluster (i.e., using the second part of the vector $M^{(\tilde{S}_{t,n},o)}$):

$$\tilde{z}^o_{t+1|t,n} = A^o \tilde{z}^o_{t|t,n} + B^o U^{(\tilde{S}_{t,n})} + w^o_t \tag{16}$$

where $A^o \tilde{z}^o_{t|t,n}$, takes the GS information from $\tilde{z}^o_{t|t,n}$ and makes null its time derivatives. $B^o U^{(\tilde{S}_{t,n})}$ encodes the time derivative information at time t. $U^{(\tilde{S}_{t,n})}$ contains the derivative-related part of the variable $M^{(\tilde{S}_{t,n},o)}$. The variable ω^o_t is a zero-mean Gaussian distribution representing the noise of the model.

STEP 4. Update phase. At all time steps, except for the first one, the video and odometry states are updated. The video update is carried out first. For each particle, the C matrix of the associated cluster is used, i.e., $C^{(\tilde{S}_{t,n})}$. We denote in Alg. this step as $KF\_update(z_{t|t-1,n}, \Sigma_{t|t-1,n}, C^{(\tilde{S}_{t,n})}, a_t, R^v)$ because of the use of the traditional KF update equation, starting from the predicted mean $z_{t|t-1,n}$ and covariance $\Sigma_{t|t-1,n}$ at time instant $t-1$, with the pseudo-observation matrix $C^{(\tilde{S}_{t,n})}$; the pseudo-observation is the state a_t, with uncertainty R^v.

To perform the odometry state update, an observation of the vehicle's position would be required, but is unavailable. To address this limitation, the supplemental matrices D and E were introduced. These matrices allow the derivation of an approximate estimate $\hat{z}^o_{t,n}$ for the odometry state, based on the latent state $z_{t,n}$ (see Eq. (13)). An odometry "update", denoted as $\tilde{z}^o_{t|t,n}$, is then obtained following the conventional Kalman Filter equations. The update process incorporates $\hat{z}^o_{t,n}$ as the odometry observation. Note that this operation is not an actual update but rather a combination of two predictions, one using the odometry prediction model defined by $(M^{(\tilde{S}_t,o)}, Q^{(\tilde{S}_t,o)})$ and the other employing the cross-modality model defined by matrices $(D^{(\tilde{S}_t)}, E^{(\tilde{S}_t)})$. Here, we adopt the name "update" because we combine the two predictions through the standard KF update equations. The image-level variables $\hat{x}^o_{t,n}$ and $x^o_{t|t,n}$, corresponding to $\hat{z}^o_{t,n}$ and $\tilde{z}^o_{t|t,n}$ are then obtained through the observation model in Eq. (6).

STEP 5. Anomaly calculation. Several anomalies can be computed after the update phase, on the different levels of the CDBN. We do not consider them in this paper.

STEP 6. Particles resampling. We reweight the particles by giving higher weight to particles belonging to clusters with a higher likelihood and with a lower anomaly. As a reweighting anomaly, we use $diff_{a,n}$. Particles are resampled when the filter's effective size N_{eff} is below a defined threshold [32]. We propose to use two separate thresholds. The first one, $N_{th,l}$, is used in the initial part of the trajectory. After the first resampling, it is replaced by a second threshold, $N_{th,h}$, which is employed for the rest of the tracking. At the start of a trajectory, the algorithm needs to decide in which location of the overall space the agent is situated and might need more time - and so a lower threshold - to correctly do it. Conversely, a higher threshold can be used for the rest of the tracking. This reasoning does not apply if the vehicle always starts from the same position: in this case, there is no need to use a lower threshold at the beginning of the trajectory, because we approximately know where the vehicle is located.

STEP 7. Particles restarts. Through the anomalies, the algorithm can detect whether its localization prediction is diverging from the actual path that the vehicle is following. If this is the case, it needs to correct its localization.

4. Experimental setup

4.1. The datasets used in our architecture

We evaluated the method on the following datasets.

Icab dataset [58]. In the training set (6,098 frames), a vehicle performs the Perimeter Monitoring of a courtyard. In the abnormal test set (6,558 frames), the vehicle must stop to let a pedestrian cross its path. An issue of the dataset, for performing the localization, is the symmetry of the courtyard on the four sides (see the first four images in Fig. 6). The training data are divided into 4,500 for actual training and 1,598 for validation.

Egocart dataset [42,45]. In the train and test sets (13,360 vs. 6,171 frames), a shopping cart moves in an empty retail store. We divide the 13,360 images into 10,259 for training and 3,101 for validation. The Egocart dataset has other issues compared to the iCab dataset. Firstly, its visual complexity is higher. Secondly, the cart turns at many intersections between the aisles, and takes many alternative routes in the store.

The FM and LAM drone datasets. These are two indoor drone datasets that we build to test our method. A DJI FLY 2S drone is employed. The positional data used during training is extracted from the onboard drone sensors: with a KF, we combine the localization through ArUco markers and Visual Inertial Odometry. Then, we localize the drone using a more precise external sensor, an OUSTER OS1 lidar: the drone is tracked in the lidar point cloud. During training, the localization from the onboard sensors is used; during testing, we instead adopt the localization from the lidar as ground truth. We have two scenarios (normal and abnormal) for each dataset. In the Frontal Motion (FM) normal scenario, the drone takes off, turns left, moves forward for around six meters, moves backward, turns right, and lands at the same point of its departure. In the abnormal scenario, pedestrians move either on the side of the drone or in front of it. When pedestrians appear in front of the drone, it stops to let them pass and then continues its motion. In the LAteral Motion (LAM) normal scenario, the drone takes off, moves left for six meters, then moves right, and lands at the same point of its departure. In the abnormal scenario, static abnormalities (i.e., squared boards of black or gray color) appear in the scene. The FM dataset includes 18,253 train frames, 2,518 validation frames, and 34,920 testing frames; the LAM dataset includes 17,368 train frames, 1,736 validation frames, and 20,309 testing frames. To summarize, the two datasets are different in terms of anomalies (moving pedestrians vs. static objects), and motion type (frontal vs. lateral) which affects the environment area observed from the camera. Moreover in the FM case the drone always follows the same path, whereas in the LAM case the drone moves on five equidistant paths (see Fig. 8). The localization in the second scenario is more difficult due to the visual similarity between the five paths. Fig. 7 displays images from the FM and LAM datasets. Fig. 8 shows the training positions of FM and LAM.

Table 1
Details about the models trained for each dataset. The [.] symbol denotes the size of the variable.

	Icab	Egocart	Drone (FM)	Drone (LAM)
$[x_t]$	64x64	192x108	152x84	152x84
kernels	$(3,3)$	$(3,3)$	$(3,3)$	$(3,3)$
channels	$32,64,128$	$16,32,64,64,128$	$16,32,32,64,64$	$16,32,32,64,64$
$[a_t]$	32	128	16	16
$[z_t]$	8	20	8	8
K	45	249	79	75

Model implementation details on each dataset are reported in Table 1. Dataset images are reduced to the size $[x_t]$.

4.2. Comparison with our previous method

The proposed approach and the work in [25] mainly differ in the information used as input to the GNG clustering. From here on, we denote the method in [25] as the First Version (FV), and the one proposed here as the Second Version (SV). The clusters of the FV are built using odometry data only, whereas the clusters of the SV are obtained employing both odometry and video data. As mentioned in the introduction, in the SV, the fusion of the two sensor modalities is performed at a lower level than in the FV (i.e., by joining odometry and video in the clustering process). This is also evident when observing Fig. 3. These differences lead to some advantages and disadvantages in each version.

Firstly, the SV is expected to perform better than the FV. The first executed actions at each time instant are the particle-independent operations. These include two steps: encoding the image with the VAE and calculating the probability of being in each cluster ($\boldsymbol{\alpha}_{v,t}$), based on the VAE bottleneck. Thus, a first estimate of the probability of being in each cluster is obtained only considering the content of the images. The clusters in the SV of the method are built using the image information too. The video vocabulary is, thus, more precise and explainable. As a result, the first estimate of the cluster at time t will tend to be more accurate in the SV compared to the FV. As expected, final localization results are shown to be better in the SV.

The FV has, instead, the advantage of being more modular. The odometry vocabulary is learned, without using the video data, and then the video block is learned too. In the SV, learning the odometry vocabulary involves the video data too. This has two consequences. First, when performing continual learning, if video anomalies are detected, both the video and odometry vocabularies must be relearned (in the more modular FV, only the video vocabulary must be relearned). Second, the CG-KVAE learning phase can become more restricted too: instead of learning the VAE parameters and KVAE matrices together from the start, the VAE parameters must be learned first (unlike what was suggested in [28]). Another minor advantage of the first version is that the transition matrix tends to be less entropic, which potentially leads to the need of less particles to track all the hypotheses.

4.3. Comparison with other state-of-the-art methods

We compare our approach to a set of state-of-the-art methods. Firstly, we employ methods based on pre-trained models that were tested on the Egocart dataset in [42]. When discussing the results on Egocart, we will report a selection of methods in [42] that achieved the best and worst results. Both Image Retrieval (IR) and Regression (REG) methods are reported. As mentioned above, for comparison on

Fig. 6. Example of images in the iCab dataset.

(a) (b) (c) (d)

Fig. 7. Images from the FM and LAM scenarios. (a) and (b) show a viewpoint shared by both scenarios. (c) and (d) belong to the FM and LAM scenarios, respectively.

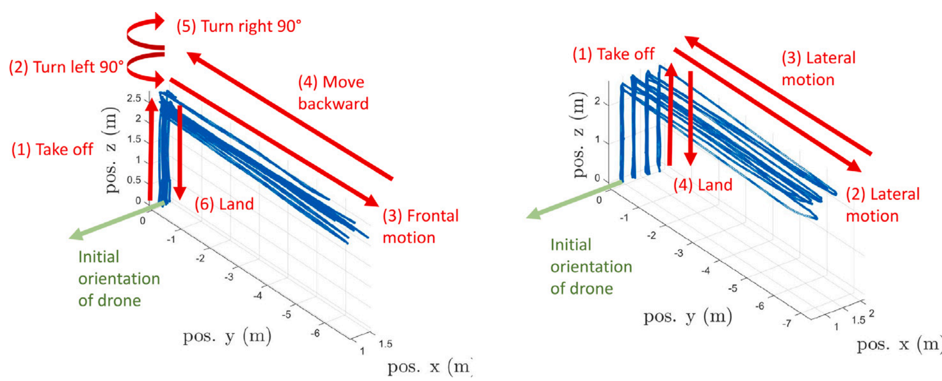


Fig. 8. Training positions covered by the FM (left) and LAM (right) drone datasets.

the Icab, FM, and LAM datasets, we selected the IR-IV3, IR-TC-IV3, and REG-PNET-RGB-POS-IV3 methods. We replicated the approaches and tested them on our three datasets.

Furthermore, we have added three methods: IR-VAE, IR-TC-VAE, and REG-ENC. In IR-VAE, as a first step, a VAE is trained to reconstruct the training images. A database of the latent states obtained applying a VAE on the training data is built. During testing, each image x_t is given as input to the VAE, extracting the latent state a_t. The closest latent state in the training database is found; x_t is finally assigned the position of this latent state. This method is, therefore, similar to methods such as IR-IV3. The difference is that the latent state is extracted from a VAE built solely on the training dataset, instead of using a model pre-trained on ImageNet (a classification dataset with more than 1.28 million images). The structure of the VAE is the same employed in our approach. The IR-TC-VAE case adds a temporal constraint, such that retrieval can be performed only on the images of the training set that do not have a position further away of 4 meters (for iCab and Egocart) and of 0.5 meters (for the drone datasets) from the position estimated at the previous testing time instant. This case is similar to IR-TC-IV3. Finally, REG-ENC is a regression method that trains a CNN to directly predict the position from one image. The used CNN has a structure similar to the VAE encoder, but a ReLU activation, a dropout layer ($p = 0.5$), and a Fully Connected (FC) layer are also added after the bottleneck feature a_t (the variance is, instead, not learned). The FC layer gives the position as output. For this reason, we denominate this method REG-ENC, as the used CNN is the same as the encoder of the VAE, with the addition of three layers. Furthermore, the same training choices (e.g., learning rate, optimizer) are employed as for the VAE. This method is similar to REG-PNET-RGB-POS-IV3. The reason for adding these three methods is to have a fairer comparison. All the methods employed in [42] start from models pre-trained on ImageNet, whereas our method does not. Potentially, also our approach could start from a pre-trained model. However, this is not completely coherent with our framework: pre-trained models are trained through classification, which is supervised. Instead, in our framework, we aim to learn concepts in an unsupervised and continual way (through anomalies).

5. Experimental results

In this paper, compared to our previous works [27,59], we are able to achieve simultaneous localization of our system. As this is a novel contribution, the results focus on this capability and on the study of the explainable vocabulary.

We first examine, in Section 5.1, the localization results of the SV compared to the FV. As noticed in Section 4.2, we expected the new approach to perform better, and we verified it. After an ablation study involving the FV and SV schemes, the latter is also compared to other methods. The clustering optimization is analyzed too. Furthermore, in Section 5.2, we answer some important questions while comparing the two approaches, such as: "(Q1) how do the clustering quality metrics change in the two cases?", "(Q2) how do the clustering means and covariances change?", "(Q3) in which case is the clustering vocabulary better from the image point of view?", and "(Q4) how does the evolution between clusters change?" As the SV builds a better vocabulary that uses both odometry and video data, we expect it to display better letters (or symbols) from the image point of view (Q3). This means also a better video clustering metric q^a (Q1), which should lead to a higher localization accuracy. However, as a consequence of the trade-off, we expect worse odometry letters and a worse velocity metric q^v. The cluster means and covariances change consequently (Q2). In addition, due to the space being more fragmented as a result of applying clustering on both odometry and video, we expect to have more frequent passages between clusters and, thus, a more entropic transition matrix. In Section 5.3 we also discuss how the parameters chosen for the CMJPF

Table 2
Comparison between the FV and SV of the proposed method. The localization error is expressed in meters.

Dataset	FV		SV	
	Mean err.	Median err.	Mean err.	Median err.
Egocart	2.08	0.76	1.65	0.96
iCab	1.06	0.78	0.98	0.75
Drone FM	0.21	0.13	0.23	0.14
Drone LAM	1.45	0.56	0.87	0.38

can affect the final results. Section 5.4 reviews the algorithm's memory requirements and related computational time. Finally, in Section 5.5 we present an attention analysis study, highlighting the image areas that the network focuses on, particularly during the localization and prediction processes.

5.1. Localization results

Comparison between FV and SV. Table 2 compares the localization results of both the FV and SV approaches on the four datasets (in meters). The results refer to the particles which survived resampling until the end of the trajectory. The SV tends to display better results, especially when looking at the mean error. A significant improvement can be observed using the Egocart (43 cm on the mean error, whereas the median error increases by 20 cm) and drone LAM cases (58 cm on the mean error and 18 cm on the median error). A minor improvement is present for the iCab dataset (14 cm on the mean error, whereas the median error increases by 6 cm). In the FM case, instead, the SV provides better results, but the difference is minimal (2 cm for the mean error and 1 cm for the median error).

Ablation study considering the FV and SV approaches. Here we present some ablation results we obtained, considering both the FV and SV approaches.

Table 5 compares the localization results of our method, when both the FV and SV approaches are applied to the iCab, Egocart, FM, and LAM datasets. The mean and median localization errors are shown in meters. The results refer to particles that survived the resampling until the end of the trajectory.

The first row related to each dataset reports the localization error from the $\hat{x}^o_{t,n}$ estimations (i.e., directly from the video DBN prediction) and from the $x^o_{t|t,n}$ estimations (i.e., after performing the odometry 'update' phase). Therefore, the second row involves the use of the link $P(\tilde{z}^o_{t+1}|\tilde{z}^o_t)$ as well, whereas the first row does not. The second row represents the final result of our method, which displays better performance in all four cases (see also Table 2).

Comparisons between the SV and other methods. We compare the proposed method with other state-of-the-art methods. Tables 3 and 4 show the localization errors obtained from methods that do not use pre-trained models (such as ours) and methods that do (see Section 4.3). In our architecture, we derive the localization error by computing the Mean Squared Error (MSE) over the survivor particles. We perform these comparisons to evaluate if the method can reach a localization accuracy comparable to the one of the state-of-the-art methods. However, we must also note that the proposed method does not only perform localization. Instead, it is integrated into a wider theoretical framework for self-aware autonomous systems, which includes anomaly detection and explainability. Conversely, the methods proposed in the literature were developed for localization purposes only.

The tables show that, on the Egocart dataset, the proposed method outperforms IR-PNET-VGG16 and REG-SVR-PNET-RGB-VGG16. However, it obtains worse results than the other methods that use pre-trained models. On the other hand, the model performs significantly better than REG-ENC and IR-TC-VAE, but worse than IR-VAE.

In the iCab case, due to the symmetry of the environment, better results are obtained with those methods that also enforce temporal constraints/reasoning (in particular, IR-TC-IV3 and the proposed method). The proposed method performs much better than the three methods (IR-VAE, IR-TC-VAE, and REC-ENC) that do not employ a pre-trained model and use the same VAE (or encoder). The reason behind the failure of these models falls in the high visual symmetry of the dataset: the VAE features corresponding to opposite sides of the courtyard are very similar. However, through the PF, it is possible to consider diverse hypotheses in parallel for several time instants, and then select which ones to keep.

In the FM scenario, similar results are obtained with the different methods. The only exception is the REG-ENC approach, which performs worse than all the other ones. Our method is coherent with the other ones, performing slightly worse on the mean error and slightly better on the median error. Finally, in the LAM case, our approach performs slightly worse than the methods with pre-trained models and IR-VAE, better than REG-ENC, and very similarly to IR-TC-VAE.

Of the six methods presented in all tables, the IR-TC pre-trained methods (i.e., IR-TC-IV3 and IR-TC-VGG16) always outperform the proposed approach, except for the FM scenario, where the results are similar. The REG-ENC method consistently performs worse than our approach on all datasets. It is worth noting, however, that IR-TC-IV3 and IR-TC-VGG16 employ pre-trained models, whereas REG-ENC, like our approach, does not.

In summary, our method outperforms methods that were not pre-trained on large datasets. Furthermore, it provides similar results with methods pre-trained on large datasets that do not enforce temporal constraints when adopted on datasets with high symmetry in the environment (such as the iCab dataset). However, it performs worse than methods pre-trained on large datasets that additionally enforce temporal constraints.

To conclude, the proposed method performs within a similar range as other state-of-the-art VBL methods. Nevertheless, it provides further features compared to them:

- a structure coherent with a self-awareness framework;
- the coupled odometry and visual generative maps which allow the extraction of anomalies for both explainability and continual learning capabilities.

Explainable anomaly detection is, indeed, a field that has recently gained attention (although this is not one of the main objectives of this paper). When performing anomaly detection, it is desirable to determine where or when an abnormal event was, together with its cause, e.g., what model rules were broken. This is crucial in the field of Autonomous Vehicles, as unexplainable autonomous cars are less easily accepted by the public [60]. In the framework adopted in this work, a further reason to use interpretable models such as DBNs can be traced back to the six basic capabilities that an agent should have to be considered self-aware (see also Section 2.2). As observed, once anomalies are detected, a new model must be trained, and this allows the car to improve its learning capabilities. However, sometimes not the entire model needs to be retrained, but only a part of it. Understanding the anomaly reason (or, simply, at what abstraction level it occurs) allows us to detect which part of the models must be retrained. In this work, coupling odometry and image data enables us to extract specific anomalies that arise when the combination of the two data modalities (in a new scenario) is different from the one collected in a previously encountered scenario. In the end, anomalies are signals that can be further analyzed and processed. These noisy signals can be observed by self-aware filters, which can function as anomaly sensors. These filters not only track the evolution of the state of a dynamic system by estimating posterior probabilities, but also allow us to identify anomalies, using a certain model. We could potentially develop new filters by analyzing these anomalies, which represent errors between

Table 3
Comparison of methods that do not use pre-trained models. The localization error is expressed in meters.

Method	Egocart		iCab		Drone (FM)		Drone (LAM)	
	Mean err.	Median err.	Mean err.	Median err.	Mean err.	Median err.	Mean err.	Median err.
IR-VAE	1.60	0.32	23.00	23.00	0.20	0.16	0.47	0.25
IR-TC-VAE	3.61	0.39	23.00	23.00	0.20	0.16	0.86	0.33
REG-ENC	8.59	7.66	23.88	22.78	0.89	0.76	1.83	1.14
Proposed (SV)	**1.65**	**0.96**	**0.98**	**0.75**	**0.23**	**0.14**	**0.87**	**0.38**

Table 4
Comparison of methods that use pre-trained models. The localization error is expressed in meters. For details of the methods and other approaches tested on Egocart, see [42].

Method	Egocart		iCab		Drone (FM)		Drone (LAM)	
	Mean Err.	Median Err.	Mean Err.	Median Err.	Mean Err.	Median Err.	Mean Err.	Median Err.
IR-IV3	0.73	0.28	1.28	0.61	0.18	0.16	0.32	0.20
IR-TC-IV3	–	–	0.72	0.60	0.18	0.16	0.33	0.20
IR-PNET-VGG16	2.17	1.38	–	–	–	–	–	–
IR-TC-VGG16	0.52	0.28	–	–	–	–	–	–
IR-TR-TC-VGG16	0.44	0.29	–	–	–	–	–	–
REG-SVR-PNET-RGB-VGG16	1.96	1.54	–	–	–	–	–	–
REG-PNET-RGB-POS-IV3	0.42	0.29	1.52	1.15	0.24	0.20	0.74	0.71

Table 5
Ablation study results. The table shows the localization error in meters.

Dataset	FV		SV	
	Mean err. (m)	Median err. (m)	Mean err. (m)	Median err. (m)
Egocart ($\hat{x}^o_{t,n}$)	2.34	0.86	1.83	1.07
Egocart ($\mathbf{x}^\mathbf{o}_{t\|t,n}$, final)	**2.08**	**0.76**	**1.65**	**0.96**
iCab ($\hat{x}^o_{t,n}$)	1.15	0.79	1.17	0.84
iCab ($\mathbf{x}^\mathbf{o}_{t\|t,n}$, final)	**1.06**	**0.78**	**0.98**	**0.75**
Drone FM ($\hat{x}^o_{t,n}$)	0.21	0.13	0.23	0.14
Drone FM ($\mathbf{x}^\mathbf{o}_{t\|t,n}$, final)	**0.21**	**0.13**	**0.23**	**0.14**
Drone LAM ($\hat{x}^o_{t,n}$)	1.45	0.57	0.88	0.39
Drone LAM ($\mathbf{x}^\mathbf{o}_{t\|t,n}$, final)	**1.45**	**0.56**	**0.87**	**0.38**

Table 6
Localization error on iCab (in meters), with the SV, varying the number of clusters.

Number of clusters	Mean err.	Median err.	Th_r
8	4.98	2.09	~ 0.75
27	1.13	0.83	~ 0.45
45	0.98	0.75	~ 0.35
116	1.10	0.74	~ 0.10
150	1.05	0.68	~ 0.075

predictions and observed data, as they are linked to the specific model adopted. This process could be carried out incrementally, with new filters being added over time (using explainable DBNs) as the agent faces new situations.

Results of the SV with different clustering levels. Table 6 shows the localization error obtained with 8, 27, 45, 116, and 150 clusters. These cases correspond to a threshold Th_r (see Section 3.3) of around 0.75, 0.45, 0.35, 0.10, and 0.075, respectively. The threshold range 0.10–0.35 is a good trade-off between accurate velocity prediction, visual content homogeneity, and high spatial extension, with a preference for the first two factors. We can observe that, in this range, the localization error maintains similar values. A much higher threshold leads to under-clustering and to a severe degradation of performance, as it can be observed using a 0.75 threshold. This is a bad choice from the localization performance point of view, as this corresponds to giving much more priority to having few big clusters than to having accurate velocity predictions and visual content homogeneity inside each cluster. Highly inaccurate velocity predictions and visual content heterogeneity in the clusters lead to imprecise models that do not provide good localization results. Using a lower threshold, performance remains stable, with the disadvantage of an increase in memory requirements and computational complexity, caused by the need to store a larger vocabulary and to perform comparisons against a greater amount of clusters.

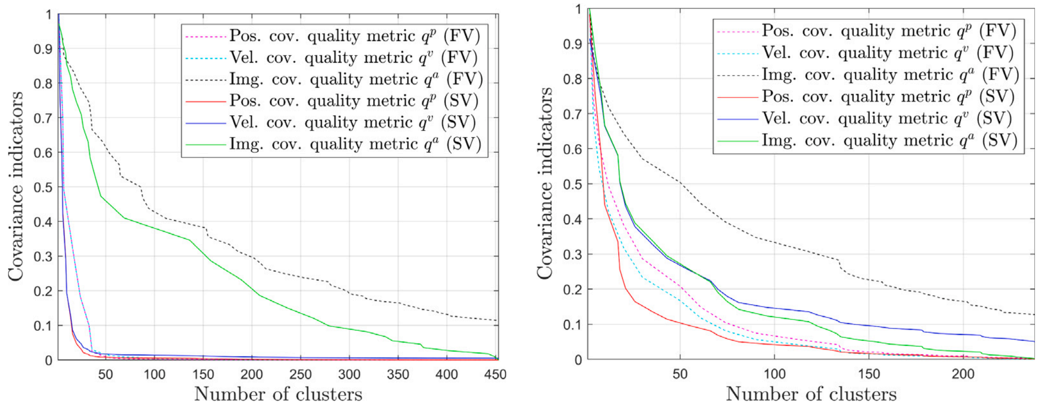
5.2. Analysis of the clustering vocabulary and its explainability

We now analyze the differences between the two versions, considering the learned clustering vocabulary. The choice of the vocabulary is fundamental because the localization performance and explainability depend on how well the learned vocabulary letters represent the data. We address the following questions presented at the beginning of this section.

How do the clustering quality metrics change in the two cases? We have seen in Section 3.3 that the optimal clustering level can be selected using three quality metrics q^p, q^v, and q^a. Let us now examine how these quality metrics change when changing the number of clusters, taking the iCab and LAM scenarios as possible examples.

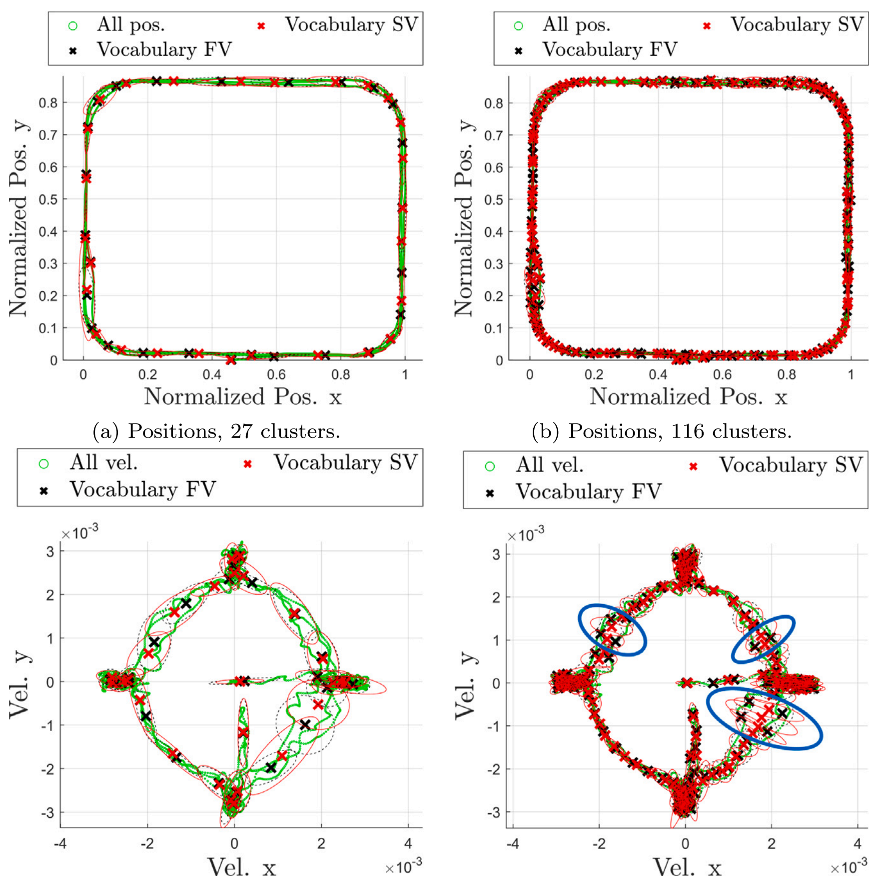
The evolution of the quality metrics, depending on the number of clusters, is displayed in Fig. 9(a) (considering the iCab dataset), for the FV and SV. The plots show the metrics normalized on the same scale. This normalization is carried out because the same odometry GSs and VAE latent states are used in the two versions. We can observe in the figure how, in the SV, both the velocity and image covariance quality metrics tend to descend faster, meaning that fewer clusters are necessary to obtain similar levels of velocity accuracy and image content homogeneity. However, after a certain clustering level (~ 50 clusters), the velocity covariance metric of the SV always remains higher than the same FV metric, and decreases very slowly. The reason behind this behavior can be explained by the trade-off between the velocity accuracy and the image homogeneity. There might be very similar images, with varying rates of velocity change, that end up being clustered together. This concept will be discussed further in the next section.

Fig. 9(b) displays the quality metrics for the LAM dataset. In this case too, the image covariance quality metric q^a decreases much quicker in the SV approach. Conversely, the velocity covariance quality metric q^v decreases more slowly.



(a) (b)

Fig. 9. Clustering quality metrics on the iCab (a) and LAM (b) datasets with different clustering levels for FV and SV.



(a) Positions, 27 clusters. (b) Positions, 116 clusters.

(c) Velocities, 27 clusters. (d) Velocities, 116 clusters.

Fig. 10. Cluster means and extensions for the iCab dataset, on the position space (upper part) and velocity space (lower part), with 27 clusters (on the left) and 116 clusters (on the right).

How do the clustering means and covariances change? We take one example (the iCab) to show the variation of the image clustering on the odometry space. In Figs. 10(a) and 10(b), we display the positional cluster centers for the FV (black) and SV (red) approaches, when using 27 and 116 clusters, respectively. The extension of the clusters is also represented by ellipses with axes proportional to the standard deviation of the clusters. Figs. 10(c) and 10(d) are related to the same two cases, but display velocity plots.

It is worth noting how, in the case with 27 clusters, many clusters have similar positional and velocity centers in the approaches. Some clusters in the SV, however, are centered in different areas to provide a better image homogeneity. This is more evident in the case with 116 clusters: observe the areas circled in blue in Fig. 10(d). In the FV, the clusters in this area are distributed to favor a more precise velocity prediction. Therefore, the two velocity "pathways" in these areas get assigned to different clusters. Conversely, in the SV, the clusters join points on the two "pathways". The reason behind this is the trade-off between the velocity accuracy and image homogeneity. This plot thus explains the behavior of the q^v quality metric in the SV compared to the FV, discussed in the previous section and illustrated in Fig. 9(a). It

Table 7
Evaluation indices that highlight the method's capability to retrieve the original cluster using a single image.

Dataset	Ratio correct cluster (better ↑)	MSE odom. dist. (m) (↓)	Bhatta odom. dist. (↓)
Icab, 27 clusters, FV	0.7113	**6.6973**	**114.1005**
Icab, 27 clusters, SV	**0.7368**	7.3099	142.4723
Icab, 45 clusters, FV	0.6918	**6.0767**	**194.3291**
Icab, 45 clusters, SV	**0.7813**	6.1775	259.6680
Icab, 116 clusters, FV	0.6804	4.9142	603.0445
Icab, 116 clusters, SV	**0.8697**	**3.3763**	**190.6402**
Egocart, FV	0.5959	3.0040	14.9611
Egocart, SV	**0.8089**	**1.3744**	**4.0040**
FM, FV	0.3990	0.2116	2.3175
FM, SV	**0.7637**	**0.0899**	**0.5567**
LAM, FV	0.5570	0.2013	2.0470
LAM, SV	**0.7904**	**0.0863**	**0.3401**

Table 8
Evaluation indices showing the entropy of the transition matrix and TTMs, and the average time spent in one cluster before moving to the next one.

Dataset	Trans. Mat. entropy	TTMs mean entropy	Mean instants spent in a cluster
Icab, 27 clusters, FV	**0.7118**	**0.1073**	**54.0854**
Icab, 27 clusters, SV	0.8589	0.1207	35.9837
Icab, 45 clusters, FV	**0.4803**	**0.0560**	**33.9773**
Icab, 45 clusters, SV	0.5696	0.0576	28.1887
Icab, 116 clusters, FV	**0.2511**	0.0152	**17.0038**
Icab, 116 clusters, SV	0.2755	**0.0145**	13.3482
Egocart, FV	0.1852	0.0094	52.7416
Egocart, SV	**0.1720**	**0.0070**	**60.8092**
FM, FV	0.4257	**0.0425**	36.3727
FM, SV	**0.3937**	0.0452	**42.8511**
LAM, FV	0.3971	**0.0279**	**9.1113**
LAM, SV	**0.3960**	0.0345	6.2284

is worth noting that this section does not aim to provide a quantitative comparison, but to qualitatively comment (with one example) how the trade-off between image homogeneity and velocity prediction accuracy influences the distribution of the clusters.

In which case is the clustering vocabulary better from the image point of view? As discussed in Sections 3.4 and 4.2, the first operations performed in the tracking process are the particle-independent ones. A first estimate of the probability of being in each cluster is computed using a single image data point. As before, we compare here the performance of the FV and SV.

Table 7 displays three metrics for this purpose. All metrics are only related to the particle-independent calculations and, therefore, are computed using a single image data point (independently of other data points and without considering the rest of the tracking). The values in the first column ("ratio correct cluster") are calculated as follows. First, we find the probability $\boldsymbol{\alpha}_{v,t}$ of being in each cluster, given an image data point. The cluster displaying the highest probability is selected and compared with the "true cluster". With "true cluster" we identify the odometry cluster closest to the odometry point. Comparing every assignment with the "true clusters", we can compute the ratio of correct cluster assignments, as displayed in the table. The second and third columns of the table show the average MSE and Bhattacharyya distances between the center of the predicted cluster and the "true cluster". The table presents these performance indices for the iCab dataset (on three clustering levels), and on the Egocart, FM, and LAM datasets, with the FV and SV. For each compared couple, the best index is highlighted in bold. The ratio in the first column is consistently better (i.e., higher) in the SV. The indices in the second and third columns tend to have better (i.e., lower) values in the SV as well. The SV approach, therefore, starts working with an improved estimation of the cluster probabilities vector $\boldsymbol{\alpha}_{v,t}$. This better estimation derives from using a better clustering, with lower image covariance. This is the case of the SV, as highlighted by the q^a metric in Fig. 10.

How does the evolution between clusters change? (i.e., how are the transition matrices?) In the SV approach, the clusters are built considering both odometry and video. This can lead to a more entropic transition matrix. The reason is that a zone covered by a single cluster in the FV (due to its motion homogeneity), could instead be split into several clusters in the SV (due to its visual heterogeneity). Table 8 illustrates this concept on the different datasets. In the first and second columns, we show the transition matrix entropy and the mean entropy across the TTMs. The last column displays the mean number of time instants spent in a cluster before moving to the next one. The entropies of the matrices tend to be slightly higher in the SV. Coherently with this, in the SV, the agent tends to spend less time in each cluster and, therefore, to move more often between clusters. For each compared couple, we highlight in bold the case with lower entropy and higher time spent in a cluster. Transition matrices that are more entropic might lead to the need for a higher number of particles because they generate more hypotheses of transition between clusters. This can be seen as a small disadvantage of the SV.

5.3. Parameter choice analysis

In this section, we analyze how the parameters selected for the CMJPF could influence the results of the validation set.

Fig. 11 displays the localization results obtained on the validation set of the iCab and Egocart datasets, with different parameters. Subfigures (a) to (d) and (f) to (i) consider variations of $N_{th,l}$, $N_{th,h}$, and m_v. The changing parameter is defined on the x-axis of the plots. The mean final distance obtained with each parameter is displayed as the heigh of the colored columns, with the standard deviation shown as an error bar. The error bar was thresholded to zero at the lower end when necessary. Each color represents a different method for calculating the localization error. Blue and red correspond to the localization computed from the weighted average of the particles using $\hat{x}^o_{t,n}$ and $x^o_{t|t,n}$, respectively; cyan and magenta, instead, refer to the use of particles surviving resampling, as in Table 3. The first row corresponds to Icab, and the second to Egocart.

We can notice how, in the Icab case, where no intersections are present, using the particles with their weights gives the best result. Conversely, in scenarios with many intersections, such as the Egocart dataset, considering the estimation of the particles remaining after full resampling provides better results. At an intersection, two paths might initially have equally high weights, but the less likely one will be eliminated at the end.

Fig. 11b shows that better results are obtained with a lower $N_{th,l}$ on the Icab dataset, which reflects how the environment is structured: in a highly symmetrical courtyard, waiting longer before the first particle resampling allows us to pick the right side with a higher probability. Fig. 11c shows the same case, displaying all the points with their respective errors. It is worth noting how cases with errors of almost 30 meters are located at the very top of the figure: the vehicle is predicted to be on the opposite side of the courtyard. It happens 2 times for $N_{th,l} = 1.01$ and 5 times for $N_{th,l} = 1.05$. In contrast, considering the Egocart dataset, a higher $N_{th,l}$ provides better results, as it can be observed in Fig. 11g. This can be attributed to two characteristics of the Egocart dataset: it has a low frame rate and it has many turns that lead to the environment changing quickly. Consequently, resampling after a longer time, also at the beginning, can be detrimental because wrong path estimations need to be eliminated promptly. In addition, the Egocart dataset does not have the strong symmetry of the Icab dataset, which made a lower $N_{th,l}$ useful.

A lower temperature parameter m_v (Figs. 11d and 11i) has a similar impact to those of the resampling thresholds $N_{th,l}$ and $N_{th,h}$, as it

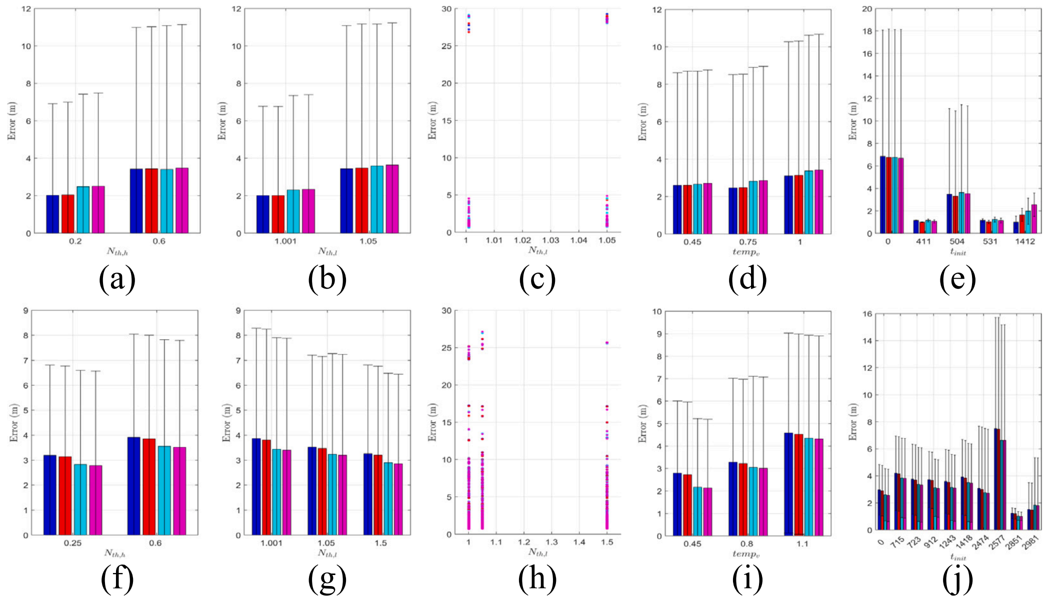


Fig. 11. Localization error on the validation dataset choosing different values of the parameters, for Icab (a-e) and Egocart (f-j) cases. (a,f): $N_{th,h}$, (b,g) and (c,h): $N_{th,l}$, (d,i): m_v, (e,j) refer to different starting points of tracking..

Table 9
Memory requirements considering each dataset.

	Icab	Egocart	Carla	Drone (FM)	Drone (LAM)
Memory (MB)	3.24(3.13)	20.7(6.5)	0.85(0.79)	1(0.94)	1(0.97)

Table 10
The last column on the right displays the algorithm's computational cost with varying settings. The changed settings are displayed from the second to the fourth column and include: the number of particles (N), the method to calculate the distance $d_{v,t}^{(\tilde{S})}$ between video states and video clusters, the adoption of the restart mechanism ("Restarts").

Dataset	N	$d_{v,t}^{(\tilde{S})}$	Restarts	Time/img (ms)
Icab	5	D_B	Yes	25
	25	D_B	Yes	65
	50	D_B	Yes	115
	50	MSE	Yes	109
	50	MSE	No	108
	50	D_B	No	114
Egocart	150	D_B	Yes	640
	125	D_B	Yes	550
	125	MSE	Yes	504
	125	MSE	No	503
	125	D_B	No	549

distributes the particles on the possible hypothesis, which is beneficial in a scenario with many intersections as in the Egocart dataset. Figs. 11e and 11j are not related to a trainable parameter, but to the different instants that were used as starting points. When evaluating the best choice of parameters, the tracking is started from several starting points across the trajectories. This provides a better choice of the parameter $N_{th,l}$ if the training and validation trajectories always start from the same area but the testing trajectories might not.

This testing phase, considering several parameters, was conducted on a grid of 120 combinations for Icab and 540 for Egocart.

5.4. Memory requirements and computational time

We now analyze the algorithm's memory requirements and computational time, considering the Icab and Egocart models. The algorithm is implemented in Python using the Pytorch library. The experiments were performed using an Intel® CoreTM i7K 8700K Processor and a dual NVIDIA® GeForce® GTX 1080 Ti with 11 GB RAM GDDR5X each.

Table 9 shows the memory required to store the overall model. In the Icab case, 3.24 MB are necessary, with 3.10 MB allocated for the VAE parameters and the matrices A, B, C, D, and E.

The distance $d_{v,t}^{(\tilde{S})}$ (see Section 3.4) can be calculated in different ways. We proposed using the Bhattacharya distance D_B between $M^{(\tilde{S},a)}$ and $Q^{(\tilde{S},a)}$. If, instead, the MSE between $M^{(\tilde{S},a)}$ and a_t is employed, it is not necessary to store the matrix $Q^{(\tilde{S},a)}$. The total memory requirement is reduced to 3.13 MB. The minor gain is due to the presence of only 22 clusters and to the small dimension of the latent state z_t.

In the Egocart case, the memory required for storing the overall model is 20.7 MB, and can be reduced to 6.5 MB, resulting in a substantial gain, as 249 clusters are used. The localization results are similar in the two cases: a mean error of 2.08 and median error of 0.76 with D_B, a mean error of 1.98, and median error of 0.97 for the MSE case. The comparison is performed with the FV approach. The model is smaller than those based on Inception V3 [50] (93 MB, 129 MB, 258 MB) and VGG-16 [61] (512 MB) used in [42]. No additional training data information needs to be stored.

As for as the computational time is concerned, the clustering levels and the number of particles employed in our CMJPF filter are related one to each other. If there are many clusters, and these clusters are all adjacent, it is generally expected that more particles need to be used in order to obtain reliable results. This is because, with a higher number clusters, the model needs a higher set of particles to accurately capture the distinctions between them. The increased number of particles helps in better representing the complexity of the data, ensuring that each cluster is sufficiently represented to avoid inaccuracies. Table 10 shows the computational cost of the algorithm, for both the Icab and Egocart models, varying the number of particles, using either D_B or MSE for calculating $d_{v,t}^{(\tilde{S})}$, and adopting particle restarts or not adopting it. First, we observe that using MSE instead of D_B reduces the computational cost of 6 ms in the Icab case with 50 particles (N), and of 46 ms in the Egocart case with 125 particles. The computational cost of particle resampling, instead, is negligible: approximately 1 ms in both cases. In the Icab case, there is a fixed cost of 15 ms, plus 2 ms for each added particle; in the Egocart case, these two costs are 100 ms and 3.6 ms, respectively. This table aims to give a more detailed understanding of the trade-offs involved in choosing different numbers of particles and clustering levels, offering valuable insights into the efficiency and

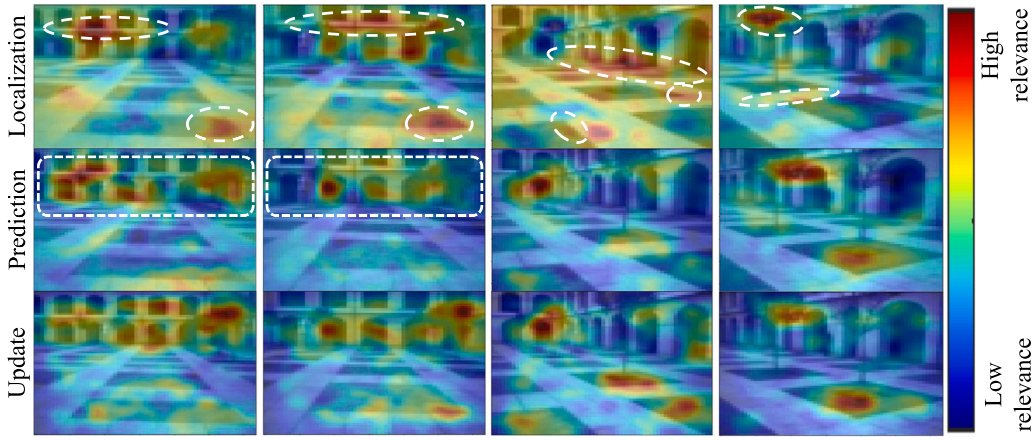


Fig. 12. Attention analysis over the Icab dataset: which image features does the network focus its attention on when performing localization, prediction, and update?.

scalability of our approach. In addition, it is worth noticing that if a transition matrix is more entropic (defining how the clusters are connected), the number of particles required to compute the final localization will be higher, and consequently the computational cost will also increase. In our case, this evaluation is not straightforward, as the dimensions of the images in the various datasets could also influence the final computational times.

5.5. Attention analysis

At this stage, we analyze the image information that the network focuses on during the localization and prediction processes. By doing so, we investigate the meaning of the z_t state. There are two different ways to visualize where NNs are concentrating their attention: *(A)* methods that change the input of the model to find which areas, when perturbed, generate the most significant changes in the output [62, 63]; *(B)* methods that visualize how the network internally responds to an input [64,65]. Most of these methods are designed for image classification.

Since in our model the convolutional layers are followed by FC layers and a set of matrices, we do not have a 2D feature space, as in most cases, of the second type. Consequently, we chose a simple method of the first type. We loop over the training data points: for each data point x_t, filtering is performed over $[x_{t-\eta}, x_{t-1}]$, with η being a chosen time window. Then, the update at t is performed in two ways: (i) using the actual observation x_t; (ii) sliding a set of occlusion windows over x_t. For each occlusion window, we compute the difference in the update $z_{t|t}$ and in the localization prediction obtained with (i) and (ii). The predictions $z_{t+1|t}$ calculated through the A matrices are also compared. The differences obtained are used to build three attention images for each data point.

Fig. 12 shows the explanation maps derived considering parts of the Icab dataset. A color closer to red means that a higher difference was obtained between the two cases, i.e., (i) and (ii). The rows are related to localization, prediction, and update, respectively. For localization, the network appears to retain information about some elements, such as the corners of the floor pattern, horizontal lines of the colonnade, and trees (circled in white in the first row of Fig. 12). For prediction, the model seemed to concentrate on the colonnade. This could be explained by the fact that this element of the image is more easily encoded in a linear way (the colonnade gets bigger as the car approaches) compared to more complicated patterns, such as the horizontal lines of the floor. Finally, a combination of the localization and update spots is highlighted for the update phase.

6. Conclusions

We proposed a DBN-based model that allows an agent to localize itself, which is integrated into a self-awareness framework for autonomous systems. The proposed method combines DL with traditional signal processing methods for filtering and tracking. We focused on examining the localization accuracy of the method, also evaluating the explainability of its vocabulary. Several important questions are presented and discussed. The proposed method performs better than the one in [25] and in a comparable range with other state-of-the-art approaches. We observed that a better video clustering metric q^a is obtained. During the clustering optimization step, we observed a trade-off between the desire for larger clusters on one hand, and the need for precise dynamics and homogeneous visual content within each cluster on the other. This trade-off leads to different cluster means and covariances and to a more entropic transition matrix. These answers allow us to explore the improved vocabulary obtained in the new approach, which explains the higher performance of the system in localizing itself in the environment.

It is worth noting that one limitation of the approach proposed in this work is the number of parameters. The grid search can be very time-consuming, even when the same threshold is applied for all anomalies. Therefore, in the future, a more efficient method for selecting the parameters could be developed. Reasonings could be drawn using the topology of the map (as we observed, maps with frequent intersections require a lower $N_{(th,h)}$ or the visual repetitions in the data (datasets with many visual repetitions in positions far from each other require a lower $N_{(th,h)}$) or other characteristics of the data and of learned vocabulary. Only video data are used during the testing phase in the proposed method. Future work could consider the addition of a second sensor to use during both the training and testing phases, such as the IMU sensor. This would increase localization performance, especially in cases where the video could not be reliable (e.g., shadows). Extensions with more than two sensors could also be developed. For example, our VBL method could be improved by integrating other sensors. We could suppose that, while GPS data are never (or rarely) observable, both camera and IMU data are available at all time instants. This is a frequent condition when agents navigate indoor environments.

As previously mentioned, there are six self-awareness steps designed to develop a cognitive system that continuously learns from new situations. One of these capabilities, called model creation, considers the generation of a new model leveraging the abnormal data. In this way, the new model can minimize the prediction error associated with data that was previously classified as abnormal. When new situations are detected, new models are learned, resulting in the availability of multiple parallel models. During subsequent testing phases, the method can consequently switch between these models, based on their respective probabilities for the given data. This incremental learning ability of the agent could be included in a future approach.

CRediT authorship contribution statement

Giulia Slavic: Writing – original draft, Validation, Investigation. **Pamela Zontone:** Writing – original draft, Validation, Investigation. **Lucio Marcenaro:** Writing – original draft, Validation, Investigation. **David Martín Gómez:** Writing – original draft, Validation, Investigation. **Carlo Regazzoni:** Writing – original draft, Validation, Investigation.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: All uniGe authors reports financial support was provided by European Union. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research was partially funded by the European Union's Horizon Europe research and innovation programme under the Grant Agreement No. 10112113, and by the European Union - NextGenerationEU and by the Ministry of University and Research (MUR), National Recovery and Resilience Plan (NRRP), Mission 4, Component 2, Investment 1.5, project "RAISE - Robotics and Al for Socio-economic Empowerment" (ECS00000035).

Data availability

Data will be made available on request.

References

[1] R.A. Epstein, E.Z. Patai, J.B. Julian, H.J. Spiers, The cognitive map in humans: spatial navigation and beyond, Nat. Neurosci. 20 (2017) 1504–1513.
[2] D. Kumaran, E.A. Maguire, Match-mismatch processes underlie human hippocampal responses to associative novelty, J. Neurosci. 27 (32) (2007).
[3] P. Zontone, A. Affanni, A. Piras, R. Rinaldo, Stress recognition in a simulated city environment using skin potential response (SPR) signals, in: 2021 IEEE International Workshop on Metrology for Automotive, MetroAutomotive, 2021, pp. 135–140.
[4] T. Aminosharieh Najafi, A. Affanni, R. Rinaldo, P. Zontone, Driver attention assessment using physiological measures from EEG, ECG, and EDA signals, Sensors 23 (4) (2023).
[5] P. Zontone, A. Affanni, A. Piras, R. Rinaldo, Convolutional neural networks using scalograms for stress recognition in drivers, in: 2023 31st European Signal Processing Conference, EUSIPCO, 2023, pp. 1185–1189.
[6] C. Stiller, F. Puente León, M. Kruse, Information fusion for automotive applications – An overview, Inf. Fusion 12 (4) (2011) 244–252.
[7] N. Piasco, D. Sidibé, C. Demonceaux, V. Gouet-Brunet, A survey on Visual-Based Localization: On the benefit of heterogeneous data, Pattern Recognit. 74 (2018) 90–109.
[8] A. Kendall, M. Grimes, R. Cipolla, PoseNet: A convolutional network for real-time 6-DOF camera relocalization, in: IEEE Int. Conf. on Computer Vision, 2015, pp. 2938–2946.
[9] J. Brejcha, M. Čadík, State-of-the-art in visual geo-localization, Pattern Anal. Appl. 20 (3) (2017) 613–637.
[10] A.R. Zamir, A. Hakeem, L.V. Gool, M. Shah, R. Szeliski (Eds.), Large-Scale Visual Geo-Localization, Springer International Publishing, 2016.
[11] F. Walch, C. Hazirbas, L. Leal-Taixé, T. Sattler, S. Hilsenbeck, D. Cremers, Image-based localization with spatial LSTMs, in: IEEE Int. Conf. on Computer Vision, 2017.
[12] G. Lu, X.-I. Wong, Taking me to the correct place: Vision-based localization for autonomous vehicles, in: 2019 IEEE International Conference on Image Processing, ICIP, 2019, pp. 2966–2970.
[13] A. Irschara, C. Zach, J.-M. Frahm, H. Bischof, From structure-from-motion point clouds to fast location recognition, in: 2009 IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 2599–2606.
[14] T. Sattler, B. Leibe, L. Kobbelt, Efficient & effective prioritized matching for large-scale image-based localization, IEEE Trans. Pattern Anal. Mach. Intell. 39 (9) (2017) 1744–1756.
[15] P. Chakravarty, A. Zhang, R. Jarvis, L. Kleeman, Anomaly detection and tracking for a patrolling robot, in: Proc. of the Australiasian Conf. on Robotics and Automation, 2007, pp. 1–9.
[16] G. Slavic, M. Bracco, L. Marcenaro, D.M. Gómez, C. Regazzoni, P. Zontone, Joint data-driven analysis of visual-odometric anomaly signals in generative Ai-based agents, in: 2024 IEEE International Conference on Acoustics, Speech, and Signal Processing Workshops, ICASSPW, 2024, pp. 264–268.
[17] Y. Fang, H. Min, X. Wu, W. Wang, X. Zhao, B. Martinez-Pastor, R. Teixeira, Anomaly diagnosis of connected autonomous vehicles: A survey, Inf. Fusion 105 (2024) 102223.
[18] K. Friston, B. Sengupta, G. Auletta, Cognitive dynamics: From attractors to active inference, Proc. IEEE 102 (4) (2014) 427–445.
[19] S. Haykin, J.M. Fuster, On cognitive dynamic systems: Cognitive neuroscience and engineering learning from each other, Proc. IEEE 102 (4) (2014) 608–628.
[20] A.R. Damasio, The Feeling of What Happens: Body and Emotion in the Making of Consciousness, Harcourt Brace, 1999.
[21] C.S. Regazzoni, L. Marcenaro, D. Campo, B. Rinner, Multisensorial generative and descriptive self-awareness models for autonomous systems, Proc. IEEE 108 (7) (2020) 987–1010.
[22] D.P. Kingma, M. Welling, Auto-encoding variational Bayes, in: Int. Conf. on Learning Representations, 2014.
[23] Z. Ghahramani, Learning dynamic Bayesian networks, in: Adaptive Processing of Sequences and Data Structures: Int. Summer School on Neural Networks, Springer Berlin Heidelberg, 1998, pp. 168–197.
[24] J. Gibson, L. Carmichael, The Senses Considered as Perceptual Systems, Houghton Mifflin, 1966.
[25] G. Slavic, P. Marin, L. Marcenaro, D.M. Gomez, C. Regazzoni, Simultaneous localization and anomaly detection from first-person video data through a coupled dynamic Bayesian network model, in: IEEE Int. Conf. on Advanced Video and Signal-Based Surveillance, 2022, pp. 1–8.
[26] M. Baydoun, D. Campo, V. Sanguineti, L. Marcenaro, A. Cavallaro, C. Regazzoni, Learning switching models for abnormality detection for autonomous driving, in: Int. Conf. on Information Fusion, 2018, pp. 2606–2613.
[27] G. Slavic, A.S. Alemaw, L. Marcenaro, D. Martín Gómez, C. Regazzoni, A Kalman variational autoencoder model assisted by odometric clustering for video frame prediction and anomaly detection, IEEE Trans. Image Process. 32 (2023) 415–429.
[28] M. Fraccaro, S. Kamronn, U. Paquet, O. Winther, A disentangled recognition and nonlinear dynamics model for unsupervised learning, in: Conf. on Neural Information Processing Systems, 2017, pp. 3601–3610.
[29] W. Zhang, B. Natarajan, On the performance of Kalman filter for Markov jump linear systems with mode mismatch, Circuits Systems Signal Process. 40 (4) (2021) 1720–1742.
[30] A.P. Gonçalves, A.R. Fioravanti, J.C. Geromel, Markov jump linear systems and filtering through network transmitted measurements, Signal Process. 90 (10) (2010) 2842–2850.
[31] R.E. Kalman, et al., A new approach to linear filtering and prediction problems, J. Basic Eng. 82 (1) (1960) 35–45.
[32] J. Elfring, E. Torta, R. van de Molengraft, Particle filters: A hands-on tutorial, Sensors 21 (2) (2021) 438.
[33] R. Neapolitan, Learning Bayesian networks, 2003.
[34] D. Koller, N. Friedman, Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning, The MIT Press, 2009.
[35] S. Särkkä, Bayesian filtering and smoothing, in: Institute of Mathematical Statistics Textbooks, Cambridge University Press, 2013.
[36] S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, third ed., Prentice Hall, 2010.
[37] N.D. Dutt, C.S. Regazzoni, B. Rinner, X. Yao, Self-awareness for autonomous systems, Proc. IEEE 108 (7) (2020) 971–975.
[38] P.R. Lewis, A. Chandra, S. Parsons, E. Robinson, K. Glette, R. Bahsoon, J. Torresen, X. Yao, A survey of self-awareness and its application in computing systems, in: 5th IEEE Conference on Self-Adaptive and Self-Organizing Systems Workshops, 2011, pp. 102–107.
[39] A. Morin, Levels of consciousness and self-awareness: A comparison and integration of various neurocognitive views, Conscious. Cogn. 15 (2006) 358–371.
[40] S. Kounev, J.O. Kephart, A. Milenkoski, X. Zhu (Eds.), Self-Aware Computing Systems, Springer International Publishing, 2017.
[41] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S.E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: IEEE Conf. on Computer Vision and Pattern Recognition, IEEE Computer Society, 2015, pp. 1–9.
[42] E. Spera, A. Furnari, S. Battiato, G.M. Farinella, EgoCart: a benchmark dataset for large-scale indoor image-based localization in retail stores, IEEE Trans. Circuits Syst. Video Technol. 31 (2021) 1253–1267.

[43] A. Kendall, R. Cipolla, Geometric loss functions for camera pose regression with deep learning, 2017, [Online]. Available: https://arxiv.org/abs/1704.00390.
[44] J. Wu, L. Ma, X. Hu, Delving deeper into convolutional neural networks for camera relocalization, in: 2017 IEEE International Conference on Robotics and Automation, ICRA, IEEE Press, 2017, pp. 5644–5651.
[45] E. Spera, A. Furnari, S. Battiato, G.M. Farinella, Egocentric shopping cart localization, in: Int. Conf. on Pattern Recognition, 2018, pp. 2277–2282.
[46] S.J. Lee, D. Kim, S.S. Hwang, D. Lee, Local to global: Efficient visual localization for a monocular camera, in: IEEE Winter Conf. on Applications of Computer Vision, 2021, pp. 2230–2239.
[47] S. Brahmbhatt, J. Gu, K. Kim, J. Hays, J. Kautz, Geometry-aware learning of maps for camera localization, in: IEEE Conf. on Computer Vision and Pattern Recognition, 2018, pp. 2616–2625.
[48] A. Valada, N. Radwan, W. Burgard, Deep auxiliary learning for visual localization and odometry, in: IEEE Int. Conf. on Robotics and Automation, 2018, pp. 6939–6946.
[49] F. Xue, X. Wang, Z. Yan, Q. Wang, J. Wang, H. Zha, Local supports global: Deep camera relocalization with sequence enhancement, in: IEEE/CVF Int. Conf. on Computer Vision, 2019, pp. 2841–2850.
[50] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the inception architecture for computer vision, in: IEEE Conf. on Computer Vision and Pattern Recognition, 2016, pp. 2818–2826.
[51] K. Yousif, A. Bab-Hadiashar, R. Hoseinnezhad, An overview to visual odometry and visual SLAM: Applications to mobile robotics, Intell. Ind. Syst. 1 (2015) 289–311.
[52] H. Iqbal, D. Campo, L. Marcenaro, D. Martin Gomez, C. Regazzoni, Data-driven transition matrix estimation in probabilistic learning models for autonomous driving, Signal Process. 188 (2021) 108170.
[53] B. Fritzke, A growing neural gas network learns topologies, in: Conf. on Neural Information Processing Systems, 1994, pp. 625–632.
[54] Y.-L. He, X.-L. Zhang, W. Ao, J.Z. Huang, Determining the optimal temperature parameter for softmax function in reinforcement learning, Appl. Soft Comput. 70 (2018) 80–85.
[55] J. Holmström, Examensarbete DV 3 2002-0830 growing neural gas experiments with GNG, GNG with utility and supervised GNG, 2002, [Online]. Available: https://api.semanticscholar.org/CorpusID:17388986.
[56] H. Iqbal, D. Campo, M. Baydoun, L. Marcenaro, D.M. Gomez, C. Regazzoni, Clustering optimization for abnormality detection in semi-autonomous systems, in: Int. Workshop on Multimodal Understanding and Learning for Embodied Applications, 2019, pp. 33–41.
[57] H. Iqbal, D. Campo, G. Slavic, L. Marcenaro, D.M. Gómez, C.S. Regazzoni, Optimization of probabilistic switching models based on a two-step clustering approach, in: IEEE Workshop on Signal Processing Systems, 2020, pp. 1–6.
[58] P. Marın-Plaza, et al., Stereo vision-based local occupancy grid map for autonomous navigation in ros, in: Int. Joint Conf. on Computer Vision, Imaging and Computer Graphics Theory and Applications, 2016, pp. 703–708.
[59] G. Slavic, M. Baydoun, D. Campo, L. Marcenaro, C. Regazzoni, Multilevel anomaly detection through variational autoencoders and Bayesian models for self-aware embodied agents, IEEE Trans. Multimed. 24 (2021) 1399–1414.
[60] L.A. Dennis, M. Fisher, Verifiable self-aware agent-based autonomous systems, Proc. IEEE 108 (7) (2020) 1011–1026.
[61] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: Int. Conf. on Learning Representations, 2015, pp. 1–14.
[62] M.D. Zeiler, R. Fergus, Visualizing and understanding convolutional networks, in: European Conference on Computer Vision, 2014, pp. 818–833.
[63] R.C. Fong, A. Vedaldi, Interpretable explanations of black boxes by meaningful perturbation, in: IEEE Int. Conf. on Computer Vision, 2017, pp. 3449–3457.
[64] R.R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra, Grad-CAM: Visual explanations from deep networks via gradient-based localization, Int. J. Comput. Vis. 128 (2) (2020) 336–359.
[65] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, W. Samek, On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation, PLoS ONE 10 (7) (2015).