Bayesian Structure Learning for the Locating of an Unknown Number of Static Objects

by

Mink Verschure

In partial fulfilment of the requirements for the degree of
Master of Science
at Delft University of Technology,
to be defended publicly on 23-7-2025.

Faculty:	Mechanical Engineering
Department:	Cognitive Robotics

Mentors / Supervisors:	Martijn Wisse

Graduation committee:	Martijn Wisse
	Reza Sabsevari
	Arkady Zgonnikov

Keywords:
Bayesian
Structure Learning
Object Locating

Abstract

This paper presents a method which is capable of creating an object centered world description based upon consecutive measurements about an unknown number of static objects using Bayesian inference. The objects are represented by a two dimensional position, with the aim of adding more attributes in future works. This objective is reformulated into a clustering problem which is then solved using a structure learning method. It is implemented using RxInfer [1] which uses the message passing algorithm in combination with factor graphs to perform Bayesian inference. The results indicate a promising performance of the structure learning model, but also show signs that the object representation has been over simplified. The future works section provides guidance on how the model complexity can be increased by adding additional attributes in order to improve performance.

Bayesian Structure Learning for the Locating of an Unknown Number of Static Objects

Report
Master Thesis
TU Delft, 15-07-2025

Author
Mink Verschure
4906020

Supervisor
Martijn Wisse

Abstract—**This paper presents a method which is capable of creating an object centered world description based upon consecutive measurements about an unknown number of static objects using Bayesian inference. The objects are represented by a two dimensional position, with the aim of adding more attributes in future works. This objective is reformulated into a clustering problem which is then solved using a structure learning method. It is implemented using RxInfer [1] which uses the message passing algorithm in combination with factor graphs to perform Bayesian inference. The results indicate a promising performance of the structure learning model, but also show signs that the object representation has been over simplified. The future works section provides guidance on how the model complexity can be increased by adding additional attributes in order to improve performance.**

I. Introduction

Labour shortages are on the rise due to an ageing society. In the Netherlands, it is expected that the labour force will shrink with 7% in the period 2015-2035 [2]. Already these shortages are visible in branches such as construction and healthcare. The field of robotics proposes a partial solution to this problem. Well developed robots can assist workers to improve their productivity or autonomously perform jobs for which no labour is available. For example, the warehouse industry has proven to be a place where robots can efficiently be deployed [3]. However, the environment of a warehouse is altered to better suit the behaviour of the robots in operation. Sometimes to such a degree that human workers face unsafe conditions [4]. In industries like construction and healthcare, the environments cannot be adapted to suit the robotic needs, and it is the complexity of these environments that proves to be problematic in the development of robotics for these industries. A good example of this is the limited success in the development of self-driving cars despite the enormous effort from numerous car manufacturers [5]. In contrast, humans can operate almost effortlessly in all of these environments. Thus, the advancement of robotics in certain fields might be inspired by the operating mechanism of humans.
For some time now, neuroscientists have been describing the human brain as a Bayesian statistical inference machine [6]. This model formulates a variational Bayesian process that explains the two fundamental occupations of any organic system: interpreting observations and performing actions. Within the context of robotics, this variational approach, also known as active inference, has already been used to construct an adaptive controller for robotic manipulators [7]. However, the field of robotics has yet to adapt active inference inspired Bayesian methods for the interpretation of observations. Here a method is formulated to perform the cognitive task associated with interpreting observations from multiple objects.

The goal of this paper is to present a method which is capable of creating an object centered world description based upon consecutive measurements about an unknown number of static objects using Bayesian inference. The objects considered here will only have two attributes, namely a two dimensional position, with the aim of adding more attributes in future works. The presented method will therefore need to be of flexible nature, such that complexity can be added when desired.
Scientific relevance can be motivated by a principle first design approach. The only constraint on the method is the use of Bayesian inference and the only assumption is in the description of the objects. This results in a sensor agnostic solution which can be modified for different intentions.

This paper will contain the following elements. Section II will show the analogy between the goal stated above and a clustering problem. This allows the problem formulation to be rewritten as a clustering problem. Following this, a Bayesian inference method called message passing will be explained in section III. Section IV will continue to expand the mathematical context needed to understand the method by introducing a structure learning method. The exact clustering method as well as the process used to evaluate this method are given in section V. Section VI presents the results and the discussion is written in section VII. The paper is wrapped up in sections VIII and IX containing some possible guidance for future works as well as the conclusion.

II. Problem Formulation and Data Creation

This section discusses how the presented method is solving a clustering problem. This is best visualized by the structure of the data used to access the performance of the presented method. A robot gathering information about a limited region of their surroundings can be modeled by considering a small patch of an image as a measurement about that image. Thus, the concept of surroundings, or scene, is approximated with a single image.

Figure 1 is an example of a scene that will be used to obtain the results given in section VI. Patches are cropped from this image, on which YOLO11 [8] is used to perform object detection. This is visualized in figure 2. The center from each of these object detections is interpreted as a measurement. A number of these measurements from multiple patches can be visualized over the original image, resulting in what is shown in figure 3. Note that this visualization requires there to be a known transformation between coordinates in the patch frame and coordinates in the original image frame. Since these patches are generated, this transformation is known. Section V-B will specify how the transform was obtained during the evaluation of the method.

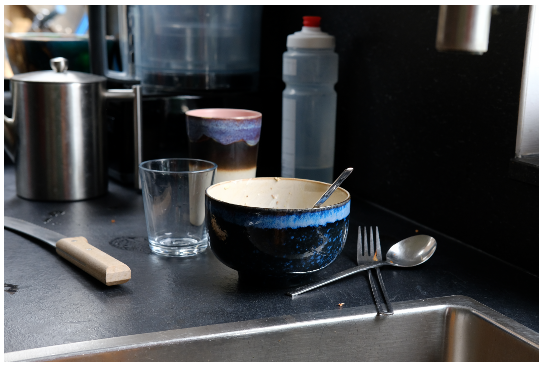


Fig. 1. An image that will serve as a scene for testing the method.

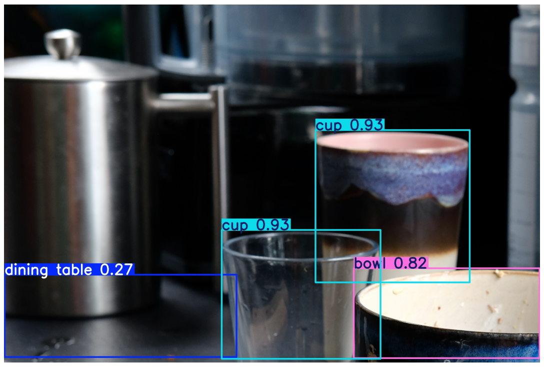


Fig. 2. A patch of the image displayed in figure 1. YOLO11 is used to detect objects and visualize their bounding boxes.

Figure 3 should convince the reader that the objective as formulated in the introduction translates to a clustering problem. Each object in a scene can be thought of as radiating observations which end up in close proximity, forming clusters as result. Therefore, to achieve the goal of this paper, a method is required that is capable of assigning a measurement to an existing cluster or introduce a new cluster if the measurement originated from a previously unobserved object.

III. Bayesian Inference and Message Passing

The solution to the clustering problem is programmed using a package called RxInfer. RxInfer implements a Bayesian inference method which relies on factor graphs and the message passing algorithm to perform inference [1]. To understand the workings of RxInfer, both of these concepts will briefly be introduced.

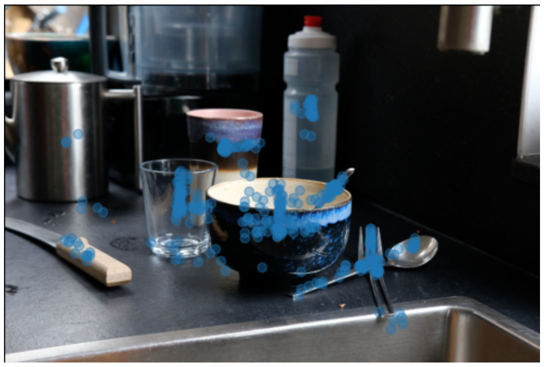


Fig. 3. This is the original image overlaid with data points gathered from multiple patches. Each data point represents the center of a bounding box as detected by YOLO11.

A. Factor Graphs

Factor graphs are a form of graphical models which explicitly represent how the joint probability of a probabilistic model decomposes into a product of factor functions [9]. In the particular notation used here, each factor function is represented by a node on a graph. The edges of this graph that link these factor functions together then represent the random variables present in the probabilistic model. Figure 4 is an example of such a factor graph. It represents a simple probabilistic model composed of two latent variables s_1 and s_2 and an observable variable y.

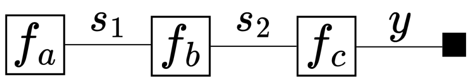


Fig. 4. An example of a factor graph consisting of two latent variables s_1 and s_2 and an observable variable y. These variables are associated with the edges of the graph that connect the factor functions f_a, f_b and f_c together. The small black square represents a special kind of factor function introduced when a specific value $y = \hat{y}$ is observed. Its factor function is of the form $\delta(y - \hat{y})$.

In general, the joint distribution of a factor graph is given by [9]:

$$p(x_1, \ldots, x_n) = \frac{1}{Z} \prod_a f_a(\mathbf{x}_a) \tag{1}$$

In this equation, Z is a normalization constant such that the resulting distribution is normalized and $\mathbf{x}_a$ represents the subset of random variables that f_a is a function over. Thus the joint distribution of the factor graph depicted in figure 4 can be written as

$$p(s_1, s_2, y) \propto f_a(s_1) f_b(s_1, s_2) f_c(s_2, y) \tag{2}$$

One thing still unexplained is the small black square in figure 4. Technically it also represents a factor function just like f_a and f_b. It represents a special kind of factor function indicating that the random variable y has been observed [10].

If $\hat{y}$ represents the observed value, the black square will have a factor function of the form $\delta(y-\hat{y})$. This clamps the random variable y to the observed value $\hat{y}$. Effectively, this changes the unobserved probabilistic model $p(x, y)$ into an *observed* probabilistic model, with a factorization as

$$p(s_1, s_2|\hat{y}) \propto f_a(s_1) f_b(s_1, s_2) f_c(s_2, y)\delta(y - \hat{y}) \tag{3}$$

B. Message Passing

Message passing, as implemented by RxInfer, uses these factor graphs to provide a scalable Bayesian inference method [1]. To motivate its advantages, a brief introduction of Bayesian inference is needed.

In a probabilistic model of the form $p(s, y) = p(s)p(y|s)$, Bayesian inference is concerned with finding the posterior probability $p(s|\hat{y})$. Here s represents a set of latent variables present in the model. Bayesian inference revolves around the well-known rule of Bayes. [10]

$$\underbrace{p(s|\hat{y})}_{\text{posterior}} = \frac{\overbrace{p(\hat{y}|s)}^{\text{likelihood}}\ \overbrace{p(s)}^{\text{prior}}}{\underbrace{p(\hat{y})}_{\text{model evidence}}} \tag{4}$$

Bayes' rule describes how the posterior is related to the likelihood, the prior and the model evidence. Though simplistic in form, a practical direct implementation of Bayes' rule can prove to be troublesome for two reasons. Firstly, where the likelihood and the prior are often known within a probabilistic model [10], the model evidence has to be calculated through the following integral

$$p(\hat{y}) = \iint p(\hat{y}|s)p(s)ds \tag{5}$$

This describes a integral over all possible values of the latent states s which often becomes intractable for higher dimensions of s [7]. Secondly, often the interest is not in obtaining a full posterior over s, but over a marginal posterior over a single latent variable s_i [11]. From $p(s|\hat{y})$ the marginal posterior can be calculated as

$$p(s_i|\hat{y}) = \iiint p(s|\hat{y})ds_{\setminus i}dy \tag{6}$$

It may be evident that equation 6 suffers from the same problems due to the dimensionality of s as equation 5.

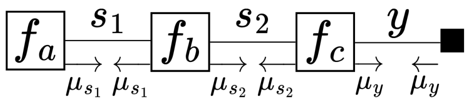
The message passing algorithm provides a method for calculating these marginal posteriors which only relies on low dimensional integrals over local variables [11]. Conceptually it does this by sending messages over the edges of a factor graph as discussed in section III-A. Mathematically, it is the factorization presented in equation 1 that allows for equation 6 to be simplified [11]. For example, with the factorization given in equation 3, the marginal posterior $p(s_2|\hat{y})$ can be calculated as

$$\begin{aligned} p(s_2|\hat{y}) &\propto \int f_a(s_1)f_b(s_1, s_2)f_c(s_2, y)\delta(y - \hat{y})ds_1dy \\ &= \underbrace{\int \overbrace{f_a(s_1)}^{\overrightarrow{\mu_{s_1}}} f_b(s_1, s_2)ds_1}_{\overrightarrow{\mu_{s_2}}} \cdot \underbrace{\int f_c(s_2, y)\overbrace{\delta(y - \hat{y})}^{\overleftarrow{\mu_y}} dy}_{\overleftarrow{\mu_{s_2}}} \end{aligned} \tag{7}$$

The groupings highlighted by the braces in this equation can be interpreted as messages being sent over a factor graph as is visualized in figure 5. Equation 7 shows that the marginal posterior of s_2 can be written as $p(s_2|\hat{y}) \propto \overrightarrow{\mu_{s_2}}\overleftarrow{\mu_{s_2}}$, i.e. the product of two messages travelling in opposite direction over the same edge. Here it is important to realize that messages are always a function of the variable corresponding to the edge they are travelling over [12]. In general, the calculation of a marginal posterior over any latent variable s_i can be written as [10]

$$p(s_i|\hat{y}) = \frac{\overrightarrow{\mu_{s_i}}\overleftarrow{\mu_{s_i}}}{\int \overrightarrow{\mu_{s_i}}\overleftarrow{\mu_{s_i}}ds_i} \tag{8}$$

Figure 5 below expands the previously used example of a factor graph with the messages that are send over the graph.



f_a
s_1
f_b
s_2
f_c
y
$\overrightarrow{\mu_{s_1}}$ $\overleftarrow{\mu_{s_1}}$
$\overrightarrow{\mu_{s_2}}$ $\overleftarrow{\mu_{s_2}}$
$\overrightarrow{\mu_y}$ $\overleftarrow{\mu_y}$

Fig. 5. This figure represents the same factor graph as displayed in figure 4, but the messages travelling over the edges of the graph are added.

In general, message are of the form [10].

$$\overrightarrow{\mu_{s_i}} = \int f_a(\mathbf{s}_a) \prod_{\substack{j \in \varepsilon(a) \\ j \neq i}} \overleftarrow{\mu_{s_j}} d\mathbf{s}_{a\setminus i} \tag{9}$$

Where $\overrightarrow{\mu_{s_i}}$ represents an outgoing message of factor f_a over edge s_i and $\varepsilon(a)$ is the set of all edges connected to node a. This allow $\prod_{\substack{j \in \varepsilon(a) \\ j \neq i}} \overleftarrow{\mu_{s_j}}$ to be interpreted as the product of all incoming messages on factor f_a except for the incoming message travelling over edge s_i. Since equation 9 defines one message based upon other messages, they display a recursive behaviour. This recursion finds natural termination points in factors describing prior distributions and factors describing observed variables.

This method of message passing is often called *Believe Propagation* or *Sum Product* message passing [13]. Under strict conditions, such a acyclic factor graph and factors belonging to the same family of exponential functions, the messages will have analytic form and the method described here will result in an exact Bayesian inference solution [14]. In practice it might be difficult to find these analytical solutions, such that numerical integration is needed. In this case, the message passing algorithm can be reformulated to minimize the Bethe free energy [15].

IV. Structure Learning

To deal with scenes that contain an unknown number of objects, an adaptive solution is required. Structure learning is such a solution. In the context of clustering, it is capable of assigning new measurements to existing clusters or initiate new clusters [16]. The structure learning method used here relies on the message passing algorithm in addition to a mixture node. Before further discussing structure learning, the working of this mixture node must be explained.

A. The Mixture Node

The mixture node is introduced in [17] with the goal of performing automated model comparison in factor graphs. Conceptually, the mixture node combines K probabilistic models which all contain similar, or overlapping, factors and variables $f_o(s_o, y_o)$. These K models are referred to *individual* model. A variable m is introduced to perform the selection over the indiviual models. m is a normalized vector of length K with every element $m_k \in \{0, 1\}$. The individual models are indexed and represented as $p(s_k, y_k|m_k = 1)$. A mixture model, containing all individual models is then defined as [17]

$$\begin{aligned} p(s, y, m) &= p(m) \prod_{k=1}^{K} p(s_k, y_k|m_k = 1)^{m_k} \\ &= p(m) f_o(s_o, y_o) \prod_{k=1}^{K} \left(\frac{p(s_k, y_k|m_k = 1)}{f_o(s_o, y_o)} \right)^{m_k} \end{aligned} \quad (10)$$

This equation indicates that the overlapping part of all the individual models can be isolated and treated separately. The step taken in equation 10 can be visualized as is done in figure 6. The left part indicates the overlapping part of K individual models which are combined in the right part using the mixture model.

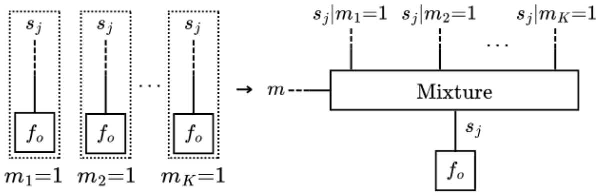


Fig. 6. The left part indicates the overlapping part of K individual models which are combined in the right part using the mixture model. This figure is inspired by a figure presented in [17].

The exact definition of the messages sent along this factor graph are not given here. They can be found in [17]. Conceptually, the message sent over s_j to f_o is a weighted sum of the messages sent over $s_j|m_k$=1 to the mixture node. The weights in this sum are the values of m_k. The real power of the mixture model is that messages are also defined such that m is adjusted to give greater weights to models that provide a better explanation of the variable s_j.

On its own, the mixture node can be used to formulate a model capable of performing inference in a setting where the number of objects is known. Such a model, capable of inferencing over two one-dimensional objects, is constructed below. The resulting model serves as a base model which is extended with structure learning to deal with an unknown number of objects. In this model a measurement will only contain information about the position of one object and it is unknown from which object this measurement originated. The factorization of an individual probabilistic model for a single object can be written as $p(x_i, y) = f_a(x_i) f_b(x_i, y)$, where x_i represents the position of object i and y a measurement about this position. If then $f_a(x_i)$ is taken to represent a prior believe about the position of an object $p(x_i)$, and $f_b(x_i, y)$ describes the likelihood distribution $p(y|x_i)$, the individual models can be described as

$$\begin{aligned} p(x_1, y|m_1{=}1) &= p(y|x_1, m_1{=}1)p(x_1) \\ p(x_2, y|m_2{=}1) &= p(y|x_2, m_2{=}1)p(x_2) \end{aligned} \quad (11)$$

Note that thus far there are no shared factors, as should be the case for a system where neither the objects themselves, nor the measurements influence each other. The shared factor is introduced by the act of taking the measurement, and the uncertainty it introduces about the origins of the measurement. Including the factor function responsible for describing this measurement, the observed individual models can be written as

$$\begin{aligned} p(x_1|m_1 = 1, \hat{y}) &\propto \delta(y - \hat{y}) p(y|x_1, m_1 = 1) p(x_1) \\ p(x_2|m_2 = 1, \hat{y}) &\propto \delta(y - \hat{y}) p(y|x_2, m_2 = 1) p(x_2) \end{aligned} \quad (12)$$

These models have an overlapping observation factor. The mixture model as defined in equation 10 also require a prior belief about our selection variable. In this case, the selection variable models the probabilities of the measurement belonging to either object 1 or 2. Since it is unknown from which object the measurement originated $p(m_1{=}1) = p(m_2{=}1) = 0.5$. This prior can be described by a categorical distribution as $p(m) = \text{Cat}([0.5, 0.5])$ such that $m_1 = m_2 = 0.5$. Following equation 10, the probabilistic mixture model for this example can be formulated as.

$$\begin{aligned} p(x_1, x_2, m|\hat{y}) \propto \delta(y - \hat{y}) p(m) [(p(y|x_1, m_1 = 1)p(x_1)]^{m_1} \\ \cdot [p(y|x_2, m_2 = 1)p(x_2)]^{m_2} \end{aligned} \quad (13)$$

A factor graph of this mixture model is presented in figure 7. This mixture model is capable of inferring the likelihood of a measurement belonging to a certain objects through the calculation of $p(m|\hat{y})$, as well as calculating the posterior believes about the positions of the objects $p(x_1|\hat{y})$ and $p(x_2|\hat{y})$.

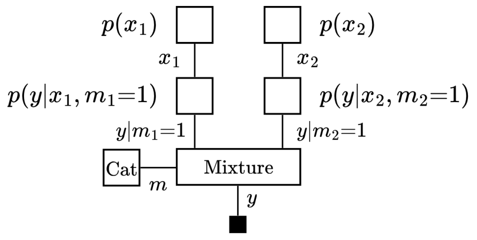


Fig. 7. This figure displays a factor graph which utilizes a mixture node to relate a single measurement to two objects.

B. Infinite Mixture

In general, structure learning allows not only for inferencing about individual model parameters, but it additionally allows for the inferencing of the model structure itself [16]. What will follow is an explanation that is limited to the conceptual workings of a structure learning method as presented in [16].

Fundamentally, the structure learning method describes an infinite mixture model. It combines this mixture model over an infinite amount of clusters with the update rule associated with the Dirichlet process and a constraint upon the selection variable. To make it feasible to handle an infinite amount of clusters, all the clusters are considered either *active* or *inactive*. This categorization allows for all the inactive clusters to be grouped together and be treated as a single cluster. In this context, the process of structure learning can be thought of as moving a cluster from the inactive group to the active group, thereby increasing the number of detected clusters by one.

The first part central in the ability to increase the number of active clusters is the Dirichlet process. The model discussed in [16] reformulates the selection variable into a cluster assignment probability c_n. It is modelled by a categorical distribution given some event probabilities π: $p(c_n|\pi) = \text{Cat}(c_n|\pi)$. These event probabilities π are in turn distributed according to a Dirichlet distribution. The update rules of the Dirichlet distribution effectively turn π into an occurrence counter. An example value of π could be $[2, 1, \alpha]$. This would indicate that the cluster associated with the first index was observed twice and the cluster associated with the second index is only observed once. Special attention is needed for the value represented by α. This is a concentration parameter related to the chance of a measurement belonging to the group of inactive clusters. The value of α therefore influences how likely the model is to activate a new cluster. Should a measurement activate another cluster, the value of π would change to $[2, 1, 1, \alpha]$.

The second important part of the structure learning model is a constraint which describes that each measurement is only capable of being assigned to a single cluster. This is enforced by setting a constraint on the approximate marginal distribution $q(c_n)$

$$q(c_n) = \delta[c_n - e_k]$$
$$\text{s.t. } k = \arg\max_k \overrightarrow{\mu_{c_n}}(c_n = e_k)\overleftarrow{\mu_{c_n}}(c_n = e_k) \quad (14)$$

This constraint ensures that inference is only performed for the cluster which best explains the measurement. It is a necessary constraint to limit the complexity of the clustering problem with an unknown number of objects. Should it not be enforced, the non-zero probability of any measurement belonging to the group of inactive clusters would be enough to activate a new cluster. One would end up with just as many clusters as measurements.
Finally, to make inference possible, a prior over the parameters of inactive clusters must be defined. This prior is sometimes called the base distribution and is notated as G.

V. Method

The previous sections discussed most of the necessary prior knowledge to the model that will be introduced here. This model comprises of two structure learning models linked by a single cluster assignment variable. The model will be evaluated by comparing its performance on different scenes. This section will provide a detailed description of the model as well as elaborate on the test process.

A. The Model

As briefly stated, the current implementation of the model consists of two structure learning models side by side. This means that one cluster selection variable is used for both models. Each individual structure learning model handles a different attribute of the objects, in this case the two coordinates of the objects on our image plane u and v. A full factor graph of the model with K activated clusters is visualized in figure 8.

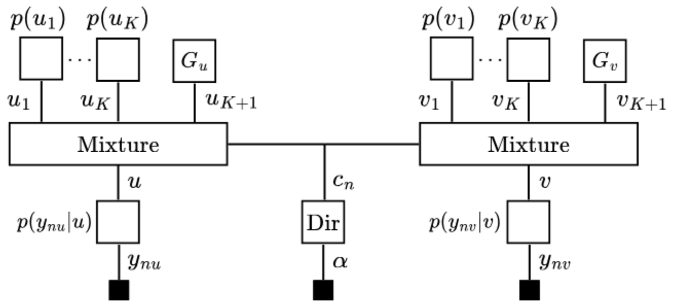


Fig. 8. This figure is a factor graph of the model that will be used for clustering when K clusters are activated. It is composed from two structure learning models as discussed in section IV, which share the selection variable c_n. In this graph u_k and v_k belong to the two coordinates of the same cluster k.

In this model, u_k and v_k represent the two coordinates of the same cluster k. Prior believes about these coordinates are written as $p(u_k)$ and $p(v_k)$. The base distributions are marked by G_u and G_v. The likelihood functions $p(y_{nu}|u)$ and $p(y_{nv}|v)$ are of the form $\mathcal{N}(u, z_u)$ and $\mathcal{N}(v, z_v)$ respectively. Here, $\mathcal{N}$ is taken to be a normal distribution defined by a mean and variance. Lastly, the cluster selection variable c_n has the same definition and dependency on α as explained in section IV-B. The specific values of z_u, z_v and α, as well as the distributions G_u and G_v will be given in section V-B.

Two distinct design choices need some motivation. To start, the comparison of a sub graph containing a single mixture node from figure 8 and the graph presented in figure 7, reveals a slight discrepancy. As mentioned, figure 7 represents a mixture model to infer the positions of two one-dimensional objects. The observation was made that the overlapping part of the individual models is the factor function describing the observed value. However, figure 8 displays a graph where the location of the mixture node in the graph serves as an indication of overlapping likelihood functions in the individual models. This seems to be in contradiction with the claims made in section IV-A. To clarify, it is the enforcement of the constraint formalized in equation 14 that effectively causes the mixture node to collapse into a single individual model. As mentioned, this ensures that a measurement only affects the selected cluster. It is this behaviour that makes it possible to consider an overlapping likelihood function and in doing so simplify the model.

A second design choice that needs motivation is the use of two structure learning models. The two models share identical graphs, for they describe two identical phenomena: each represents the inference over a single coordinate. If multivariate versions of the distributions were used this would greatly simplify the graph. This is how the original paper introduced the structure learning method [16]. In order to demonstrate the flexibility of the method, the choice was actively made not to do this. If a single cluster selection variable is able to be used for two structure learning models, each representing a distinct attribute of an object, future works have a straight forward method of increasing the number of attributes. Section VIII will continue with this train of thought.

B. Test Process

A description of the test process requires the discussion of four different elements: the scene selection, data generation, model parameters and performance metrics. This section provides the details for these elements in addition to a fifth section which describes the additional testing done to evaluate the model behaviour for increased levels of noise.

1) Scene Selection & Data Generation: Testing the method is done using the five different scenes presented in figure 9. These scenes where chosen to represent a wide range of challenges for the model. Scene 1 is chosen as an easy scene. The objects are well separated and all lie on a 2D plane orthogonal to the viewing direction. Scenes 2 and 3 are chosen for the variety in object sizes they contain with different levels of object sparsity. Scene 4 is included to present a chaotic scene with many overlapping objects. Finally, the fifth scene is included for the many objects it contains.

From these scenes, the data is generated as described in section II. The data generation method requires a transformation

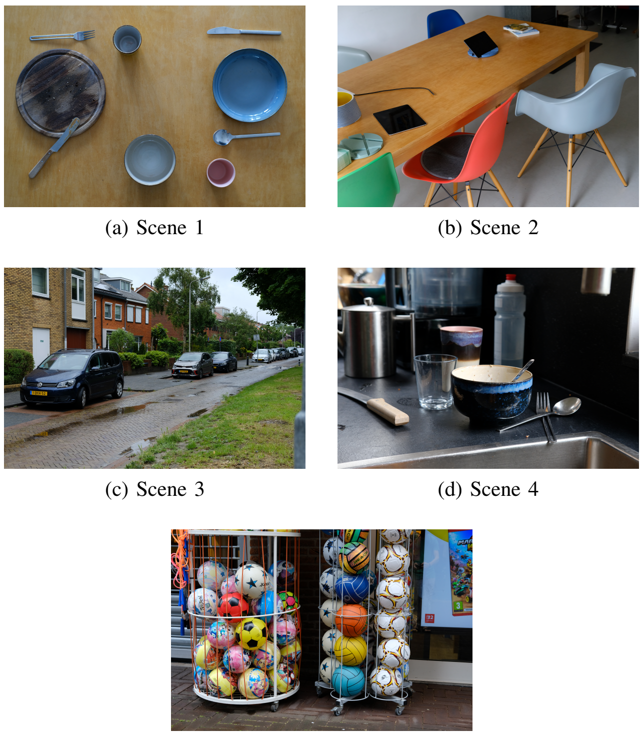

(a) Scene 1
(b) Scene 2
(c) Scene 3
(d) Scene 4
(e) Scene 5

Fig. 9. The five different scenes used for evaluating the model.

to be known between the patch coordinates and the scene coordinates. Within the context of robotics, this is analogous to transforming measurements to a world frame using odometry data. To mimic uncertainties in the odometry data, three different levels of Gaussian distributed noise are added to the known and exact transformation during data creation. These levels are: no noise, noise with a standard deviation of 20px and noise with a standard deviation of 50px. Any future reference of "added noise" refers to this process of adding noise to the transform between coordinates in the patch frame and coordinated in the original image frame. The data for each scene was created using 100 randomly located patches.

2) Model Parameters: To allow for a more general definition of the parameters mentioned in section V-A, the coordinates of the scene are scaled such as to have values in the range $[0, 1]$. Then $G_u = G_v = \mathcal{N}(0.5, 1)$ for the uninitialized clusters. A uniform distribution on the interval $[0, 1]$ would be a more appropriate choice for G_u and G_v, but using a normal distribution allows for the solution to be an exact Bayesian solution as mentioned in III. The measurement noise parameters z_u and z_v are adjusted for performance depending on the scene without added noise. The discussion in section VII provides a motivation for why this is done. The parameter α as introduced in section IV was set at a value of $10^{\frac{1}{100}}$. Some pretesting indicated that this value yielded good results. The scene did not seem to matter for a good value of α.

3) Performance Metrics: To measure the performance of the model, the inference results are compared to the objects detected by running YOLO over the entire image. These detected objects are considered to be the ground-truth objects. The comparison between the inference results and the ground-truth objects is done in two steps. Firstly, subsets of centers from inferred clusters and ground-truth objects are categorized in three categories: ”correct”, ”merged” and ”split”. A correct cluster indicates that a corresponding pair is found between the inference results and the ground-truth. A merged cluster indicates that measurements originating from two or more ground-truth objects are inferred to belong to a single cluster. This setting is scored by counting one correct cluster and one or more merged clusters. A split cluster indicates that measurements form one ground-truth object resulted in multiple clusters during inference. This setting is scored by counting one correct cluster and one or more split clusters. The relative occurrences of these categories are presented. In the second step, the distance between each pair of inferred cluster center and ground-truth object is calculated. The distribution of these distances is presented with a boxplot and serves as a secondary indication of model performance.

4) Noise Dependency: In addition to what is described above in section V-B1, the performance metrics are evaluated for additional settings to observe how the model performs for different levels of added noise. To this end scenes 3 and 4 are scored with the additional noise levels of 75, 100, 125, 150 and 200px. The hypothesis is that adding noise will gradually decrease the performance of the model.

VI. Results

The results consists of three parts. To start, the inference results for four scenes is presented. These inference results contain the inferred clusters as well as a visualization of which data points are assigned to which clusters. Secondly, the performance metrics are visualized for each setting mentioned in section V. To aid in the interpretation of these performance metrics, additional figures which include ground-truth detections are provided. Finally, the results for testing the noise dependency of the model are visualized.

A. Inference Results

Figures 10 to 13 display the results of the clustering method on the four selected scenes. Each figure contains the data points, coloured such that points assigned to the same cluster have the same colour. The center of each cluster is marked with a black cross. Each figure has an additional black cross in the exact center of the image with no points assigned to it. This marks the center of the distribution describing the uninitiated clusters. The reported number of identified clusters will not include this uninitiated cluster.

Figure 10 shows a scene of some parked cars along a street. It can be seen that 6 distinct clusters are identified. The three closest cars as well as a traffic sign above the third car are clustered correctly. All measurements originating from cars further down the street are assigned to the same cluster. This is an example of merged clusters.

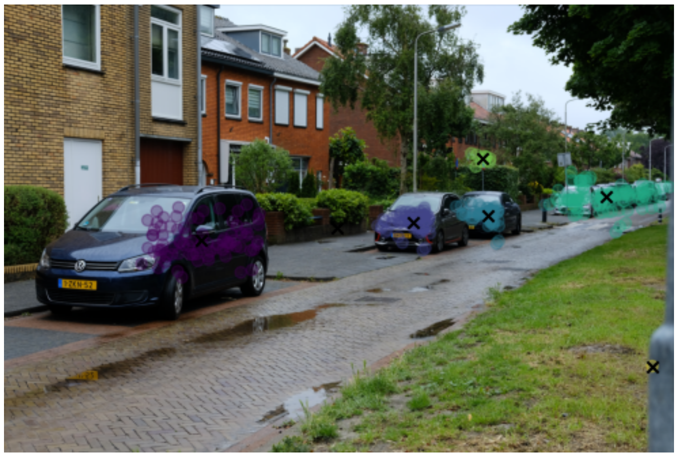


Fig. 10. A scene of parked cars along a street. The added noise on the transform between the patch coordinates and the scene coordinates has a standard deviation of 50px. The measurement uncertainty during inference is set at $1/1400$ relative to the image size. 6 Clusters are identified during inference.

Figure 11 displays a scene of some dishes next to a sink. During inference 12 clusters are found. Remarkably, the spoon sticking out from the bowl is correctly assigned a separate cluster from the bowl itself. The fork and spoon lying on the counter are separated less well and measurements originating from the glass are split into two clusters. The cluster identified by the yellow points in the bottom right corner belong to the sink.

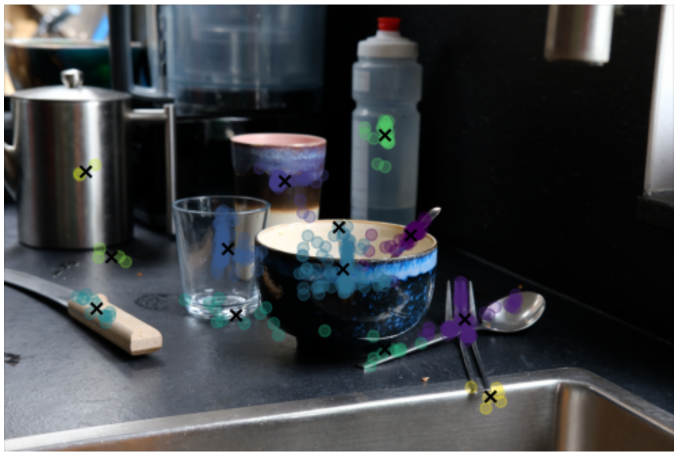


Fig. 11. A scene of some dirty dishes next to a sink. The added noise on the transform between patch coordinates and scene coordinates has a standard deviation of 20px. The measurement uncertainty during inference is set at $1/2000$ relative to the image size. 12 Clusters are identified during inference.

Figure 12 shows the scene of a dinner table as seen from the top down. 13 Clusters are identified during inference. Firstly, no measurements seem to originate from the left plate. Secondly, a lot of measurements seem to be centered in the middle of the image. This can be explained by YOLO detecting an entire patch as a ”dinner table”. Data points originating from this ”dinner table” are split over six clusters. Finally, Some clusters seem to be sharply defined such that only measurements originating form objects associated with these clusters are assigned to them. This seems to be the case for the fork, cup and knife located in the upper part of the figure. Other clusters seem to be less sharp. They seem to include measurements belonging to other objects than the object with which the cluster is associated. This can be observed for the knife, bowl, cup and spoon located in the bottom part of the figure.

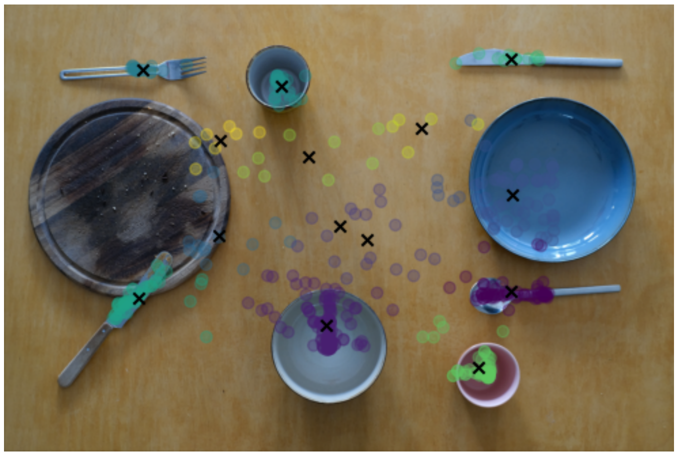


Fig. 12. A scene of a set dinner table as seen from the top down. The added noise on the transform between patch coordinates and scene coordinates has a standard deviation of 20px. The measurement uncertainty during inference is set at $1/800$ relative to the image size. 13 Clusters are identified during inference.

Figure 13 shows the scene of some football-sized balls stored in racks. During inference 39 clusters were identified. This figure shows inference in scene with a much larger number of objects than the other scenes. What is interesting in this particular figure, is that some partially occluded balls in the rack are also correctly clustered.

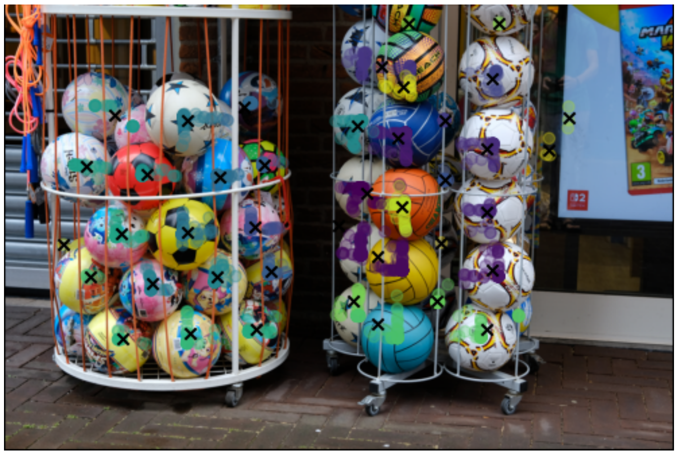


Fig. 13. A scene of two racks filled with football-sized balls. There is no noise added to the transform between patch coordinates and the scene coordinates. The measurement uncertainty during inference is set at $1/5000$ relative to the image size. 39 Clusters are identified during inference.

B. Performance Metrics

The performance metrics evaluated for each of the five scenes with the discussed three levels of noise are given in figure 14. The relative occurrences of clusters marked ”correct”, ”merged” or ”split” are displayed in the top graph. The bottom graph contains several box-plots to represent the distribution of distances between inferred clusters marked ”correct” and the corresponding ground-truth objects. It can be observed that each scene obtains similar scores, seemingly not directly correlated with the amount of noise added. The best results were obtained for scenes 4 and 5. The model scored worst on scenes 2 and 3. To aid in the interpretation of figure 14 as will be done in section VII, figure 15 displays both the locations of inferred clusters, represented by the blue dots, as well as the location of ground-truth detected objects, represented by the red dots. Most pairs of inferred clusters and ground-truth objects are in close proximity of each other. There are some inferred clusters for which no corresponding ground-truth object is detected.

C. Noise Dependency

The results of tests done to determine the performance of the model for different amounts of noise added are presented in figure 16. For both scenes, performance metrics start to worsen for added noise with a standard deviation greater than 50px. The relative occurrences of clusters labelled ”correct” decrease with a simultaneous increase in the number of clusters labelled ”split”. In addition to this, the distributions describing the distances between clusters labelled ”correct” and ground-truth objects seem to gravitate to higher values as the amount of added noise is increased.

VII. Discussion

The results from section VI showcase the method being applied to a broad range of scenes. Considering this method performs 2 dimensional clustering about objects placed in a 3 dimensional world, the results can be interpreted to be surprisingly good. Figures 10 to 13 show that obvious clusters are correctly identified. Evidently the model is capable of handling scenes with multiple objects, proving the usefulness of the structure learning model. Some remarks on the performance indicated by the results are worth discussing. The results of the noise dependency tests are discussed at the end of this section.

The first remark is about the effect of hallucinated objects on the model. It is known that YOLO sometimes hallucinates the existence of objects [18], and more obviously, sometimes misses them. Though missed object detections are hard to identify and possibly easily solved by multiple observation over roughly the same area, hallucinations are harder to detect and solve. These hallucinations are present in the data and lead to the activation of new clusters or are assigned to existing clusters. Figure 11 shows an example of a cluster being initiated due to hallucinations in the left part of the figure between the knife and the modern style teapot. Something similar happens in figure 13 with the most right clusters in

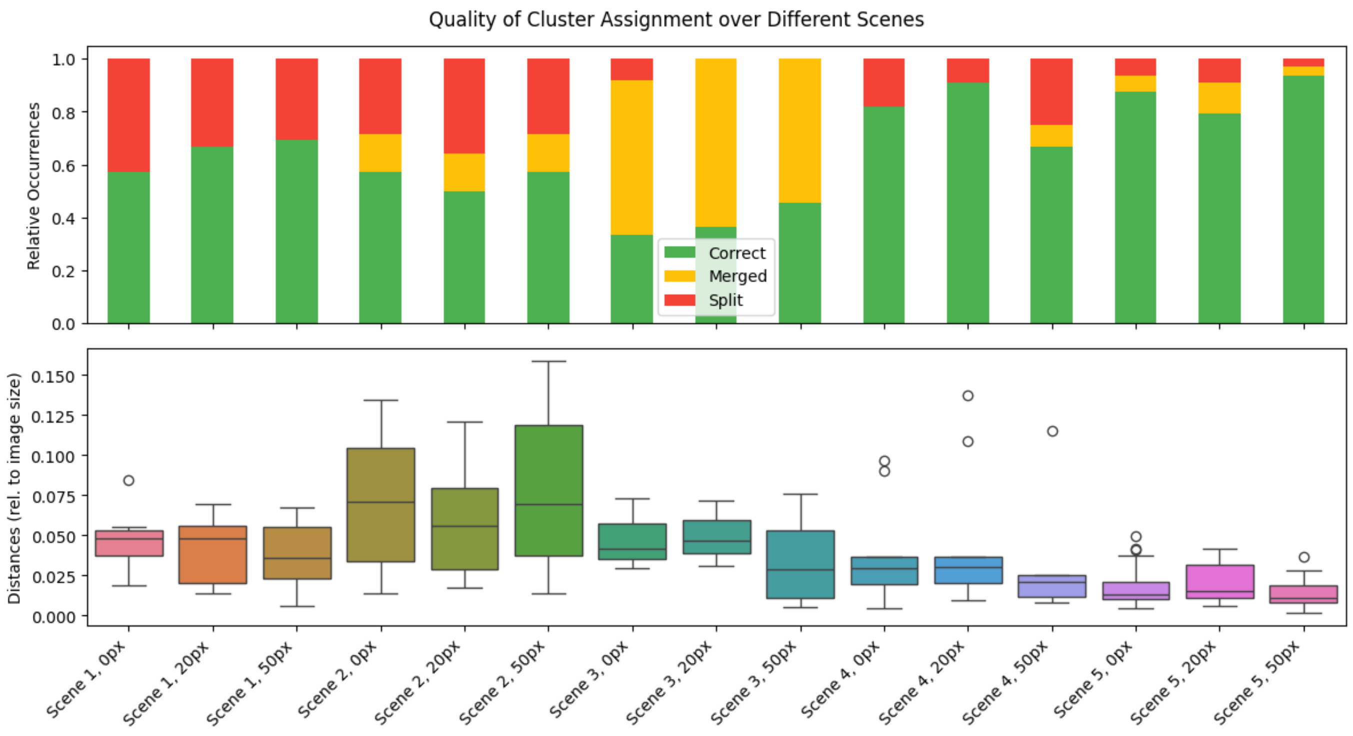


Fig. 14. The performance metrics evaluated for each scene and noise level as indicated by the labels on the horizontal axis. The top graph visualizes the relative occurrence of correctly inferred clusters as well as merged and split clusters. A merged cluster indicates that two ground-truth objects are inferred to belong to a single cluster. This setting is scored by counting one correct cluster and one merged cluster. A split cluster indicates that one ground-truth object is resulted in multiple clusters during inference. This setting is scored by counting one correct cluster and one split cluster. The bottom graph displays the distributions of distances between pairs of corresponding ground-truth objects and clusters marked ”Correct”. The distance is presented relative to the image size.

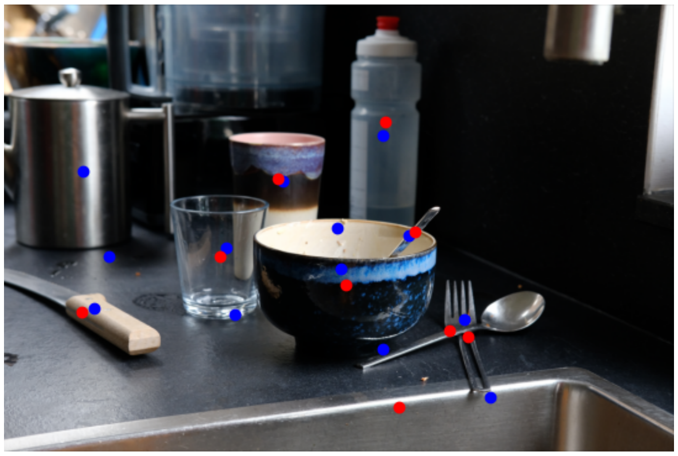


Fig. 15. The inference results of scene 4 together with the ground-truth object detection. The centers of the inferred clusters are represented by blue dots. The centers of the ground-truth detected objects are presented in red. The add added noise on the transform between patch coordinates and scene coordinates has a standard deviation of 20px. An additional blue dot is present in the center of the scene. This dot represents the uninitiated clusters.

the figure. Since these hallucinations originate from YOLO, the model technically still does a good job when clustering them together. Section VIII will briefly discuss the desirable behaviour of the model to be able to distinguish hallucinated cluster from actual objects.

The second point worth discussing relates the definition of measurement uncertainty with object size in combination with occlusion. Traditionally, measurement uncertainty is a measure describing a statistical upper bound on the difference between a reported value and an actual value [19]. But within the context of the method and application described in this paper, some ambiguity seeped in. A traditional interpretation still holds: when YOLO detects an object, creates a bounding box and then reports the center of this bounding box, the reported center of the object will deviate from the actual center of the object. Consecutive measurements, each with minor insignificant changes, will report a value close to, but not exactly the same as the previous reported value. But with the introduction of partial observations and when dealing with finitely small objects, a second interpretation of measurement uncertainty is introduced. This is best explained by an example. Lets look at figure 10 and specially the car on the left. In some patches the entire car will be visible, with a reported center located roughly in the middle of the car. In other patches either the front or the rear of the car will be occluded, resulting in the reported centers being shifted to one of both sides. Thus introducing a secondary, much greater, spread of measurements which is directly related to the apparent size of the object. The current implementation

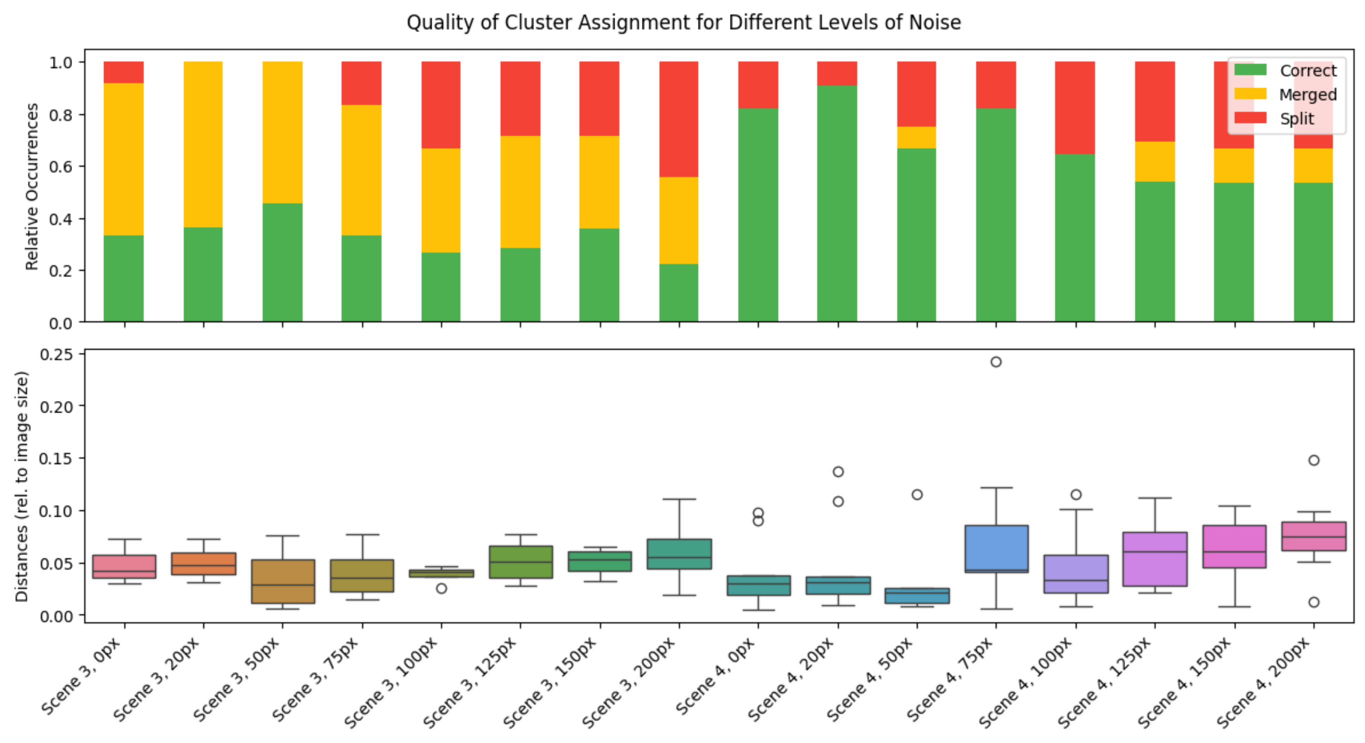


Fig. 16. The performance metrics evaluated for scenes 3 and 4 were noise is added to the transform with a standard deviation varying between $0 - 200$px. The top graph visualizes the relative occurrence of correctly inferred clusters as well as merged and split clusters. A merged cluster indicates that two ground-truth objects are inferred to belong to a single cluster. This setting is scored by counting one correct cluster and one merged cluster. A split cluster indicates that one ground-truth object is resulted in multiple clusters during inference. This setting is scored by counting one correct cluster and one split cluster. The bottom graph displays the distributions of distances between pairs of corresponding ground-truth objects and clusters marked ”Correct”. The distance is presented relative to the image size.

sets one measurement uncertainty for all measurements within a scene, which imposes a limitation that is clearly visible in figure 12. The dinner table in figure 12 is much larger than any other object in the scene, thus the spread in its measurements is much greater. When setting the measurement uncertainty such that the smaller objects are well clustered, the measurements of the table are incorrectly spread over multiple clusters. Section VIII will provide a conceptual solution to the problem this poses.

Finally, two main observations can be made from figure 14: performance varies greatly between scenes and there is hardly any difference in performance for the same scene at different noise levels. Coincidently, the previous topic of discussion provides a suitable explanation for both of these observations. The first observation leads to the belief that the performance in a scene is related to the variance in object size present within this scene. The scenes that resulted in the worst performance, scenes 3 and 4, were the scenes with the greatest variance in objects. In contrast, scenes 4 and 5 resulted in the best performance and contained much less variance in object sizes. The second observation might be a sign that the added noise in the transform is relatively small compared to the variance in measurements introduced due to the size of objects.

A. *Noise Dependency*

The results form figure 16 fall in line with the expectation of gradually decreasing performance as the amount of added noise increases, but only partially. As is discussed previously and already visibly in figure 14, the performance seems to be steady for the first 2 or 3 settings of each scene. Only after this does the performance start to degrade. The results of this experiment are inconclusive. Further research is warranted for a more comprehensive understanding of the models performance in the presence of noise.

VIII. Limitations and Future Works

Looking back on the method discussed in this paper in combination with the obtained results, nothing more than a critical review and suggestions for improvement are suitable. This section will facilitate this review and provide some speculation on how dynamics might be introduced to this model.

A. *Model Complexity*

A clear limitation in the method as described in section V is the fact that only the two dimensional position of the objects is taken into account for clustering. Rather unsurprisingly, and verified by the results, objects cannot be abstracted down in to a two dimensional position and still be consistently separated into distinct objects. As discussed in section VII, this clustering method clearly struggles when objects vary in size or get to close to one another. The argument that is made here, is that both of these issues might be solved by increasing the model complexity.

To start, increasing the number of attributes which define an object might directly correlate to a better performance. In the clustering sense, every added attribute would add another axis over which objects can be different. This could increase the distance between cluster centers allowing for a more accurate reconstruction. Some obvious attributes to add would be the size, type and colour of an object. During the work on the results presented in this paper, extensive efforts were made to allow for the inference over discretely distributed attributes. This would allow for the model to be extended with an attribute for the objects' class. The results of these efforts are presented in the appendix.

In addition to this, the likelihood function can be adjusted for better performance when dealing with objects of different sizes. The observations in section VII clearly state that there is a relation between object size and the spread of the measurements. If the size attribute is added to the model, the likelihood models for the position measurements could be adjusted to take into account the believe about the size of the object.

B. Dynamics

The inclusion of dynamics might come with some challenges. A question that arises might be: if in the stationary case the problem is about identifying clusters, how would this translate to a dynamic environment? These clusters would become traces describing a trajectory throughout the space created by the span of all considered attributes. There would need to be a prior believe about where the trace is expected to go which would depend on where the trace has been observed and on other traces around it.

This may paint a mental image of how the problem is altered by the addition of dynamics, but it does not provide any practical guidance on how to approach it. For this, work done in the field active inference might prove useful. The active inference framework, as described in [6], already implements dynamic systems using generalized coordinates [21]. With the use of multi variate distributions, the mathematics of generalized coordinates might be applied in a factor graph based solution.

IX. Conclusion

This paper set out to provide a method which is capable of creating and object centered world description based upon consecutive measurements about an unknown number of static objects using Bayesian inference. This was successfully achieved to a varying degree. The results show that the structure learning model discussed in section IV is fundamentally the right tool for the job as obvious clusters are often correctly inferred. This also implies that the analogy between the problem statement and a clustering problem seems to hold. The conclusion can be drawn that the structure learning model provides an elegant Bayesian technique for this inference problem due to the usage of the message passing algorithm. However, a more critical analysis of the results, as is performed in section VII, seems to indicate that the specific implementation of the structure learning model evaluated in this paper has oversimplified the representation of objects. Using only two dimensional coordinates as the measurements results in mediocre performance when objects are located in close proximity of each other or when the apparent size of the objects differs to much. The value of this paper is therefore in the predicted ease of expanding this model to accommodate for more attributes associated with objects. Finally, section VIII shares some thoughts on how model complexity can be increased with the expected result of improved performance.

References

[1] Bagaev, D., Podusenko, A., & De Vries, B. (2023). RxInfer: A Julia package for reactive real-time Bayesian inference. Journal of Open Source Software, 8(84), 5161.

[2] Vermeulen, M. (2020, May 21). Dit is een oplossing voor vergrijzing, armoede én mensensmokkel. Maar bijna niemand wil eraan. De Correspondent. https://decorrespondent.nl/11262/dit-is-een-oplossing-voor-vergrijzing-armoede-en-mensensmokkel-maar-bijna-niemand-wil-eraan/d846ef45-05a5-0f72-359e-5e4ecea4a6d6

[3] Bogue, R. (2016). Growth in e-commerce boosts innovation in the warehouse robot market. Industrial Robot: An International Journal, 43(6), 583-587.

[4] U.S. Department of Labour (2023). US Department of Labor finds Amazon exposed workers to unsafe conditions, ergonomic hazards at three more warehouses in Colorado, Idaho, New York. https://www.dol.gov/newsroom/releases/osha/osha20230201-0

[5] De black box van Tesla - Zembla - BNNVARA. (n.d.). Zembla. https://www.bnnvara.nl/zembla/artikelen/de-black-box-van-tesla

[6] Friston, K., Kilner, J., & Harrison, L. (2006). A free energy principle for the brain. Journal of physiology-Paris, 100(1-3), 70-87.

[7] Pezzato, C., Ferrari, R., & Corbato, C. H. (2020). A novel adaptive controller for robot manipulators based on active inference. IEEE Robotics and Automation Letters, 5(2), 2973-2980.

[8] Khanam, R., & Hussain, M. (2024). Yolov11: An overview of the key architectural enhancements. arXiv preprint arXiv:2410.17725.

[9] Wainwright, M. J., & Jordan, M. I. (2007). Graphical models, exponential families, and variational inference. Foundations and Trends® in Machine Learning, 1(1–2), 1-305.

[10] Bagaev, D. V. (2022). Reactive Probabilistic Programming for Scalable Bayesian Inference.

[11] Bagaev, D., & de Vries, B. (2023). Reactive message passing for scalable Bayesian inference. Scientific Programming, 2023(1), 6601690.

[12] Yedidia, J. S., Freeman, W. T., & Weiss, Y. (2003). Understanding belief propagation and its generalizations. Exploring artificial intelligence in the new millennium, 8(236–239), 0018-9448.

[13] Yedidia, J. S., Freeman, W. T., & Weiss, Y. (2005). Constructing free-energy approximations and generalized belief propagation algorithms. IEEE Transactions on information theory, 51(7), 2282-2312.

[14] Winn, J., Bishop, C. M., & Jaakkola, T. (2005). Variational message passing. Journal of Machine Learning Research, 6(4).

[15] Winn, J. M. (2004). Variational message passing and its applications.

[16] van Erp, B., Nuijten, W. W., & de Vries, B. (2024, September). Online Structure Learning with Dirichlet Processes Through Message Passing. In International Workshop on Active Inference (pp. 91-104). Cham: Springer Nature Switzerland.

[17] van Erp, B., Nuijten, W. W., van de Laar, T., & de Vries, B. (2023). Automating model comparison in factor graphs. Entropy, 25(8), 1138.

[18] He, W., Wu, C., Cheng, C. H., Huang, X., & Bensalem, S. (2025). Mitigating hallucinations in YOLO-based object detection models: A revisit to out-of-distribution detection. arXiv preprint arXiv:2503.07330.

[19] Hughes, I., & Hase, T. (2010). Measurements and their uncertainties: a practical guide to modern error analysis. OUP Oxford.

[20] Lee, M. D., & Cummins, T. D. (2004). Evidence accumulation in decision making: Unifying the “take the best” and the “rational” models. Psychonomic bulletin & review, 11(2), 343-352.

[21] Balaji, B., & Friston, K. (2011). Bayesian state estimation using generalized coordinates. Signal processing, sensor fusion, and target recognition XX, 8050, 716-727.

APPENDIX

This appendix describes the steps taken to expand the model to handle objects that contain a discretely distributed attribute such as the objects’ class. Before adding a new mixture node to the structure learning model, individual models capable of performing inference over a single discretely distributed variable have been explored. Two such models will be discussed. One model represents a naïve approach, where as the other describes a more complex approach capable of modelling structured class confusion.

In its most basic form, a measurement about the class of an object contains a single value, i.e. measuring either ”bike” or ”chair”. A categorical distribution can be used to describe distribution over possible measurement. The shape of this categorical distribution is of course determined by the class of the measured object.

This is the extent of the naïve model. A variable l is used to describe the class, or label, of the object, which influences the measurement y_l as $p(y_l|l) = Cat(y_l|l)$. A prior is needed for the variable l. This prior is modelled by a Dirichlet distribution, for this is a natural prior of a categorical distribution. Figure 17 shows the factor graph of the naïve model.

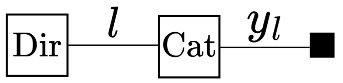


Fig. 17. The factor graph of the naïve model for inference over discretely distributed variables. The observable variable y_l represent a measurement of the l. In turn, the variable l represents the discrete variable of interest, possibly associated with the class, or label, of an object.

This model is considered naive for it does not explicitly model the variance introduced by the means of measurement. Say for example that the method used for measuring the class of the objects is known to confuse the classes ”bicycle” and ”motorbike”. Then measuring ”bicycle” should simultaneously make it more likely to assume that the object is a ”bicycle” or a ”motorbike”.

The second model is capable of taking into account this structured class confusion. It achieves this by implementing a mixture node over the variables β_1, …, β_N as is seen in figure 18. Each β_n describes the expected distribution of measured classes if the true class is identified by index n. Together, all these variables effectively describe the confusion matrix of the class detection model. As a whole, the mixture node can be though of as to produce the expected spread of measurements given the selection variable l. This selection variable can then be interpreted as the class of the objects.

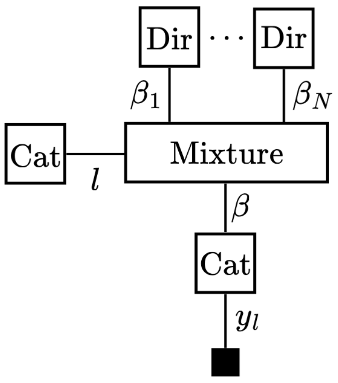


Fig. 18. The factor graph of a more complex model for inference over discretely distributed variables. It utilizes a mixture node over the variables $\beta_1, \ldots, \beta_N$ to allow for the modelling of the confusion matrix of the method used to make observation y_l. Each β_n describes what the model might confuse the class identified by n with. The mixture node can be though of as to produce the expected spread of measurements given the selection variable l. This selection variable can then be interpreted as the class of the objects.

Implementation of these models was successful, but the addition of them to the structure learning model was not. At the time of writing, RxInfer appears to not yet have the necessary infrastructure for the specific factor graph required. However, an extensive documentation and a helpful community are available to aid in the task of the creation of this infrastructure.