2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Relational Object Tracking and Learning

Davide Nitti[1], Tinne De Laet[2], Luc De Raedt[3]

***Abstract*— We propose a relational model for online object tracking during human activities using the Distributional Clauses Particle Filter framework, which allows to encode commonsense world knowledge such as qualitative physical laws, object properties as well as relations between them. We tested the framework during a packaging activity where many objects are invisible for longer periods of time. In addition, we extended the framework to learn the parameters online and tested it in a tracking scenario involving objects connected by strings.**

I. INTRODUCTION

Robots that can help humans in everyday tasks have to be able to estimate the world state during manipulation tasks and human activities. This is important for decision making as well as learning. In this paper we investigate object tracking, a hard task as one needs to recognize the objects and estimate the pose in the scene. On top of this, the objects may be invisible for a longer period of time. For example, objects inside a moving box will be hard to track unless we can use and encode knowledge about the concept "inside", the object type "container" and how these containers interact with other objects over time. For example, if we put objects in a container, they will follow the container's position and they will fall on the floor whenever we rotate the container. A motion model that encodes such knowledge can better track hidden objects and is helpful to determine whether an appearing object is new or is one of the hidden objects that already existed in the belief state (data association problem). This principle is extendable beyond tracking. Indeed, knowledge about human activities, objects, places, their properties and how they are related, can help the robot to understand human activities, make complex plans and communicate in natural language.

Relational representations, based on first-order logic, can be used to define such background knowledge as they allow to naturally describe objects, properties, relations and implication rules. Probabilistic programming languages [1] and statistical relational learning techniques (SRL) [2] have combined such representations with uncertainty reasoning, and have been successful in many application areas ranging from natural language processing to bioinformatics. Even though there are many formalisms described in the statistical relational learning and probabilistic programming literature, only few of these representations have been applied to robotics, especially in an online setting. The main challenges are the computational cost and the discrete nature of most of the existing relational languages. Indeed, it is unsurprising that a richer representation has a higher computational cost; so, the challenge is to find a compromise between expressiveness and performance. The second issue is that most probabilistic relational languages are restricted to discrete (often even binary) random variables. In addition, there is the symbol grounding problem that needs to be tackled: relational frameworks define and manipulate symbols and relations, we need to ensure that these symbols correspond to physical objects in the real world and that the relations capture the intended meaning.

Recently, probabilistic relational languages that support continuous and discrete random variables have been proposed [3], [4], [5]. The models developed in this paper are based on a relational probabilistic language called Distributional Clauses (DC) [3] and its dynamic extension DDC [6]. Inference is performed using the Distributional Clauses Particle Filter (DCPF). This framework exploits (context specific) independence assumptions in the relational model to speed up inference and supports a growing state space.

Building on that work, our first contribution in the present paper is a model for online object tracking that exploits relational knowledge, and that is applied to an object packaging scenario. The knowledge about the world (such as object types and spatial relations) is used to encode qualitative physical laws in the state transition model. This type of representation can also help to make belief states interpretable for humans.

As a second contribution, we investigate the use of online learning techniques for the DCPF framework. In particular, we show how to integrate online state estimation with parameter learning. This is useful to estimate parameters of the model or static properties, without manual tuning. We test the learning strategies in the "string scenario", an object tracking scenario where the objects are connected by strings, and where we need to estimate the objects' positions together with the string lengths.

This paper is organized as follows: we first review the related work on tracking and relational approaches in robotics (Section II), introduce background notions of logic programming and the DCPF framework used in the paper (Section III). Then we describe online learning algorithms in Section IV. After that we describe the packaging scenario (Section V) followed by the proposed learning algorithms

*This work is supported by the European Community's 7th Framework Programme, grant agreement First-MM-248258.

[1] Davide Nitti is supported by the IWT (Agentschap voor Innovatie door Wetenschap en Technologie). Department of Computer Science, KU Leuven, Belgium. davide.nitti@cs.kuleuven.be

[2] Tinne De Laet was a Postdoctoral Fellow of the Fund for Scientific Research–Flanders (F.W.O.) in Belgium. Faculty of Engineering Science, KU Leuven, Belgium. tinne.delaet@kuleuven.be

[3] Department of Computer Science KU Leuven, Belgium. luc.deraedt@cs.kuleuven.be

applied to the string scenario (Section VI). Finally we present experiments (Section VII) and conclude (Section VIII).

II. Related Work

One of the most used motion models for tracking is the constant Nth-order-derivative model [7]. This model is linear and filtering can be performed with a Kalman filter. A more complex model is the switching state space model [7], where the dynamics can stochastically switch from one regime to another, where each regime is usually a linear model. These simple models are sufficient for several applications, but they are not sufficient to encode knowledge about objects properties and relations between those. Indeed, those models lack a rich representation language such as first-order logic and an inference procedure that exploits this knowledge.

An alternative approach to object tracking uses physics engines to accurately simulate physical laws. However, this solution is not applicable if the behaviour of an object/person is not describable by the physical laws implemented in those engines. In addition, to perform probabilistic inference, e.g. sampling, those physical simulations should be executed many times, making the process slow and preventing online usage. Moreover, physical engines still lack a way to encode relational background knowledge about the world, for example, knowledge to derive spatial relations from object positions, or object categories and their properties. In contrast, a relational framework, such as the one proposed here, is more flexible, and unifies the logical representation of the world and the probabilistic model. In addition, the time performance suffices for online applications (e.g., 0.5-1.5 ms per particle in the tested scenarios). Nonetheless, this flexibility comes with the burden of specifying the background model.

Some relational approaches have been proposed for state estimation and tracking. For example, the relational particle filter of Manfredotti et al. [8], [9] uses relations such as 'walking together' in people tracking to improve prediction and tracking. These results demonstrate the importance of relations in tracking and activity recognition. Nonetheless, the language DC and the DCPF framework are more flexible and general [6]. For example, Manfredotti et al. assume observations conditionally independent of the relations in the state. Moreover, in their approach a relation can be true or false. In contrast, DCPF does not make a real distinction between attributes and relations, i.e., each random variable has a relational representation, regardless of the distribution. This allows parametrization and template definition for any kind of random variable.

The most notable relational languages for robotics are KNOWROB [10] and the physics engine approach for planning [11]. KNOWROB is a framework to integrate knowledge about the world that is useful to perform a variety of tasks. The knowledge is represented in a description logic, the OWL language. To overcome the binary nature of this relational representation, KNOWROB uses computables, arbitrary procedures (even entire programs) that process the robot-internal data structures (e.g., to evaluate the relation on from object pose estimations). In contrast, our approach tries to integrate as much as possible within the same DCPF framework; indeed continuous and complex data structures can be directly defined in DCPF; nonetheless external procedures can be called and integrated as predicates in the language if desired. Another example is [11], which uses a physical engine for prediction and a relational language for knowledge representation and planning.

III. Background

Our representation language and inference procedures are based on logic programming. We now introduce the key notions: A *clause* is a first-order formula with a head, an atomic formula, and a body, a conjunction of atomic formulas or their negation. For example, the clause

$$\texttt{inside}(\texttt{A},\texttt{B}) \leftarrow \texttt{inside}(\texttt{A},\texttt{C}),\texttt{inside}(\texttt{C},\texttt{B})$$

states that for all A, B and C, A is inside B if A is inside C and C is inside B (transitivity property). A, B and C are logical variables.

A *ground atomic formula* is a predicate applied to a list of terms that represents objects. For example, $\texttt{inside}(1,2)$ is a ground atomic formula, where `inside` is a predicate, sometimes called relation, and $1, 2$ are symbols that refer to objects.

A *literal* is an atomic formula or a negated atomic formula. A clause usually contains non-ground literals, that is, literals with logical variables (e.g. $\texttt{inside}(\texttt{A},\texttt{B})$). A substitution θ, applied to a clause or a formula, replaces the variables with other terms. For example, for $\theta = \{A = 1, B = 2, C = 3\}$ the above clause becomes:

$$\texttt{inside}(1,2) \leftarrow \texttt{inside}(1,3),\texttt{inside}(3,2)$$

and states that if $\texttt{inside}(1,3)$ and $\texttt{inside}(3,2)$ are true, then $\texttt{inside}(1,2)$ is true. In Distributional Clauses [3], the traditional logic programming formalism is extended to define random variables. A *distributional clause* is of the form $\texttt{h} \sim \mathcal{D} \leftarrow \texttt{b}_1,\ldots,\texttt{b}_\texttt{n}$, where the $\texttt{b}_\texttt{i}$ are literals and $\sim$ is a binary predicate written in infix notation. The intended meaning of a distributional clause is that each ground instance of the clause $(\texttt{h} \sim \mathcal{D} \leftarrow \texttt{b}_1,\ldots,\texttt{b}_\texttt{n})\theta$ defines the random variable $\texttt{h}\theta$ as being distributed according to $\mathcal{D}\theta$ whenever all the $\texttt{b}_\texttt{i}\theta$ hold, where θ is a substitution. The term $\mathcal{D}$, which represents the distribution, can be non-ground, i.e. values, probabilities or distribution parameters can be related to conditions in the body. Furthermore, a term $\simeq(d)$ constructed from the reserved functor $\simeq/1$ represents the value of the random variable d. Consider the following clauses (rules):

$$\texttt{n} \sim \texttt{poisson}(6). \quad (1)$$

$$\texttt{pos}(\texttt{P}) \sim \texttt{uniform}(1,10) \leftarrow \texttt{between}(1,\simeq(\texttt{n}),\texttt{P}). \quad (2)$$

Clause (1) states that the number of people `n` is governed by a Poisson distribution with mean 6; clause (2) models the position $\texttt{pos}(\texttt{P})$ as a random variable uniformly distributed from 1 to 10, for each person `P` such that $\texttt{between}(1,\simeq(\texttt{n}),\texttt{P})$

succeeds. Thus, if the outcome of n is 2, there will be 2 independent random variables pos(1) and pos(2).

A distributional clause is a powerful template to define conditional probabilities: the random variable h has a distribution $\mathcal{D}$ given the conditions in the body $\mathtt{b}_1, \ldots, \mathtt{b}_n$ (referred also as body). Furthermore, it supports continuous random variables in contrast with the majority of the statistical relational languages. Indeed, a ground atom (i.e., tuple of a relation) represents a random variable. Therefore, throughout the paper, ground atoms and random variables are considered interchangeable. The dynamic version of this language (Dynamic Distributional Clauses) is used to define the prior distribution, the state transition model and the measurement model in a particle filter framework called DCPF.

A particle filter [12] is a Monte-Carlo technique to perform filtering in temporal models. Filtering consists in computing the probability density function $p(x_t|z_{1:t}, u_{1:t})$, where x_t is the current state, $z_{1:t}$ is the sequence of observations, and $u_{1:t}$ the actions (inputs) performed from time step 1 to t.

DCPF performs particle filtering in a relational domain, where particles $x_t^{(i)}$ are interpretations, i.e. sets of ground facts for the predicates and the values of random variables that hold at time t. This relational language is useful for describing objects and their properties as well as relations between them. Probabilistic rules define how those relations affect each other with respect to time.

IV. Online Parameter Learning

A simple solution to perform state estimation and parameter learning with particle filters consists of adding the static parameters in the state space: $\hat{x}_t = \{x_t, \theta\}$, therefore the posterior distribution $p(\hat{x}_t|z_{1:t})$ is described as a set of particles $\{x_t^{(i)}, \theta^{(i)}\}$. However, this solution produces poor results due to the degeneracy problem. Indeed, the parameters are sampled in the first step and left unchanged, thus after few steps the parameter samples $\theta^{(i)}$ will degenerate to a single value that will remain unchanged. Better strategies have been proposed and are summarized in [13], however online learning in nonlinear models remains an open problem. We focus on two techniques that have a limited computational cost: artificial dynamics [14] and resample-move [15]. Both methods introduce diversity among the particles to solve the described degeneration problem. The first method adds dynamics to the parameter θ:

$$\theta_{t+1} = \theta_t + \epsilon_{t+1},$$

where ϵ_{t+1} is artificial noise with a small and decreasing variance over time. This strategy has the advantage to be simple and fast, nonetheless it modifies the original problem and requires tuning [13]. We will show that this technique is suitable for the scenarios considered in this paper. The second method is the resample-move that adds a single MCMC step to rejuvenate the parameters in the particles. There are several variants of this technique, the most notable are the Storvik's filter [16] and Particle Learning by Carvalho et al. [17].

To understand these approaches, consider the following factorization of the joint distribution of interest:

$$p(x_{0:t}, \theta|z_{1:t}) = p(\theta|x_{0:t}, z_{1:t})p(x_{0:t}|z_{1:t}) =$$
$$= p(\theta|x_{0:t-1}, z_{1:t-1})p(x_{0:t-1}|z_{1:t-1})\frac{p(z_t|x_t, \theta)p(x_t|x_{t-1}, \theta)}{p(z_t|z_{1:t-1})}$$

Thus, in addition to the standard propagation and weighing steps, both algorithms perform a Gibbs sampling step that samples a new parameter value $\theta_t^{(i)}$ from the distribution $p(\theta|x_{0:t}^{(i)}, z_{1:t}) = p(\theta|s_t^{(i)})$ where $s_t^{(i)}$ is the sufficient statistics of the distribution. This distribution can be recursively updated as follow:

$$p(\theta|x_{0:t}, z_{1:t}) \propto p(\theta|x_{0:t-1}, z_{1:t-1})p(z_t|x_t, \theta)p(x_t|x_{t-1}, \theta)$$
$$= p(\theta|s_{t-1})p(z_t|x_t, \theta)p(x_t|x_{t-1}, \theta) \propto p(\theta|s_t) \quad (3)$$

This leads to a deterministic sufficient statistics update

$$s_t = S(s_{t-1}, x_t, x_{t-1}, z_t).$$

Storvik's filter algorithm goes as follows:

- propagate: $x_t^{(i)} \sim q(x_t|x_{t-1}^{(i)}, \theta_{t-1}^{(i)}, z_t)$
- resample particles with weights:

$$w_t^{(i)} = \frac{p(z_t|x_t^{(i)}, \theta_{t-1}^{(i)})p(x_t^{(i)}|x_{t-1}^{(i)}, \theta_{t-1}^{(i)})}{q(x_t^{(i)}|x_{t-1}^{(i)}, \theta_{t-1}^{(i)}, z_t)}$$

- propagate (deterministically) sufficient statistics:

$$s_t^{(i)} = S(s_{t-1}^{(i)}, x_t^{(i)}, x_{t-1}^{(i)}, z_t^{(i)})$$

- sample $\theta_t^{(i)} \sim p(\theta|s_t^{(i)})$

The Particle Learning proposed by Carvalho et al. [17] is an optimization of the Storvik's filter since it adopts the auxiliary particle filter [18] and an optimal proposal distribution q. Resample-move strategies do not change the problem as in the artificial dynamics, and have been proved successfully in several classes of problems [17], [19], [20]. However, they suffer of the sufficient statistics degeneracy problem that may produce an increasing error in the parameter posterior distribution [21].

V. Packaging scenario

The goal of this scenario is to track objects moved by a human during a packaging activity with boxes (Fig. 1a-d). The framework should be able to keep track of objects inside boxes. To solve this problem we defined a model in Dynamic Distributional Clauses where the **state** consists of the position, the velocity and orientation of all objects, plus the relations between them. The relations considered are left, right, near, on, and inside plus object properties such as color, type and size.

Whenever a new object is observed its pose is added to the state together with the all the derived relations.

We also modelled the following physical principles in the **state transition model**:

1) if an object is on top of another object, it cannot fall down;

2) if there are no objects under an object, the object will fall down until it collides with another object or the table;
3) an object may fall inside the box only if it is on the box in the previous step, is smaller than the box and the box is open-side up.
4) if an object is inside a box it remains in the box its position follows that of the box as long as it is open-side up.
5) if the box is rotated upside down the objects inside will fall down with a certain probability

As an example consider property (3). If an object ID is not inside a box, is on top of a box B with rotation $\texttt{yaw}_\texttt{t}(\texttt{B}) > 0$ (open-side up) and the object ID is smaller than the box B, then it can fall inside the box with probability 0.8 in the next step. This can be modelled by the following clause:

$$\begin{aligned}\texttt{inside}_{\texttt{t+1}}(\texttt{ID},\texttt{B}) \sim \texttt{finite}([0.8{:}\texttt{true}, 0.2{:}\texttt{false}]) \leftarrow \\ \texttt{not}(\simeq(\texttt{inside}_\texttt{t}(\texttt{ID},\texttt{C})) = \texttt{true}),\ \texttt{on}_\texttt{t}(\texttt{ID},\texttt{B}), \\ \texttt{type}(\texttt{B},\texttt{box}),\ \simeq(\texttt{yaw}_\texttt{t}(\texttt{B})) > 0,\ \texttt{smaller}(\texttt{ID},\texttt{B}). \quad (4)\end{aligned}$$

That is to say, a particle at time t with two objects 1 and 2, where $\texttt{on}_\texttt{t}(2,1), \texttt{type}(1,\texttt{box})$, $\texttt{type}(2,\texttt{cup})$, $\simeq(\texttt{inside}_\texttt{t}(2,1)) = \texttt{false}$, $\texttt{smaller}(2,1)$, $\simeq(\texttt{yaw}_\texttt{t}(1)) > 0$ hold; the body of clause (4) is true for $\theta = \{ID = 2, B = 1\}$, therefore the random variable $\texttt{inside}_{\texttt{t+1}}(2,1)$ at time $t+1$ will be sampled from the distribution $[0.8 : \texttt{true}, 0.2{:}\texttt{false}]$. The predicate smaller is true when the dimensions of the first object are smaller than the second object. To apply the clause, the box yaw orientation needs to be positive. This assumes that yaw is represented as a radiant angle in the range $[-\pi, \pi]$, positive in the quadrants I and II. According to the coordinate system used the yaw of a box object is positive whenever is open-side up.

Furthermore, if A is inside B at time t, the relation holds at $t+1$ with probability close to one (clause omitted). If an object is inside the box, we assume that its position is uniformly distributed inside the box:

$$\begin{aligned}\texttt{pos}_{\texttt{t+1}}(\texttt{ID})_\texttt{x} \sim \texttt{uniform}(\simeq(\texttt{pos}_{\texttt{t+1}}(\texttt{B}))_\texttt{x} - \texttt{D}_\texttt{x}/2, \\ \simeq(\texttt{pos}_{\texttt{t+1}}(\texttt{B}))_\texttt{x} + \texttt{D}_\texttt{x}/2)) \leftarrow \\ \simeq(\texttt{inside}_\texttt{t}(\texttt{ID},\texttt{B})) = \texttt{true}, \texttt{size}(\texttt{B},\texttt{D}_\texttt{x},\texttt{D}_\texttt{y},\texttt{D}_\texttt{z}).\end{aligned}$$

We only showed the x dimension and omitted the object's velocity for ease of exposition. To model the position and the velocity of objects in free fall we use the rule:

$$\begin{aligned}&\texttt{pos\_vel}_{\texttt{t+1}}(\texttt{ID})_\texttt{z} \sim \texttt{gaussian} \\ &\left(\begin{bmatrix} \simeq(\texttt{pos}_\texttt{t}(\texttt{ID}))_\texttt{z} + \Delta\texttt{t} \simeq(\texttt{vel}_\texttt{t}(\texttt{ID}))_\texttt{z} - 0.5\texttt{g}\Delta\texttt{t}^2 \\ \simeq(\texttt{vel}_\texttt{t}(\texttt{ID}))_\texttt{z} - \texttt{g}\Delta\texttt{t} \end{bmatrix}, cov \right) \\ &\leftarrow \texttt{not}(\simeq(\texttt{inside}_\texttt{t}(\texttt{ID},\_)) = \texttt{true}), \\ &\quad \texttt{not}(\texttt{on}_\texttt{t}(\texttt{ID},\_)), \simeq(\texttt{held}_\texttt{t}(\texttt{ID})) = \texttt{false}.\end{aligned}$$

It states that if the object ID is neither 'on' nor 'inside' any object, and is not held, the object will fall with gravitational acceleration g, where we specify only the position and velocity for the coordinate z. For the coordinates x and y the rule is similar but without acceleration. The gravitational force can be compensated by a human or a robot that holds the object. In that case there is no acceleration and we use a constant position model. The variable $\texttt{held}_\texttt{t}(\texttt{ID})$ indicates whether the object is held or not, we sample this variable from a Bernoulli distribution with probability 0.6 to be true, whereas the relative clause is omitted for brevity.

The measurement model is the product of Gaussian distributions around each object's position $\texttt{pos}_\texttt{t}(\texttt{i})$ (thereby assuming that the measurements are i.i.d.):

$$\texttt{obsPos}_{\texttt{t+1}}(\texttt{ID}) \sim \texttt{gaussian}(\simeq(\texttt{pos}_{\texttt{t+1}}(\texttt{ID})),\ cov).$$

In addition, we assume objects inside a box invisible, thus the probability to observe an object inside a box is assumed null. Therefore, to improve the performance we consider the observation in the body of clauses that define the inside proposal distribution.

The commonsense inside concept is transitive, therefore we defined a transitive inside relation $\texttt{tr\_inside}_\texttt{t}(\texttt{A},\texttt{B})$ from $\texttt{inside}_\texttt{t}(\texttt{A},\texttt{B})$:

$$\begin{aligned}&\texttt{tr\_inside}_\texttt{t}(\texttt{A},\texttt{B}) \leftarrow \simeq(\texttt{inside}_\texttt{t}(\texttt{A},\texttt{B})) = \texttt{true}. \\ &\texttt{tr\_inside}_\texttt{t}(\texttt{A},\texttt{B}) \leftarrow \\ &\quad \simeq(\texttt{inside}_\texttt{t}(\texttt{A},\texttt{C})) = \texttt{true}, \texttt{tr\_inside}_\texttt{t}(\texttt{C},\texttt{B}).\end{aligned}$$

To make the particle filter more efficient we can use the optimal proposal distribution $p(x_{t+1}|x_t, z_{t+1})$ and the corresponding weight $p(z_{t+1}|x_t)$. Given a complex nonlinear transition model those distributions are not easy to compute analytically, therefore we adopt suboptimal solutions. Let assume that the state is $x_t = \{a_t, b_t\}$ and the observations depends only on b_{t+1}, then the weight can be written as:

$$w_{t+1}^{(i)} = \frac{p(z_{t+1}|b_{t+1}^{(i)}) p(b_{t+1}^{(i)}|a_{t+1}^{(i)}, x_t^{(i)}) p(a_{t+1}^{(i)}|x_t^{(i)})}{q(x_{t+1}^{(i)}|x_t^{(i)}, z_{t+1})}$$

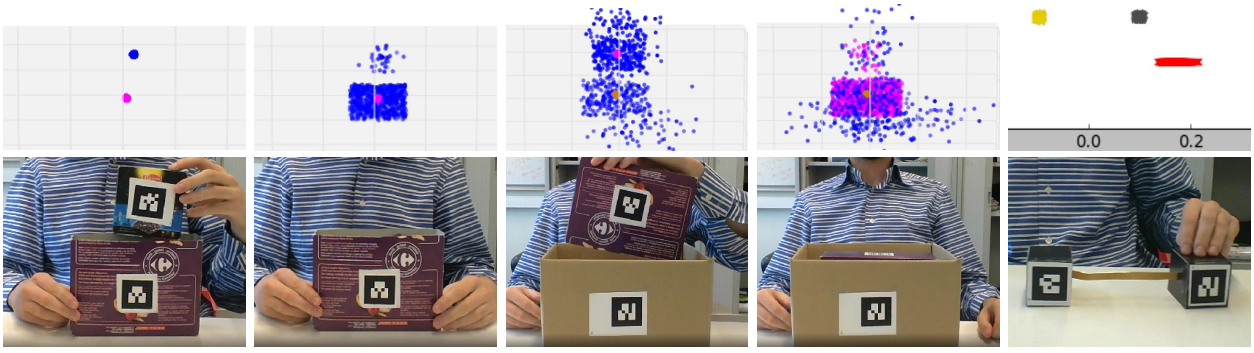
Thus setting the proposal distribution to

$$q(x_{t+1}|x_t^{(i)}, z_{t+1}) = p(b_{t+1}|a_{t+1}, x_t^{(i)}, z_{t+1}) p(a_{t+1}|x_t^{(i)})$$

we obtain a weight $w_{t+1} = p(z_{t+1}|a_{t+1}^{(i)}, x_t^{(i)})$. If $p(b_{t+1}|a_{t+1}, x_t)$ is a linear Gaussian transition model or discrete we can easily compute the above (sub)optimal proposal and relative weight after sampling a_{t+1}. We consider the objects' positions as the set b_t, while the remaining states define the set a_t. In this scenario, the relation "inside" is also added to b_t. Thus, we developed an auxiliary particle filter for the DCPF framework to improve the performance.

VI. Online Learning for DCPF

To illustrate the proposed learning algorithms we consider the string scenario. In this scenario we have a table with several objects possibly connected by strings (Fig. 1e). The goal is to track the objects moved by a human (or a robot), and estimate online the length of the strings between objects. This problem involves static parameters, that is, the strings' length for each object pair. This makes the problem difficult as explained in section IV.



(a) cube on the box (b) cube inside the box (c) rotated box on a beige box (d) cube and box inside the beige box (e) String scenario

Fig. 1: (a-d) packaging scenario experiments. The bottom images represent moments of the experiment, while the top images show the corresponding estimated objects' positions, where each colored point represents an object in a particle. The cube is represented in blue, the small box in fuchsia and the big box in beige (e) string scenario. The top figure represents the estimated objects' positions (yellow and grey), and the estimated string length in red.

The main **state variables** are the object positions, the strings' length for each pair of objects, and the object ID moved by a human or robot (if any). Whenever the distance of the objects is greater than the string length that connects them, each object is pulled in the direction of the other. Therefore, we apply a displacement proportional to the absolute difference between the distance and the string length. Otherwise, if the string is longer than the distance, such displacement is not applied. Multiple displacements can be applied to the same object, in that case the total displacement is the sum of the simple components. However, whenever the object is held we assume the displacement 0. We assume each string length $\texttt{string}(\texttt{A},\texttt{B})$ that connects objects A and B, ranges from 0 to 1.

For the artificial dynamics technique we defined a uniform prior:

$$\texttt{string}_0(\texttt{A},\texttt{B}) \sim \texttt{uniform}(0,1).$$

While the transition model defines the artificial dynamics:

$$\texttt{string}_{t+1}(\texttt{A},\texttt{B}) \sim \texttt{gaussian}(\simeq(\texttt{string}_t(\texttt{A},\texttt{B})), \frac{\bar{\sigma}^2}{\texttt{T}^\texttt{X}}),$$

where T is the time step, X is a fixed exponent (set to 2 in this paper) and $\bar{\sigma}^2$ is a constant that represents the initial variance. Initially the variance is high, thus the particle filter can "explore" the parameter space, after some steps the variance decreases in the hope that the parameter converges to the real value.

We also developed a variant of the Storvik's filter for learning. Equation (3) shows how to update the parameter posterior and then the sufficient statistics for each particle. However, this formulation needs a conjugate prior for the parameter likelihood. For a complex distribution this may be hard or even impossible. To overcome this problem we add $\hat{\theta}_t$ to the state that represents the current sampled "parameter" value, while θ is the parameter to estimate, e.g., the mean of $\hat{\theta}_t$. We also assume the state x_t depends only on $\hat{\theta}_t$: $p(x_t|x_{t-1},\hat{\theta}_t,\theta) = p(x_t|x_{t-1},\hat{\theta}_t)$. For example, in the string object scenario, $\hat{\theta}_t = \texttt{string}_t(\texttt{A},\texttt{B})$ while $p(\hat{\theta}_t|\theta)$ can be a Gaussian with mean θ and a fixed variance. Thus the posterior becomes:

$$\begin{aligned} p(x_{0:t},\theta,\hat{\theta}_{0:t}|z_{1:t}) &\propto p(\theta,\hat{\theta}_t,\hat{\theta}_{0:t-1},x_{0:t-1}|z_{1:t-1}) \\ & p(z_t|x_t,\theta,\hat{\theta}_t)p(x_t|x_{t-1},\theta,\hat{\theta}_t) \\ =& p(\hat{\theta}_t|\theta)p(\theta|s_{t-1})p(\hat{\theta}_{0:t-1},x_{0:t-1}|z_{1:t-1}) \\ & p(z_t|x_t)p(x_t|x_{t-1},\hat{\theta}_t) \end{aligned}$$

Therefore, we replace (3) with:

$$p(\theta|\hat{\theta}_{0:t},x_{0:t},z_{1:t}) = p(\theta|s_t) \propto p(\theta|s_{t-1})p(\hat{\theta}_t|\theta).$$

At this point we can avoid sampling θ as required by the Storvik's filter, but sample $\hat{\theta}_t$ from the marginal distribution:

$$p(\hat{\theta}_t|s_{t-1}) = \int p(\hat{\theta}_t|\theta)p(\theta|s_{t-1})d\theta.$$

For ease of exposition we assumed the measurement model is independent of θ, but the principle remains unchanged.

If the objects are held and moved by a human, we need to estimate also which object is free and which one is held. To simplify the problem we assume the human can move at most one object at a time. Thus we added the variable $\texttt{move}_t$ in the state that indicates the object ID held/moved or zero for none. The prior distribution is:

$$\begin{aligned} \texttt{move}_0 \sim \texttt{uniform}([0,0|\texttt{L}]) \leftarrow \\ \texttt{findall}(\texttt{ID},\simeq(\texttt{object}(\texttt{ID})),\texttt{L}). \end{aligned}$$

That is a uniform distribution over all objects ID plus zero counted two times. Therefore the probability of no object being held is twice the probability that a given object is held. In addition, to make more probable remaining in the same state, we defined the following transition model:

$$\begin{aligned} \texttt{move}_{t+1} \sim \texttt{uniform}([0,\simeq(\texttt{move}_t),\simeq(\texttt{move}_t)|\texttt{L}]) \leftarrow \\ \texttt{findall}(\texttt{ID},\simeq(\texttt{object}(\texttt{ID})),\texttt{L}). \end{aligned}$$

Thus the previous state is twice as probable as the other values. Whenever $\texttt{move}_t$ has the value id the corresponding object transition model will have a noise variance double the one of the other objects, e.g. for the axis x:

$$\texttt{pos}_{t+1}(\texttt{ID})_x \sim \texttt{gaussian}(\simeq(\texttt{pos}_t(\texttt{ID})_x), \sigma^2) \leftarrow \simeq(\texttt{move}_t) = \texttt{ID}. \quad (5)$$

$$\texttt{pos}_{t+1}(\texttt{ID})_x \sim \texttt{gaussian}(\simeq(\texttt{pos}_t(\texttt{ID})) + \simeq(\texttt{totDisp}_t(\texttt{ID})_x), \frac{\sigma^2}{2}) \leftarrow \simeq(\texttt{move}_t) \neq \texttt{ID}.$$

If the object is not held we take in account the eventual displacements caused by pulled strings. Thus, $\texttt{totDisp}_t(\texttt{ID})$ is the sum of all displacements $\texttt{displacement}_t(\texttt{ID}, \texttt{C})$ applied to object ID and caused by the string that connects ID with the object C. We use the auxiliary particle filter and suboptimal proposal as in the packaging scenario. For example, the proposal that replaces (5) is:

$$\begin{aligned}&\texttt{pos}_{t+1}(\texttt{ID})_x \sim \texttt{gaussian}(\texttt{Mean}, \texttt{Var}) \leftarrow \\ &\quad \texttt{Var is } (\sigma^{-2} + \omega^{-2})^{-1}, \\ &\quad \texttt{Mean is Var}(\simeq(\texttt{obsPos}_t(\texttt{ID})_x)\omega^{-2} + \simeq(\texttt{pos}_t(\texttt{ID})_x)\sigma^{-2}), \\ &\quad \simeq(\texttt{move}_t) = \texttt{ID}. \quad (6)\end{aligned}$$

Where ω^2 is the variance of the Gaussian measurement model.

VII. Experiments

This section answers the following questions:

- **(Q1)** Does the DCPF framework obtain the correct results with the described models?
- **(Q2)** How do the learning algorithms perform?
- **(Q3)** Is the DCPF suitable for real-world applications?

All algorithms were implemented in YAP Prolog and C++, and run on a laptop Intel Core i7. The objects are marked with AR tags so that their position and orientation can be easily recognized by a camera. We assume the type, size and color are known for each object, however integrating the output of a classifier is straightforward. We also encoded the static object 'table', therefore when an object is not held will fall down until it collides with another object or the table.

We first performed preliminary tests to check the robustness of the models and the performance of the described learning algorithms on simple models with simulated data. This empirical evaluation showed that Storvik's filter and Carvalho's improvement converged to the expected value and are sufficiently stable. The degeneration problems described in [21] was observed when the parameter to learn is a covariance (e.g., of a Gaussian) in the transition and/or measurement model. In this case the results may appear biased, caused by an error accumulation, especially for heavily noisy observations. On the other hand, whenever the parameter to learn has a relatively "peaked" likelihood, this issue was not observed.

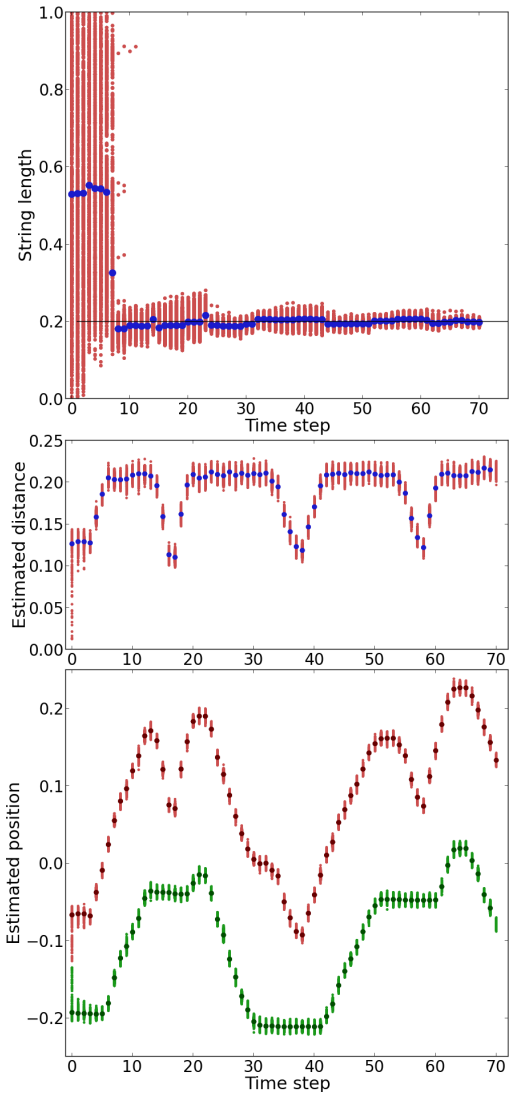


Fig. 2: String scenario with two objects using artificial dynamics. The top image shows the string length particles in red at each step, while the average is plotted in blue. The ground truth (0.2 m) is in black. The central image shows the estimated distance between the objects for each particle in red and the average in blue. The bottom image shows the estimated positions of the two objects in red and green, where the average per step is spotted in dark. The distance is measured in meters.

For the packaging scenario we performed several test-cases. The first case consists of taking a box, putting an object inside the box, moving the box and then rotating the box upside down. The second test-case consists of putting an object inside a small box, putting the small box in a bigger box, then moving the bigger box and rotating it upside down. Finally, we put an object inside a small box and then rotate the box on top of another box, the object inside has to fall inside the other object. For this scenario we used 600 particles. The results showed that the model is stable, correctly tracks the objects, and successfully estimates the

transitive relation inside (Q1). For example, whenever we put an object in a box the DCPF estimates that it is inside the box or still outside with a small probability. Similarly, when we rotate a box upside down the object that was inside falls outside and goes inside the box below. One issue that was encountered is when the objects are moved rapidly, in this case the filter keeps track of the visible objects but may loose the particle diversity of the invisible objects. To avoid this problem the variance in the state transition model needs to be increased.

A relational representation allows to perform complex queries exploiting background knowledge. This is useful to convert the belief state in a readable form and to communicate with humans. For example, in DCPF we can ask how many objects there are in the box with the respective probability for each object, or if there is a blue cube in the beige box. The second query would be the conjunction:

$$(\texttt{object}_\texttt{t}(\texttt{A}), \texttt{type}(\texttt{A}, \texttt{cube}), \texttt{color}(\texttt{A}, \texttt{blue}),$$
$$\simeq(\texttt{inside}_\texttt{t}(\texttt{A}, \texttt{B})) = \texttt{true}, \texttt{type}(\texttt{B}, \texttt{box}), \texttt{color}(\texttt{B}, \texttt{beige})).$$

The answer is the probability that the query is true. Alternatively we can list all objects pair (A,B) that satisfy the query with the respective probability.

The filter performance for this scenario depends on the number of objects, with two objects the framework spends around 0.7-1.0 ms per particle for one step, while with three objects it needs around 1-1.5 ms per particle.

Then we tested the two described learning algorithms in the string scenario: artificial dynamics and the proposed Storvik's filter variation. We performed 15 trials for each of the two algorithms and for two and three objects. In each trial we randomly pulled and pushed one object at a time. The results of the experiments are summarized in Table I and were performed using 800 particles. In the experiments with two objects both learning strategies perform well (Q2).

However, learning becomes challenging with three objects, that is, with three parameters to learn (the string length for each object pair). In several trials not all parameters were correctly estimated, nonetheless the particles often converged to a solution that has a high likelihood for the provided sequence of observations. Indeed, there are different objects configurations that can produce a similar behavior when moved. With three objects artificial dynamics performs better than Storvik's filter variation. In addition, artificial dynamics is always faster than Storvik's filter variation, this is due to the sufficient statistic update needed in the latter.

Learning in this scenario is challenging because the object being moved needs to be estimated. Indeed, supplementary experiments showed better results when the object held/moved is known at each step (Table I). In this case the two learning algorithms have similar success rate. Another reason that makes learning hard is the fact that the parameter likelihood is high in a small region near the real parameter, but constant for values greater than the distance of the objects. Therefore, the parameter can remain stuck with values greater than the real length of the string.

An example with two objects using artificial dynamics is showed in Fig. 1e and the state estimation at each step is showed in Fig. 2. In this example the objects were moved on the same line for ease of representation. During the first steps the string length distribution is uniform between 0 and 1, and the string is not stretched. Then an object is pulled, thus the distance between the objects reaches the maximum and the other object starts moving in the direction of the other. In those steps only the particles with values close to the real string length survive. Afterwards, the object is pushed towards the other one with no effect on the string length estimation. During the remaining steps the artificial dynamics variance decreases and the particles converge to the ground truth.

TABLE I: String scenario results. Correct: the mean of each estimated parameter converged to the ground truth up to an error of 2 cm. Partially correct: the mean of some of the estimated parameters converged to the ground truth up to an error of 2 cm. Wrong: the mean of none of the estimated parameters converged to the ground truth. Action known: the object moved is known at each step.

			Parameter learning			
Algorithm	Objects	Action known	Correct	Partially correct	Wrong	ms / particle
Artificial dynamics	2	NO	13/15		2/15	0.3-0.6
	3	NO	7/15	8/15	0/15	0.6-0.8
	3	YES	10/15	5/15	0/15	0.6-0.8
Storvik's filter variation	2	NO	14/15		1/15	0.8-1.2
	3	NO	4/15	10/15	1/15	1.1-1.5
	3	YES	9/15	4/15	1/15	1.1-1.5

The videos of the experiments are available at `https://dtai.cs.kuleuven.be/ml/systems/DC/tracking.html`

VIII. CONCLUSIONS

We proposed a model for online tracking with a relational representation and two learning strategies. The framework was tested in two scenarios. The experiments show encouraging results, which are promising for robotics applications. One of the advantages of a relational framework like DCPF is the flexibility and generality of the model with respect to a particular situation. Indeed, whenever a new object appears all the respective properties and relations with other objects are implicitly defined (but not necessary computed and added to the particle). In addition, the expressivity of the language helps to bridge the gap between robotics and the high-level symbolic representation used in Artificial Intelligence.

The performance is acceptable, but could be improved to scale well with high-dimensional states. Indeed, each particle represents the entire state, therefore inference can be computationally intensive for a high number of objects and relations. Nonetheless, DCPF exploits the structure of the model and partial particles to speed up inference and improve the performance [6]. Finally, both learning strategies tested in this framework perform well for a limited number

of parameters. More sophisticated strategies need to be investigated for a higher number of parameters.

REFERENCES

[1] L. De Raedt, P. Frasconi, K. Kersting, and S. Muggleton, Eds., *Probabilistic Inductive Logic Programming, Theory and Applications*, ser. Lecture Notes in Artificial Intelligence. Springer, 2008.

[2] L. Getoor and B. Taskar, *An Introduction to Statistical Relational Learning*. MIT Press, 2007.

[3] B. Gutmann, I. Thon, A. Kimmig, M. Bruynooghe, and L. De Raedt, "The magic of logical inference in probabilistic programming," *Theory and Practice of Logic Programming*, 2011.

[4] J. Wang and P. Domingos, "Hybrid markov logic networks." in *AAAI*, vol. 8, 2008, pp. 1106–1111.

[5] M. A. Islam, C. R. Ramakrishnan, and I. V. Ramakrishnan, "Inference in probabilistic logic programs with continuous random variables," *CoRR*, vol. abs/1112.2681, 2011.

[6] D. Nitti, T. D. Laet, and L. D. Raedt, "A particle filter for hybrid relational domains," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[7] T. De Laet, "Rigorously bayesian multitarget tracking and localization," Ph.D. dissertation, May 2010.

[8] C. Manfredotti, "Modeling and inference with relational dynamic bayesian networks," in *Advances in artificial intelligence*. Springer Berlin Heidelberg, 2009, pp. 287–290.

[9] L. Cattelani, C. Manfredotti, and E. Messina, "A particle filtering approach for tracking an unknown number of objects with dynamic relations," *Journal of Mathematical Modelling and Algorithms in Operations Research*, pp. 1–19, 2012.

[10] M. Tenorth and M. Beetz, "Knowrob: A knowledge processing infrastructure for cognition-enabled robots," *The International Journal of Robotics Research*, vol. 32, no. 5, pp. 566–590, 2013.

[11] L. Mösenlechner and M. Beetz, "Using physics- and sensor-based simulation for high-fidelity temporal projection of realistic robot behavior," in *19th International Conference on Automated Planning and Scheduling (ICAPS'09)*., 2009.

[12] A. Doucet, S. Godsill, and C. Andrieu, "On sequential monte carlo sampling methods for bayesian filtering," *STATISTICS AND COMPUTING*, vol. 10, no. 3, pp. 197–208, 2000.

[13] N. Kantas, A. Doucet, S. S. Singh, and J. M. Maciejowski, "An overview of sequential monte carlo methods for parameter estimation in general state-space models," in *15th IFAC Symposium on System Identification*, vol. 15, 2009, pp. 774–785.

[14] T. Higuchi, "Self-organizing time series model," in *Sequential Monte Carlo Methods in Practice*, ser. Statistics for Engineering and Information Science, A. Doucet, N. Freitas, and N. Gordon, Eds. Springer New York, 2001, pp. 429–444.

[15] W. R. Gilks and C. Berzuini, "Following a moving targetmonte carlo inference for dynamic bayesian models," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 63, no. 1, pp. 127–146, 2001.

[16] G. Storvik, "Particle filters for state-space models with the presence of unknown static parameters," *Signal Processing, IEEE Transactions on*, vol. 50, no. 2, pp. 281–289, 2002.

[17] C. M. Carvalho, M. S. Johannes, H. F. Lopes, and N. G. Polson, "Particle learning and smoothing," *Statistical Science*, vol. 25, no. 1, pp. 88–106, 2010.

[18] M. K. Pitt and N. Shephard, "Filtering via simulation: Auxiliary particle filters," *Journal of the American statistical association*, vol. 94, no. 446, pp. 590–599, 1999.

[19] C. M. Carvalho, H. F. Lopes, N. G. Polson, and M. A. Taddy, "Particle learning for general mixtures," *Bayesian Analysis*, vol. 5, no. 4, pp. 709–740, 2010.

[20] H. F. Lopes, C. M. Carvalho, M. Johannes, and N. G. Polson, "Particle learning for sequential bayesian computation," *Bayesian statistics*, vol. 9, pp. 175–96, 2010.

[21] C. Andrieu, A. Doucet, and V. B. Tadic, "On-line parameter estimation in general state-space models," in *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on*, 2005, pp. 332–337.