UNIVERSITÀ DI MILANO - BICOCCA
FACOLTÀ DI SCIENZE MATEMATICHE, FISICHE E NATURALI
DIPARTIMENTO DI INFORMATICA, SISTEMISTICA E COMUNICAZIONE (DISCO)
DOTTORATO DI RICERCA IN INFORMATICA

XXII CICLO

MODELING AND INFERENCE WITH RELATIONAL DYNAMIC BAYESIAN NETWORKS

A dissertation presented
by
Cristina Elena Manfredotti

in partial fulfillment of the requirements for the degree of
DOCTOR of PHILOSOPHY
in
Computer Science

October 2009

Advisor: **Prof. Enza Messina**
Co-advisor: **Prof. David Fleet**
Tutor: **Prof. Domenico G. Sorrenti**
PhD Programme Coordinator: **Prof. Stefania Bandini**

a Te, il mio copilota

Abstract

Many domains in the real world are richly structured, containing a diverse set of agents characterized by different set of features and related to each other in a variety of ways. Moreover, uncertainty both on the objects observations and on their relations can be present. This is the case of many problems as, for example, multi-target tracking, activity recognition, automatic surveillance and traffic monitoring.

The common ground of these types of problems is the necessity of recognizing and understanding the scene, the activities that are going on, who are the actors, their role and estimate their positions. When the environment is particularly complex, including several distinct entities whose behaviors might be correlated, automated reasoning becomes particularly challenging. Even in cases where humans can easily recognize activities, current computer programs fail because they lack of commonsense reasoning, and because the current limitation of automated reasoning systems. As a result surveillance supervision is so far mostly delegated to humans.

The explicit representation of the interconnected behaviors of agents can provide better models for capturing key elements of the activities in the scene. In this Thesis we propose the use of relations to model particular correlations between agents features, aimed at improving the inference task. We propose the use of Relational Dynamic Bayesian Networks, an extension of Dynamic Bayesian Networks with First Order Logic, to represent the dependencies between an agent's attributes, the scene's elements and the evolution of state variables over time. In this way, we can combine the advantages of First Order Logic (that can compactly represent structured environments), with those of probabilistic models (that provide a mathematically sound framework for inference in face of uncertainty).

In particular, we investigate the use of Relational Dynamic Bayesian Networks to represent the dependencies between the agents' behaviors in the context of multi-agents tracking and activity recognition. We propose a new formulation of the transition model that accommodates for relations and present a filtering algorithm that extends the Particle Filter algorithm in order to directly track relations between the agents.

The explicit recognition of the relationships between interacting objects can improve the understanding of their dynamic domain. The inference algorithm we develop in this Thesis is able to take into account relations between interacting objects and we demonstrate with experiments that the performance of our relational approach outperforms those of standard non-relational methods.

While the goal of emulating human-level inference on scene understanding is out of reach for the current state of the art, we believe that this work represents an important step towards better algorithms and models to provide inference in complex multi-agent systems.

Another advantage of our probabilistic model is its ability to make inference *online*, so that the appropriate cause of action can be taken when necessary (*e.g.*, raise an alarm). This is an important requirement for the adoption of automatic surveillance systems in the real world, and avoid the common problems associated with human surveillance.

Keywords: Multi Target tracking, Probabilistic Relational Models, Bayesian Filtering, Particle Filtering.

Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

$2TBN$ two-time-slice BN fragment

$2TRBN$ two-time-slice RBN fragment

BN Bayesian Network

CPD Conditional Probability Distribution

CPT Conditional Probability Table

DBN Dynamic Bayesian Network

FOL First-Order Logic

FOPT First-Order Probabilistic Tree

K-PRM Probabilistic Relational Model introduced in (Friedman, Getoor, Koller, & Pfeffer, 1999)

PF Particle Fitler

PM Probabilistic Model

PRM Probabilistic Relational Model

RBN Relational Bayesian Network

RDBN Relational Dynamic Bayesian Network

ROC Receiver Operator Characteristic

RPF Relational Particle Filter

Modeling and Inference with Relational Dynamic Bayesian Networks

Chapter 1

Introduction

There are finer fish in the sea than have ever been caught.

Irish proverb

Many domains in the real world are richly structured, containing a diverse set of objects characterized by attributes and related to each other in a variety of ways. A central aspect of human intelligence is the ability to make inference in these structured environments using abstract knowledge. For example, human reasoning is able to easily infer the participants and their role in a particular activity or situation and it is able to recognize the activity itself.

The *context* is often a key element that facilitates our understanding of the world around. Imagine, for instance, a scene where someone in the street is waving his hand. It can either be that the subject is greeting someone, perhaps a friend, or that is hailing a taxi. While we, humans, are very good at making this kind of distinctions, automated reasoning encounters great difficulties.

When the context is particularly complex, including several distinct entities whose actions might be correlated, automated reasoning becomes particularly challenging. Imagine, for instance, a road traffic scenario where driving behaviors are dependent on a quantity of variables, as road and traffic conditions, time, etc. Detecting the *relations* between the cars (who is traveling together with who, the traffic due to an important match in the nearby stadium) we can identify suspicious behaviors and support traffic monitoring.

In several applications, as for example surveillance systems, it is important to provide *online* reasoning, so that the appropriate cause of action can be taken when necessary (*e.g.*, raise an alarm).

As another example, consider the problem of the surveillance of a big port that use a sensor network to monitor movements in the harbor. Criminals engaged in illicit trades on approaching boats try to minimize exposure to the port authorities. The port's sensor system might be able only to catch a fraction of the boats trajectories, or identify a fraudulent activity when it is too late for intervention; moreover, weather conditions could possibly limit the reliability of the sensors.

Under noisy observations condition, an automated reasoner needs to make use of all the information available in order to assess the most probable situation both in terms of individual

attributes (in our example, the most likely position of the boats) and joint attributes or relations (the connection between the boats: legal exchange, illegal encounter, no connection).

Indeed, complex contexts reasoning are also characterized by uncertainty not only on objects' observation but also on their relations.

In this work, we focus on multi-target tracking for activity recognition, in particular we study how to use explicit recognition of the relationships between interacting objects to improve the understanding of their dynamic model. The proposed approach has been validated on two different scenarios: a traffic monitoring system and a harbor surveillance system.

1.1 Relational multi-target tracking

Traditional (positional) *tracking* is defined as the problem of associating an object moving in a scene with its most likely trajectory over time. If performed *online* it requires to make such association at each time step. When more than one object is present in the scene, we have to deal with the problem of *multi target tracking*. Multi target tracking is the problem of jointly tracking a (possibly unknown) number of objects.

In this work we consider, in addition to the positions and the object's attributes, relations that represent joint properties of the objects. *Relational multi target tracking* is the problem of associating a set of objects or agents[1] with a full specification of the evolution of the value of their attributes and relations over time. Relational tracking is a paradigm first introduced in (Guibas, 2002) that we think can be seen as a general abstraction for many problems of context understanding.

In our work we model the relations in the context as a set of First-Order Logic (FOL) predicates. In any given situation, the state of the system is characterized by the evaluation of these predicates. In domains as sport, different players often move towards a specific coordinated action. In this case, the state represents the players' position, the type of action (*e.g.*, move on the side, cross in the center and shoot) and the participants (the players). The relations are not usually observed directly[2] as, for instance, we cannot recognize the type of action by simply looking at a single still frame extracted from a video. Instead, relations are inferred using the history of past observations and prior knowledge. Because of the uncertainty of observations (as motivated in the previous section), we represent our knowledge probabilistically, maintaining *beliefs* (conditional probabilities of the state given the observations) and updating them upon the acquisition of new information.

Furthermore, probabilistic inference can provide information that can be used to reason about the most likely course of action that will happen next. Returning to the sport example, the observations of previous phases of the game, combined with prior knowledge about playing habits, can be used to recognize the beginning of a particular pattern, and predict future moves.

An important contribution of this work is to show how modeling relations is useful with respect to two different goals:

[1] In this work we use the terms object, target or agent quite interchangeably; however we might use the term agent to underline the ability of proactive and deliberative reasoning.

[2] This will be discussed in details in Chapter 3

- Relations can improve the efficiency of the positional tracking. The information contained in the relationships can improve prediction, resulting in a better estimation of objects' trajectories with respect to the state of the art algorithms.

- Relations can be monitored as a goal in itself. Reasoning about relationship between different moving objects can be helpful to identify a particular activity. This is the case in many applications like traffic prediction or consumer monitoring, anomaly detection or activity recognition.

The achievement of these goals is based on the use of tools that extend the state of the art of probabilistic relational reasoning to dynamic domains. To this aim, we use *Relational Dynamic Bayesian Networks* (RDBNs) (see Chapter 2) as a formalism to model objects and relations between moving objects in the domain. In our relational dynamic Bayesian network-based model, relationships are considered as random variables whose values may change over time. While tracking the objects in the domain, we also track the evolution of their relationships, using a novel algorithm called *Relational Particle Filter* (RPF) (see Chapter 3).

1.1.1 Relational Dynamic Bayesian Networks

Logical and probabilistic reasoning have been traditionally seen as very different fields by Artificial Intelligence community. first-order logic systems can deal with rich representations but they cannot treat uncertainty. On the other hand, probabilistic models can deal well with uncertainty in many real-world domains, but they operate on a propositional level, and cannot scale to cases where several instances are present. Moreover, logic languages give an advantage in terms of expressivity.

Recently a lot of interest has arise towards approaches that integrate these two types of models; a prominent example is the work of Jaeger (Jaeger, 1997) on Relational Bayesian Networks (RBNs). A relational Bayesian network is a probabilistic graphical model whose nodes represent first-order logic predicates and whose probability distribution takes into account first-order logic quantifiers.

However in many situations the state evolves over time. As far as we know, not much work has been done to incorporate logical reasoning into dynamic domains; inference in such domains has been carried on only in propositional terms, for instance using Dynamic Bayesian Networks (DBNs) (Murphy, 2002).

In this Thesis we present relational dynamic Bayesian networks that are an extension of dynamic Bayesian network to first-order logic [3].

A relational dynamic Bayesian network is defined as a couple of relational Bayesian networks: the first provides the prior of the state of the relational domain, the second gives the probability distribution between time steps.

[3]The authors are aware of the works of Sanghai, Weld and Domingos on Relational Dynamic Bayesian Networks; however the paper presenting their work has been retracted. Refer to: http://www.aaai.org/Library/JAIR/Vol24/jair24-019.php

1.1.2 Relational Particle Filter

To accomplish both the task of tracking related multiple targets and recognizing complex activities, in this Thesis, we introduce a novel inference algorithm able to track both the position of the objects in the scene and their possible relations.

We extend the particle filter algorithm to deal with relations, introducing a new algorithm called *Relational Particle Filter* (RPF). A particle filtering technique recursively implements a Monte Carlo sampling on the belief over the state of the domain. In order to deal with the increased complexity of the state due to the introduction of relations, we adopt a particular state representation that factors the state in two parts: the *state of the attributes* and the *state of relations*. Our relational particle filtering takes advantage of this factorization and implements a two phases sequential Monte Carlo sampling.

1.2 Context Modeling

Context interpretation and context-based reasoning have been shown to be key factors in the development of algorithms for object recognition. In this field the context is the scene where objects are and the knowledge about it, is expressed by the beliefs over the scene (see (Derek Hoiem & Hebert, 2006) and (Elidan, Heitz, & Koller, 2006) as examples). Knowing the scene can improve the task of objects recognition; the knowledge about the identity of the objects improves the belief over the scene.

In this work we loosely consider the concept of context as "what is happening around the object we are tracking". We take advantage of the knowledge about what is happening in the scene (which "relations" are believed to be true in the scene) to improve the tracking and of the knowledge about the state of the objects to improve our knowledge about the relation between the objects in the scene (i.e. the context).

In the last years, computer vision has mostly dealt with the recognition of activities composed by the sequence of simple movements (Yan Ke & Hebert, 2007): in this Thesis we show how reasoning about relations between objects and/or the sequence of single different actions can help us in recognizing more complex activities.

To understand how relations can be used for context modeling, we describe the two scenarios that have been used as validating examples in this Thesis.

1.2.1 Scenario 1: traffic monitoring

Consider several vehicles traveling on a one-lane highway along which several highway entrances and exits are present. We want to track the vehicles, which are moving non-deterministically so that the future speed - and thus future position - cannot be exactly predicted by knowing the current state. As we have a limited number of possibly faulty and noisy sensors, we want to exploit the information that we can acquire from recognizing common behaviors due to relations.

The goal is to be able to track moving objects taking into account relations between them. For example, a vehicle moving at very high speed will eventually have to slow down if the cars

in front are moving substantially slower. Or we might want to monitor which cars are likely to be traveling together (because on a trip together or delivering to the same place). The value of the relation $TravelingTogether(X, Y)$ for a given X and Y cannot be computed on the basis of the current values of the other variable values. We need, instead, to infer this relation from the scene and from previous observations, and reason about our *beliefs* that two cars are traveling together.

A simple prior definition of this probability might express that two cars are very likely to be traveling together if they have the same size and color and enter at the same entrance in temporal proximity.

During the tracking, we update the belief with increased or decreased evidence about the fact that car X and car Y are traveling together. For example, the update should satisfy the following intuition:

- if car X exits but not car Y, the belief they are traveling together is greatly decreased: two cars that take different directions are not usually traveling together
- if X and Y are at a great distance for a long period; the belief probability decreases with respect to the number of time steps in which they are far away: the longer and the farther away, the less likely they are to travel together
- the closer X and Y are, the more likely the belief to travel together increases

Furthermore for this relation we can express the correlation between objects in the same relation: the observation that two vehicles are behaving similarly, produces evidence that they are in relation ($TravelingTogether$), but once we are quite sure that two vehicles are traveling together we can use this belief to predict that they will behave similarly in the future. We can then anticipate the behavior of all components of a group, predicting the value of other variables and relations.

These intuitive patterns for belief update are given by a precise and sound probabilistic semantics in the graphical model that we use.

1.2.2 Scenario 2: harbor surveillance system

Consider the problem of monitoring the approaches to a harbor from the sea and in particular the problem of detecting any behavior that might indicate that a ship represents a security risk or a law infringement. Monitoring the coast is complicated by the sparse, irregular, imprecise, and not always reliable nature of the surveillance data. Of course, the problem becomes even worst when multiple ships are approaching the coast.

Taking into account relations can improve the tracking. For example, if we know that a couple of ships are sailing together because in a tour together or because they belong to the same company (*i.e.*, if we have a certain belief over their relation), we know they will have a similar behavior or a similar motion and this will help us in tracking them. On the other hand, if we know there are multiple boats approaching the coast, we presume they will avoid collision, so we can predict their behavior such that they will not come too near one to the other.

Taking into account the relations between objects allows us to recognize complex activities like, for example, the "rendezvous" between ships. The activity of rendezvous is the activity of two ships that stop or travel slowly together to exchange goods. Common surveillance systems cannot detect the good that has been exchanged and have to detect those encounters from the behavior of the two ships.

A priori probability of two ships doing a rendezvous can be learned from data. During the tracking, we update the belief with increased or decreased evidence about the fact that two boat are involved in a rendezvous or not. For example, given two ships (X and Y) just entered the scene, $p(rendezvous(X, Y) = true)$ should satisfy the following intuition:

- if the distance between boat X and boat Y increases, the belief they are doing a rendezvous greatly decreased: two boats should be close to do a rendezvous
- if boat X decreases its speed but not boat Y the belief they are doing a rendezvous decreased: to do a rendezvous, two boats have to decrease their speed at almost the same time

Dealing with relations between moving objects allows us to distinguish the activity of rendezvous from the "pick up" (a vessel dropping a package into the water, that is quickly found and picked up by another vessel). Both these encounters have the common pattern of the two ships that approach each other and subsequently go apart, but in the rendezvous activity the two ships travel for a while together. Studying relations between ships allows us to recognize each of these two incidents and distinguish both of them from two ships that are avoiding each other, when one stop to let the other pass.

Furthermore, once we are quite sure that two boats are (or are not) involved in an encounter, we can use this belief to predict their future behavior.

1.3 Objectives and Contributions

This Thesis has the goal of studying how it is possible to reason with relations between moving objects in the context of multi-target tracking. An important part is devoted to literature review in both fields of probabilistic reasoning (and in particular relational reasoning) and computer vision. We mainly focus on the concept of relations in dynamic domains.

One of the main contributions of the Thesis is the development of an inference algorithm able to handle with relations between moving objects. The algorithm is a two-phases sequential Monte Carlo technique that samples the probability of the state of the objects given the previous state in two steps: the first step predicts the state of the objects'attributes and the second deals with the prediction of the relations between them. The key point is to divide the state of the relational domain in *state of the attributes* and *state of relations* and make the state of relations being probabilistically independent by the state of the attributes at the previous time-step.

A large part of this work concentrates on the validation of these techniques in different scenarios. In particular we show some results in the domain of traffic monitoring and activity recognition.

We evaluate the performance of the proposed method comparing it to a method that uses a standard sequential Monte Carlo technique and to heuristic algorithms that make use of static rules. Results show that our technique improves the ability of detecting anomalous behaviors without increasing the computational cost of the system. We also validate the hypothesis (discussed before) that relational reasoning gives us predictions that improve positional tracking.

1.4 Overview

In the following, we present the organization of the Thesis. This chapter has introduced the basic ideas and the motivations of this work. The remain of the Thesis can be divided in two parts. In the first part, we start describing the problem of relational reasoning and the problem of reasoning in dynamic domains, introducing the proposed modeling approach based on relational dynamic Bayesian networks (Chapter 2). Then we introduce the inference problem and our relational particle filter algorithm (Chapter 3).

In the second part of this Thesis, we discuss possible applications of our model compared with the state of the art (Chapter 4) and we evaluate our approach on different scenarios (Chapter 5); finally, we describe possible improvements (Chapter 6) considering other possible applications and draw our conclusions.

Chapter 2 We present the state of the art for reasoning with relations in uncertain domains. We define first-order logic, probabilistic relational models and dynamic Bayesian networks. Finally we introduce relational dynamic domains and relational dynamic Bayesian networks.

Chapter 3 We address the problem of inference in relational dynamic domains introducing our relational particle filtering algorithm.

Chapter 4 In this chapter we consider particular applications and discuss the fundamental problems and challenges posed by the design of activity recognition and surveillance systems, reviewing relevant works from the computer vision field.

Chapter 5 This chapter presents the results obtained applying our method to both the problem of traffic monitoring and harbor surveillance.

Chapter 6 We provide a brief summary of the contributions and limitations of this Thesis and we discuss promising future research directions.

Chapter 2

Modeling uncertainty in Relational Dynamic Domains

I'm Winston Wolfe. I solve problems.

from the movie "Pulp fiction"

Uncertainty is a fundamental and irreducible aspect of our knowledge about the world; probabilistic models provide a natural, sound and coherent foundation for its representation.

In this chapter we present a novel framework to model uncertainty in dynamic relational domains. The uncertainty about the state of the world can be modeled with a joint distribution for a set of random variables representing the attributes of the objects in our world. In principle we could just list all the complete instantiations of the objects' attributes and specify a probability for each one (this is the "atomic" or "naive" representation); as long as the probabilities we specify add up to one, then this specification will indeed define a unique distribution. However, this approach is not generally feasible for real-world scenarios: the number of cases grows exponentially with the number of variables. This is a problem both computationally, because the model requires exponential space and time to answer queries, and statistically, because the number of probabilities to estimate from data will be exponentially large.

Probabilistic graphical models, instead, allow a compact representation of the uncertainty about the state of the world. They provide a graphical structure that shows the dependencies between objects' attributes and constraint the probabilistic model only on this dependencies.

We present a probabilistic graphical model able to take into account relations in dynamic domains. In this chapter we first review the literature about probabilistic graphical models for static and dynamic domains; then we review probabilistic relational graphical models, that support first-order logic; finally we extend the latter to model dynamic domains defining relational dynamic Bayesian networks.

2.1 Probabilistic Graphical Models

Probabilistic graphical models are graphs in which nodes represent random variables, and arcs represent conditional dependence assumptions. These models provide a compact representation

of the joint probability distribution of the set of random variables representing the world in a compact and natural way.

There are two main kinds of graphical models: undirected and directed. Undirected graphical models, also known as Markov networks or Markov random fields (Chellappa & Jain, 1993), are more popular with the physics and vision communities. Directed graphical models (Computer, Russell, Pearl, & Russell, 1994), also known as Bayesian networks, belief networks, generative models, causal models, etc. are more popular with the Artificial Intelligence and Machine Learning communities. It is also possible to have a model with both directed and undirected arcs, which is called a chain graph (Studeny & Bouckaert, 1998).

While in a directed graphical model an arc from A to B can be informally interpreted as indicating the existence of a causal dependency between A and B, in an undirected graphical model this would show the simple existence of a (symmetric) connection between the two variables. Since it is a common sense rule to think about the past "causing" the future, directed graphical model can more naturally be extended to model dynamic domains and for this reason in this Thesis we will use them to model relations between objects in dynamic domains.

In the following, we first introduce Bayesian networks and dynamic Bayesian networks (for problems in static and dynamic domains) then we introduce relations in static domains introducing relational Bayesian networks. Finally, we extend relational Bayesian networks to dynamic domains introducing relational dynamic Bayesian networks that are a new framework to model relations between moving objects using first-order logic.

Relational dynamic Bayesian networks extend dynamic Bayesian networks with first-order logic as Bayesian networks has been extended to relational Bayesian networks, combining the representative power of first-order logic to reason about moving objects in the world.

2.1.1 Bayesian Networks

Bayesian Networks (BNs) (Pearl, 1986) encode the joint probability distribution of a set of variables, $x_1, \cdots, x_n$, exploiting independence properties. We will introduce BNs with the following simple example, first used by Pearl in (Pearl, 1986).

Example 1 *Suppose I have a home alarm system that is designed to be triggered by would-be burglars, but can also be set off by small earthquakes, which are common where I live. If my alarm goes off while I am at work, my neighbors John and Mary may call to let me know.*

My beliefs about this scenario can be formalized with a probability distribution over the product space of five variables: $Burglary$ (represented by letter B), $Earthquake$ (E), $Alarm$ (A), $JohnCalls$ (J), and $MaryCalls$ (M). Each of these variables is Boolean, taking values in the set $\{T, F\}$. Figure 2.1 shows a BN for this example. A BN consists of two parts,

1. the BN structure and
2. the Conditional Probability Distributions (CPDs).

Hence directed cycles are disallowed, the BN structure is a directed acyclic graph with a node for each random variable. Random variables represent objects' attribute in the domain.

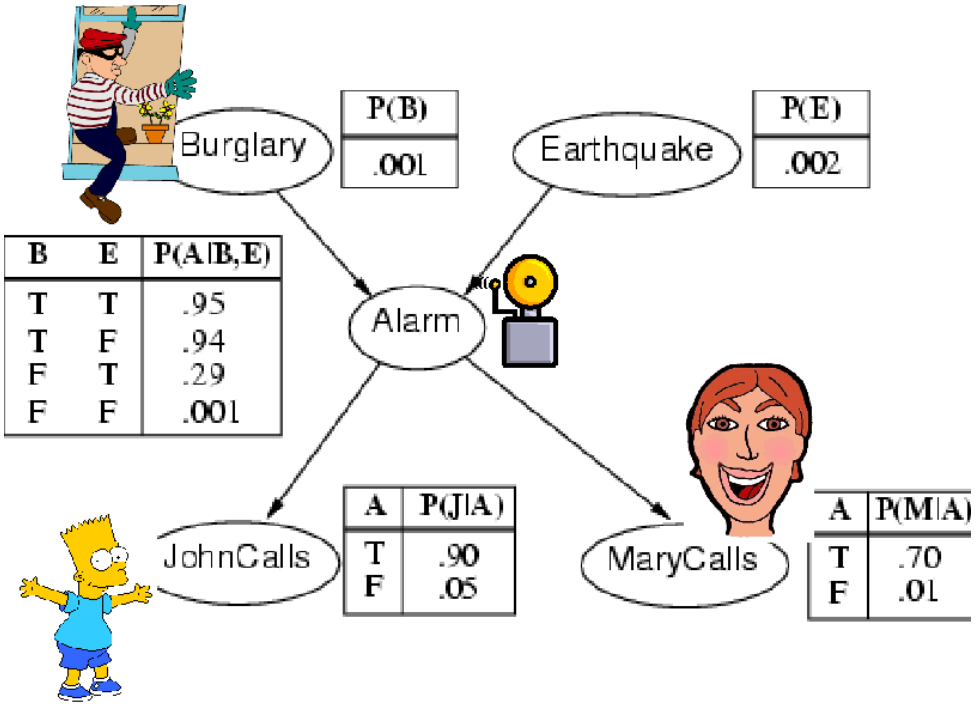


Figure 2.1: A BN for the Example 1, including the BN structure and conditional probability tables (figure from (Russell & Norvig, 2002)).

The nodes with an arc to x are the *parents* of x. We will denote the set of parents of a variable x in the BN B as $Pa_B(x)$. An edge in the graph represents the dependency of an object's attributes (or variable) from its parents.

In our example the variable $Alarm$ depends on the variables $Burglary$ and $Earthquake$, we will say:

$$Pa(A) = \{B, E\}. \tag{2.1}$$

For each variable x, B specifies a CPD for x given $Pa_B(x)$. The structure of the network encodes the assertion that each node is conditionally independent of its non-descendants given its parents. The probability of an arbitrary event $X = (x_1, \cdots, x_d)$ can then be computed as $p(X) = \prod_{i=1}^{d} p(x_i|Pa_B(x_i))$. A formal definition of BN is the following:

Definition 1 *A BN is a direct acyclic graph which nodes are conditionally independent of its non-descendants given its parents (this is also called* local Markov property*).*

If we topologically order the nodes (parents before children) as $1, \cdots, N$, we can write the joint distribution as follows (Russell & Norvig, 2002):

$$\begin{aligned} p(x_1, ..., x_N) &= p(x_1)p(x_2|x_1)p(x_3|x_1, x_2)\cdots p(x_N|x_1, \cdots, x_{N-1}) \\ &= \prod_{i=1}^{N} p(x_i|x_{1:i-1}) \\ &= \prod_{i=1}^{N} p(x_i|Pa_B(x_i)) \end{aligned} \tag{2.2}$$

where $x_{1:i-1} = (x_1, \cdots, x_{i-1})$. The first line follows from the chain rule of probability (see Appendix A), the second line is the same as the first, and the third line follows because node x_i is independent of all its ancestors, $x_{1:i-1}$, given its parents. In our example,

$$\begin{aligned} p(B = T, E = F, A = T, J = F, M = T) = \\ p(B = T)p(E = F)p(A = T|B = T, E = F)p(J = F|A = T)p(M = T|A = T). \end{aligned} \tag{2.3}$$

When x_i and all its parents can assume a finite set of discrete values, a CPD for x_i can be represented as a Conditional Probability Table (CPT) with a row for each instantiation of $Pa_B(x_i)$. This is illustrated in Figure 2.1. Note that in this example, the CPTs contain only 20 probability values. In fact, since the values in each row of each CPT must sum to one, this representation has only 10 free parameters. By contrast, a table listing probabilities for all 32 instantiations of these 5 binary variables would have 31 free parameters. Thus, even for this small example, the BN is considerably more compact than an atomic representation. The advantage of a BN increases with the number of variables: while an explicit representation of a joint distribution for n k-ary variables has k^{n-1} parameters, a BN representation in which each variable has at most m parents has only $O(nk^m)$ parameters.

A BN can be used to reason about any attribute of the objects in the domain, given any set of observations. It can thus be used for a variety of tasks, including classification (Friedman, Geiger, & Goldszmidt, 1997), prediction (Jansen, Yu, Greenbaum, Kluger, Krogan, Chung, Emili, Snyder, Greenblatt, & Gerstein, 2003), and decision making (wu Liao, Wan, & Li, 2008). For instance, imagine we observed that both John and Mary call, which is the probability of the variable $Burglary$ to be true? We can compute the probability of the variable $Burglary$ to be true as follow:

$$p(B = T|, J = T, M = T) = \alpha \sum_{E} \sum_{A} p(B = T)p(E)p(A|B = T, E)p(J = T|A)p(M = T|A), \tag{2.4}$$

where we marginalized (see Appendix A) over the variable A and E to compute the probability of each value of that variable. To compute this expression, we have to add four terms (one for each possible combination of the values of the variable $Alarm$ and $Earthquake$) each computed by multiplying five numbers using the probability tables in Figure 2.1. The probability of the $burglary$ being true given that both John and Mary called is 0.00059236.

The probabilistic semantics also gives a strong foundation for the task of learning models from data. Techniques currently exist for learning both the structure and the parameters, for dealing with missing data and hidden variables, and for discovering causal structure.

2.2 Modeling sequential data

Most of the events that we meet in our everyday life are not detected based on a particular point in time, but they can be described through a multiple states of observations that yield a judgement of one complete final event. Statisticians have developed numerous methods for reasoning about temporal relationships among different entities in the world. This field is generally known as time-series analysis. Time-series is a sample realization of a stochastic process, consisting of a set of observations made sequentially over time.

Time is also an important dimension in the field of artificial intelligence and reasoning. However, BNs do not provide direct mechanism for representing temporal dependencies. In attempting to add temporal dimension into the BN models various approaches has been suggested. Between others, hidden Markov models and Kalman filter models are popular models because they are simple and flexible. For example, hidden Markov models have been used for speech recognition and bio-sequence analysis, and Kalman filter models have been used for problems ranging from tracking planes and missiles to predicting the economy. However, hidden Markov models and Kalman filter models are limited in their "expressive power". Hidden Markov models constrain the state to be represented as a single random variable, Kalman filter models constrain the probability distributions to be Gaussian.

Dynamic Bayesian Networks (DBNs) generalize hidden Markov models by allowing the state to be represented in factored form and generalize Kalman filter models using arbitrary probability distributions.

2.2.1 Dynamic Bayesian Networks

DBNs are an extension of BNs for modeling dynamic domains. In a DBN, the state depends on the time t and is represented by a set of random variables $X_t = (x_{1,t}, ..., x_{d,t})$. The state at time t depends on the states at previous time steps.

Typically, we assume that each state only depends on the immediately preceding state (*i.e.*, the system is *first-order Markov*), and thus we need to represent the probability distribution $p(X_t|X_{t-1})$. This can be done using a two-time-slice BN fragment ($2TBN$):

Definition 2 *A $2TBN$ is a BN that contains variables from X_t whose parents are variables from X_{t-1} and/or from X_t, and variables from X_{t-1} without their parents.*

A $2TBN$ (B_t) defines $p(X_t|X_{t-1})$ by means of a directed acyclic graph as follows:

$$p(X_t|X_{t-1}) = \prod_{i=1}^{N} p(X_{i,t}|Pa_{B_t}(X_{i,t})). \tag{2.5}$$

The nodes in the first slice of a $2TBN$ do not have any parameters associated with them, but each node in the second slice of the $2TBN$ has associated a CPD, which defines $p(x_{i,t}|Pa_{B_t}(x_{i,t}))$ for all $t > 1$. The distribution given by a $2TBN$ can be divided in two:

- the *inter-slice distribution*, that models the probability of variables in X_t with parents at time $t - 1$ and

- the *intra-slice distribution* that models the probability of variable in X_t with parents in the same time slice.

We assume that the parameters of the CPDs are time-invariant, *i.e.*, the model is time-homogeneous.

Typically, we also assume that the process is stationary, *i.e.*, the transition models for all time slices are identical: $B_1 = B_2 = ... = B_t = B_{\rightarrow}$.

Definition 3 *A DBN is defined to be a pair of BNs (B_0, $B_{\rightarrow}$), where*

- *B_0 represents the initial distribution $p(X_0)$, and*
- *$B_{\rightarrow}$ is a $2TBN$, which defines the distribution $p(X_t|X_{t-1})$.*

The set X_t is commonly divided into two sets: the unobserved state variables S_t and the observed variables Z_t. The observed variables Z_t are assumed to depend only on the current state variables S_t. The joint distribution represented by a DBN can then be obtained by unrolling the $2TBN$:

$$p(S_0, ..., S_T, Z_0, ..., Z_T) = p(S_0)p(Z_0|S_0) \prod_{t=1}^{T} p(S_t|S_{t-1})p(Z_t|S_t) \tag{2.6}$$

where $p(S_0)p(Z_0)$ is the distribution given by B_0 and $\prod_{t=1}^{T} p(S_t|S_{t-1})p(Z_t|S_t)$ highlights the intra-slice $p(Z_t|S_t)$ and the inter-slice $p(S_t|S_{t-1})$ distributions:

$$p(X_t|X_{t-1}) = p(S_t|S_{t-1})p(Z_t|S_t) \tag{2.7}$$

To show the different parts of a DBN we consider the following oversimplified example (Russell & Norvig, 2003);

Example 2 *Suppose you are the security guard at some secret underground installation. You want to know whether it is raining today, but your only access to the outside world occurs each morning when you see the director coming in with, or without an umbrella.*

In this example,

- the intra-slice distribution is represented by the probability that the director has taken the umbrella if it is raining (or not),
- the inter-slice distribution is given by the probability of a rainy day given the weather of the previous day.

For each day t, the set Z_t contains a single observed variable: U_t, whether the director takes the umbrella or not. The set of the unobserved state variables contains a single variable: R_t, whether it is raining or not. In Figure 2.2 the DBN is reported and the $2TBN$s are highlighted.

Note that the term dynamic means we are modeling a dynamic system, not that the network changes over time.

DBNs are a good tradeoff between expressiveness and tractability, and include the vast majority of models that have been proved successful in practice.

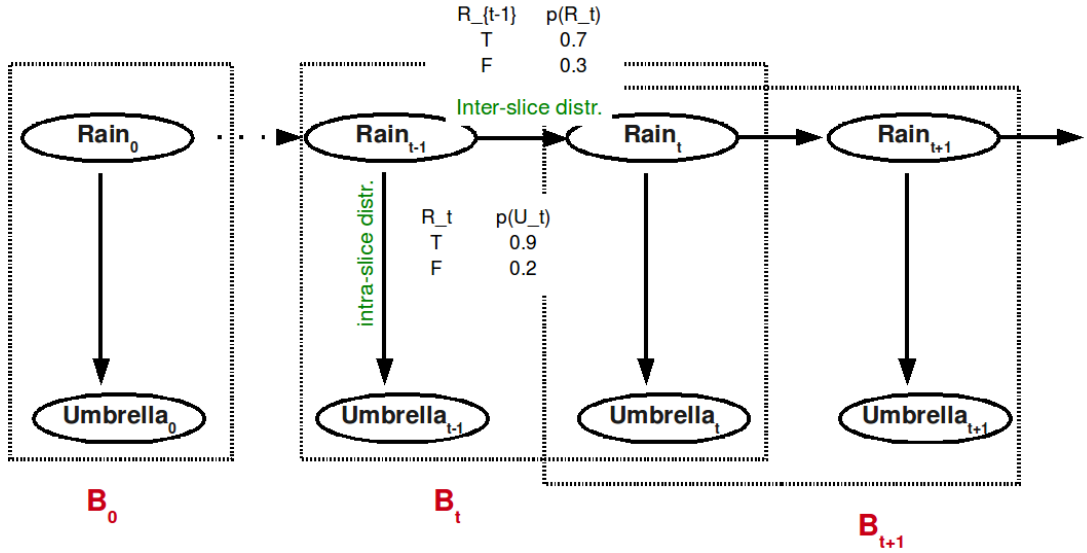


Figure 2.2: A DBN for the Example 2. In the figure, the the intra-slice and inter-slice distributions are reported together with the $2TBN$. (figure from (Russell & Norvig, 2002)).

2.3 Modeling relations

One of the main limitations of BNs is that they represent the world in terms of a fixed set of variables. Consider the Example 1) and consider the case in which I have more than two neighbors and they have neighbors themselves Figure 2.3: we need to explicitly represent each neighbor as a variable with its specific CPT. Indeed, graphical models are incapable of reasoning explicitly about classes of objects (*e.g.*, class $Neighbor$), and thus cannot represent models over domains where the set of entities and the relations between them are not fixed in advance. They are propositional, as opposed to first-order: in other words, they do not support quantification over objects. As a consequence, BNs are limited in their ability to model large and complex domains.

Probabilistic Relational Models (PRMs) are a language for describing probabilistic models based on the significantly more expressivity of first-order logic. They allow the domain to be represented in terms of object classes, their properties (or attributes), and the relations between them. These models represent the uncertainty over the properties of an entity, representing its probabilistic dependence both on other properties of that entity and on properties of related entities.

2.3.1 First-Order Logic

First-order logic (FOL) is a formal language interpreted by mathematical structures. FOL is a system of deduction that extends propositional logic by allowing quantification over classes of a given domain (*the universe*). *Objects*, *relations* and *quantifiers* are the three main components of FOL.

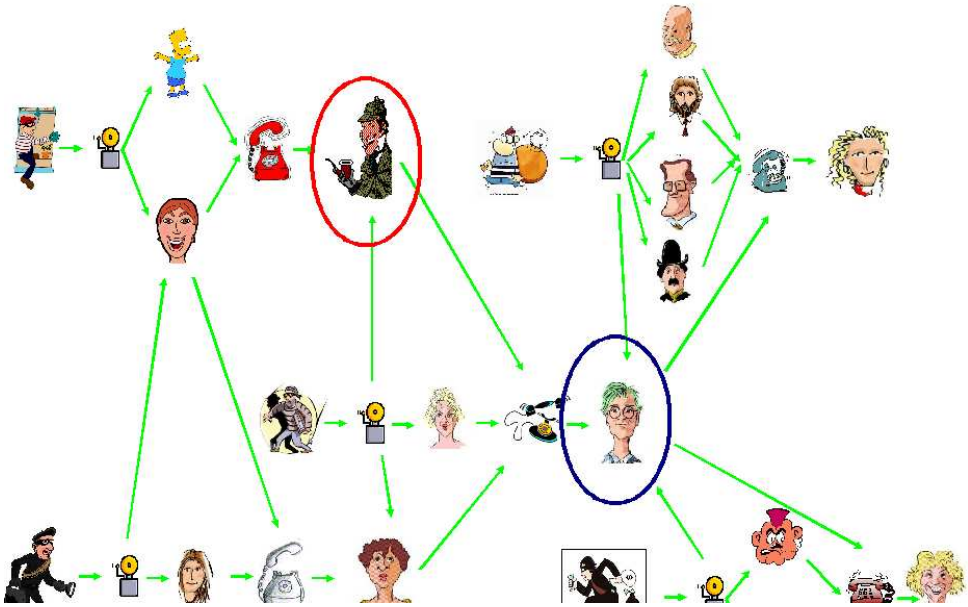


Figure 2.3: A BN for the Example 1 extended to relational domains. If we have more than 2 neighbors we have to instantiate a variable for each neighbor. *Thanks to Mark Chavira for providing us with this image.*

Murphy in (Murphy, 2002) states that: "Objects (objects classes) are basically groups of attributes which "belong together", c.f.r. a structure in a programming language, once completely instantiated (grounded) they give rise to a particular object in the domain". Object classes are characterized by *attributes* and are related one another through relations. *Propositions* over objects can be expressed by quantifiers.

While propositional logic deals with simple declarative propositions, FOL additionally covers predicates and quantification. Take for example the following sentences: "John is my neighbor", "Mary is my neighbor". In propositional logic these will be two unrelated propositions, denoted for example by p and q. In FOL however, both sentences would be connected by the same attribute: $x.MyNeighbor$, where $x.MyNeighbor$ means that x is one of my neighbors. When $x = John$ we get proposition p, and when $x = Mary$ we get proposition q. Such a construction allows for a much more powerful logic when quantifiers are introduced. Consider for example the quantifier "for every" ($\forall$): "$\forall\ x$, if $x.MyNeighbor \rightarrow x.CallMe$", enounce a proposition that is valid for each x..

Without quantifiers, every valid argument in FOL is valid in propositional logic, and vice-versa.

The vocabulary of the FOL is composed of

1. *Constants*: symbols usually used to represent objects or their attributes; they are often denoted by lowercase letters at the beginning of the alphabet a, b, c,... .

2. *Variables*: symbols that range over the objects; they are often denoted by lowercase letters

at the end of the alphabet x, y, z,... . Both the constants and the variables can be typed, in which case the variables take on values only of the corresponding type.

3. A set of *functions*, each of some valence ≥ 1 that fixes the number of inputs it can take. Functions take objects as input and return object, and are often denoted by lowercase letters f, g, h,...

4. *Predicates*: symbols used to represent relations between objects in the domain or attributes of objects which are often denoted by uppercase letters P, Q, R,... . Each predicate symbol is associated with an arity. A ground predicate is a predicate with constant as arguments (*i.e.,* not variables).

An *interpretation* for a relational domain, assigns a semantic meaning to each object, function and relation in the domain. Each ground predicate is associated with a true value in an interpretation.

Existential quantifiers in FOL are handled by checking whether the predicate is true for any object in the current state of the domain.

A *term* l in the FOL can be

- a constant symbol, as a, b, 0, 1
- a variable, as for example x, y
- a function of valence n applied to n terms $f(l_1, \cdots, l_n)$.

A *first-order formula* assumes one of the following forms:

1. $R(l_1, \cdots, l_n)$ where R is a predicate of arity n and l_i are terms,
2. $\neg F$ or $(F' \wedge F'')$ or $(F' \vee F'')$ where F, F' and F'' are first-order formula,
3. $\exists x F(x)'$ or $\forall x F(x)'$, where x is a variable and $F(x)'$ is a first-order formula,
4. $\sharp(= n)xF(x)$ or $\sharp(< n)xF(x)$ or $\sharp(> n)xF(x)$, where x is a variable, $F(x)$ a first-order formula and n an integer.

2.3.2 Relational Domain

A relational domain contains a set of objects with relations between them.

Definition 4 *A* relational domain *is a set of constants, variables and predicates that represent the objects and their relations in the domain.*

The set of all true ground predicates can be represented explicitly as tuples in a relational database. This corresponds to the *state* of the world.

Definition 5 *The state of a relational domain (*relational state*) is the set of all the ground predicates that are true.*

In an uncertain domain, the truth value of a ground predicate can be uncertain and the value can potentially depend on the values of other ground predicates. These dependencies can be specified using a BN on the ground predicates. However, the number of such ground predicates is exponential in the size of the domain and hence the explicit construction of such a BN would be infeasible.

Relational Bayesian networks were introduced to compactly represent the uncertainty in this setting.

2.3.3 Relational Bayesian Networks

A Relational Bayesian Network (RBN) specifies dependencies between predicates at the first-order level by using first-order expressions which include existential and universal quantifiers.

Definition 6 *Given a relational domain, a RBN (RB) is a graph that, for every FOL predicate R, contains:*

- *A node in the graph.*
- *A set of parents $Pa_{RB}(R) = \{R_1, ..., R_l\}$ which are a subset of the predicates in the graph.*
- *A conditional probability model for $p(R|Pa_{RB}(R))$ which is a function with range $[0, 1]$ defined over all the variables in $Pa_{RB}(R)$.*

We come back to the previous example and modify it to explain the differences between BNs and RBNs.

Example 3 *Suppose I live in a building where each owner has a home alarm system that is designed to be triggered by would-be burglars, but can also be set off by small earthquakes, which are common where we live. If one of these alarms goes off while the owner is at work, his neighbors may call him to let him know. The neighbors have an uncertain knowledge about whose alarm went off and they are less likely to call when there is noise or when they are not paying attention.*

The objects in this relational domain can be represented by the variables: $Earthquake$, $Burglar$, $House$, $Neighbor$. Each object has some attributes that can characterize their instantiation: for the object $House$, it can be $AlarmRinging$ indicating if the alarm of the house is ringing, for the object $Neighbor$, it can be $AttentionDegree$ and $NoiseAround$, describing the reliability of the neighbor. The type space of $Neighbor$ is $person$ and the attribute $Neighbor.AttentionDegree$ ranges over the set of constants $\{High, Low\}$. Moreover there will be some relations between objects: *e.g.,* the relation $Enter$ of arity 2 will represent the relation of a $Burglar$ to enter an $House$, the relation $ToCall$ will relate a $Neighbor$ to the $House.Howner$ which he will eventually call if he hears an alarm. Figures 2.4 and 2.5 report the objects and the relations of the domain.

Following Definition 6 the RBN for this problem will be a graph which for every predicate R (representing both objects' attributes or relations) contains a node in the graph and

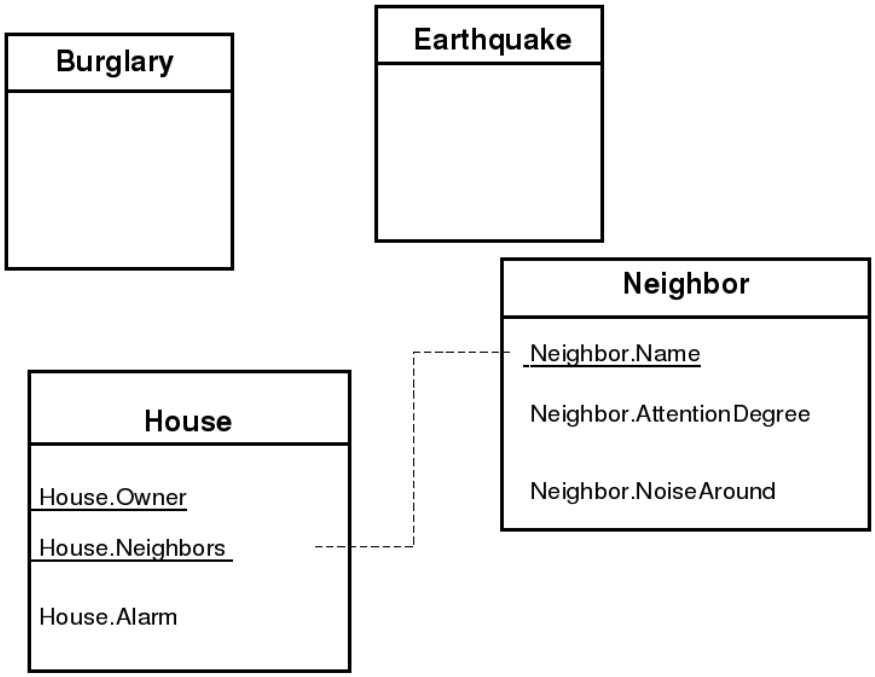


Figure 2.4: The objects and the attributes of the relational domain of the example 3. We show the objects as usually done for relational data bases, the dashed line refers to foreign keys.

a set of parents $Pa(R)$ that "cause" the value of the predicate R. *E.g.*, the parents of the relation $ToCall(Neighbor, House.Owner)$ will be $Neighbor.AttentionDegree$, $Neighbor.NoiseAround$ and $House.AlarmRinging$ (Figure 2.6 reports this RBN).

The RBN in the example presents more nodes than the BN of Figure 2.1 and it encapsulates much more information, in fact it can be used to explain the dependencies in each neighborhood we want to consider independently from the number or the type of neighbors an owner has got.

A RBN defines a BN on the ground predicates in the relational domain. It has not to be acyclic but its complete instantiation defining a BN has to[1]. For every ground predicate $R(c_1, ..., c_m)$ a node is created together with its parents'nodes obtained by instantiating the predicates which appear in $Pa_{RB}(R)$. The conditional model for a ground predicate is, therefore, restricted to the particular ground predicate and its parents. Thus, a RBN gives a joint probability distribution on the state of the relational domain.

To avoid cycles appearing in the BN obtained after grounding it is necessary to restrict the set of parents of a predicate assuming an ordering. The ordering $\prec$ between the ground predicates is given by the following rules:

1. $R(x_1, \cdots, x_n) \prec R'(x'_1, \cdots, x'_n)$ if $R \prec R'$

[1]This means that an attribute of an object can depend by the same attribute of another object of the same class; this leads to a cycle at the object level that reveals to be not a cycle at the grounding level.

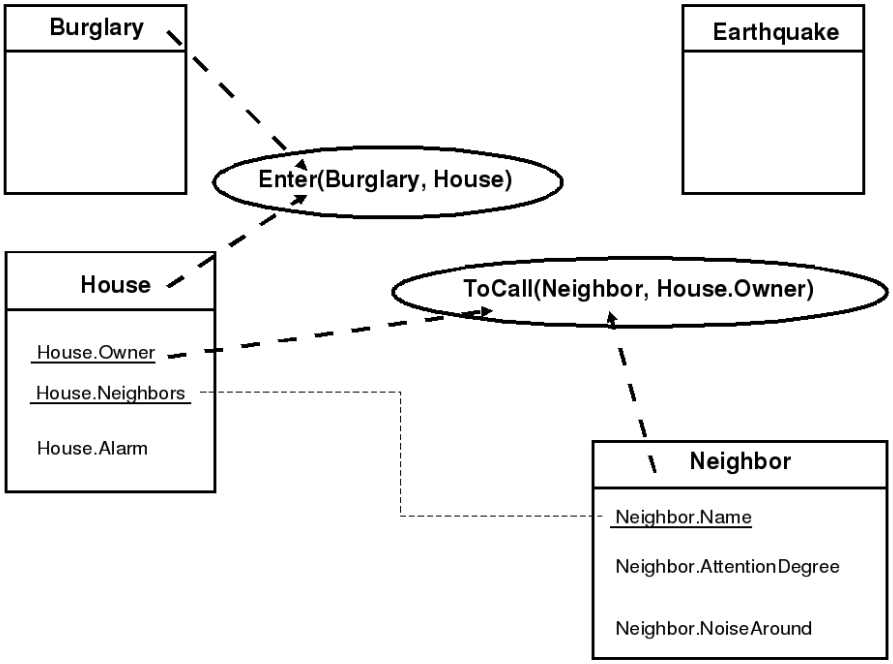


Figure 2.5: The relations of the relational domain of the example 3. With dashed bolt lines we represent which objects participate in which relations.

2. $R(x_1, \cdots, x_n) \prec R(x'_1, \cdots, x'_n)$ if $\exists i : x_i \prec x'_i$ and $x_j = x'_j, \forall j < i$

The set of parents of a predicate in a RBN is restricted as follows:

- The parent set $Pa(R)$ can contain a predicate R' only if either $R' \prec R$ or $R' = R$
- If $Pa(R)$ contains R then, during the grounding, $R(x_1, \cdots, x_n)$ has parents $R(x'_1, \cdots, x'_n)$ only if $R(x'_1, \cdots, x'_n) \prec R(x_1, \cdots, x_n)$.

This ordering implies that in the resulting BN each ground predicate can only have higher order ground predicates as parents.

The conditional model can be any first-order conditional model and can be chosen depending on the domain, the model's applicability and the easy of use. We will use first-order probabilistic trees.

First-Order Probabilistic Trees

The most general way to model the conditional model is to use an arbitrary CPT that can represent any possible distribution.

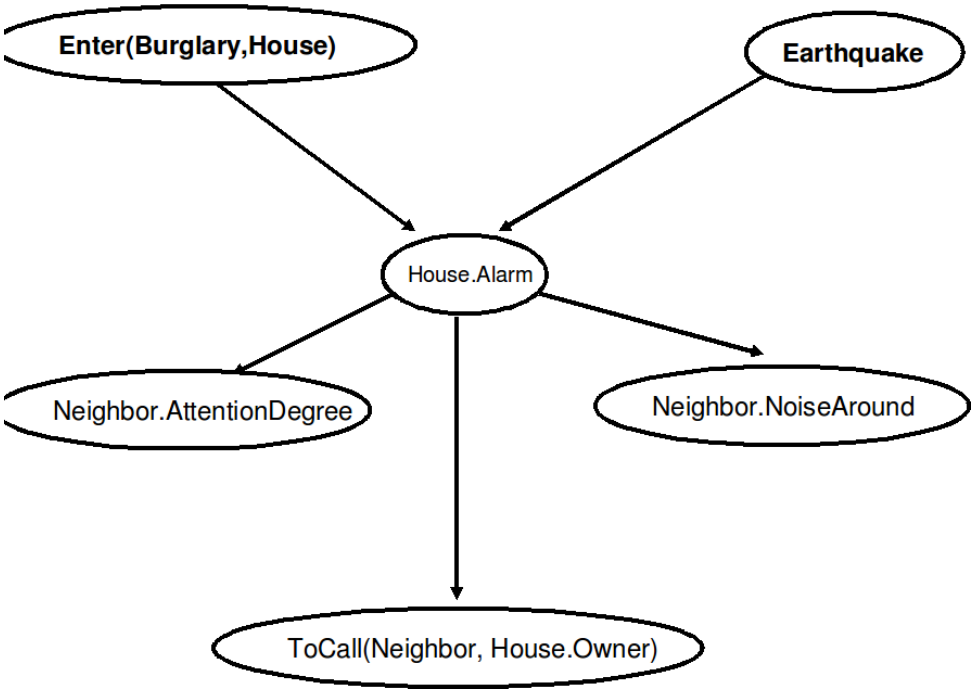


Figure 2.6: The RBN for the example 3.

Generally a CPT representation has an high memory cost: because the number of entries is exponential in the number of relations and attributes of the domain. Indeed, given n objects each with k attributes of d possible values and r binary relations, the state of the attributes requires d^{nk} cases, while each binary relation associates 2 possible values (true or false) to any pair of objects and there are n^2 pairs. In total the entries of the required CPT would be $d^{nk}2^{n^2r}$.

For this reason, it is generally preferred to have a compact representation of the CPTs. A way to encode this probability is to use a *First-Order Probabilistic Tree* (FOPT). FOPTs, also called first-order decision diagrams (C. Wang & Khardon, 2008), are probabilistic trees whose nodes are first-order logic formulas.

Definition 7 *Given a predicate R and its parents $Pa(R)$, a FOPT is a tree where:*

- *each interior node (k) is associated with a first-order logic formula F_k whose arguments are a subset of $Pa(R)$,*
- *each child of k corresponds to either the true or false outcome of F_k*
- *the leaves are associated with a probability distribution function over the possible values of R.*

A FOPT's node can contain a formula with free variables and quantifiers/aggregators over them. Moreover, the quantification of a variable is preserved throughout the descendants, *i.e.*, if a variable x is substituted by a constant c at a node n, then x takes c as its value over all the

descendants of n. To avoid cycles in the network, quantified variables in a FOPT range only over values that precede the child node's values in the ordering. The function at the leaf gives the probability of the ground predicate being true.

Just like a BN is completely specified by providing a CPT for each variable, a RBN is completely specified by having an FOPT for each first-order predicate.

2.4 Related Works

In this section we review the relevant works done on relational probabilistic modelization. As mentioned in the introduction (Chapter 1), a lot of work has been done to incorporate FOL reasoning and Bayesian uncertainty. The definition of RBN we introduced is most closely related to the one of *Relational Bayesian Network* given by Jaeger in (Jaeger, 1997) even if he constrains CPDs to be combination functions (such as noisy-or) while we use FOPTs. He presents a sophisticated scheme for combination functions, including the possibility of their nesting.

In (Friedman et al., 1999) and in (Koller, 1999) Probabilistic Relational Models (PRMs) are defined with the formalism of frame systems used as a starting point. The language of frames, similar also to relational databases, consists of defining a set of classes, objects and their attributes. PRMs add probabilities to frame systems by specifying a probability distribution for each attribute of each class as a generic CPT. The only difference from our definition of RBN is the fact that parents of an attribute that are attributes of related classes are reached via some *slot chain*. A slot chain in a frame-based system performs the same function of a foreign key in a relational database. A slot chain can be viewed as a sequence of foreign keys enabling one to move from one table to another. In our definition of RBN no restriction is imposed over the reachability of the nodes.

RBNs as defined in this chapter subsume PRM (Friedman et al., 1999) in fact, replacing the attributes of a PRM by FOL predicates would lead to define PRMs as a particular example of RBNs (see Appendix B).

On the other hand, Domingos *et al.*, (Domingos, Kok, Lowd, Poon, Richardson, Singla, Sumner, & Wang, 2008) represent uncertainty in the domain by the use of undirected graphs as Markov logic networks and focus on the inference task. Markov logic networks are a recent and rapidly evolving framework for probabilistic logic that has a very simple semantics while keeping the expressive power of FOL. Markov logic networks consist of a set of weighted first-order logic formulas and a universe of objects. Its semantics is simply that of a Markov network whose features are the instantiations of all these formulas given the universe of objects. Markov logic networks are a powerful language accompanied by well-supported software (called Alchemy) which has been applied to real domains. Its major drawback is due to its impossibility to represent quantification over objects, replaced by the disjunction of their grounding (this is possible because the domains are assumed to be finite). For this reason dealing with very large networks (as dynamic networks generally are) for Markov logic networks is very difficult.

Kersting and DeRaedt (Kersting & Raedt, 2000) introduce Bayesian logic programs to provide a language which is as syntactically and conceptually simple as possible while preserving

the expressive power of the works presented so far. According to the authors, this is necessary to understand the relationship between all these approaches, and the fundamental aspects of probabilistic relational models.

Milch (Milch, 2006) introduced BLog (Bayesian Logic) that provides a language that uses FOL to extend inference over set of objects belonging to the same class. However, he does not seem to take into account the objects' movement nor the relations that influence it.

There has been very limited work on extending relational models to dynamic domains. Dynamic object-oriented Bayesian networks (Friedman, Koller, & Pfeffer, 1998) combine DBNs with object-oriented Bayesian networks, a predecessor of PRMs. Unfortunately, no efficient inference methods were proposed for dynamic object-oriented Bayesian networks.

Glesner and Koller (Glesner & Koller, 1995) proposed the idea of adding the power of FOL to DBNs. However, they only give procedures for constructing flexible DBNs out of first-order knowledge bases, and consider inference only at the propositional level. Relational Markov models (Anderson, Domingos, & Weld, 2002) and logical hidden Markov models (Kersting & Raiko, 2005) are an extension of hidden Markov models to first-order domains and as hidden Markov models present the shortcoming of being able to model only single-variable states.

2.5 Introducing Relations in Dynamic Domains

One of the purposes of this Thesis is the introduction of relations in dynamic domains. We want to extend DBNs with FOL as BN has been extended to RBNs. In this way we will combine the representative power of FOL to reason about moving objects in the world.

While in the previous section we defined the state of a relational domain, in this section we consider relational domains in which the state evolves with time, these are called *dynamic relational domains*.

2.5.1 Relational Dynamic Bayesian Networks

Relational Dynamic Bayesian Networks (RDBNs) extend RBNs to model dynamics in relational domains. To define relational dynamic Bayesian networks, we have first to define dynamic relational domains.

Dynamic relational domains are relational domains where the state can change at every time step. In a dynamic relational domain a ground predicate can be true or false depending on the time step. Therefore we have to add a time argument to each predicate: $R(x_1, ..., x_n, t)$, where t is a non-negative integer variable and indicates the time step.

Definition 8 *A dynamic relational domain is a set of constants, variables, and predicates that can change their value with time.*

As done for the relational domain, we can define the (relational) state of a dynamic relational domain as follows:

Definition 9 *The state of a dynamic relational domain at time t is the set of all the ground predicates in the domain that are true at time t.*

We now introduce *Relational Dynamic Bayesian Network* (RDBN) that model uncertainty in a dynamic relational domain.

Following the definition of DBN reported in Section 2.2.1, to define a RDBN we have first to define a *two-time-slice RBN* ($2TRBN$).

Definition 10 *A $2TRBN$ is a graph which given the state of the domain at time $t-1$ gives a distribution on the state of the domain at time t. It contains*

- *predicates at time t (R_t) whose parents are predicate at time $t-1$ and/or t, and*
- *predicates at time $t-1$ without their parents.*

As a DBN is defined as a pair of BNs, a RDBN can be defined as a pair of RBNs:

Definition 11 *A RDBN is a pair of networks $(BR_0, BR_{\rightarrow})$, where BR_0 is an RBN with all $t=0$ and $BR_{\rightarrow}$ is a $2TRBN$.*

- *BR_0 represents the probability distribution over the state of the Relational Domain at time 0.*
- *$BR_{\rightarrow}$ gives the probability distribution on the state of the domain at time t given the state of the domain at time $t-1$.*

An RDBN gives rise to a DBN in the same way that a RBN gives a BN. At time t a node is created for every ground predicate and edges added between the predicate and its parents (if $t > 0$ then the parents are obtained from $BR_{\rightarrow}$, otherwise from BR_0). The conditional model at each node is given by the conditional model restricted to the particular grounding of the predicate.

Let us consider another very simple example.

Example 4 *Imagine you are monitoring the movements of a group of persons and you want to know who is friend with who. You are given observations about each person's location each day (for simplicity we assume a single observation each day, and fixed number of possible places: park, cinema, theater; we assume also that observations are acquired with a sensor placed at the entrance of each place). The assumption is that friends are more likely to go together to one place, rather than non-friends. At the same time, in this toy example, people will prefer to variate their activities, so if one is going to the park on a given day, he will be more likely to go to the cinema or the theater the next day. We can also accommodate individual specific preferences, as the fact that one agent prefers going to the cinema, while another prefers going to the park.*

In this example, we have objects $Person$ which are characterized by some attributes as $Location(t)$, $Preference$. The attribute $Location$ changes during time while $preference$ is fixed. Moreover, between objects it can exist the relation of $Friend$, that relates two objects that are friend (in Figure 2.7 we report the object $Person$ and the $2TRBN$ for our example). The probability distribution of the state of the domain at time t given the state of the domain at time $t-1$ is specified by the probability distribution $P(x_t|x_{t-1}, Friend)$. Where x_t represents

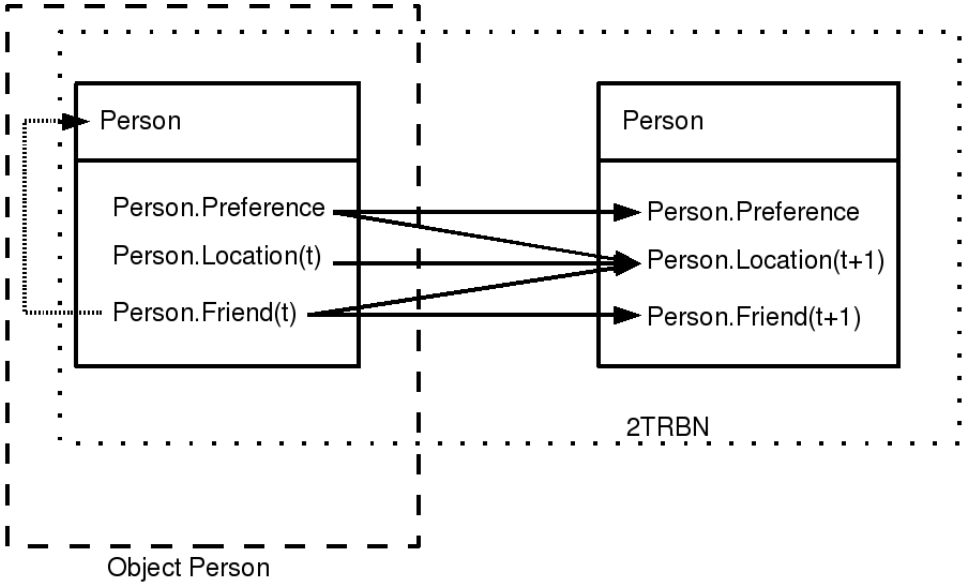


Figure 2.7: The $2TRBN$ for Example 4 is depicted. On the left the object $Person$ is reported.

the agents' $Location$ at time t, and $Friend$ is a 0-1 characteristic matrix representing all the friendship relations in the domain (a cell (i, j) has value 1 if agent i is friend of agent j).

Even in the case of noiseless observations (we observe who is going to the theater on a given evening), the friendship relation ($Friend$) is not directly observable. However, by using inference we can maintain a belief distribution (the probability distribution representing our guesses, and the respective certitude factor, about the unknown features of the system) about the friendship relation, represented by a table of probabilities, whose cells (i, j) indicate the probability that agent i is friend with agent j. These probabilities will be initially set to a prior (for instance, 0.5) and then updated after each observation. So if, for instance, agent a goes to the cinema alone and agent b and agent c both goes to the park, the probability (belief) of agent a being friend with either agent b or c will decrease a little bit, while the probability of agent b being friend with agent c will increase.

These decreases or increases of our beliefs are dictated by the observations and the transition model. More precisely, the current belief can be obtained using the Bayes' theorem, integrating over all possible values that the unknown features could take (in this case, all possible values of the tables representing the friendship relation). This exact approach however does not scale well: if we consider 5 agents, this already means integrating over 2^{10} possible combinations of values for the table representing the relation. If we complicate the model assuming uncertain observation (with some non-zero probability, the sensor might say that agent a is at the cinema when in fact he is at the park), the number of cases to consider would be even larger. It is then easy to see that, as the model becomes more complex (multiple relations in the model), as the observation model becomes more uncertain (fewer elements are observable), as the transition model becomes more complex, or as the number of agents increases, the exact approach would be infeasible, as it has exponential complexity. In the next chapter we will discuss how

to achieve tractable inference by considering probabilistic trees to compactly represent the transition model, and particle filtering for monte-carlo for inference.

2.5.2 Discussion

Introducing RDBNs to model the world offers two major advantages:

1. we are able to take directly into consideration relations between the agents in the domain;
2. we can model the behavior of an infinite class of objects in a compact way.

For example, in the scenario of the harbor introduced in the previous chapter (Section 1.2.2) and suppose the only information we have are relative to the position and the type of the boats.

A DBN-based framework would model each boat in the scene with a random variable. If a new boat enters the scene, it would be necessary to construct a new (different) DBN. Moreover, the inter-slice distribution that, gives the probability distribution on the state of the domain given the previous state, would model the behavior of each boat independently, without taking into account the existence of possible correlations between them.

A RBN-based framework would, instead, model each predicate of a class of objects with a random variable. For this reason, if a new boat enters the scene it would not be necessary to change the representation, because this framework is able to reason about classes of objects and not only about particular objects. Dependencies between variables at the same time-step will be given by the type of relation that can exist between boats. The inter-slice distribution will model how the state of the domain can change with respect to the relations that exist between the boats.

In this way a RBN-based framework would be able to model sequences of an arbitrary length of states (as DBN does) and a not known a priori number of objects. Moreover, taking directly into account relations between objects, it will be able to probabilistically model and tracking the object behavior recognizing that on line.

2.6 Summary

The major contribution of this chapter is the introduction of relational dynamic Bayesian networks (RDBNs). RDBNs are FOL-based probabilistic graphical models. They extend both RBNs to model dynamic domains (as DBNs extend BNs) and DBNs with the representative power of FOL (as RBNs extend BNs). The last section (Section 2.5.2) showed that RDBNs can be more compact than DBNs in representing a domain and more effective in dealing with objects' behavior.

In the next chapter (Chapter 3) we will introduce the problem of inference in relational dynamic domains, introducing an algorithm that takes advantage of the knowledge about relations between objects to infer objects'position and doing it online with relations recognition. In the remain of this work we will deal in particular with the tasks of activity recognition and multi objects tracking.

Chapter 3

Inference in Dynamic Relational Domains

Not being able to control events, I control myself; and I adapt myself to them, if they do not adapt themselves to me.

Michel de Montaigne

Reasonable people adapt themselves to the world. Unreasonable people attempt to adapt the world to themselves. All progress, therefore, depends on unreasonable people

George Bernard Shaw

In this chapter we present a novel algorithm that can tackle inference in dynamic relational domains. In particular, we consider the estimation of the relational state of a system that changes over time using a sequence of noisy measurements (or observations) of some variables of the system.

In the first part of this chapter we describe the general problem of inference and we show how it is tackled in non-relational domains. In the second part we introduce our relational particle filter algorithm that is able to track relations.

3.1 Systems that evolve over time

A dynamic system can be represented by a state-space model. A state-space model is represented by some underlying hidden state of the world (the state vector) that generates the observations and evolves with time. A state-space model, usually, consists of two equations, one that models the dynamic of the state vector and the other that models the observed state variables. The state vector contains all relevant information required to describe the system under investigation. For example, in tracking problems, this information could be related to the kinematic characteristics of the target. Alternatively, in an econometrics problem, it could be related to monetary flow, interest rates, inflation, etc.

A state s_t will be called *complete* if it is the best predictor of the future state of the system[1]. Completeness entails that knowledge of past states carry no additional information that would

[1]Recall that this is an assumption already taken introducing DBNs

help us predict the future more accurately. Temporal processes that meet these conditions are commonly known as satisfying the *Markov property*.

In an online setting, the goal is to infer the hidden state given the observations up to the current time, $z_{1:t}$, we can define our goal as computing the probability distribution over the state variable conditioned on all past measurements; this is called the *belief* of the state:

$$bel(s_t) = p(s_t|z_{1:t}) \tag{3.1}$$

The measurement vector represents (possibly noisy) observations that are related to the state vector. The measurement vector is generally (but not necessarily) of lower dimension than the state vector.

The evolution of the state is governed by probabilistic laws. In general, the state s_t is governed stochastically from the state s_{t-1} . Thus, it makes sense to specify the probability distribution from which s_t is generated. At first glance, the emergence of the state s_t might be conditioned on all past states; hence, the probabilistic law characterizing the evolution of the state might be given by a probability distribution of the following form: $p(s_t|s_{0:t-1}, z_{1:t-1})$. An important insight is the following: if the state s is complete then it is a sufficient summary of all that happened in previous time steps. In particular, s_{t-1} is a sufficient statistic of all previous measurements up to the point time t. In probabilistic terms, this insight is expressed by the following equality:

$$p(s_t|s_{0:t-1}, z_{1:t-1}) = p(s_t|s_{t-1}). \tag{3.2}$$

The conditional independence expressed in Equation 3.2 is the primary reason why the algorithms we will present in this chapter are computational tractable.

One has also to model the process by which the observations are being generated from the state. Again, if s_t is complete, we have an important conditional independence:

$$p(z_t|s_{0:t}, z_{1:t-1}) = p(z_t|s_t). \tag{3.3}$$

In other words, the state s_t is sufficient to predict the measurement z_t.

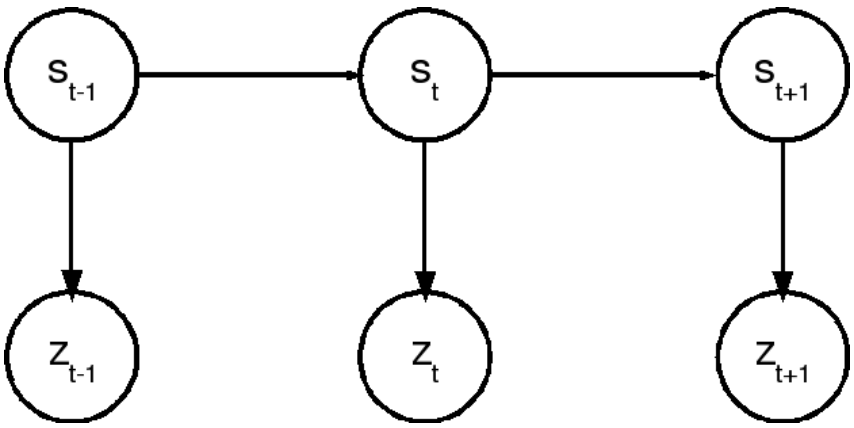


Figure 3.1: The DBN that characterizes the evolution of the states and measurements.

We can say that, in order to analyze and make inference about a dynamic system, any state-space model must define

- a *prior*, $p(s_0)$,
- a *state-transition probability*, $p(s_t|s_{t-1})$, to predict future observations given all the observations occurred to the present, and
- a *measurement model*, $p(z_t|s_t)$, to relate the noisy measurement to the state (sometimes also called *observation model*.

The state transition probability and the measurement model together describe the dynamic stochastic system of the domain. Figure 3.1 illustrates the evolution of the states and measurements defined through those probabilities. The state at time t is stochastically dependent on the state at time $t-1$. The measurement z_t depends stochastically on the state at time t. Such a temporal generative model can be represented by a DBN where the state transition model is the inter-slice distribution and the measurement model the intra-slice distribution. Since we are dealing with relational domains, we will say that the system will be represented with a RDBN.

3.1.1 Bayes Filter

The probabilistic state-space formulation and the requirement for the updating of information on receipt of new measurements are ideally suited for the Bayesian approach that provides a rigorous general framework for dynamic state estimation problems. In this approach to dynamic state estimation, one attempts to construct the posterior probability density function of the state based on all available information, including the set of received measurements.

In online analysis, an estimate is required every time that a measurement is received. The Bayes filter algorithm is the most general method for calculating the belief distribution from measurements data. The Bayes filter is recursive, that is, $bel(s_t)$ at time t is calculated from the belief $bel(x_{t-1})$ at time $t-1$. Received data can be processed sequentially rather than as a batch; the advantage is that it is not necessary to store the complete data set nor to completely reprocess previous observation if a new measurement becomes available.

In the Bayes filter algorithm the belief of the state is computed after the acquisition of the measurement z_t. In the *prediction step*, $\widetilde{bel}(x_t)$ predicts the state at time t based on the previous belief state, before incorporating the measurements at time t:

$$\widetilde{bel}(s_t) = p(s_t|z_{1:t-1}) = \int p(s_t|s_{t-1})bel(s_{t-1})ds_{t-1} \tag{3.4}$$

Computing $bel(x_t)$ from $\widetilde{bel}(x_t)$ is called *update*: at time t, a measurement z_t becomes available, and this may be used to update the prediction using the *Bayes' law* (see Appendix A):

$$
\begin{aligned}
bel(s_t) &= \frac{p(z_t|s_t, z_{1:t-1})p(s_t|z_{1:t-1})}{p(z_t|z_{1:t-1})} \\
&= \frac{p(z_t|s_t)\widetilde{bel}(s_t)}{p(z_t|z_{1:t-1})}
\end{aligned} \tag{3.5}
$$

where the likelihood function $p(z_t|z_{1:t-1})$ (that can be view as a normalization factor and will be often substituted by the letter α) is defined by the measurement model:

$$
\begin{aligned}
p(z_t|z_{1:t-1}) &= \int p(z_t|s_t, z_{1:t-1})p(s_t|z_{1:t-1})ds_t \\
&= \int p(z_t|s_t)\widetilde{bel}(s_t)ds_t
\end{aligned} \tag{3.6}
$$

The prediction stage uses the state-transition probability to predict the state belief forward from one measurement time to the next. Since the state is usually subject to unknown disturbances (modeled as random noise), prediction generally translates, deforms, and spreads the state distribution. In the update stage (Equation 3.5), the measurement z_t is used to modify the prediction to obtain the required belief of the current state. This is achieved using Bayes theorem, which is the mechanism for updating knowledge about the target state in the light of extra information from new data (a sketch of this process is given in Figure 3.2).

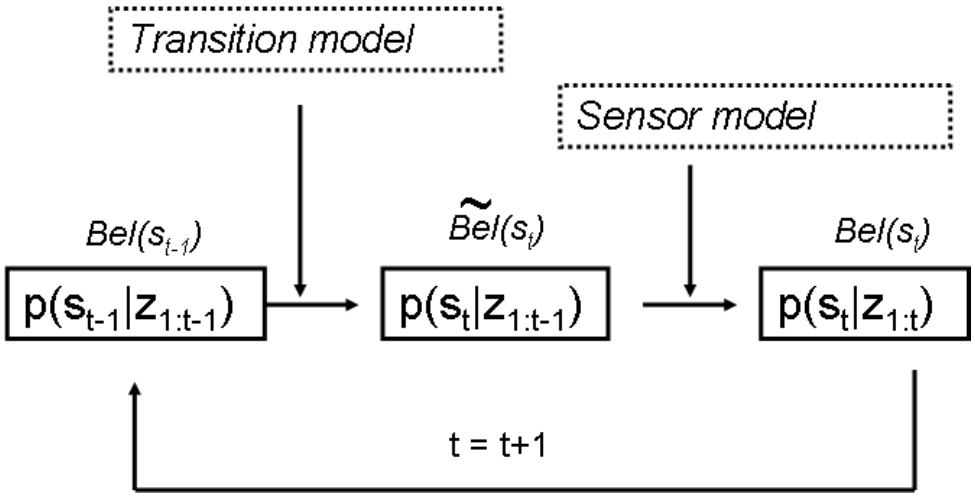


Figure 3.2: Graphical sketch of the Bayes filter iteration.

We assume that the transition and observation models are the same for all time[2]; the models are said to be *time-invariant* or *homogeneous* (without this assumption, we could not model infinitely long sequences of data).

[2]Also this assumption was already taken introducing DBNs.

3.1.2 Relational State

In the previous chapter (Chapter 2) we illustrated how a relational domain represents the objects by means of constants or variables and the relations between objects and their attributes by means of predicates. In FOL predicates represent both relations and attributes. At the purpose of this Thesis we have to differentiate the two. With reference to relational data bases, we will call *attributes* (or attribute's predicates when there is the risk of confusion) the predicates that refer to an object's attribute and *relations* (or relation's predicates) the predicates that refer to relations between objects. We will maintain the term predicate to refer to both the relations and the attributes. We will refer to ground relations or ground attributes when the respective predicates are grounded.

Therefore the relational state s can be divided in two parts: the *state of the attributes* s^a and the *state of the relations* s^r. We will write $s = [s^a, s^r]$.

The state of the attributes assigns to each object's attribute a value in its domain. Similarly, the state of relations associates to each ground relation a truth value. The grounding of the predicates is done over the objects present in the domain.

We define the relations to be *unobservable*: reasoning about relations can be done by inference over objects' attributes and their evolution over time, but it is not possible to measure a relation directly (for example, we can measure the position of a boat but we will have to infer from the positions of different boats if any of them is breaking any low). We will say relations are *intensional predicates* because their value can only be inferred and cannot be directly observed. Objects'attributes are *extensional predicates* because their value can be directly evaluated by a low-level pre-processing module (Minker & Seipel, 2002).

We think that this assumption is quite reasonable and, in most situations, natural. Anyway, if a measurable relation exists, it is always possible to define an equivalent model where only state attributes are measurable (*i.e.*, it is always possible to define an object's attribute that is equivalent to the original measurable relation).

When dealing with relations we should take them into account in both the measurements and the transition model, in the following we introduce these two models for relational dynamic domains.

3.1.3 Measurements model

The state is often observed by a noisy measurement system that can introduce uncertainty in the Domain. Given a certain measurement system (*e.g.*, a radar) a measurement model is defined that gives the probability of the state at time t given the measurements obtained at the same time: it is appropriate to think of measurements (z_t) as noisy projections of the state (s_t).

Since the part of the state relative to relations, s^r, is not directly measurable: we can define

$$p(z_t|s_t) = p(z_t|s_t^a, s_t^r) = p(z_t|s_t^a) \tag{3.7}$$

as the observation z_t is independent by the relations between objects. In other words, this measurement model only depends on the part of the state relative to the attributes.

3.1.4 Relational Transition model

The state-transition model $p(s_t|s_{t-1}) = p(s^a_t, s^r_t|s^a_{t-1}, s^r_{t-1})$ is a joint probability of the state of all objects and relations (Figure 3.3).

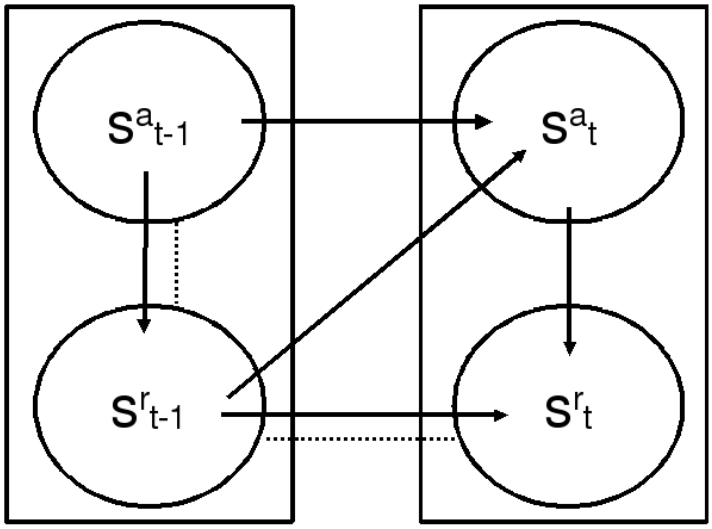


Figure 3.3: The relational transition model for the relational domain. The arrows mean probabilistic dependence: relations are stochastic functions of the attributes, relations at time t depends by their history (s^r_{t-1}) and the attributes at time t. The attributes at time t depends by the whole story of the state (relations and attributes). We assume the relational state to be complete.

In this work we will assume that the state of relations s^r does not evolve with respect to the previous attributes but only conditioned on its previous values and the actual objects instantiations. This assumption simplifies the transition model without loosing in generality: information about the previous state of the attributes are included in the respective state of relations.

In this particular circumstance we have to deal with a transition model that is a composition of two distributions: $p(s^a_t|s^a_{t-1}, s^r_{t-1})$ and $p(s^r_t|s^r_{t-1}, s^a_t)$

Through the Bayes' rule, the transition model $p(s^a_t, s^r_t|s^a_{t-1}, s^r_{t-1})$ can be written as

$$p(s^a_t, s^r_t|s^a_{t-1}, s^r_{t-1}) = p(s^a_t|s^a_{t-1}, s^r_{t-1})p(s^r_t|s^a_{t-1}, s^r_{t-1}, s^a_t), \tag{3.8}$$

given the independence of s^r_t from s^a_{t-1} given s^a_t we can write:

$$p(s^a_t, s^r_t|s^a_{t-1}, s^r_{t-1}) = p(s^a_t|s^a_{t-1}, s^r_{t-1})p(s^r_t|s^r_{t-1}, s^a_t), \tag{3.9}$$

that is the product of the two transition models introduced above.

Often the state of an object at time t depends by the state of the attributes of other objects at the previous time step; in this case it is necessary to establish an *order* ($\prec$) on the objects.

In our settings, the *order* on the objects is scenario-dependent: in a traffic monitoring we assume that the furthermost object from the camera is *preferred* according to $\prec$; in a harbor surveillance system the order, instead, follows from the "rules of the road" for boats. The introduction of this ordering while allowing the inference of those objects'state that depend by the relations with other objects, disallows cycles in the network (as explained in Chapter 2).

3.2 Particle Filtering

The recurrence relations given in Equation 3.4 and in Equation 3.5 form the basis for the recursive Bayesian filter. This recursive propagation of the posterior density is only a conceptual solution in fact, generally, it is not possible to determined the posterior analytically. Solutions do exist in a restrictive set of cases, including the Kalman filter model.

The Particle Filtering (PF) algorithm is a Monte Carlo method that forms the basis for most sequential Monte Carlo filters developed over the past decades. It is a technique for implementing a recursive Bayesian filter by Monte Carlo simulations. The key idea is to represent the required posterior density function by a set of random samples with associated weights and to compute estimates based on these samples and weights. As the number of samples becomes very large, this Monte Carlo characterization becomes an equivalent representation to the usual functional description of the posterior distribution, and the PF filter approaches the optimal Bayesian estimate (Arulampalam, Maskell, & Gordon, 2002).

In a PF algorithm, the samples of the posterior distribution are called particles and are denoted with

$$\chi_t := s_t^{[1]}, s_t^{[2]}, \cdots, s_t^{[M]}. \tag{3.10}$$

Each particle $s_t^{[m]}$ (with $1 \leq m \leq M$) is a concrete instantiation of the state at time t. Put differently, a particle is a hypothesis as to what the true world state may be at time t. Here M denotes the number of particles in the particle set χ_t. In practice, the number M is often a large number. In some implementation M is a function of t or other quantities related to the belief $bel(s_t)$.

The intuition behind PF is to approximate the belief $bel(s_t)$ by the set of particles in χ_t. Ideally, the likelihood for a state hypothesis s_t to be included in the particle set χ_t shall be proportional to its Bayes filter posterior $bel(s_t)$

$$s_t^{[m]} \propto p(s_t|z_{1:t}) = bel(s_t) \tag{3.11}$$

As a consequence, the denser a subregion of the state space is populated by samples, the more likely it is that the true state falls into this region.

3.2.1 Importance Sampling

Let $\{\chi_t, \{\omega_t^{[m]}\}_{m=1}^M\}$ characterize the belief of the state $bel(s_t)$, where the set of particles χ_t has associated weights $\{\omega_t^{[m]}, m = 1, ..., M\}$ and the weights are normalized such that $\sum_m \omega_t^{[m]} = 1$. Then, the posterior density at time t can be approximated as

$$bel(s_t) \approx \sum_{m=1}^{M} \omega_t^{[m]} \delta(s_t - s_t^{[m]}) \tag{3.12}$$

where $\delta()$ is the Dirac function. We therefore have a discrete weighted approximation to the true posterior, $bel(s_t) = p(s_t|z_{1:t})$.

The weights are chosen using the principle of *importance sampling*. This principle relies on the following. Suppose $p(s) \propto \pi(s)$ is a probability density from which it is difficult to draw samples but for which $\pi(s)$ can be evaluated. In addition, let $s^{[m]} \sim q(s)$, $m = 1, ..., M$ be samples that are easily generated from a proposal $q(\cdot)$ called *importance density*. Then, a weighted approximation to the density $\pi(\cdot)$ is given by

$$p(s) \approx \sum_{m=1}^{M} \omega^{[m]} \delta(s - s^{[m]}) \tag{3.13}$$

where

$$\omega^{[m]} \propto \frac{\pi(s^{[m]})}{q(s^{[m]})} \tag{3.14}$$

is the normalized weight of the ith particle.

Therefore, if the samples $s_t^{[m]}$ were drawn from an importance density $q(s_t|z_{1:t})$, then the weights in Equation 3.12 are defined by Equation 3.14 to be

$$\omega_t^{[m]} \propto \frac{p(s_t^{[m]}|z_{1:t})}{q(s_t^{[m]}|z_{1:t})} \tag{3.15}$$

3.2.2 Basic Algorithm

Returning to the sequential case, at each iteration, one could have samples constituting an approximation of $p(s_{t-1}|z_{1:t-1})$ and want to approximate $p(s_t|z_{1:t})$ with a new set of samples.

The PF algorithm constructs the belief $bel(s_t)$ recursively from the belief $bel(s_{t-1})$ one time step earlier. Since beliefs are represented by sets of particles, this means that PF constructs the particle set χ_t recursively from the set χ_{t-1}. A pseudo-code description of the most basic variant of this algorithm is given by the following Algorithm 1.

The input of this algorithm is the particle set χ_{t-1}, along with the most recent measurement z_t. The algorithm then first constructs a temporary particle set $\widetilde{\chi}_t$ that represents the belief $\widetilde{bel}_t$. It does this by systematically processing each particle $s_{t-1}^{[m]}$ in the input particle set χ_{t-1}. Subsequently, it transforms these particles into the set χ_t, which approximates the posterior distribution $bel(s_t)$.

Algorithm 1: Pseudo code for the PF basic algorithm

1. $\chi_t = PF(\chi_{t-1}, z_t)$
2. $\widetilde{\chi}_t = \chi_t = 0$;
for all $m = 1 : M$ **do**
   4. Sample $s_t^{[m]} \sim p(s_t|s_{t-1}^{[m]})$;
   5. $\omega^{[m]} = p(z_t|s_t^{[m]})$;
   6. $\widetilde{\chi}_t = \widetilde{\chi}_t + \langle s_t^{[m]}, \omega^{[m]} \rangle$;
for all $m = 1 : M$ **do**
   9. Draw i with probability $\propto \omega^{[i]}$;
   10. Add $s_t^{[i]}$ to χ_t;

The real "trick" of the PF algorithm occurs in line 9 and 10 in Algorithm 1, where the *resampling* step is implemented. In the resampling step the algorithm draws, with replacement, M particles from the temporary set $\widetilde{\chi}_t$, transforming a particle set of M particles into another particle set of the same size.

With the resampling process, the distribution of the particles changes: whereas before the resampling step, they were distributed according to $\widetilde{bel}(s_t)$, after the resampling they are distributed (approximately) according to the posterior $bel(s_t) \propto \widetilde{bel}(s_t)p(z_t|s_t)$. In fact, the resulting sample set usually contains many duplicates, since particles are drawn with replacement. The particle not contained in χ_t are the particles with lower importance weights.

Thus, the resampling step has the important function of forcing particles back to the posterior $bel(s_t)$. There are different way to implement the resampling step: the one implemented in Algorithm 1 is called *simple random sampling* and draws the particles with probability given by their importance weigh. In the next subsection we present a different procedure that is easier to implement than the simple random sampling and provides smaller variance.

3.2.3 Residual Sampling

The resampling step has been introduced to overcome the *degeneracy problem*. The degeneracy problem is the problem where, after a few iterations, all but a particle has negligible weights.

With the resampling step, particles that have small weights are eliminated and the algorithm concentrates on particles with large weights diminishing the Monte Carlo variation of the particles (Berzuini, Best, Gilks, & Larizza, 1997).

Residual Resampling is a resampling technique that can replace the simple random sampling providing favorable computation time and diminishing particles' Monte Carlo variation (Liu & Chen, 1998). It consists of the following steps:

1. Retain $k^{[m]} = \lfloor M\omega^{[m]} \rfloor$ copies of $s_t^{[m]}$ for each $1 \leq m \leq M$.
2. Let $M_r = M - \sum_{m=1}^{M} k^{[m]}$.
3. Draw M_r samples from $\widetilde{\chi}_t$ with probabilities proportional to $M\omega_t^{[m]} - k^{[m]}$ for each $1 \leq m \leq M$.

4. reset the weights to $1/M$.

The residual sampling does not seem to have disadvantages in any aspects.

A pseudo-code for the variant of the PF algorithm that makes use of the residual sampling is given by the following Algorithm 2.

Algorithm 2: Pseudo code for the PF algorithm with residual resampling

1. $\chi_t = PF(\chi_{t-1}, z_t)$
2. $\widetilde{\chi}_t = \chi_t = 0$;
for all $m = 1 : M$ **do**
   4. Sample $s_t^{[m]} \sim p(s_t|s_{t-1}^{[m]})$;
   5. $\omega^{[m]} = p(z_t|s_t^{[m]})$;
   6. $\widetilde{\chi}_t = \widetilde{\chi}_t + \langle s_t^{[m]}, \omega^{[m]} \rangle$;
for all $m = 1 : M$ **do**
   9. $k^{[m]} = \lfloor M\omega^{[m]} \rfloor$;
   10. Add $k^{[m]}$ copies of $s_t^{[m]}$ to χ_t;
   11. $\omega_r^{[m]} = M\omega^{[m]} - k^{[m]}$;
13. $M_r = M - \sum_{m=1}^{M} k^{[m]}$;
for all $m = 1 : M_r$ **do**
   15. Draw i with probability $\propto \omega_r^{[i]}$;
   16. Add $s_t^{[i]}$ to χ_t;

3.3 Relational Particle Filter

Given our subdivision of the relational state in s^a (state of the attributes) and s^r (state of the relations), we want to express the belief $bel(s_t)$ and the prediction $\widetilde{bel}(s_t)$ in terms of s^a and s^r. The belief of the relational state is:

$$bel(s_t) = p(s_t^a, s_t^r|z_{1:t}) \tag{3.16}$$

A Bayesian filter algorithm requires to compute the belief distribution from measurement data as:.

$$bel(s_t) = \alpha\, p(z_t|s_t^a, s_t^r) \int p(s_t^a, s_t^r|s_{t-1}^a, s_{t-1}^r) bel(s_{t-1}) ds_{t-1} \tag{3.17}$$

Following the assumption that the relational part of the state s^r is not measurable (see Section 2.5), the observations depend exclusively on the attributes: $p(z|s^r, s^a) = p(z|s^a)$. The previous equation becomes:

$$bel(s_t) = \alpha\, p(z_t|s_t^a) \int p(s_t^a, s_t^r|s_{t-1}^a, s_{t-1}^r) bel(s_{t-1}) ds_{t-1}. \tag{3.18}$$

The prediction $\widetilde{bel}(s_t)$ can be written as:

$$\widetilde{bel}(s_t) = p(s_t^a, s_t^r | z_{1:t-1}) = \int p(s_t^a, s_t^r | s_{t-1}^a, s_{t-1}^r) bel(s_{t-1}) ds_{t-1} \tag{3.19}$$

Considering the relational transition model introduced in Equation 3.9, we can write $\widetilde{bel}(s_t)$ as:

$$\widetilde{bel}(s_t) = p(s_t^a, s_t^r | z_{1:t-1}) = \int p(s_t^a | s_{t-1}^a, s_{t-1}^r) p(s_t^r | s_{t-1}^r, s_t^a) bel(s_{t-1}) ds_{t-1}, \tag{3.20}$$

that allows us to easily implement the filtering step.

The intuition behind our algorithm is the following. At each time step we have samples constituting an approximation of $p(s_t^a | s_{t-1}^a, s_{t-1}^r)$ and we want to approximate $p(s_t^a, s_t^r | s_{t-1}^a, s_{t-1}^r)$ with a new set of samples. Since the transition model is such that $p(s_t^a, s_t^r | s_{t-1}^a, s_{t-1}^r) = p(s_t^a | s_{t-1}^a, s_{t-1}^r) p(s_t^a | s_{t-1}^a, s_{t-1}^r)$ we can obtain samples $[s_t^{a,[m]}, s_t^{r,[m]}] \sim p(s_t^a, s_t^r | s_{t-1}^a, s_{t-1}^r)$ by augmenting each of the existing samples $s_t^{a,[m]} \sim p(s_t^a | s_{t-1}^a, s_{t-1}^r)$ with the new state of relations $s_t^{r,[m]} \sim p(s_t^a | s_{t-1}^a, s_{t-1}^r)$.

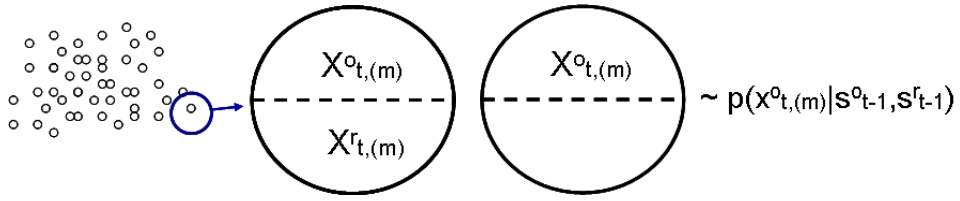
We introduce in Algorithm 3 our Relational Particle Filter (RPF) algorithm able to take advantage of the decomposition introduced in the relational transition model.

Algorithm 3: Pseudo code for the RPF algorithm

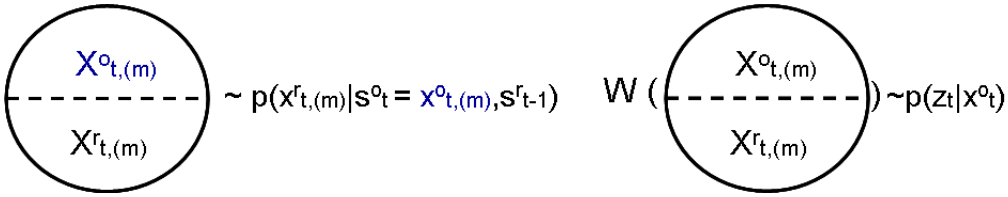
1. $\chi_t = RPF(\chi_{t-1}, z_t)$
2. $\widetilde{\chi}_t = \chi_t = 0$;
for all $m = 1 : M$ **do**
4. Sample $s_t^{a;[m]} \sim p(s_t^a | s_{t-1}^{a;[m]}, s_{t-1}^{r;[m]})$; Hypothesis for the state of the attributes
5. Sample $s_t^{r;[m]} \sim p(s_t^r | s_t^{a;[m]}, s_{t-1}^{r;[m]})$; Hypothesis for the state of the relations
6. $\omega^{[m]} = p(z_t | s_t^{a;[m]})$; Weights computation
7. $\widetilde{\chi}_t = \widetilde{\chi}_t + \langle [s_t^{a;[m]}, s_t^{r;[m]}], \omega^{[m]} \rangle$;
for all $m = 1 : M$ **do**
10. $k^{[m]} = \lfloor M\omega^{[m]} \rfloor$; Residual Resampling step:
11. Add $k^{[m]}$ copies of $[s_t^{a;[m]}, s_t^{r;[m]}]$ to χ_t;
12. $\omega_r^{[m]} = M\omega^{[m]} - k^{[m]}$;
14. $M_r = M - \sum_{m=1}^{M} k^{[m]}$;
for all $m = 1 : M_r$ **do**
16. Draw i with probability $\propto \omega_r^{[i]}$;
17. Add $[s_t^{[a;i]}, s_t^{[r;i]}]$ to χ_t;

A particle ($s_t^{[m]}$) is a representation of the state, for this reason, in our setting, it is divided in two parts: the part of the attributes $s_t^{a,[m]}$ and the part relative to relations $s_t^{r,[m]}$ (see Figure 3.4(a)). The part of the particle relative to the attributes is sampled according to the first part of the relational transition model (Line 4), subsequently the part of the particle relative to the relations is sampled according to the second part of the relational transition model (Line 5).

When the measurement is acquired, particles are weighted according to the sensor model (Line 6). The measurement model takes into account only the part of the particle relative to the attributes, since the particle is composed by two parts, also the part relative to the relations is weighted. After the weighting step, particle are resampled following the Residual Resampling procedure (Line 9). A sketch of the sampling and weighting steps is given in Figure 3.3.



(a) Particle representation. (b) First step of hypothesis.



(c) Second step of hypothesis. (d) Particle weighting.

Figure 3.4: Cartoon representation of the proposed algorithm.

3.3.1 Mathematical Derivation of the RPF

To derive the RPF mathematically we refer to (Thrun, Burgard, & Fox, 2005). We think of particles as samples of the state sequences:

$$[s_{0:t}^{a;[m]}, s_{0:t}^{r;[m]}] = [s_0^{a;[m]}, s_0^{r;[m]}], [s_1^{a;[m]}, s_1^{r;[m]}], \cdots, [s_t^{a;[m]}, s_t^{r;[m]}] \tag{3.21}$$

It is easy to modify the algorithm accordingly: simply append to the particle $[s_t^{a;[m]}, s_t^{r;[m]}]$ the sequence of state samples from which it was generated $[s_{0:t-1}^{a;[m]}, s_{0:t-1}^{r;[m]}]$. This relational particle filter calculates the posterior over all state sequences:

$$bel(s_{0:t}^a, s_{0:t}^r) = p(s_{0:t}^a, s_{0:t}^r | z_{1:t}) \tag{3.22}$$

instead of the belief $bel(s_t^a, s_t^r) = p(s_t^a, s_t^r | z_{1:t})$. This definition is needed to derive the RPF algorithm given in Algorithm 3.

The posterior $bel(s^a_{0:t}, s^r_{0:t})$ is obtained by:

$$\begin{aligned}
bel(s^a_{0:t}, s^r_{0:t}) &= p(s^a_{0:t}, s^r_{0:t}|z_{1:t}) \\
&= \alpha p(z_t|s^a_{0:t}, s^r_{0:t}, z_{1:t-1})p(s^a_{0:t}, s^r_{0:t}|z_{1:t-1}) \\
&= \alpha p(z_t|s^a_t, s^r_t)p(s^a_{0:t}, s^r_{0:t}|z_{1:1-t}) \\
&= \alpha p(z_t|s^a_t, s^r_t)p(s^a_t, s^r_t|s^a_{0:t-1}, s^r_{0:t-1}z_{1:1-t})p(s^a_{0:t-1}, s^r_{0:t-1}|z_{1:1-t}) \\
&= \alpha p(z_t|s^a_t, s^r_t)p(s^a_t, s^r_t|s^a_{t-1}, s^r_{t-1})p(s^a_{0:t-1}, s^r_{0:t-1}|z_{1:1-t}) \\
&= \alpha p(z_t|s^a_t, s^r_t)p(s^a_t|s^a_{t-1}, s^r_{t-1})p(s^r_t|s^r_{t-1}, s^a_t)p(s^a_{0:t-1}, s^r_{0:t-1}|z_{1:1-t})
\end{aligned}$$

The absence of the integral signs is the result of maintaining all states in the posterior.

The derivation is now carried out by induction. The initial condition is trivial to verify. Assume that our first particle set is obtained by sampling the prior $p(s^a_0, s^r_0)$. Let us assume that the particle set at time $t-1$ is distributed according to $bel(s^a_{0:t-1}, s^r_{0:t-1})$. For the m-th particle $[s^{a;[m]}_{0:t-1}, s^{r;[m]}_{0:t-1}]$ in the input set, the sample $s^{a;[m]}_t$ is generated from the proposal distribution:

$$p(s^a_t|s^a_{t-1}, s^r_{t-1})bel(s^a_{0:t-1}, s^r_{0:t-1}) \tag{3.23}$$

and the sample $[s^{a;[m]}_t, s^{r;[m]}_t]$ is generated according to the proposal distribution:

$$p(s^r_t|s^r_{t-1}, s^a_t)p(s^a_t|s^a_{t-1}, s^r_{t-1})bel(s^a_{0:t-1}, s^r_{0:t-1}) \tag{3.24}$$

with $s^a_t = s^{a;[m]}_t$.

To compute weights we use

$$\omega^{[m]}_t = \frac{\text{target distribution}}{\text{proposal distribution}} \tag{3.25}$$

where

$$\text{target distribution} = \eta p(z_t|s^a_t, s^r_t)p(s^a_t|s^a_{t-1}, s^r_{t-1})p(s^r_t|s^r_{t-1}, s^a_t)bel(s^a_{0:t-1}, s^r_{0:t-1}) \tag{3.26}$$

and

$$\text{proposal distribution} = p(s^r_t|s^r_{t-1}, s^a_t)p(s^a_t|s^a_{t-1}, s^r_{t-1})bel(s^a_{0:t-1}, s^r_{0:t-1}). \tag{3.27}$$

$$\omega^{[m]}_t = \frac{\eta p(z_t|s^a_t, s^r_t)p(s^a_t|s^a_{t-1}, s^r_{t-1})p(s^r_t|s^r_{t-1}, s^a_t)bel(s^a_{0:t-1}, s^r_{0:t-1})}{p(s^r_t|s^r_{t-1}, s^a_t)p(s^a_t|s^a_{t-1}, s^r_{t-1})bel(s^a_{0:t-1}, s^r_{0:t-1})} \tag{3.28}$$

from which follows:

$$\omega^{[m]}_t = \eta p(z_t|s^a_t, s^r_t) \tag{3.29}$$

The constant η plays no role since the resampling takes place with probability proportional to the importance weights. By resampling particles with probability proportional to $\omega^{[m]}_t$, the resulting particles are distributed according to the product of the proposal and the importance weights $\omega^{[m]}_t$:

$$\eta\omega^{[m]}_t p(s^r_t|s^r_{t-1}, s^a_t)p(s^a_t|s^a_{t-1}, s^r_{t-1})p(s^a_{0:t-1}, s^r_{0:t-1}|z_{0:t-1}) = bel(s^a_t, s^r_t) \tag{3.30}$$

Algorithm 3 follows from the simple observation that if $[s_{0:t}^{[m],a}, s_{0:t}^{[m],r}]$ is distributed accordingly to $bel(s_{0:t}^a, s_{0:t}^r)$ then the relational state sample $[s_t^{[m],a}, s_t^{[m],r}]$ is (trivially) distributed accordingly to $bel(s_t^a, s_t^r)$. The mathematical derivation of the filter, ensures the convergence of the filter for $M \uparrow \infty$.

3.4 Conclusion

In this chapter we presented the general problem of inference in dynamic domains, introducing the Bayes filter, that is the most general algorithm for this problem. To face the increased complexity posed by relational domains, we introduced one of the major contributions of the Thesis, our *Relational Particle Filter* algorithm. This algorithm computes the belief of the state taking into account relations between objects, and it is capable of track the relations as well.

In the last part of this chapter we derived the algorithm mathematically, proving that our reasoning is sound and the algorithm converges.

We will now focus our attention on two principal problems: the problem of activity recognition and the problem of multi target tracking. In the next chapters we first introduce these problems, reviewing related works and then we show how our RPF outperforms the state of the art algorithms.

Chapter 4

Anatomy of an Activity Recognition System

You'll never find your gold on a sandy beach,//You'll never drill for oil on a city street.//I know you're looking for a ruby in a mountain of rocks.// But there ain't no coupe de ville hiding //at the bottom of a cracker jack box.

Jim Steinman

The techniques developed in this work can be used in a variety of domains ranging from biosequence analysis, where different genes participating in the same interaction can be related, to economy prediction, where the price is related to the trend of the demand of related shares. We focus on behavior and scene understanding applications and in particular we will deal with systems for *vision-based activity recognition*. In this chapter we describe the challenges of designing a vision-based activity recognition system in all its main components and present the state of the art and our approach.

4.1 Vision-based Activity Recognition Systems

An *activity recognition system* aims to recognize the actions and the goals of one or more agents from a series of observations on the agents' positions, attributes and the environmental conditions. In particular, *vision-based activity recognition* consists in tracking and understanding the behavior of the agents through videos taken by a camera or a number of cameras. Vision-based activity recognition has found many applications such as human-computer interaction, user interface design, robot learning, and surveillance, among others.

A vision-based activity recognition system consists of multiple modules

1. The *motion detection* module has the goal to detect the objects in the scene that are moving significantly (*i.e.,* distinguish from background motion that is not of interest like leafs moving on a tree).

2. In the *tracking* module the objects detected in the previous step are associated with the "path" they are traveling. If only one object is moving in the scene this task if straightforward. If there are multiple objects in the scene the task becomes more complex (a *data association* step is required) and if the objects are interacting the complexity of the tracking task grows a lot.

3. The *activity recognition* module associates each (or groups) of the detected paths with a particular activity giving a meaning to the motion of the objects. Activity recognition can be exploited *online* with the tracking (*i.e.*, at each time step, a measurement is acquired, the state of the domain is filtered given the hypothesis done over the domain by the tracker and the belief over the activity computed) or off-line, when a sufficient amount of knowledge about the domain has been acquired.

In some cases, applications are also required to raise an alarm when a particular dangerous or forbidden situation arise. These systems, called *anomaly detection systems*, may use activity recognition in order to decide when rise an alarm and are discussed in (Section 4.5). In Figure 4.1 a sketch of the interactions of these module is reported.

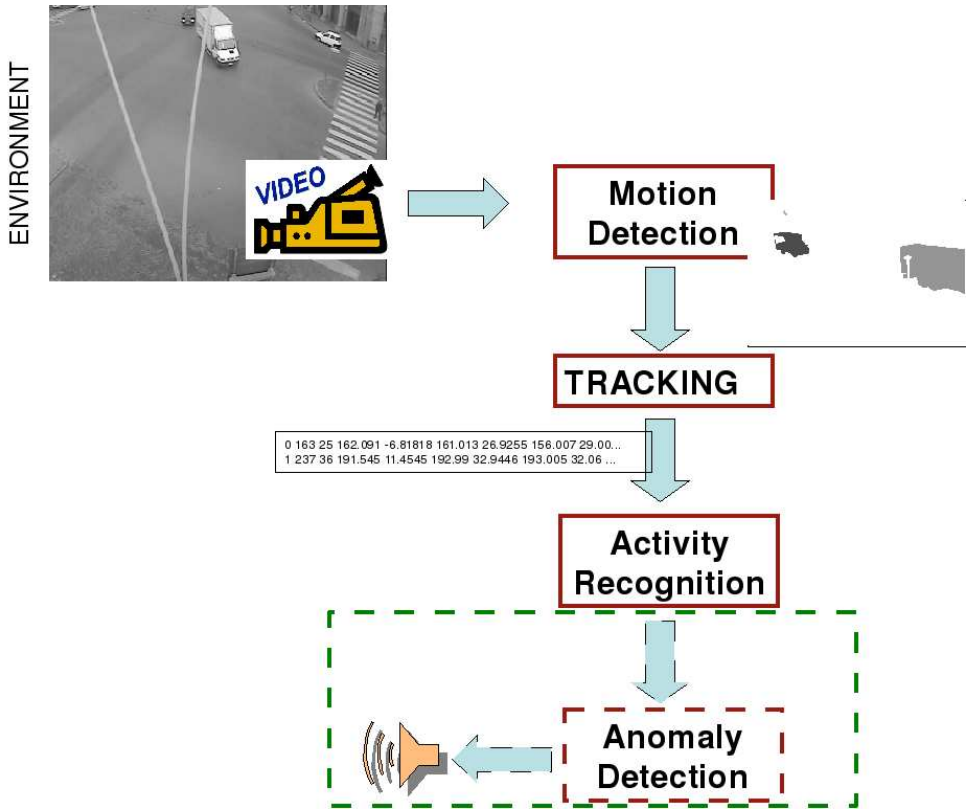


Figure 4.1: Graphical sketch of the activity recognition modules iteration.

In this Thesis we use probabilistic relational models to both improve tracking thanks to the prior about the ongoing activity and feed-back the knowledge acquired about the state to the activity recognition module to compute the belief over the activity online.

In the following sections we describe each layer of a vision-based activity recognition system, discuss the design challenges, review existing works and propose our solutions.

4.2 Motion Detection

Motion detection is the activity of extracting the pixels of the moving objects in the scene from the images of a video. Most of the approaches presented in this section rely on the output of a robust motion detection step and our method for tracking and activity recognition relies on the same hypothesis.

Motion detection methods are used to locate the presence (or absence) of motion in a given animated scene. We refer here to a specific class of motion detection methods for video surveillance (McKenna, Jabri, Duric, & Wechsler, 2000), traffic monitoring (B.Gloyer & T.Kailath, 1995) and others using a static camera.

4.2.1 Traditional Approaches to Motion Detection

Several methods have been proposed to automatically detect the objects'motion in an image sequence. They can be classified in two major categories: techniques that compares each new frame to a model of the scene background, called *background subtraction* techniques and techniques based on the difference of consecutive frames, called *inter frame difference* or *temporal difference techniques*.

Background subtraction bases the detection of moving objects on the difference between the current frame and a reference frame, often called *background* image. This implies that the background image has to be reliable, *i.e.*, it has to be an image of the scene without moving objects. This turns into the need of computing and updating a background model, which could account for changes in light conditions or small movements of the scene and has to face with a trade off: if the background model adapts too slowly to changes in the scene, then we will construct a very wide and inaccurate model that will have low detection sensitivity. On the other hand, if the model adapts too quickly, this will lead to two problems: the background model may include the target themselves, as their speed cannot be neglected with respect to the background variations, and it may lead to poor estimation of the motion.

The techniques used to model the background can be classified in two major groups:

- Non-parametric approaches and
- Parametric approaches.

As an example of *non-parametric* approaches to background modeling, consider the model presented in (Elgammal, Harwood, & Davis, 2000) where the density function of the distribution of each pixel in the scene is estimated at any moment of time given only very recent (n frames) history information. With the median filter approach, proposed in (R.Cutler & L.Davis, 1998) and in (R.Cucchiara & A.Prati, 2003), one computes each pixel of the background image as the average of the corresponding pixels in the n previous images. We also mention (Spagnolo, Leo, D'Orazio, Caroppo, & Martiriggiano, 2006) where the pixels' energy information is

exploited in order to distinguish static points from moving ones: a low energy content means that the considered point is a static one and the corresponding statistics are included in the background model, whereas high energy points, corresponding to foreground cannot contribute to the model. Non-parametric approaches are based on the assumption that a pixel is part of the background image for a certain amount of time (at least $\frac{n}{2}$ frames).

A first class of *parametric* algorithms uses *statistical approaches* to model background pixels. In (Stauffer & Grimson, 1999) a generalized mixture of Gaussians is used to model complex non-static background, however the presence of foreground objects during the learning phase could heavily alter the reliability of the model, as under sudden light changes (Cheung & Kamath, 2004). Moreover, these methods are computationally intensive and their parameters require careful tuning. A different class is composed by the approaches that use *filters* for temporal analysis. In (Luong, Weber, Koller, & Malik, 1995) the authors use a Kalman filter approach for modeling the state dynamics for a given pixel. In (Doretto, Chiuso, Wu, & Soatto, 2003), instead, an autoregressive model was proposed to capture the properties of dynamic scene. An improvement of this algorithm was implemented by (Monnet, Mittal, Paragios, & Ramesh, 2003) to address the modeling of dynamic background and perform foreground detection. The common assumption of these filter-based techniques is that the observed time series is independent at each pixel.

To reduce the background changes, the temporal difference approach detects motion by taking the absolute difference of consecutive images (this technique is also known as *single difference* (C. Zhang, 2003)). These approaches present the advantages of requiring much lower computational effort than the background subtraction methods and avoids errors typically due to the use of a particular background model and of using a very up-to-date image of the scene as background. The disadvantage of the temporal difference approach is that the image used as background includes the moving objects as well, therefore frame difference is liable to generate large areas of false foreground, known as *ghosts*. Moreover, it may miss the detection of that pixels that stop motion in the image frame (problem known as *foreground aperture*). To solve the ghost issue the "*double difference*" has been proposed in (Yoshinari & Michihito, 1996) as a variation on this method. This approach computes a threshold difference between frames at time t and $t-1$ and between frames at time $t-1$ and $t-2$, combining them with a logical AND. However, if the moving objects have not enough texture this procedure does not allow an accurate motion detection and the object position is not estimated in real time.

In the last years background subtraction and temporal difference has been integrated in new methods to fix the drawbacks each method present. One of these algorithm has been described in (Collins, Lipton, & Kanade, 1999), this algorithm exploits image difference between frames at time t and $t-1$ and the difference between t and $t-2$ to erase ghosting; it also keep in memory a background model to solve the foreground aperture problem. Another algorithm proposed as the integration of two different techniques is the one proposed in (Migliore, Matteucci, & Naccari, 2006) where an image of background is updated according to the result of the single difference on the current frame. Despite these approaches obtain good results, they spend a high computational effort to solve problems introduced by the integration of the two methods.

4.2.2 Context-aware Motion Detection

In this section we describe our approach to motion detection.

Context-based reasoning has been shown to be a key factor for computer vision in the development of algorithms for object recognition (Derek Hoiem & Hebert, 2006), (Elidan et al., 2006). These algorithms express the knowledge about the scene around the objects with a probability distribution and use it to accomplish the task of object recognition given their context. In (Heitz & Koller, 2008), for example, a system that leverages "context" toward improving detection is presented. The method clusters image regions based on their ability to serve as context for the detection of objects. This method automatically groups regions based on their appearance and computes the relationships between regions to detect objects in the image.

Another approach that models the context for the object recognition task is that presented in (Copsey & Webb, 2002). In this paper the uncertain causal relationships between the objects and their environment is modeled with a Bayesian Networks (BNs) to recognize multiple military targets. In particular they do that taking into account measurement information on the targets, measurements on the terrain around the target and the knowledge about certain characteristics of the target (preference in hiding, ability to traverse different types of terrain, ...)

As far as we know, algorithms for motion detection have not considered context interpretation. In (Archetti, Manfredotti, Messina, & Sorrenti, 2006b) we presented an approach that detects moving objects performing a temporal difference method differentiating between foreground and ghost areas with an heuristic. Here we describe this heuristic in terms of *context reasoning*.

Context to improve moving objects detection

The method we proposed is based on the difference between consecutive frames. Traditionally, such difference is computed as the absolute value of the difference in the intensity (as in Equation 4.1):

$$SDiff_t(x,y) = \begin{cases} |I_t(x,y) - I_{(t-1)}(x,y)| & \text{if } |I_t(x,y) - I_{(t-1)}(x,y)| > \overline{T} \\ 0 & \text{otherwise} \end{cases} \quad (4.1)$$

In $SDiff$ motion is detected in two areas, one due to the image position the object had at time $(t-1)$ (ghost), and the other due to the object position in the current frame (foreground). The two instances have similar image intensity in $SDiff$, and the possibility of distinguish between the two is lost.

In Figure 4.3, another relevant area, called *foreground aperture*, is emphasized. It is formed by the overlapping of the target positions in $I_{(t-1)}$ and I_t and its weightless depends by the motion of the target in the image with respect to the frame rate. The foreground aperture $(F.A.)$ area is characterized by pixels where the intensity is close to zero, due to the effect of the subtraction of pixels more or less at the same intensity level: in this area no motion can be detected.

Our method is based on the use of a Single Difference technique without computing its absolute value. We will call it *signed single difference*. We propose to use the signed single

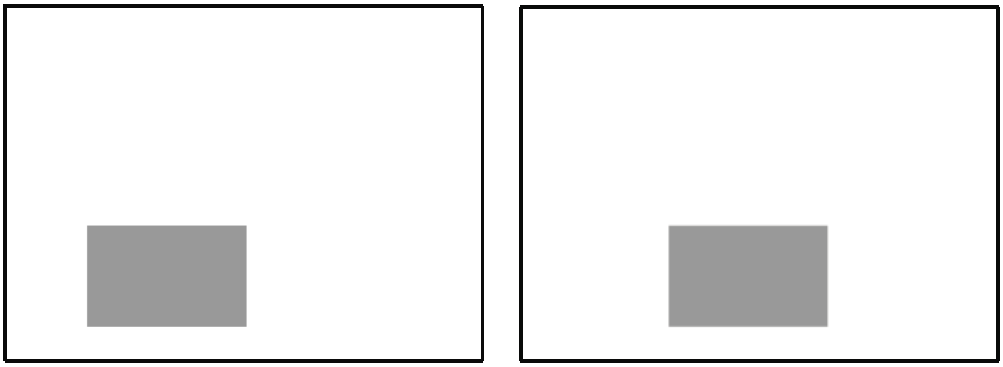


Figure 4.2: Left: Image at time $(t-1)$, I_{t-1}. Right: Image at time t, I_t.

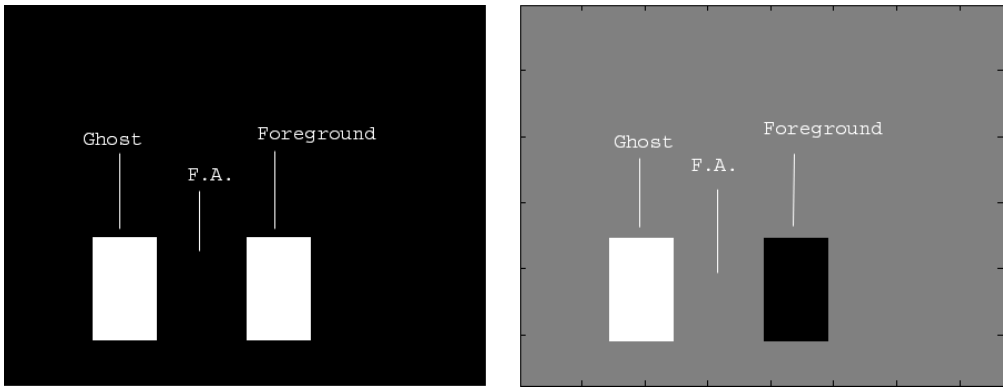


Figure 4.3: Left: SDiff: both foreground pixels and ghost pixels are set to 1 in the motion image shown. Right: SSDiff: ghost and foreground pixels have different intensity.

difference and to separate pixels of positive intensity from pixels of negative intensity: this separation will be used to discriminate the foreground from the ghost, as shown in Figure 4.3 right. Given two consecutive frames, $I_{(t-1)}$ and I_t (Figure 4.2, left and right) we consider the signed single difference, $SSDiff_t(x,y)$:

$$SSDiff_t(x,y) = \begin{cases} (I_t(x,y) - I_{(t-1)}(x,y)) & \text{if } |I_t(x,y) - I_{(t-1)}(x,y)| > \overline{T} \\ 0 & \text{otherwise} \end{cases} \quad (4.2)$$

We are left with some ghost and some foreground areas, which we want to discriminate. In Figure 4.4 left a simple example with one foreground and one ghost area is shown.

We now reason about the *context* of each detected blob (set of pixels assigned to a moving object). The context of a moving object is the background of the image. Our idea is based on the observation that around each blob area we have background, but *inside* an area detected by a blob:

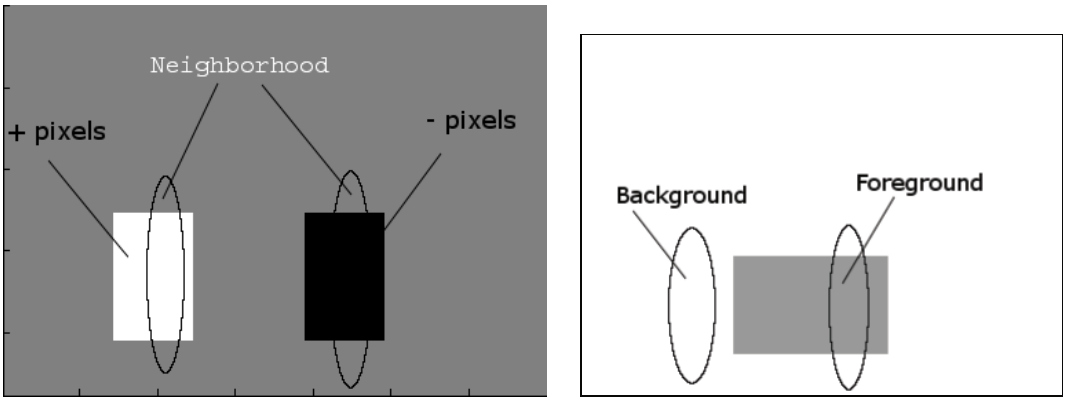


Figure 4.4: Left: The context (or neighborhood) for the heuristic are defined in the $SSDiff$ image. Right: The descriptor for the neighborhood are evaluated in the current image, I_t.

- if it is *foreground* there is something *different* from the context: this is an area detected as foreground that is indeed foreground;
- if it is *ghost* (*i.e.*, background) there is something *similar* to the context all around the blob: this is an area detected as foreground that is indeed background.

Similarity and difference can be defined in terms of color, light, etc.

Relational Reasoning to infer Foreground Aperture

Once we detect the foreground and the ghost areas, we are left with areas of background that can be faulty detected.

A *foreground aperture area* is an area in the image between a blob of ghost and a blob of foreground that has been detected as background but it is indeed foreground. We can use a First-Order Logic (FOL) relation to detect such areas.

We observed that, for each blob of foreground that is formed by positive pixels in the $SSDiff$, the *related* blob of ghost is formed by negative pixels.

With respect to the main theme of this Thesis (relational reasoning) we can consider this intuition from a "relational" perspective. We can define this relation (between a positive foreground and a negative ghost or vice-versa) based on the blobs'distance, their pattern in the image or their behavior in the previous n frames.

We can detect the foreground aperture areas as those areas between the couples foreground and ghost that are most probably related.

4.3 Multi-target Tracking

The problem of multi-target tracking is the problem of associating a (possibly unknown) number of moving objects with their most likely trajectories (sequences of positions) over time. If performed *online* it requires to make such associations at each time step.

Online tracking subsumes inference: it is the problem of infer the state of all the moving objects in the domain given the observations up to the current time.

A lot of work has been done in the past years to deal with the problem of finding the tracks of an unknown number of agents moving in the scene.

For non-interacting targets, the classical multi-target tracking literature approaches the problem by performing a data-association step after the acquisition of the measurement. The multiple hypothesis tracker (Reid, 1979) and the joint probabilistic data association filter (Fortmann, Bar-Shalom, & Scheffe, 1983) are the most influential algorithms in this class. These multi-target tracking algorithms have been used extensively in the context of computer vision. Some examples are the use of nearest neighbor tracking in (Deriche & Faugeras, 1990), the multiple hypothesis tracker in (Cox & Leonard, 1994), and the joint probabilistic data association filter in (Rasmussen & Hager, 2001).

Some of the works that deal with multi-target tracking integrate the low level motion detection step from a sequence of frames with the tracking. This is called *tracking-by-detection* and integrates the motion detection module with the tracking, taking advantage from the feedback connection between the two modules to improve both tasks. The underlying idea is to derive higher-level semantic information from the motion detection module and feed it back to the tracking module in order to improve performance there. Having an hypothesis over the future positions of the moving objects in the scene, will decrease the complexity of the motion detection. Between others an example of this approach is the work of Zhao *et al.*, (Zhao, Nevatia, & Wu, 2008).

In (Zhao et al., 2008) a model-based approach to interpret the image observations by multiple partially occluded human hypotheses in a Bayesian framework is presented. They define a joint image likelihood for multiple humans based on their appearance, the visibility of their body obtained by reasoning about occlusions and foreground/background separation. Their "data driven Markov chain Monte Carlo" sampling method performs inference using image observations as proposal probabilities.

In (Okuma, Taleghani, de Freitas, Little, & Lowe, 2004) the mixed particle filter algorithm is combined with Adaboost algorithm to learn, detect and track objects of interest. The mixture particle filter algorithm assigns a mixture component to each object, the Adaboost algorithm generates the proposal distribution of each particle.

These approaches are appropriate when targets behave independently, and the problem is a problem of "visual confusion".

The works presented in (Isard & MacCormick, 2001) and in (MacCormick & Blake, 2000) focus on the observation model, both these methods seem to "embed" the concept of relation in the dynamic model. In (Isard & MacCormick, 2001) the fact that two objects cannot occupy the same spot in the scene is used to constrain the object recognition module. In (MacCormick & Blake, 2000) reasoning about the fact that objects nearer to the camera are more likely to be recorded by it is used to detect of two objects which is occluding and which one is occluded.

Both of these approaches do not give an explicit description of the relations and they do not treat with them in a probabilistic way: relations are used as "rules of thumb" to constrain the reasoning over the system.

In (Khan, Balch, & Dellaert, 2004) a multiple hypothesis particle filter is described able to

track targets that are influenced by the proximity and/or the behavior of other targets. They do that defining a Markov random field at each time step to model the interactions between objects. They argue that taking advantage of the knowledge about the interaction between objects greatly help tracking two targets that pass close to one another. Their goal is only to track multiple objects, they do not take advantage of the knowledge about the interaction between objects to learn (or detect) what are the objects doing in the scene.

Mixed-states models, that we discuss next, are the models for tracking that most closely relate to our approach.

4.3.1 Mixed-state models

There are scenarios in which the tracking task is complex (and we provided some examples in the introduction) due to the unpredictability of the behavior of the targets: in these cases a unique motion model is not enough to predict the future state of the system, different motion models would be necessary to handle the targets possible behaviors. *Mixed-state* models allow automatic switching between multiple motion models as a natural extension of the tracking process.

The work presented in (Isard & Blake, 1998) allows a mixed-state object representation combining continuous-valued shape parameters with a discrete label encoding which of a discrete set of motion models is in force. While the inference task is performed, the discrete label says which model must be used to predict the future (continuous) state. In the prediction step, model switching is performed when necessary.

Consider, for example, a mixed state model for tracking two billiard balls. When the balls are far apart (first model) the system tracks the balls independently (*i.e.*, predicts their position using simple cinematic model), instead when they are close (second model) the system considers a transition model that predicts collisions or bounces.

In a mixed-state models the label of the transition model relates the prediction of the future state only to the time step previous to the current one, this assumption makes the prediction over the motion model unrelated with the current state, this may lead to a possible delay for the tracker to correctly approximate the objects' behavior. In the example, at time t the distance between two billiard balls is computed taking into account the positions of the balls at time $t-1$ and the switching between the two models is done for the prediction of the state at time $t+1$; this leads to a delayed in the approximation of the state of the two balls. In our method, instead, (see next paragraph) we use relations as the representation of the motion model in force. The state of relations does depend by the previous relational state and by the current state of attributes, the prediction step takes into account which relations are true to make its hypothesis over the next state, that results in being more up-to-dated with respect to a mixed-state model approach. Moreover, the use of FOL relations (as opposed to a list of possible motion models) generalizes our models to different domains.

4.3.2 *Relational* Multi-target Tracking

In this Thesis we deal with *relational tracking* a framework first presented by Guibas in (Guibas, 2002) that allows to track multiple agents taking into account the correlation between their behavior. In relational tracking one considers, in addition to the positions and the object's attributes, relations that represent joint properties of the objects and associates a set of objects with a full specification of the evolution of the value of their attributes and relations over time.

While tracking the state of attributes (*i.e.,* "usual" state of the domain, refer to Section 3.1.2) we track also the relations between objects at the same time. The goal of relational tracking is that of finding the trajectories over time of an (unknown) number of objects as well as the evolution of the relations between them. We can say that relational tracking subsumes activity recognition.

A complete description of the relations existing between objects gives us a description of the activity they are involved in. Tracking relations means being able to say at each time step the activity (described as set of true relations) in which each object in the scene is involved. In this Thesis we do not deal with the recognition of a single action performed by a single person but we focus on recognizing activities that reflect semantic goals.

4.4 Activity Recognition

Activity recognition is the problem of model and detect specific, dynamic interaction between moving objects (sometimes referred as agents) or part of objects. In the past years, the concept of *activity* ranged from single-agent, short-duration action to the semantic goal (or intention) of a group of targets. Recognizing single-agent, short-duration activities is also called event detection (Laptev, Caputo, Schüldt, & Lindeberg, 2007) and has the goal of identify and localize spatio-temporal patterns in videos. *Complex* activities have been often defined as temporally extended activities that can be fragmented in simple ones.

4.4.1 Traditional Approaches in Activity Recognition

In (Hongeng, Nevatia, & Brémond, 2004) simple activities (*single threads*) are characterized by being executed by a single actor and multi-agent events (complex activities) are represented by a number of single threads related by temporal constraints. Single threads are recognized from the characteristic of the trajectories of the actors using Bayesian methods. Complex activities are recognized by propagating temporal constraints of single threads in a temporal logic network.

We will focus on actions that see the *interaction* between objects, instead of focusing on complex actions that are temporal sequence of simple one.

In (Ivanov & Bobick, 2000), (Moore & Essa, 2002) and (Ryoo & Aggarwal, 2006) the interactions between different agents is recognized. These methods recognize single-actor, simple activities and organize the detected simple activities with a stochastic grammar free parsing to recognize complex activities. The recognition task is decoupled in two levels: a lower level that detects single simple activities that are the inputs for the stochastic context-free grammar used as a "bag of words" to interpret the structure of the system.

The complexity of this interactions are still temporal based. Ivanov and Bobick report the example of a domain where the problem is to recognize the activities that can occur in a parking lot of a building. The interaction between a person exiting the building and a car entering the parking lot is recognized as "pick-up" if the person approaches the car and then "disappears" from the scene. As in (Intille & Bobick, 1999), Ivanov and Bobick detect complex actions that contain many components that occur in an ordered temporal relation to one another, that generally reflect causal connection. We are, instead, interested in modeling and recognizing activities that see the *integration* and *coordination* of multiple objects.

A good example of such activities is the one of recognizing actions during a sport match, in particular some work has been done in soccer domain.

An example of these works is the one reported in (Tovinkere & Qian, 2001) where, based on tracking data, players'actions and game events are detected using a set of heuristic rules. These (propositional) rules are derived from a hierarchical entity-relationship model representing the prior knowledge of soccer events.

A first-order probabilistic model that combines multiple clues to classify human activities from video data is introduced in (Biswas, Thrun, & Fujimura, 2007). The probabilistic model is implemented as a Dynamic Markov Logic Network that groups fifteen FOL propositions. The system is applied to an office setting where only activities that involve an agent and an inanimate object are considered (talking to the phone, writing with a pen, ...). Our goal is to model FOL relations between different moving objects.

As far as we know, the nearest approach to the one proposed in this Thesis is the one introduced in (Tran & Davis, 2008) where common sense domain knowledge is represented as FOL rules and Markov logic networks are defined based on these rules.

Differently from our method, the inference task is performed off-line: they perform probabilistic inference for input queries about events of interest that already happened. We seek, instead, to perform an online probabilistic inference of both the state of the domain and the activities and leave each module sending feedbacks to the other to improve its performance. We claim that, once we recognize complex coordinated activities it will be easier to express the motion model. Our approach seek to take advantage from the tracking for the activity recognition task and from the belief over complex activity to improve the tracking.

4.4.2 Online Activity Recognition for Relational Tracking

The aim of our approach is that of tracking multiple interacting agents and recognize their coordinated activities online. For this purpose we use our RPF for tracking the objects in the domain together with their relations. The coordinated activities are the results of the set of all true relations in the domain.

For the sake of the explanation of our computational framework we divide each particle in two parts: the part of the attributes and the part of the relations; as reported in Figure 4.5.

Referring to Algorithm 3, in the first step of the prediction, the part of the particle relative to the relations plays the role of the discrete label in the mixed-states models: they encode the values of the parameters of the relational model based on which relations are true (Figure 4.6 left). In the second step of the prediction, the values of the relations are predicted according to

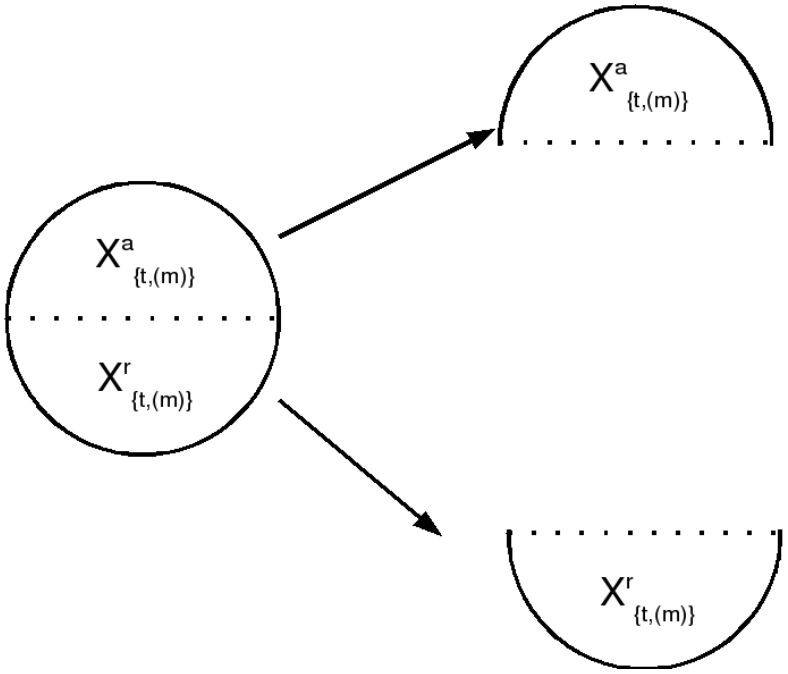


Figure 4.5: Our particles can be considered the combination of two parts, the parts of the attributes and the parts of the relations, these will cooperate in the prediction step.

their previous values and the hypothesis over the state of the attributes. Computing the prediction of the relations between the objects means to predict the activities in which the objects are involve in: this step can be called *activity prediction* (Figure 4.6 right).

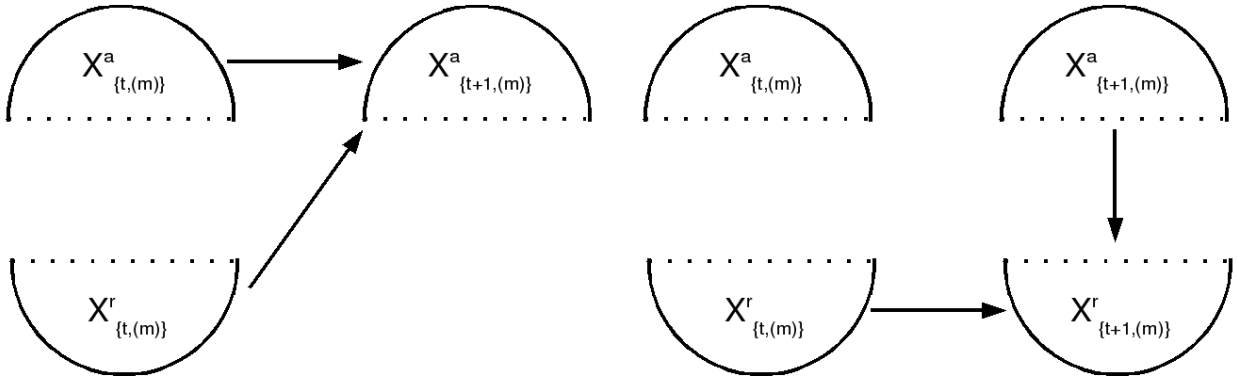


Figure 4.6: In the first step of the prediction the part of the particle relative to relations plays the role of the discrete label in the mixed-states models: encodes, of each object, which discrete model is in force. In the second step of the prediction the values of the relations are predicted according to their previous values and the hypothesis done over the state of the attributes.

In the resampling step the state is filtered: the particles are forced back to the posterior

$bel(s_t)$. With this step we filter the activities: only the most probable activities will survive the resampling step.

Using our Relational Particle Filter (RPF) for multi target tracking purposes allows us to recognize activities online. To be notice that all the activities we speak about (the single-person, short-duration, the temporal based and the coordination) can be expressed as FOL relations.

4.5 Anomaly Detection

Most of the models used for activity recognition can be used also with the purpose of anomaly detection. *Anomaly detection* is the problem of detecting the occurrence of suspicious events. Activity recognition methods can detect anomalies when these suspicious events are known or described by a model representing interactions among multiple objects moving in a scene.

In the next chapter we will often use the terms anomaly detection and activity recognition interchangeably because one of the applications we will deal with is the recognition of the activity of rendezvous between boats approaching the harbor, that is, indeed, a suspicious (or anomalous) behavior. For this reason, here we want to explain the differences between anomaly detection and activity recognition approaches.

Anomaly detection techniques can be divided in two different kind of approaches: *signature* and *statistical* anomaly detection. Signature anomaly detection has an internal "list" of *anomalous* patterns; if an ongoing activity matches a pattern in the "list", an alarm is raised. Activity recognition models can be used as signature anomaly detector. In (Archetti, Manfredotti, Matteuci, Messina, & Sorrenti, 2006a), we explained that signature anomaly detection present a principal disadvantage: since the set of anomalous patterns is based on known anomalies, new one cannot be discovered. Instead, the objective of Statistical Anomaly Detection is to establish profiles of *normal* activities: sequences of events that deviate from these profiles are considered anomalous and consequently an alarm is raised. The key point is that the statistical model is an accurate predictor of normal behavior, so if an ongoing pattern is not accurately predicted by the model, it is likely to be anomalous.

Our purpose is not to develop an anomaly detector system but an activity recognition framework. It is possible to use our RDBN-based framework to model "normal" behaviors and to use it as a statistical anomaly detector.

In the next chapter,relations that, in some sense, can also be considered "anomalies". We do that to show how using FOL relations between objects can improve the performance of both the tracking and the activity recognition. Being, our model, able to track more accurately multiple targets and the relations between them, we have reason to believe that it would be a good model to implement a statistical anomaly detector as well.

A RDBN anomaly detector would be able to take the best of the two worlds: as a statistical anomaly detector it can represent "normal" behavior probabilistically; moreover it supports the behavior modeling of known anomalous activities (that can be represented as "relations") as a signature anomaly detector. An alarm is raised to capture the attention of a human operator if either the belief associated to a known forbidden act is high or if all allowable, "normal" acts are associated with low belief (unknown anomaly).

4.6 Conclusions

In this chapter we presented the challenges associated with activity recognition from a practical perspective. An application for activity recognition is a complex system composed of many layers: motion detection, tracking and activity recognition itself. For each of these, we presented our approach compared to the state of the art.

The experimental part of this Thesis (Chapter 5) will be based on these findings.

Chapter 5

Experiments

So come up to the lab and see what's on the slab.

Frank in the movie "The Rocky Horror Picture Show"

In this chapter we present the experiments we performed to demonstrate the effectiveness of our method.

We evaluate our approach on three synthetic data sets on multiple objects moving in different domains. These agents can perform a variety of actions, some of which may require interaction between them. These experimental domains are of particular interest because the targets' behaviors are dependent on some external conditions that can be modeled with relations.

In the following we give first an outline of the experiments, we introduce the performance indicators and discuss our experimental goals. Then we present in detail each experiment.

5.1 Introduction

In the previous chapter we presented a method that uses relations to improve complex activity recognition, resulting from agents' interactions. Activity detection is performed online while tracking the state of the domain. The proposed method is based on the use of probabilistic relational models extended to model dynamic domains. The introduction of RDBNs to model dynamic domains allows a compact representation of the world, the use of FOL predicates allows the representation of the interactions of the agents.

This chapter has the goal of presenting the experimental results we obtained applying our approach to different domains. We show how modeling and tracking relations between objects as well as their state, improves the prediction ability of the tracker and the activity recognition task.

5.2 Overview of the experiments

We present the results obtained over three data sets and present them in an increasing difficulty order.

- The first data set deals with three cars moving on a one-way lane. The relation between the targets that we consider is $BeingInFront$. This relation influences the behavior of the objects in the following way: when the car x, that is in front the car y, slows down, the car y has to slow down as well. In this case, our relational tracker, taking into account the relation $BeingInFront$, predicts their positions more accurately than standard methods, this leads to an improvement of the performance of the tracking.

- In the second data set we consider a crossroad where we want to identify the cars that are traveling together (cars that are traveling together are more likely to have some common characteristics and follow a common travel pattern). The $TravelingTogether$ relation is therefore unknown to the system but can be inferred by looking at the movements of the cars.

- In the last data set we consider the automatic surveillance system of a Canadian harbor. A set of relations ($Rendezvous$, $Avoidance$, $PickUp$) model the different activities that two boats can undertake. All these activities are composed of phases or sub-activities (as moving towards the other boat, slowing down, loading goods, etc.) and timing is an important dimension. As a result, these activities are more complex than the one seen before; they require a more complex transition probability model that also depends on the type of the boats.

In all three experiments we show how the inference about the relations allows us to make better estimate. In the next section we present the performance metrics that we use.

5.3 Performance metrics

In the experiments below we show a number of statistics that evaluate the performance of our relational tracker compared to alternative standard techniques. As we have two complementary goals (improve positional tracking and identify relations) we have two families of metrics as well.

5.3.1 Positional tracking error

In the first family of metrics we consider the ability of the system of identifying the right positions of the target in a period of time. A trajectory (or track) is a sequence of positions over time. A trajectory l is generally defined as a sequence of $2D$ positions (x_t, y_t) and corresponding times, t:

$$l = \{(x_{t_1}, y_{t_1}, t_1), (x_{t_2}, y_{t_2}, t_2), \cdots, (x_{t_n}, y_{t_n}, t_n)\}. \tag{5.1}$$

In the computer vision domain, time steps can often assumed to be equal, and measured in frames. Thus, t_n may be dropped, as the subscript on the positions can be taken as time, and Equation 5.1 becomes:

$$l = \{(x_1, y_1), (x_2, y_2), \cdots, (x_T, y_T)\} \tag{5.2}$$

i.e., trajectory l is a sequence of (x_t, y_t) positions.

Our relational tracking technique makes use of relations to reason about the positions of the targets. At each time step, $bel(s_t)$ is a probability distribution (particle distribution) of the state. When considering the track identification problem, as we need to output a single position, the best guess is obtained using statistical expectation of the state projected on the part relative to the positions. We define the estimated position of a tracked target at a given time step t according to a distribution $bel(s_t)$ as:

$$(\hat{x}_t, \hat{y}_t) = (E_{bel}[x_t|s_t], E_{bel}[y_t|s_t]), \tag{5.3}$$

where s_t is the state of the world and (x_t, y_t) is the part of the state relative to the position of the targets. $(E_{bel}[x_t|s_t], E_{bel}[y_t|s_t])$ are the expected values of the state restricted to its positional values:

$$(E_{bel}[x_t|s_t], E_{bel}[y_t|s_t]) = (\frac{1}{M}\sum_{m=1}^{M} x_t^{[m]}, \frac{1}{M}\sum_{m=1}^{M} y_t^{[m]}), \tag{5.4}$$

The track obtained with a RPF is the following [1]:

$$l^{RPF} = \{(\hat{x}_1, \hat{y}_1), (\hat{x}_2, \hat{y}_2), \cdots, (\hat{x}_T, \hat{y}_T)\}. \tag{5.5}$$

We stress that this operation of averaging is only for the purpose of selecting a single candidate position (useful if we want to answer the query "where is the target now?") but the algorithm continues to propagate the belief distribution to the next step.

Comparison of Trajectories

Consider two trajectories l^A and l^B. Let positions on trajectory l^A be $l_t^A = (x_t, y_t)$ and positions on trajectory l^B be $l_t^B = (p_t, q_t)$, for each time step t. The distance between the positions at time step t is given by the Euclidian distance:

$$d(l_t^A, l_t^B) = \sqrt{(p_t - x_t)^2 + (q_t - y_t)^2}. \tag{5.6}$$

Let us define $d(l^A, l^B)$ to be the set of distances $d(l_t^A, l_t^B)$ between the trajectory l^A and l^B. A metric commonly used for tracker evaluation is the *mean* of these distances (Needham & Boyle, 2001), (Harville, 2002):

$$\mu(d(l^A, l^B)) = \frac{1}{T}\sum_{t=1}^{T} d(l_t^A, l_t^B) \tag{5.7}$$

where μ gives the average distance of two trajectories over a certain period of time.

[1] The definition is the same for the non-relational tracker; however the belief distribution will be different.

Discrete trajectories The definitions reported so far referred to a domain in which objects move in a 2D space. Of course, we can deal with discrete domains as well (this is the case of the second data set we consider, the *crossroad* domain). In a discrete domain objects move on a graph, where each node is associated with a label id, belonging to a set ID. Then, trajectories are defined as sequences of the (discrete) labels $l = \{x_1, x_2, \cdots, x_T\}$, where each $x_i \in ID$.

Given a particle distribution of a discrete position, instead of the expected value of the position, we compute the *mode*, projecting the state of the attributes on the position:

$$\hat{x}_t = mode_{bel}[x_t|s_t]. \tag{5.8}$$

The track obtained with a RPF is given by the sequence of $\hat{x}_t$:

$$l^{RPF} = \{\hat{x}_1, \hat{x}_2, \cdots, \hat{x}_T\}. \tag{5.9}$$

To compare two tracks, we have to calculate the distance between their positions at time t. Given two discrete trajectories l^A and l^B, let positions on l^A be $l^A_t = x_t$ and positions on l^B be $l^B_t = q_t$ for each time step. Since IDs are purely numeric names and are not ordered (*i.e.*, ID_4 can be nearer to ID_7 than to ID_5 in a particular ordainment) we need to consider the adjacency graph of the IDs and compute the distance over this graph. To compute the distance we use the *shortest path algorithm* (or *Dijkstra algorithm* (Sedgewick, 2001)) that returns the length of the shortest path from a node to another on a given graph.

As before, we define $d(l^A, l^B)$ to be the set of distances $d(l^A_t, l^B_t)$ between the discrete trajectory l^A and the discrete trajectory l^B. The mean distance of the two tracks over a certain period of time is computed as the entire part of the average value of the distances in that period of time:

$$\mu(d(l^A, l^B)) = \lfloor \frac{1}{T} \sum_{t=1}^{T} d(l^A_t, l^B_t) \rfloor \tag{5.10}$$

Now that we have defined the terminology and introduced the concept of track or trajectory, we can formulate the evaluation metric that we use in the rest of the chapter. The following metrics apply equivalently to discrete or continuous tracks.

Tracker evaluation

For the evaluation of the performance of a tracker implemented by a given algorithm A, with respect to its ability of identifying the right track, we compare the track (the ground truth) we want to estimate $l = \{(x_1, y_1), (x_2, y_2), \cdots, (x_T, y_T)\}$ with the track generated by the tracker $l^A = \{(\hat{x}_1, \hat{y}_1), (\hat{x}_2, \hat{y}_2), \cdots, (\hat{x}_T, \hat{y}_T)\}$. In the following we show three measures of errors.

Filtered error The *filtered tracking error* or simply the *tracking error* measures the ability of a tracker to follow a target. It is computed as the mean of the distances between the sequence of the filtered positions (the output trajectory of the tracker) and the track we want to estimate. The tracking error of an algorithm A is:

$$E^F(A) = \mu(d(l^A, l)). \tag{5.11}$$

Prediction error We also want to consider how the more informative belief distribution obtained reasoning with relations improves the predictions (at the prediction step). In this case, we consider the tuple $(\widetilde{x}_t, \widetilde{y}_t)$ made of the prediction's estimation, instead of the filtered distribution:

$$\begin{aligned}(\widetilde{x}_t, \widetilde{y}_t) &= (E_{\widetilde{bel}}[x_t|s_t], E_{\widetilde{bel}}[y_t|s_t]), \text{ for the continous case and} \\ \widetilde{x}_t &= mode_{\widetilde{bel}}[x_t|s_t], \text{ for the discrete case.}\end{aligned} \tag{5.12}$$

The *prediction error* of a given algorithm A is again defined as the mean of the distances between the sequence of these values ($\widetilde{l}^A = \{\widetilde{x}_t, \widetilde{y}_t\}$ or $\widetilde{l}^A = \{\widetilde{x}_t\}$) and the real trajectory, l:

$$E^P(A) = \mu(d(\widetilde{l}^A, l)). \tag{5.13}$$

It is possible to note that, on average, $E^F(A) < E^P(A)$, as the filtered distribution incorporates more evidence.

Tracking error with the most-likely relational assumption The relational state is the vector combining the *state of the attributes* and the *state of the relations*. At each time step, $bel(s_t)$ gives the joint probability of the state of the attributes and the state of the relations and our set of particles is the result of the pairing of different relation and position values.

The statistical estimation (used in the previously introduced error metrics) takes the belief distribution and outputs the "average case", this estimation can work well in many circumstances. However, in some cases a simple average might give an unsatisfactory estimate, or even an unrealistic result. For example, in a road domain, the most likely value of the relation might be that a car a is traveling together with a car b; however their estimated position might actually be extremely different. To overcome this problem, we consider a metric that estimates relational values first and then estimates the attributes only considering the particles consistent with the relation.

This method defines the best guess for the position of a target at a certain time step conditioned on the most probable value of the relations.

The trajectory obtained by this restriction is the sequence of the estimation of the probability distribution represented by the subset of particles obtained in the following way: at each time step we first compute the most probable value (the mathematical *mode*) of the part of the particle relative to the relations (we call it q^r); we then extract from the set of all the particles $q^{[m]} = [q^{a,[m]}, q^{r,[m]}]$ only that particles that have their relational part ($q^{r,[m]}$) equal to q^r. From this subset of particles we compute $(\hat{x}_{t,*r}, \hat{y}_{t,*r})$ or $\hat{x}_{t,*r}$ as before:

$$\begin{aligned}(\hat{x}_{t,*r}, \hat{y}_{t,*r}) &= (E_{bel}[x_t|s_t, q^r], E_{bel}[y_t|s_t, q^r]) \text{ and} \\ \hat{x}_{t,*r} &= mode_{bel}[x_t|s_t, q^r]\,.\end{aligned} \tag{5.14}$$

In this way we obtain the sequence of filtered positions:

$$\begin{aligned}l^{RPF}_{*r} &= \{(\hat{x}_{t,*r}, \hat{y}_{t,*r})\} \text{ or equivalently} \\ l^{RPF}_{*r} &= \{\hat{x}_{t,*r}\}\end{aligned} \tag{5.15}$$

and we can compute $E^F_{*r}(RPF)$ as follows:

$$E^F_{*r}(RPF) = \mu(d(l^{RPF}_{*r}, l)) \tag{5.16}$$

Other metrics Other common statistics provide quantitative information about the distribution of this errors. In this chapter we will use the mean, the standard deviation, the minimum and the maximum values to describe our results. The above statistics can be applied to the set $d(l^A, l^B)$:

- Mean $\mu(d(l^A, l^B)) = \frac{1}{T}\sum_{t=1}^{T} d(l^A_t, l^B_t)$
- Standard deviation $\sigma(d(l^A, l^B)) = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(d(l^A_t, l^B_t) - \mu(d(l^A, l^B)))^2}$
- Minimum $min(d(l^A, l^B)) =$ the smallest $d(l^A_t, l^B_t)$
- Maximum $max(d(l^A, l^B)) =$ the largest $d(l^A_t, l^B_t)$

5.3.2 Relational identification error

With respect to our second main objective, *i.e.*, to identify the relations, we consider the error relative to the classification induced by our current belief. At a particular time step t, we are given the belief distribution $bel(s_t)$ and we have to associate a value for the grounding of each binary relation. We consider two alternative ways to do that:

1. Fixing a threshold: $bel(s_t)$ assigns a probability to the true value of each relation. To associate a unique value (true or false) to each predicate, we fix a threshold th and assign the value $true$ if the probability is greater than th, else $false$.

2. Picking the most probable value: this method chooses the most probable value for each relation as a representative value for the relations modeled in s_t (this is the "most-likely" relational assumption that we also considered in the tracking error we presented in Equation 5.14). In other worlds, we pick the values that are most represented by the particles.

We call I^A the resulting *interpretation* (with one of these methods) using algorithm A. The correct interpretation I is matched against our guess and we count the mismatched cases. The comparison of guessed and true values generates a 2×2 contingency table that expresses the correct and false matches between the two sets of interpretations based on:

1. true positives, N_{tp}: the number of guesses confirmed by the correct interpretation
2. false positives, N_{fp}: the number of guesses not matched in the correct interpretation
3. true negatives, N_{tn}: the number of guesses rejected (correctly identified as not-matched)
4. false negative, N_{fn}: the number of guesses erroneously accepted as a match, while they are unmatched in the correct interpretation.

	Correct interpretation	
Guessed	positive	negative
positive	N_{tp}	N_{fp}
negative	N_{fn}	N_{tn}

Table 5.1: 2×2 contingency table

Name	Index
sensitivity	$N_{tp}/(N_{tp} + N_{fn})$
specificity	$N_{tn}/(N_{tn} + N_{fp})$
accuracy	$(N_{tn} + N_{tp})/N$
positive predictive value	$N_{tp}/(N_{tp} + N_{fp})$
false negative rate	$N_{fn}/(N_{tp} + N_{fn})$
false positive rate	$N_{fp}/(N_{fp} + N_{tn})$
negative predictive value	$N_{tn}/(N_{tn} + N_{fn})$

Table 5.2: Scoring indexes for a method of identification of the correct relation.

In Table 5.2 we report some indexes derived from the comparison between correct and guessed interpretations:

A graphical tool to evaluate the performance of a classification method is the *Receiver Operator Characteristic* (ROC) curve. The ROC curve is a graphical plot of the *sensitivity* vs *(1−specificity)* for a binary classifier system as its discrimination threshold is varied. It can also be represented equivalently by plotting the true positive rate vs the false positive rate.

True positive rate determines a classifier performance on classifying positive instances correctly among all positive samples available during the test. False positive rate, on the other hand, defines how many incorrect positive results occur among all negative samples available during the test.

A *ROC space* is defined by false positive rate and true positive rate as x and y axes respectively, which depicts relative trade-offs between true positive and false positive. Each prediction result represents one point in the ROC space.

The "ideal" prediction that is always correct would yield a point in the upper left corner of coordinate (0,1) of the ROC space, representing 100% sensitivity (no false negatives) and 100% specificity (no false positives). The (0,1) point is also called the perfect classification. A completely random guess would give a point along the diagonal line (the so-called line of no-discrimination) from the left bottom to the top right corners.

The diagonal line splits the ROC space in areas of good or bad classification. Points above the diagonal line indicate good classification results, while points below the line indicate wrong results.

Our RPF produces probability values representing the degree of belief that the objects are in relationships. If we discretize the beliefs, setting a threshold value, we will determine a point in the ROC space. Different values of the threshold correspond to different points in the ROC

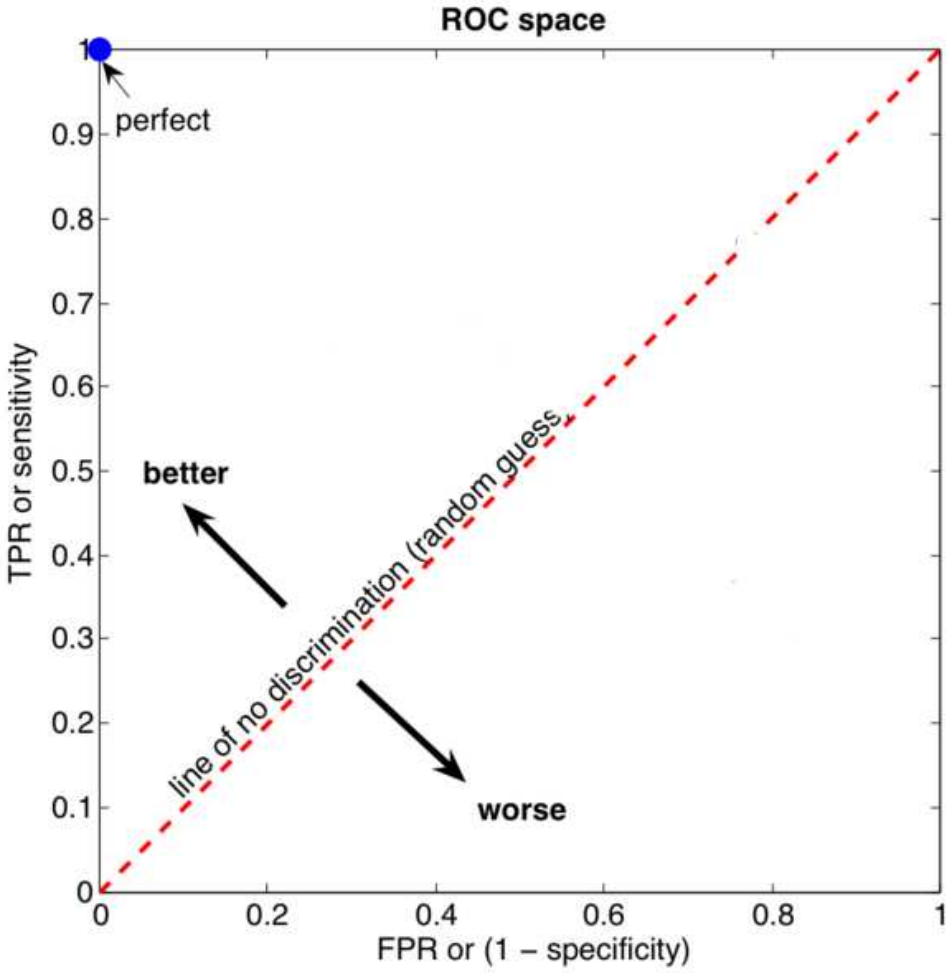


Figure 5.1: The ROC space.

space. Plotting the ROC point for each possible threshold value results in a curve.

At each time step, in the ROC curve we plot the performance of our method in identifying each of the modeled relations.

5.3.3 Experimental Goals

The hypothesis that we want to verify in the experiments are the following:

- Prediction error is lower for the RPF than for a traditional PF: $E^P(RPF) < E^P(PF)$.
- Filtered tracking error is lower for the RPF than for a traditional PF: $E^F(RPF) < E^F(PF)$ and also using the most-likely relational assumption.
- The performance of our RPF in correctly identifying relations is better than that of an

alternative rule-based classifier with respect to all the performance indicators presented before.

5.4 Exp1: one-way road scenario

Our first domain is composed of three agents moving in a one-dimensional path, each of them starting at the same time. We can imagine the data set as being obtained by a camera recording the traffic of a one-way road, and we concentrate only on a queue of three cars. The camera records the position of the cars moving behind it. In this scenario, at a certain time step, the object that leads the line slows down and therefore the other agents have to slow down as well.

This experiment is the simplest among the ones we present: we are not concerned with the understanding of what the agents are doing but we focus on tracking the positions of the cars. The relation that influences the behavior of the target in this settings is the relation of $BeingInFront$ that does not change over time. We track the objects in the scene taking into account the relation that exists between them. Our results show that a RDBN is an appropriate and effective way to model the behavior of an agent: using RDBNs our method predicts the agent's position more accurately improving the performance of standard tracking methods.

5.4.1 Experimental settings

We represent the *state of the attributes* of our relational domain at time t with the position (p_t^i) and the velocity (v_t^i) of each target (i) in the scene.

The three agents move forming a single line, their speed is not deterministically known but it is correlated as the cars behind cannot overtake the car in front. This means that by observing a variation of the speed of the first car, we can often predict a similar variation on the other cars.

We use a *dynamic model* that computes $[p_t^i, v_t^i]$ given the state at the previous time step as the following:

$$\begin{aligned} p_t^i &= p_{t-1}^i + v_{t-1}^i dt + \frac{1}{2} a\, dt^2 \text{ and} \\ v_t^i &= v_{t-1}^i + a\, dt, \end{aligned} \tag{5.17}$$

where a is a random normally distributed variable that represents the possible acceleration of an agent.

To correlate the prediction done over the system to the relation that exists between the targets we let the probability distribution given by the transition model being represented by the FOPT reported in Figure 5.2, where the dependencies between the position and velocity of two different targets are explained by the relations that can exist between them given by their relational structure.

For each target j the probability of its next state is given by a Gaussian distribution with

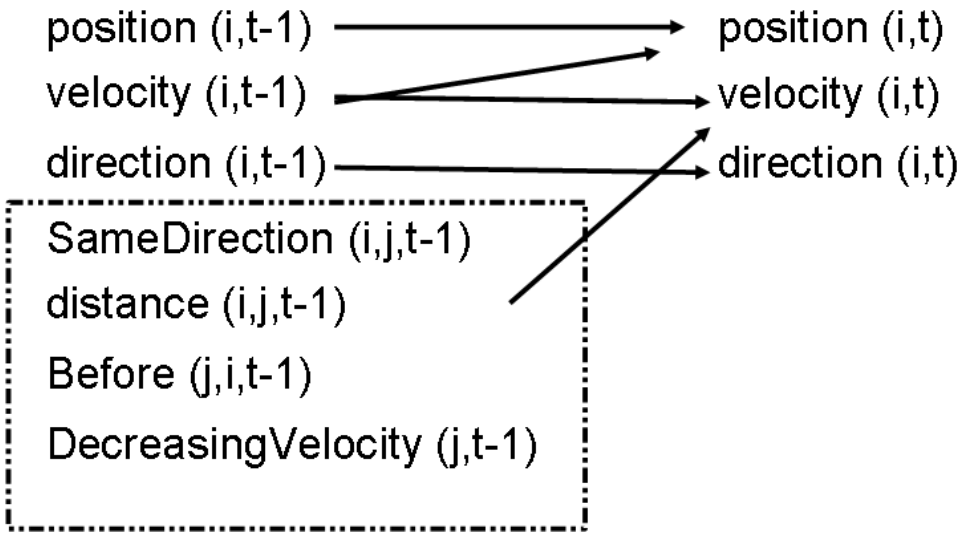


Figure 5.2: The FOPT for the objects moving on a one-way road. The dependencies between the states of two different targets are expressed by the relational structure.

mean $\mu_t = A\mu_{t-1} + a$ where A is the matrix:

$$A = \begin{pmatrix} 1 & 0 & (1 + \frac{r}{1+d_{i,j}})dt & 0 \\ 0 & 1 & 0 & (1 + \frac{r}{1+d_{i,j}})dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1. \end{pmatrix} \tag{5.18}$$

In the matrix A, $d_{i,j}$ is the distance between the target j and another target i in the scene and r is such that:

- if $\exists\, i$: $BeingInFront(i, j)$ holds and its velocity v_t^i has decreased ($v_{t-1}^i < v_{t-2}^i$): $r = \frac{v_t^i - v_{t-1}^i}{v_{t-1}^i}$.
- otherwise $r = 0$

We can define r as the ratio between the deceleration at time t of the front target and its speed at time $t-1$, we expect the target behind to decelerate as much as the target in front, with the distance acting as discount factor. It should be noticed that the proposed transition model is suitable for other kinds of relations like "being on the right/left" of another target.

As a *sensor model* we use a model that relates the measurements (z_t) with the state of the attributes (s_t^a) using a normal distribution centered in the real position of the car:

$$p(z_t|s_t^a) = \frac{1}{\sqrt{2\pi}\sigma} exp\{\frac{d(z_t, s_t^a)^2}{2\sigma^2}\}, \tag{5.19}$$

where $d(z_t, s_t^a)$ is the Euclidian distance between the measurement and the state of the attributes.

Results

In this experiments we compare the performance of our RPF to that of a PF algorithm that uses a transition model that does not take into account relations. The transition model used in the latter algorithm is a transition model based on the matrix A with r always equal to 0.

In both cases, the data association algorithm associates at each prediction the nearest measure obtained (in a certain distance range).

The RPF at each step, for each target, checks if the distance with the target before is lower than a given threshold and if the target before slowed down at the previous time step, in that case it computes the prediction of the next state with r different from 0 (refer to Equation 5.18).

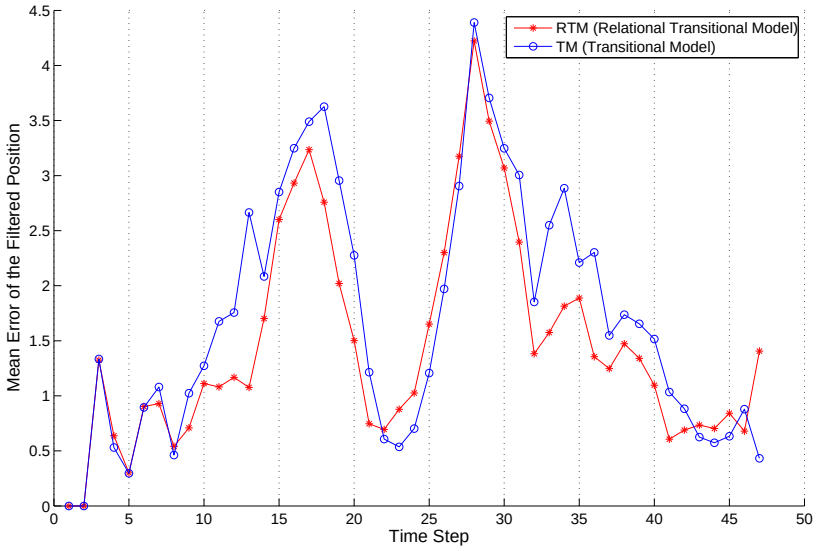


Figure 5.3: Tracking error for object 3 for each time step, with both methods (number of particles $M = 1000$ and $\sigma = 1.5$ cfr. Equation 5.19). At steps 15, 31 and 33 object 2 (that is in front object 3) slows down. At steps 16, 32 and 34 the RPF correctly expects the agent to slow down and achieves a better prediction of the trajectories in these and the following steps.

The tracking errors of the two methods for one of the three agents in this scenario are shown in Figure 5.3. Our approach achieves better results with respect to the real position of the agent than a standard approach. In particular we can notice that at the time steps in which the target slows down because the target in front slowed down, our method achieves good results and maintains this advantage over the PF method for the following steps as well.

We now compare the results considering their tracking error only for those steps in which the relation $BeingInFront$ is believed to be true and the speed of the car in front has decreased, (*i.e.*, $r \neq 0$ in Equation 5.18). We do that because it is in these cases that our relational reasoning can give an advantage; while in the other cases, both methods behave in the same way. For each of the time steps in which r differs from zero, we compute the distance between the tracked state (i.e, the filtered position given the measurement) and the true state. When the

tracks have been completely processed, we compute the average of these distances along each track obtaining the tracking error. We iterate this process 100 times for each of the two methods (RPF and PF) and we calculate the average of the tracking errors over the 100 iterations at the end of the simulations. We compare the results for number of particles (M) equal to 100 and 1000 and values of the variance of the sensor model (crf. σ in Equation 5.19) equal to 0.3, 0.5, 1, 1.5, 2 and 3.

Of course, the error for the first target (the car in front in the lane) is comparable between the two methods because to track it the system would never use the relational hypothesis. The average errors for the other two targets are reported in Table 5.3:

Tracking Error	$M = 100$		$M = 1000$	
	$E^F(RPF)$	$E^F(PF)$	$E^F(RPF)$	$E^F(PF)$
$\sigma = 0.3$				
Obj2	11.57	19.42	8.23	12.36
Obj3	12.98	20.88	8.32	15.90
$\sigma = 0.5$				
Obj2	6.34	8.50	3.68	4.95
Obj3	6.43	9.52	3.62	5.92
$\sigma = 1.0$				
Obj2	5.40	6.18	3.56	4.19
Obj3	5.18	6.09	3.42	4.57
$\sigma = 1.5$				
Obj2	5.70	6.13	4.25	5.14
Obj3	5.32	6.28	4.15	5.33
$\sigma = 2.0$				
Obj2	6.04	6.68	5.42	6.02
Obj3	5.29	6.82	4.94	5.94
$\sigma = 3.0$				
Obj2	6.14	6.91	5.96	6.74
Obj3	5.63	6.86	5.75	6.38

Table 5.3: Tracking error for the two methods, PF and RPF, for different values of σ and M.

In Table 5.3 we can see that in all the cases where the relation is true, the tracking error is lower for our RPF than for a standard PF.

In terms of execution time, the proposed approach is not more computational demanding than a standard tracker. In fact given that the two trackers has been coded in the same way, using the same data association and the same importance sampling approach; the execution time averaged over 100 iterations of the two trackers using 100 particles is for our RPF 1.61 s and the other takes 0.1 seconds less.

We can finally conclude that, in this settings, our method has shown to be more effective in terms of precision of tracking without being more computational demanding than a standard tracker.

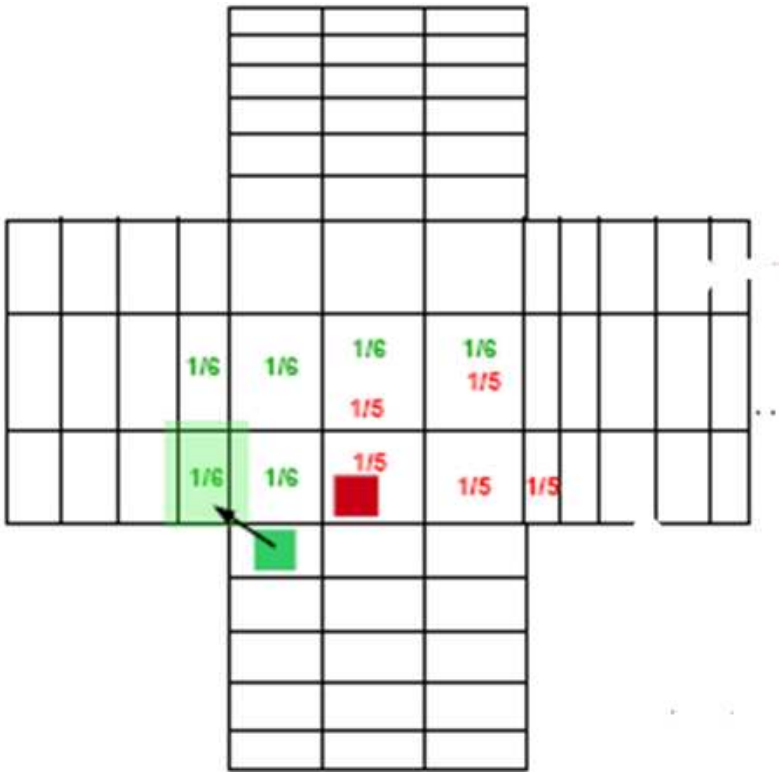


Figure 5.4: The crossroad where the simulated objects can travel together.

5.5 Exp2: identification of vehicles traveling together

The second data set contains positions of 15 objects traveling on the crossroad depicted in Figure 5.4. The crossroad is divided in 59 cells. From each cell an object can move to one of the n cells in its neighborhood[2].

Again, we can imagine this data as obtained by a camera installed right over the crossroad. The camera records the position and the color of the cars (with some noise).

In this scenario, we make the hypothesis that objects can move together if they are of the same color (to represent vans of the same company). We also make the hypothesis that if moving together, two (or more) objects will be always in a cell from which it is possible to reach (one of) the other(s) or vice-versa in one time step ("reachability" assumption). Consequently, if moving together the objects will behave similarly (*i.e.*, if one turns right also the other(s) will turn).

The relation that influences the behavior of the target is the relation of $TravellingTogether$ that does not change over time but has to be recognized by the system.

The goal of these experiments is twofold: we want to track the objects in the scene and recognize online which objects are traveling together.

[2] n take different values in order to model different speed

5.5.1 Experimental settings

We model the *state of the attributes* of the relational domain with the position (cell ID) and the color of each object in the scene:

$$s_t^a = (ID, color). \tag{5.20}$$

The *state of the relations* reports, for each object, the list of the objects that are traveling with it. Our RPF gives the probability distribution of the state of the domain at each time step.

An object i in a cell c_i at time step t can move to one of the n cells connected to c_i. We will call this set of cells $N(c_i)$. When traveling together with an object j that has moved in c_j, i can move from c_i only to that cells reachable from j in one time step (for our "reachability" assumption). Therefore, i can move only in the cells z such that $z \in N(c_i) \cap N(c_j)$.

The transition model, representing the distribution of the state at the next time step, has to be consistent with the reachability assumption. For ease of computation, we establish an order to all the objects ($j \prec i$) and we process the objects in that order. So if $j \prec i$ for the particle m, we first predict the position of the object j and then we predict the (constrained if $TravelingTogether(i, j) = true$) position of the object i.

The *probability transition model* will give equal probability to each cell consistent with the reachability assumption, conditioned to the value of the relation. We denote with $U(X)$, the uniform distribution assigning the same probability $1/ \mid X \mid$ to all elements $x \in X$, and 0 to all other elements. We now can write:

$$p(s_t^a | s_{t-1}^a, s_{t-1}^r) = \begin{cases} U(N(c_i) \cap N(c_j)) & \forall j \text{ s.t. } TravelingTogether(i, j) = true \\ U(N(c_i)) & \text{otherwise} \end{cases} \tag{5.21}$$

that gives the same probability at each possible cells to be reach by the object at the next time step.

We relate the measurements (z_t) with the state of the attributes (s_t^a) with the following *sensor model*:

$$p(z_t | s_t^a) = \frac{1}{\sqrt{2\pi}\sigma} exp\{\frac{d(z_t, s_t^a)^2}{2\sigma^2}\}. \tag{5.22}$$

Where the noise is normally distributed and the distance between the measurement and the state ($d(z_t, s_t^a)$) is computed with the Dijkstra's algorithm that gives the number of cells that the object should cross to move from the prediction to the observed state.

Results

In our domain, there are 15 moving objects (with IDs = 1, 2, ..., 15). The ground truth (unknown to the tracking system) is that objects 2, 4 and 12 travel together as well as objects 3 and 7 all other objects travel alone. We compare the tracking performance of our method with the performance of a PF that does not take into account relations. Table 5.4 reports the tracking error of the two methods with $M = 1000$ particles.

	$E^F(PF)$	$E^F(RPF)$
Objects traveling alone		
Obj1	4.7	4.6
Obj5	4.6	4.7
Obj6	4.6	2.7
Obj8	1.9	1.5
Obj9	4.6	4.6
Obj10	2.2	2.4
Obj11	3.7	1.3
Obj13	2.0	2.0
Obj14	5.9	5.8
Obj15	1.6	2.2
Objects traveling together		
Obj3	5.6	5.2
Obj7	3.8	3.5
Objects traveling together		
Obj2	3.6	3.5
Obj4	2.6	2.5
Obj12	2.7	2.1

Table 5.4: Tracking error for the two methods, PF and RPF, applied to the cross roads data set. Objects 2, 4 and 12 and objects 3 and 7 are traveling together.

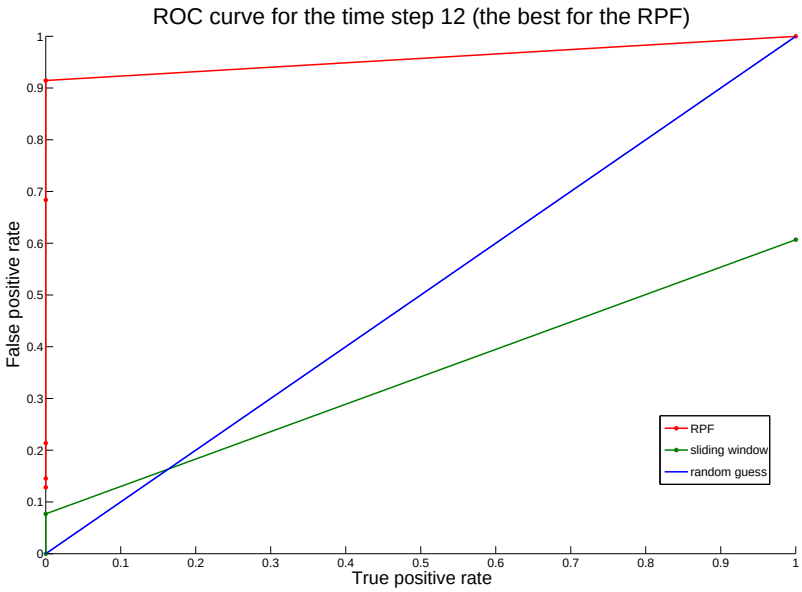


Figure 5.5: ROC curve to evaluate the performance of our method. Identification of the relation $TravelingTogether$ at time step 12. Time step 12 is the time step of best performance for our RPF.

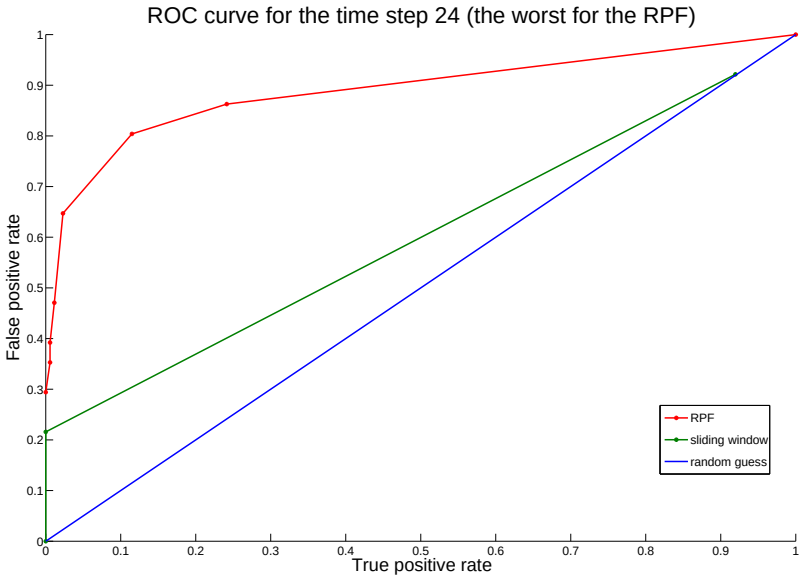


Figure 5.6: ROC curve to evaluate the performance of our method. Identification of the relation $TravelingTogether$ at time step 24. Time step 24 is the time step of worst performance for our RPF.

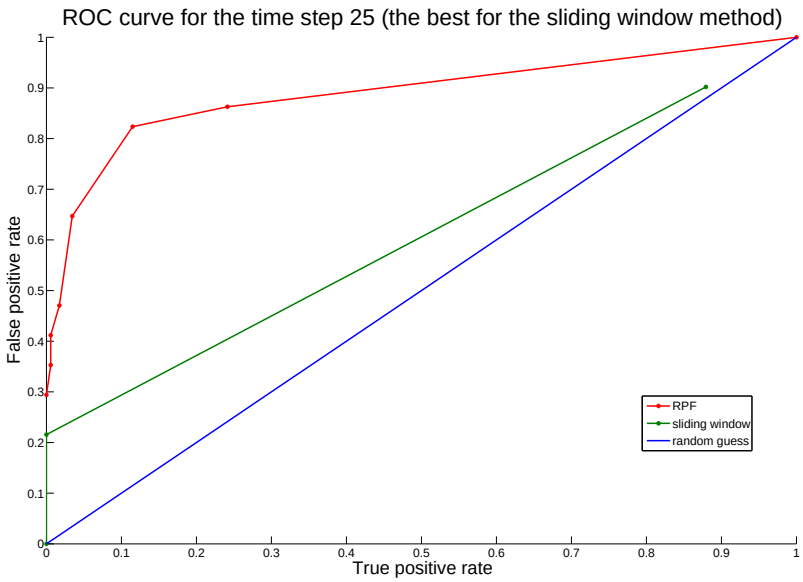


Figure 5.7: ROC curve to evaluate the performance of our method. Identification of the relation $TravelingTogether$ at time step 25. Time step 25 is the time step of best performance for the standard moving window approach.

Focusing on the agents that do travel together, we can see that RPF achieves a lower tracking error: for objects 2, 4 and 12 and for objects 3 and 7 our method gives better results[3]. This gain does not come at the expense of tracking the other agents, as both RPF and PF have comparable performance.

We compare the relation recognition performance of our RPF with a *moving window method*. This approach, at each time step, computes the distance between every couple of objects (i, j) and gives to the relation $TravelingTogether(i, j)$ the true value if the objects have the same color and their distance is lower than a given threshold. Finally it averages this value over a moving window of 10 time steps. At each time step the two methods return the probability distribution of the relation to be true. We plot the ROC curve for some of the time steps that are more characteristic. In Figures 5.5 and 5.6 we report the ROC curve for the steps of best and worst performance of our RPF compared to the performance of the moving window method at the same time steps. In Figure 5.7 we report the ROC curve for the two methods for that time step in which the moving window reaches its best performance. The ROC curves have been drawn for 100 values of threshold in the range $[0, 1]$ and overall it allows us to conclude that RPF is superior to the sliding window method for the purpose of identifying the $TravelingTogether$ relation.

[3]Remember that, for these results, instead of using the Euclidian distance we compute the distance with the Dijkstra's algorithm.

5.6 Exp3: automatic surveillance of a Canadian harbor

The last data set we use is the data set provided for the Canadian Intelligent System Challenge 2008-2009[4] about the surveillance of an harbor.

Canadian government agencies are responsible for monitoring coastal activities, and in particular for detecting any behavior that might indicate that a ship represents a security risk or a risk to break the law. One type of such behavior is when two ships rendezvous at sea -an action that is seldom necessary to meet legitimate commercial objectives, but that may indicate piracy or an exchange of contraband goods for delivery to a coastal harbor.

The goal of the Challenge was that of interpret surveillance data and identify probable incidents of two ships that are either doing:

- A $Rendezvous$: two ships stop or travel slowly together to exchange goods, or
- A $Pickup$: a larger vessel drops a package into the water that is very quickly found and picked up by a smaller vessel.

The data set contains the description of 40 events happened in the sea. Given the small number of $Pickup$ activities (only 3 elements) we concentrate here on the rendezvous activities.

The contest data includes tracking data for three classes of ship:

- Cargo Ship: large ships whose job is to travel from one port to another in the most efficient possible way. These typically travel at 17 to 25 knots and seldom change heading.
- Fisher: ships and large boats of varying sizes that travel slowly (3-5 knots) when fishing and faster (11 to 16 knots) when transiting. These boats may change direction or speed frequently as part of their normal commercial activities.
- Yacht: the contest's name for a variety of smaller craft that typically travel up and down the coast for commerce or pleasure. They may travel at 20 knots or more when weather permits, and may legitimately change heading more often than cargo ships.

The simulated ships generally follow the well-established "rules of the road" for ships[5].

For the contest, the following simplifying assumptions has been made:

1. Unambiguous identity: it is always possible to know which ship to associate with each contact report
2. No weather effects: sensors and ship dynamics are not affected by changing weather
3. No other ships: each incident involves only two ships.

The purpose of our experiments is to model the relations between ships and infer what they are doing and their path. In the following subsections we present our results on different type of models.

[4] http://www.intelligent-systems-challenge.ca/home/index.html

[5] the "rules of the road" for ships can be found for example at http://www.boatsafe.com/nauticalknowhow/boating/colregs.html.

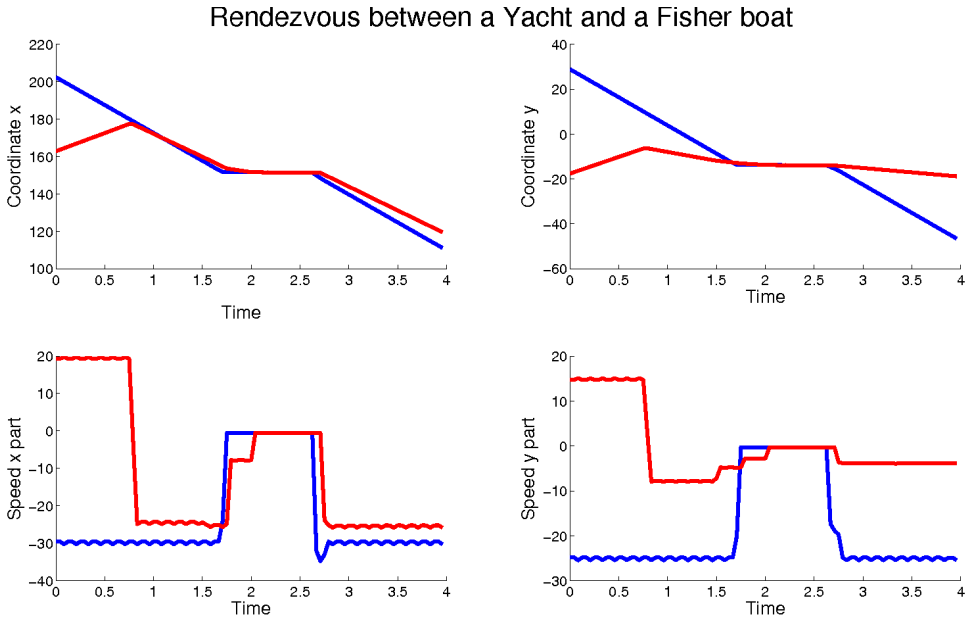


Figure 5.8: Example of Rendezvous. Of each boat the x and y coordinate and the coordinate for the speed are reported.

5.6.1 Experimental settings: Rendezvous between a Fisher and a Yacht

In the data set, for each time interval, only an event is supposed to take place at the sea (see third assumption in the previous paragraph). Our challenge is, instead, to detect activities in a multi-target environment. For this reason, we present in this section experiments that deal with couples of events occurring at the same time (i.e. recognition of 2 activities that involve 2 ships each). In particular for the experiments reported in this section we dealt with the problem of detecting a $Rendezvous$ between a Yacht and a Fisher ship. For this reason, we selected from the original data set only the events that were $Avoidance$ in general or $Rendezvous$ between a Yacht and a Fisher obtaining 20 events and we build a new data set of 120 elements, each representing tracks of four ships that can be rendezvousing or avoiding each other.

From this new data set we learnt the prior for the event $Rendezvous$ between a Fisher and a Yacht. The probability of a Fisher and a Yacht to be in relation is 33/80. From the data we observed recurring patterns that characterize the activities; we model these in the transition model so that we are able to make inference and predict the ships behavior. An example of $Rendezvous$ relation between a Yacht and a Fisher ship is given in Figure 5.8: the two ships come closer and reduce their speed till they have both nearly-zero speed, differently from a couple of ships not in relation where one maintains its speed and the other decelerates (Figure 5.9). From Figure 5.8 it is also possible to notice the three-phases which characterize the relation: ships approach each other reducing their speed in the first phase, they travel in the same direction with nearly-zero speed in the second phase and finally they go apart and at least one of them changes its speed. Our relational transition model takes into account these three different phases allowing to detect when the event starts and when it finishes (that was one of the request

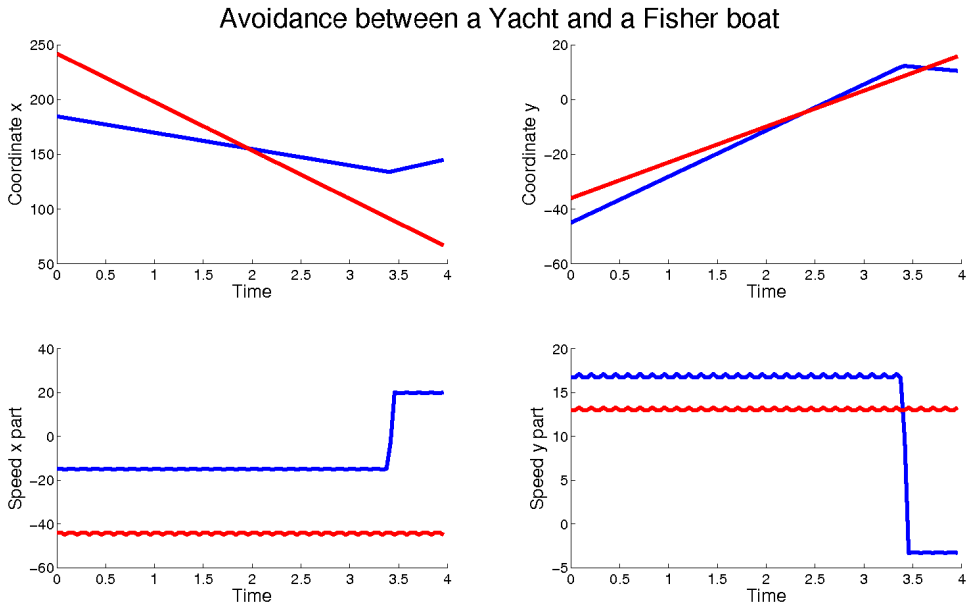


Figure 5.9: Example of Avoidance.

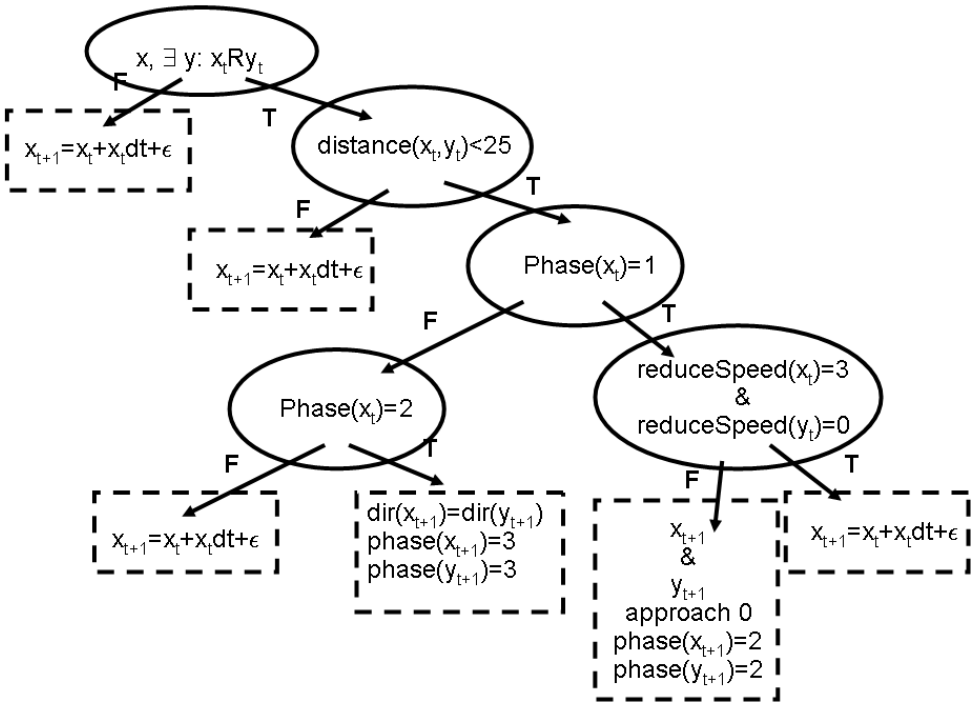


Figure 5.10: The FOPT we used to represent $p(s_t^a|s_{t-1}^a, s_{t-1}^r)$. At each time step, for each object it computes the future state given the object's relation and the phase.

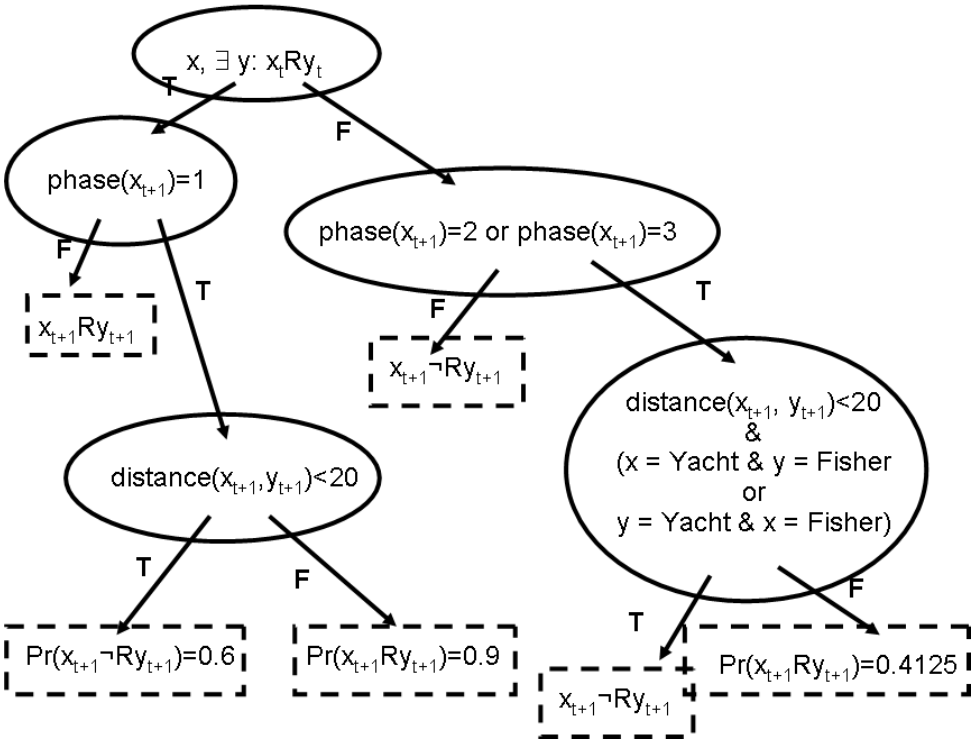


Figure 5.11: The FOPT we used to model $p(s_t^r|s_{t-1}^r, s_t^a)$. At each time step, for each object it computes the probability of the object to be in relation (or not) with another object given their attributes and the distance between them.

of the Challenge) but also allowing to understand (since a ship can be rendezvousing only another ship) if two ships can be in relation (if one of them has already finished an encounter with another ship this is not possible). Sketches of the *relational transition model* used in our experiments are given in Figure 5.10 (where the transition model for the state of the attributes is reported) and in Figure 5.11 (where the transition model for the state of the relations is reported).

The *relational state* of this domain is represented by the position, the speed and the type of each boats and by the value of the relation $Rendezvous$ between each couple of boats.

The *sensor model* is distributed accordingly to a Gaussian distribution centered on the measurements.

Results

We run the experiments on each of the 120 set of four tracks in the data set.

In the central columns of Table 5.5 we report the accuracy and the positive/negative predictive value of our method for the $Rendezvous$ detection: we assign to the relation the most probable value given the particle distribution. In the last columns we show the tracking error of our method (RPF) compared to a method that does not take into account relations (PF). We report the results divided by number of relations present -column R- and number of couple Yacht-Fish (potentially related) -column Y F- in the examined set of four boats

R	Y-F	accuracy	pos. predictive value	neg. predictive value	$E^F(PF)$ km	$E^F(RPF)$ km
2	2	$5/8$	1	-	4.4138	3.1473
1	2	$17/20$	$9/11$	$8/9$	4.3771	2.9496
1	1	$23/46$	$11/12$	$12/14$	3.2874	3.0183
0	2	$23/30$	-	1	1.4131	1.4838
0	1	$125/13$	-	1	1.0881	1.0883

Table 5.5: Results are divided by number of rendezvous relations true in the data (column R) and number of couple Yacht-Fisher (coloum Y-F). In columns TP, FP, TN and FN the number of True Positive, False Positive, True Negative and False Negative are reported respectively. In the last two columns the average tracking error for our method (RPF) and a method that does not take into account relations (PF) is reported.

5.6.2 Experimental settings: Master-Slave relation

Observing the $Rendezvous$ events in the data set we noticed a peculiar characteristic of the two boats involved in an encounter: one plays the role of *master*, while the other that of *slave*. In particular, we focus on the variation of speed of the two targets: the master-boat is the one that first decreases its speed and decides where to stop (or start going very slowly) and when the encounter is finished; the slave-boat "imitates" the behavior of the other ship (see Figure 5.8).

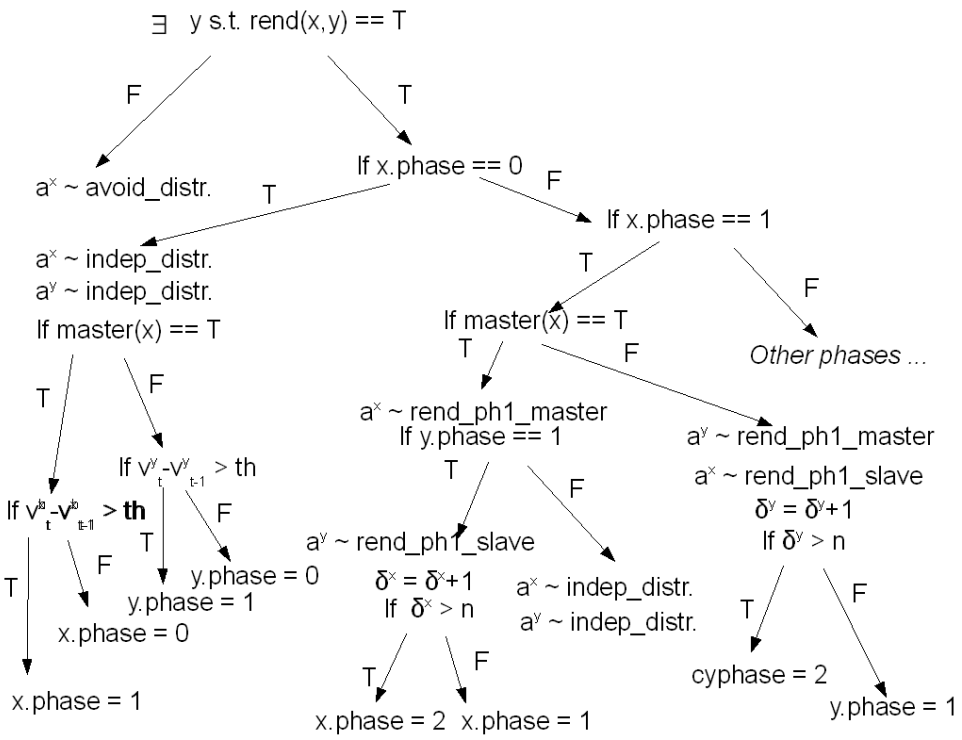


Figure 5.12: A possible FOPT for $p(s_t^a|s_{t-1}^a, s_{t-1}^r)$. At each time step, for each object it computes the future state distribution given the object's relation.

Different is the case of ships that are avoiding each other (thus not in relation according to our model), one maintains its speed and the other decelerates (see Figure 5.9).

In this section we present the experimental results obtained by applying this intuition to all the original data set. The "role" of the boat (either master or slave) is also unknown. The RPF considers both hypothesis -every particle makes an assumption about which is the master- and the importance sampling propagates the particles that make the hypothesis that better explains the observations.

We used the data set to estimate the prior for the event $Rendezvous$ between different couples of boats and the prior for a boat involved in a $Rendezvous$ to be the master-boat.

In these experiments we model the rendezvous activity as a five phases activities (as it is also possible to notice in the images):

- ships are traveling independently (we call this phase zero),
- then, the master boat decreases its speed and decides where to stop (*i.e.*, it decides where the $Rendezvous$ will take place, see time step 1.6 in Figure 5.8)
- in phase two, the slave boat approaches the master (at time steps from 1.7 to 2)
- phase three is the period in which the boats are rendezvousing: they maintain the same direction and no (or very low) velocity

- finally, the master boat decides the $Rendezvous$ is finished and go apart (we call this phase four) and both boats start traveling independently.

Our new implementation of the relational transition model takes into account these different phases allowing to detect when the event starts and when it finishes and the role of the ships involved.

We represent the state of the attributes with the position (p) and the velocity (v) of each target in the scene and use a dynamic model that computes $[p_t, v_t]$ as the following:

$$\begin{aligned} p_t &= p_{t-1} + v_{t-1}dt + \frac{1}{2}a\ dt^2 \\ v_t &= v_{t-1} + a\ dt, \end{aligned} \tag{5.23}$$

where a is a random variable whose distribution depends by the objects'type, the relation and, if the relation is true, it depends also by the role played by the object and the phase of the ongoing activity. We learnt the distribution of a from the data set.

Each particle represents the state of the domain: it represents the position and the velocity of all the objects in the scene as well as their relations. In the relational state we represent the value of the relation and, if the relation is true, the role of each boat in the scene. When sampling the particles, we take into account the *order* of the objects introduced by the master-slave relationship to predict the future state of each object taking into account the relations. Examples of the FOPTs we used to do that are reported in Figure 5.12 and in Figure 5.13.

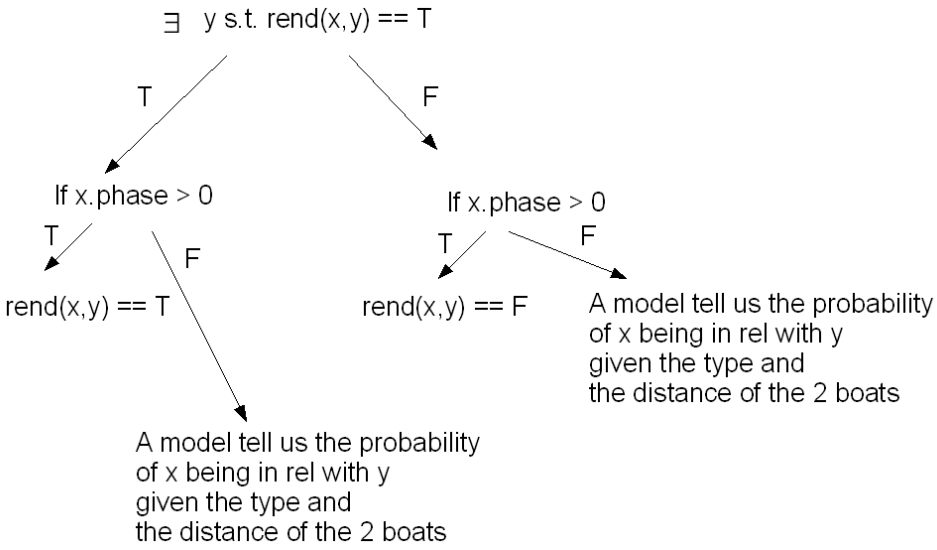


Figure 5.13: An example of FOPT for $p(s_t^r|s_{t-1}^r, s_t^a)$. At each time step, for each object it computes the probability of the object to be in relation (or not) with another object given their attributes and the distance between them.

Results

We ran the experiments on each of the 37 events in the original data set. We apply PF techniques with ten thousands particles. In Table 5.6 we show the true positive and true negative rate of our method for the rendezvous detection (we consider, again, for the value of the relation the most probable value, given the particle distribution) compared to a method that randomly chooses which boats are in relation.

	RPF	random
true positive rate	0.8947	0.4444
false positive rate	0.6111	0.4841

Table 5.6: True positive and true negative rate of our method for hte rendezvous detection compared to a method that randomly chooses which boats are in relation.

In Table 5.7, we compare some statistics of the tracking errors of our RPF and a standard method (a PF that does not take into account relations between objects). We compare the mean

	$E^F_{*r}(RPF)[km]$	$E^F(PF)[km]$
all tracks		
mean	1.8533	2.1967
max	5.6811	8.1665
min	0.001	0.001
st.dev.	2.3660	2.9716
only rendezvous		
mean	1.6178	2.6356
max	4.0405	10.3366
min	0.001	0.001
st.dev.	2.2077	2.7612
only correctly detected		
mean	1.3167	3.1435
max	3.0108	11.2724
min	0.001	0.001
st.dev.	1.6451	2.2069

Table 5.7: Some statistics for the prediction error of the two methods: our RPF and a standard PF for their average tracking error are reported averaged over all the tracks, over only the rendezvous tracks and over only that tracks which RPF correctly recognizes as rendezvous activity.

of the errors over time (for our RPF we compute the error averaged over that particles that recognize the relation as the most probable (E^F_{*r})), the minimum and maximum values of the

errors and its standard deviation (st.dev.). We report the overall results for all the tracks and the result for only the rendezvous tracks and, finally, for only the rendezvous tracks that our activity detection system correctly recognizes.

From these Tables 5.7 and 5.6 we can see that:

1. The accuracy of our relational activity recognition method is better than a simple standard method.

2. Comparing the tracking error in this Section with the results reported in the Table of Section 5.6.1, we can say that the introduction of the master-slave assumption has increased the performance of our method, generating better tracks and better exploiting the agents' behavior.

3. Moreover, the tracking error for the agents whose activity is correctly recognized is particularly low and better than a standard method.

4. The standard deviation of the tracking error of our method is lower than the one of the PF's error, this means that, during time, the error does not vary a lot giving the convergence of the method.

We evaluate the performance of our algorithm on this data set with the prediction error as well. In Table 5.8 we report some statistics for the prediction error of our RPF and a standard PF:

	$E^P_{*r}(RPF)[km]$	$E^P(PF)[km]$
mean	2.12457	2.95225
max	6.0.1905	8.27881
min	0.001	0.001
st.dev.	1.6472	2.2087

Table 5.8: Some statistics for the prediction error of the two methods: our RPF and a standard PF.

As we was expecting the prediction error is bigger than the tracking error. From the Table we can see that the prediction of our method is more accurate than that of a standard method.

5.7 Execution Time

In terms of *computational performance*, our RPF is not more computational demanding than a standard tracker. In table 5.9 we report the execution time averaged over 100 iterations of our method ($\Delta t(RPF)$) and a standard PF ($\Delta t(PF)$) for different settings of the number of particles (M) and number of possible relations (R) between the tracked objects. The computational time increases at the increasing of the number of particles and as the number of objects (two opposed to four) increases but it doe not increase much using our RPF.

M	R	$\Delta t(RPF)$	$(\Delta t(PF))$
100	1	4.9 s	4.6 s
100	2	23.3 s	23.2 s
1000	1	53.6 s	53.1 s
1000	2	105.7 s	104.3 s
10000	1	128.9 s	127.2 s
10000	2	211.4 s	208.1 s

Table 5.9: Execution time averaged over 100 iterations of our method ($\Delta t(RPF)$) and a standard PF ($\Delta t(PF)$).

5.8 Conclusion

In this chapter we presented the experiments we performed to show the performance of our RPF.

We evaluated our approach on three synthetic data sets: the first records the positions of some objects moving on a one-lane way, the second records the position and the color of a variety of objects that can be moving together or not on a crossroad, the last data set records the position of boats in a Canadian harbor for surveillance purposes.

These experimental domains are of particular interest because the targets' behaviors are dependent on some external conditions that can be modeled with relations.

We outlined the experiments and introduced the performance indicators at the beginning of the chapter. For each data set, we presented in detail the experiments we carried on and the results we obtained. Results validate the following hypothesis:

- Our RDBN based approach is able to model more accurately dynamic domains where relations between objects can be interpreted as important clues for the understanding of the on-going activities.
- The advantage of our relational tracker is particularly pronounced when the relations are correctly identified.
- Our activity recognition system, based on the recognized relations has an accuracy that overcomes simple methods.

In the next chapter we draw some conclusion and outline some future works.

Chapter 6

Conclusion

The present letter is a very long one, simply because I had no leisure to make it shorter.

Blaise Pascal.

In this chapter we review the work presented in this Thesis, highlight our achievements as well as the limitations of our work; finally, we introduce some interesting directions we would like to pursue in the future.

6.1 Contributions and limitations of this work

This Thesis proposed a novel approach to tracking multiple targets when the targets are deliberative agents that act in proactive ways and interact with each other. Modeling the interactions that the agents can undertake allows the possibility of making inference on the agents behaviors, on the activities they are carrying out and on their role. This allows to make predictions about the next actions given the current belief about the agent, considering the relationships between them and their activity.

In this work we presented a methodology that model activities and interconnected behaviors with relations, that are in turn modeled with first order logical formulas. The algorithm we presented, *relational particle filter*, is able to make inference with relations, scaling up to cases with increasing complexity.

These are our main achievements:

- We extended DBNs with first-order logic formulas defining *Relational Dynamic Bayesian Networks* (RDBNs)

- We introduced a novel tracking algorithm, *Relational Particle Filter* (RPF) for making inference on RDBNs. This algorithm is particularly suited to track multiple targets that show some relations in their movements. We showed the mathematical convergence of the algorithm: increasing the number of particles, the particle distribution becomes closer to the actual belief (the probability of the state given the priors and all the past observations).

- We discussed the problem of turning a belief distribution into an estimation of the agent's trajectory and confronted several alternatives: state estimation with the most likely relational assumption versus statistical expectation.

- We showed, with simulations, how RPF offers better estimate of the positions of the targets, exploiting the knowledge about the relations between the agents.

- We showed how it is possible to represent activities with relations, transforming the activity recognition task in the problem of tracking these relations over time. We showed that our tracking algorithm is suitable for this task because it maintains a belief distribution over the possible relations.

- We validated our activity recognition system with experiments, we showed the performance of our RPF in two domains: the identification of convoys in the area around a crossroad and monitoring activities in a Canadian harbor.

The bottom line of this work is that, by making inference about the agents behaviors, the activity that they are undertaking and their role, we can make better predictions about their actions. This in turn, once predictions are filtered using future observations, can be used to update our belief about the activities and the agents' roles.

We hope that this work will generate a considerable interest in the research community and that probabilistic relational models will be used in real life domains. As one of the main problems in this community is the lack of common benchmark data and test beds, the availability of more data sets could be a great benefit.

While the results are compelling and our formalism is theoretically sound, we acknowledge that more work is needed in order to support the adoption of relational methods in real life situations, for example to scale the algorithm to domains with a large number (hundreds or thousands) of coexisting agents, triggering an important research direction towards fast inference in RDBNs and approximated methods. Other domains offer different challenges, requiring to tailor the relational model to particular situations. In the next section we discuss some interesting future work that we consider relevant.

6.2 Current and further research directions

A number of research directions naturally arise. Some of these challenges (see next subsections) are related to problems that require quantitative advances in the inference algorithm to make it computationally more efficient (as it would be required by the adaptation to many real life applications), other require the design of a qualitatively more complex model (as application of relational inference to social networks).

Our current work is focusing on applying the presented model and inference system to more complex situations like the detection of unattended goods or the monitoring of a football match (in which the relation master-slave could be important). Moreover, as we shall see, automatic learning of the transition model is of paramount importance.

6.2.1 Detection of unattended goods

Security has become an important issue in recent years. Due to widespread fear of acts of terrorism, many governing bodies have implemented policies of control of public spaces. A common problem is the detection of unattended goods. This important problem has even attracted the interest of a number of conferences, such as the *Performance Evaluation of Tracking and Surveillance* workshop held in New York in 2006.

The problem of unattended goods detection is more difficult than one could think at first sight. First, in complex and crowded places (and with noisy sensors) as train stations or airports it can be difficult to identify the object in the first place. Second, legitimate baggages can often be left on the floor for a few seconds or even minutes, while the owner is a few steps away. We believe that a simple threshold model based on distance may not work in many cases, giving several false positives (the person just went a few steps away to check the transit map, or left the luggage close to a family member) or false negative.

We conjecture that relational modeling and reasoning can be helpful in this domain: we could model the relation of "owning" a luggage between a person and an object; the probabilistic transition model will account that a person owning a luggage will usually carry it in the hands or even left in the floor close to him for some time. But with some small (non zero) probability the luggage will be away from him or even be close to somebody else (maybe a person traveling with him). In this case, the domain needs to represent the relations between the people in the scene (considering who might be a friend of the person owing the luggage) and a RDBNs based system should reason about which person is more likely to be a friend of the luggage's owner and detect if the good has been left unattended or not.

6.2.2 Tracking football players

Our ideas can be applied to the tracking of football players. Relations can represent common game strategies, specific game patterns (as left-wing, advance, pass to center, head ball to goal) and the role of the players. Relations can then represent importance clues to the understanding of what is happening in the scene. Moreover, the relation *master-slave* introduced in the previous chapter can be an important insight of the model used to track agents in this domain. We conjecture that this relationship would show to be useful in sport domains as well. For example, you can often understand who has the ball only detecting which player has been followed by the others.

The application of our system to this domain can be important for the development of an automatic running commentary of a football match and also for the labeling of a set of videos to classify actions of players based on what has happened.

6.2.3 Relational reasoning to support Decision-making

In the future we would like to extend our work to *decision support systems*. We would like to incorporate in our inference environment decisions that can be *influenced* by the observations and can induce or *cause* a change in the domain. In this setting, an approach based on the reasoning about relations will be able to make a decision based on the correlations between the

objects in the scene and not the conjunction of the singular activities of the objects. Taking into account relations, we allow the system to see the domain as an overall world and not as a set of agents, the whole community of agents can make more smart and reasoned decisions.

6.2.4 Tracking robots

The two ideas presented before, tracking players and support decision-making, could be integrated to develop a system for the coordination of a *small league RoboCup* team.

The *small size robot soccer* is one of the *RoboCup* league divisions that focuses on the problem of intelligent multi-agent cooperation and control in a highly dynamic environment with a hybrid centralized/distributed system. During the match an overhead camera transmits video to an off-field PC that identifies and tracks the robots as they move around the field. Typically the off-field PC also performs the processing required for coordination and control of the robots.

Our idea is that of using our relational reasoning approach to track the adversaries, detecting what they are doing and their probable future actions; in this way the system can plan better strategies for the team. We think that the application of our techniques to small size robot soccer for adversarial modeling can be a very interesting and challenging application that can lead to promising results.

6.2.5 Parameter Learning

Our framework is based on the assumption that a transition model (the probability distribution of the next state given the current state and the relations) is readily available. This, however, in many practical circumstances is not the case.

A research direction of paramount importance for the practical adoption of relational methods in the real world is the study and development of algorithms suited to automatically learn the transition model from data. A number of techniques from machine learning and statistical learning could be used for this purpose.

Model uncertainty could be explicitly represented by making an hypothesis and revising it when new observed data does conflict with the current hypothesis. This idea is behind a technique called *auxiliary particle filter* that have been proposed in the literature (McKenna & Nait-Charif, 2007). In the last months, in our lab, we applied the auxiliary particle filter to financial problem, with interesting results. We hope to extend this idea to relational models, implementing an "Auxiliary Relational Particle Filter" able to learn the dependency distribution of the predicates while tracking the objects.

6.2.6 Friends matching and mobile assistants

We consider how to use relations in social networks to connect friends, and notify them about what the user is doing. The idea is to couple the versatility of social networks (used as messaging services, micro-blogging, and coordinating evening and outings) with the power of ubiquitous computing. Ubiquitous computing allows people to be always connected and is gaining

momentum with the widespread adoption of smart phones and personal assistants as PDAs.

At the moment, we are considering the integration of our system into a tool for PDA called *friend-finder*. We assume that every user has a GPS integrated on his PDA or smart phone. The application knows who are your friends and, thanks to the GPS each device is able to locate himself and can communicate its position to an application that post it on the web (to this end, the system needs to track the users positions). At this point *friend-finder* can locate your friends and give you a road-map to reach them, accounting for their moves and continuously tracking the position of each user. Moreover, it is possible that, while you move to reach a friend another friend moves as well in a position nearer to you; in this case the system would be able to suggest you to reach the second friend and together reach the first one. Or, at the contrary, if one of your friends is localized with someone else and the system recognizes that he would prefer to be alone (by observing their behavior or simply because they do not interact with other people) it could suggest you not to reach him.

This system, could also have more serious application other than entertainment. It could be useful in the future to track people in crowded environment, find lost children in big places and also help elderly persons that lost the way home.

6.3 Conclusions

A central aspect of human intelligence is the ability to make inference using abstract knowledge in structured environments that contain diverse sets of agents related to each other in a variety of ways. Current Artificial Intelligence techniques are far from matching the human capabilities of understanding complex scenes. In fact, when the environment is particularly complex, including several distinct entities whose actions might be correlated, automated reasoning becomes particularly challenging.

In this work we tackle the inference problem in complex domains by combining mathematical logic with probabilistic models. First-order logic can deal with the modelization of structured environments but it cannot treat uncertainty. On the other hand, probabilistic models can deal well with uncertainty in many real-world domains, but they operate on a propositional level, and cannot scale to cases where several instances are present.

Recently a lot of interest has arisen towards approaches that integrate these two types of models (relational Bayesian networks are an example); but not much work has been done to incorporate logical reasoning into dynamic domains; moreover inference in such domains has been carried on only in propositional terms.

In this Thesis we presented relational dynamic Bayesian networks, that are an extension of relational Bayesian networks, able to model correlated actions of different entities in dynamic domains.

Under noisy observations, an automated reasoner needs to assess the most probable situation both in terms of individual attributes and relations occurring in the scene. At this purpose, in this Thesis we developed a new algorithm for both inference and tracking,able to take into account the structure of the environment and the relations between the objects modeled by a RDBN.

In several applications, as for example surveillance systems, it is important to provide *online* reasoning, so that the appropriate cause of action can be taken when necessary (*e.g.*, raise an

alarm). Our inference algorithm, tracking the relations between objects, is a suitable tool for the online detection of those activities in which there is interaction between objects.

Appendix A

Basic Concepts in Probability

This Appendix provides basic probabilistic notions useful to understand this thesis.

Random Variables

In an uncertain domain, quantities such as sensor measurements, attributes' or relations' values are modeled as *random variables*. Random variables can assume multiple values accordingly to specific *probabilistic laws*.

Let X denote a random variable. We denote with x a specific value that X might assume. If the space of all values that X can take on is *discrete*, we write $p(X = x)$ to denote the probability that the random variable X has value x. For example in a fair coin flip, the random variable X can take on the (discrete) values heads or tails with probability $p(X = head) = p(X = tail) = \frac{1}{2}$

Discrete probabilities sum to one: $\sum_x p(X = x) = 1$. Probabilities are always non-negative: $p(X = x) \geq 0$.

Continuous spaces are characterized by random variables that can assume continuous values accordingly to a *probability density function* (PDF).

Joint Distribution

The *joint distribution* of two random variables X and Y is given by $p(x, y) = p(X = x, Y = y)$, that describes the probability of the event that the random variable X takes on the value x and that Y take the value y. If X and Y are *independent*, we have $p(x, y) = p(x)p(y)$. Often, random variables carry information about other random variables. If we already know that Y's value is y, and we want to know the probability that X's value is x conditioned on that fact, we have to compute $p(x|y) = p(X = x|Y = y)$. This probability is called *conditional probability* and is defined as

$$p(x|y) = \frac{p(x, y)}{p(y)}. \tag{A.1}$$

If X and Y are independent, we have

$$p(x|y) = \frac{p(x)p(y)}{p(y)} = p(x). \tag{A.2}$$

If X and Y are independent, Y tells us nothing about the value of X.

The unconditional or *prior probability* associated with a random variable X (p(X)) is the probability accorded to it in the absence of any other information.

Bayes' Rule

The *Bayes' rule* relates a conditional probability of the type $p(x|y)$ to the conditional probability that inverts the random variable x and y, $p(y|x)$. The Bayes' rule states that:

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)} = \frac{p(y|x)p(x)}{\sum_{x'} p(y|x')p(x')} \text{ in the discrete case,} \tag{A.3}$$

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)} = \frac{p(y|x)p(x)}{\int p(y|x')p(x')dx'} \text{ in the continuous case.} \tag{A.4}$$

If x is a quantity that we would like to infer from y, y is called the *data* (*e.g.*, a sensor measurement). The distribution $p(x)$ summarizes the knowledge we have regarding X prior to incorporate the data y. The probability $p(x|y)$ is called the *posterior probability distribution* over X. Bayes' rule provides a convenient way to compute a posterior $p(x|y)$ using the "inverse" conditional probability $p(y|x)$ along with the prior probability $p(x)$.

The denominator of Bayes' rule $p(y)$ does not depend on x. Thus, the factor $p(y)^{-1}$ will be the same for any value x in the posterior $p(x|y)$. For this reason, $p(y)^{-1}$ is often written as a normalizer in Bayes' rule variable, we will denote it with the letter α: $p(x|y) = \alpha p(y|x)p(x)$.

Marginalization

When we need to extract the distribution over some subset of variables or a single variable, we need to *marginalize* or sum out the variables other than the variables of interest. The marginalization rule for any sets of variables X and Y is given by:

$$p(X) = \sum_{y} p(X, y). \tag{A.5}$$

The distribution over X can be obtained by summing out all the other variables from any joint distribution containing X.

A variant of this rule is called *conditioning* and it involves conditional probabilities instead of joint probabilities, using the product rules:

$$p(X) = \sum_{y} p(X|y)p(y). \tag{A.6}$$

That is the *theorem of total probability*, that says that $p(x) = \sum_y p(x|y)p(y)$, in the discrete case and $p(x) = \int p(x|y)p(y)dy$, in the continuous case.

We can be interested in computing *conditional* probabilities of some variables given evidence (or observation) about others. Let X be the *query variable* (*i.e.,* the variables we are interested in computing the conditional probability of); let e be the evidence we are given and Y the remaining unobserved variables:

$$p(X|e) = \alpha \sum_y p(X, e, y), \tag{A.7}$$

where the summation is over all possible combinations of values y of the unobserved variable Y and together the variables X, E, and Y constitute the complete set of variables for the domain.

Conditional Independence

Any of the rules discussed so far can be conditioned on arbitrary random variables, such as the variable Z. For example, conditioning Bayes' rule on $Z = z$ gives us:

$$p(x|y, z) = \frac{p(y|x, z)p(x|z)}{p(y|z)}. \tag{A.8}$$

Similarly, we can condition the rule for combining probabilities of independent random variables on other variables z: $p(x, y|z) = p(x|z)p(y|z)$. Such a relation is known as *conditional independence*.

It is worth to note that conditional independence is equivalent to: $p(x|z) = p(x|z, y)$ and to $p(y|z) = p(y|z, zy)$.

Conditional independence does not imply independence, that is,

$$p(x, y|z) = p(x|z)p(y|z) \not\Rightarrow p(x, y) = p(x)p(y); \tag{A.9}$$

and absolute independence does not imply conditional independence:

$$p(x, y) = p(x)p(y) \not\Rightarrow p(x, y|z) = p(x|z)p(y|z). \tag{A.10}$$

Chain Rule

Another very useful rule in probability theory is called the *chain rule*. The chain rule computes joint probabilities from conditional probabilities: consider three random variables, X, Y, Z, the chain rule claims:

$$p(x, y, z) = p(x|y, z)p(y|z)p(z). \tag{A.11}$$

If we expand out the conditional probabilities with their definitions, we get

$$p(x, y, z) = \frac{p(x, y, z)}{p(y, z)} \frac{p(y, z)}{p(z)} p(z), \tag{A.12}$$

when written this way, we see that each terms numerator cancels the previous terms denominator, leaving us with a simple expression that $p(x, y, z)$ equals itself. The chain rule is important when one have to estimate the probability distributions of sequences of data.

Normal Distribution

An example of density function is that of the one-dimensional *normal distribution* with mean μ and variance σ^2. The PDF of a normal distribution is given by the following *Gaussian* function:

$$p(x) = (2\pi\sigma^2)^{-\frac{1}{2}} exp\{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\} \tag{A.13}$$

that assumes x to be a scalar value. Often, x will be a multi-dimensional vector. Normal distributions over vectors are called *multivariate*. Multivariate normal distributions are characterized by density functions of the following form:

$$p(x) = det(2\pi\sigma)^{-\frac{1}{2}} exp\{-\frac{1}{2}(x-\mu)^T \Xi^{-1}(x-\mu)\}, \tag{A.14}$$

where:

- μ is the mean vector and
- Ξ a positive semi-definite and symmetric matrix called the covariance matrix.

Equation A.14 and Equation A.13 are equivalent when x is a scalar value and $\Xi = \sigma^2$.

Equations A.13 and A.14 are examples of PDFs. As discrete probability distributions always sum up to 1, PDFs integrate to 1:

$$\int p(x)dx = 1. \tag{A.15}$$

Unlike a discrete probability, the value of a PDF is not upper bounded by 1.

Expectation and Covariance

The *expectation* of a random variable X is given by: $E[X] = \sum_x xp(x)$ for discrete cases and $E[X] = \int xp(x)dx$ for continuous cases. The expectation is a linear function of a random variable: $E[aX + b] = aE[X] + b$.

The *covariance* of X is obtained as $Cov[X] = E[X - E[X]^2 = E[X^2] - E[X]^2$. The covariance measures the squared expected deviation form the mean.

Appendix B

RBNs subsume PRMs

As discussed in Chapter 2, PRMs (Koller, 1999) can also be used to model uncertainty in a relational domain. However they are based on frame-based systems, and inherit the limitations of the latter.

After presenting in more detail PRMs, in this Appendix we show that RBNs subsume PRMs.

B.1 Probabilistic Relational Models

Probabilistic Relational Models (Koller, 1999)(from now one, K-PRMs [1]) are the extension to relational domain of Probabilistic Models (PM), they specify a joint distribution over a relational domain.

A *schema* for a K-PRM describes a set of classes, for each class its attributes and the set of relations between them. A K-PRM defines a probability distribution over the possible instances of a given schema. It can be thought as a PM which defines the dependency model at the class level, allowing it to be used for any object in the class and that explicitly uses the relational structure of the model. In particular, it allows the PM of an attribute to depend also on attributes of related objects.

The sequence of relationships that permits to an attribute to depend by another one of a different class is called *slot chain*. A slot chain $C.\tau$ represents the sequence of objects that are τ-relatives of an attribute ($C.A$) of the class C.

In (Getoor, Friedman, Koller, & Pfeffer, 2001) the concept of *relational schema* and *relational skeleton* are introduced. A relational skeleton for a relational schema is a partial specification (instantiation) of the schema: it specifies the objects involved in the schema and the relations between them. In (Getoor et al., 2001) Getoor *et al.*, define a K-PRM as the RBN that specifies the probability distribution over particular instantiations of a given skeleton: for each schema there are more skeletons, for each skeleton there are more completions (*i.e.*, complete instantiations). A K-PRM induces a distribution over instances that complete the skeleton.

The structure of a K-PRM is defined by associating to each attribute a set of formal parents that will be instantiated in different ways for different attributes. There are two types of formal

[1] We will call *K-PRM* the PRMs presented in (Friedman et al., 1999) to differentiate them from the more general PRMs of which RBNs are a sub-class.

parents:

- Parents of an attribute that are attributes of the same object; in this case the same conditional probabilistic model is applied to every connection child-parent.
- Parents of an attribute that are attributes of different class related through a slot chain.

For the objects part of the slot chain $C.\tau$ we must specify the probabilistic dependence of $C.A$ on the multi set $\{K.B : K \in C.\tau\}$. This dependence poses a representation problem: we need to specify the distribution of $C.A$ given a multi set of values of unknown size. This issue is addressed introducing the notion of *aggregation*

B.1.1 Aggregation

To represent the conditional probabilistic model of a connection child-parents in which child and parents are attributes of two different classes in K-PRMs an *aggregation* of the values of the parents is used. The dependence of $C.A$ on $C.\tau.B$ is interpreted as a probabilistic dependence of $C.A$ on some (deterministically computed) "aggregate property" of the multi set of these parents.

For each node, the set of parents is divided with respect to the class and the mean of the values of each object in that class computed treating each of the obtained values as an "artificial" value of a "super-parent" of the considered node. The set of parameters associated with the qualitative structure is represented by a CPTs for each attribute of each class; this parameters are shared by each objects in the class.

B.2 RBNs subsume K-PRMs

Theorem 1 *For each K-PRM representing a probability distribution over a relational domain there is an RBN representing the same distribution.*

Proof 1 *We will first convert the attribute and the reference slots in a K-PRM to predicates of a RBN, then add the corresponding edges to indicate the parents of a node, and finally prove that this does not lead to a cycle and the CPT can be converted to a FOPT.*

Each attribute $C.A$ in a K-PRM can be seen as a FOL predicate, where C is the object class name and A is the predicate name. If $C.A$ has a parent of the form $C.B$ in the relational schema, then $C.A$ has $C.B$ as parent. If $C.A$ has a parent of the form $\gamma(C.\tau.B)$ where τ is a slot chain and γ an aggregation function, then all the predicates corresponding to the attributes in the slot chain are parents of $C.A$.

*We now show that if the initial K-PRM is legal (*i.e.*, without cycles) then the RBN obtained above is also legal.*

*To prove this, we first consider the set of all certain slots in the K-PRM (*i.e.*, slots whose values are already known). In the RBN, the predicates corresponding to these are certain and these predicates can be given higher priority than any of the other predicates. The relative ordering of the predicates themselves does not matter. For the rest of the predicates we define a*

relative ordering as follows: if any predicate appears as a parent of some other predicate, the parent predicate is given a higher priority.

We have to prove that the relative ordering defined above is consistent, i.e., *if we consider the RBN graph there is no cycle among the predicates. We do this by contradiction.*

Assume that there is a cycle corresponding to predicates $R_1, \cdots, R_k$, i.e., *R_i is a parent of R_{i-1} and R_1 is a parent of R_k. Any predicate which has known values cannot appear in the cycle. If the cycle consists of only predicates which correspond to simple attributes, we can see that there will be a cycle among the attributes in the K-PRM. Therefore, the cycle must involve predicates which correspond to reference slots (that are unknown). We will assume that all the predicates in the cycle correspond to reference slots and prove that the K-PRM is illegal, leading to a contradiction. The case where some of the predicates in the cycle might correspond to simple attributes can be ruled out similarly. Let $C_i.\rho_i$ be the reference slot corresponding to the predicate R_i. Since R_i is a parent of R_{i-1} either $C_i.\rho_i$ is a parent of $C_{i-1}.\rho_{i-1}$ in the K-PRM or $C_i.\rho_i$ appears in the slot chain τ such that $C_{i-1}.\tau.B$ is a parent of $C_{i-1}.\rho_{i-1}$. In both these cases there is an edge in the K-PRM from $C_i.\rho_i$ to $C_{i-1}.\rho_{i-1}$. Hence, we can see that a cycle among the predicates corresponds to a cycle among the reference slots in the K-PRM. This implies that the K-PRM is illegal, leading to a contradiction.*

Finally, we have to prove that the CPT in a PRM can be converted to a FOPT in an RBN. It is easy to think that any row of a CPT in the K-PRM is equivalent to a FOL expression that can be expressed in terms of FOPT.

There are several advantages in using RBNs instead of K-PRMs:

- RBNs generalize K-PRMs by providing a more powerful language because based on FOL instead of frame-based systems.
- In K-PRMs, parents of a predicate are obtained by traversing chains of reference slots (*i.e.*, conjunctive expressions) in RBNs, instead, parents can be obtained via any FOL constraint. However, for the same reason, learning K-PRMs is easier.
- Modeling n-ary relationships using K-PRMs requires to break them into binary relationships, while with RBNs it is possible to model them directly choosing a proper conditional probability model (since the definition of RBNs is not constrained to the use of FOPTs).
- In K-PRMs the set of parents and the conditional models are specified using a big CPT. In RBNs, the parents and the conditional model are specified using FOPTs which can take advantage of context-specific independencies to reduce space requirements and possible speed up inference.

Bibliography

Anderson, C. R., Domingos, P., & Weld, D. S. (2002). Relational markov models and their application to adaptive web navigation. In *KDD*, pp. 143–152.

Archetti, F., Manfredotti, C., Matteuci, M., Messina, V., & Sorrenti, D. (2006a). Parallel first-order markov chain for on-line anomaly detection in traffic video surveillance. In *Crime and Security, 2006. The Institution of Engineering and Technology Conference on*, pp. 582–587.

Archetti, F., Manfredotti, C. E., Messina, V., & Sorrenti, D. G. (2006b). Foreground-to-ghost discrimination in single-difference pre-processing. In *ACIVS*, pp. 263–274.

Arulampalam, S., Maskell, S., & Gordon, N. (2002). A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, *50*, 174–188.

Berzuini, C., Best, N. G., Gilks, W. R., & Larizza, C. (1997). Dynamic conditional independence models and markov chain monte carlo methods. *Journal of the American Statistical Association*, *92*, 1403–1412.

B.Gloyer, H.Aghajan, K.-Y., & T.Kailath (1995). Video-based freeway monitoring system using recursive vehicle tracking.. pp. 173 – 180. Proceedings of SPIE.

Biswas, R., Thrun, S., & Fujimura, K. (2007). Recognizing activities with multiple cues. In *Workshop on Human Motion*, pp. 255–270.

C. Wang, S. J., & Khardon, R. (2008). First order decision diagrams for relational mdps. *JAIR*, *31*, 431–472.

C. Zhang, S-C Chen, M.-L. S. S. P. (2003). Adaptive background learning for vehicle detection and spatio-temporal tracking..

Chellappa, R., & Jain, A. (1993). *Markov random fields. Theory and application*.

Cheung, S. C., & Kamath, C. (2004). Robust techniques for background subtraction in urban traffic video. In *Video Communications and Image Processing, SPIE Electronic Imaging*, San Jose.

Collins, R., Lipton, A., & Kanade, T. (1999). A system for video surveillance and monitoring. In *American Nuclear Society 8th Internal Topical Meeting on Robotics and Remote Systems*.

Computer, J. P., Russell, S., Pearl, J., & Russell, S. (1994). Bayesian networks..

Copsey, K., & Webb, A. (2002). Bayesian networks for incorporation of contextual information in target recognition systems. In *SSPR/SPR*, pp. 709–717.

Cox, I. J., & Leonard, J. J. (1994). Modeling a dynamic environment using a bayesian multiple hypothesis approach. *Artif. Intell.*, *66*(2), 311–344.

Derek Hoiem, A. A. E., & Hebert, M. (2006). Putting objects in perspective. In *Proc. IEEE Computer Vision and Pattern Recognition (CVPR)*, Vol. 2, pp. 2137 – 2144.

Deriche, R., & Faugeras, O. D. (1990). Tracking line segments. In *ECCV*, pp. 259–268.

Domingos, P., Kok, S., Lowd, D., Poon, H., Richardson, M., Singla, P., Sumner, M., & Wang, J. (2008). Markov logic: A unifying language for structural and statistical pattern recognition. In *SSPR/SPR*, p. 3.

Doretto, G., Chiuso, A., Wu, Y. N., & Soatto, S. (2003). Dynamic textures. *International Journal of Computer Vision*, *51*(2), 91–109.

Elgammal, A. M., Harwood, D., & Davis, L. S. (2000). Non-parametric model for background subtraction. In *ECCV (2)*, pp. 751–767.

Elidan, G., Heitz, G., & Koller, D. (2006). Learning object shape: From drawings to images. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*.

Fortmann, T., Bar-Shalom, Y., & Scheffe, M. (1983). Sonar tracking of multiple targets using joint probabilistic data association. *Oceanic Engineering, IEEE Journal of*, *8*(3), 173–184.

Friedman, N., Geiger, D., & Goldszmidt, M. (1997). Bayesian network classifiers. *Machine Learning*, *29*(2-3), 131–163.

Friedman, N., Getoor, L., Koller, D., & Pfeffer, A. (1999). Learning probabilistic relational models. In *IJCAI*, pp. 1300–1309.

Friedman, N., Koller, D., & Pfeffer, A. (1998). Structured representation of complex stochastic systems. In *AAAI/IAAI*, pp. 157–164.

Getoor, L., Friedman, N., Koller, D., & Pfeffer, A. (2001). Learning probabilistic relational models. In Džeroski, S., & Lavrac, N. (Eds.), *Relational Data Mining*, pp. 307–335. Springer-Verlag.

Glesner, S., & Koller, D. (1995). Constructing flexible dynamic belief networks from first-order probalistic knowledge bases. In *ECSQARU*, pp. 217–226.

Guibas, L. J. (2002). Sensing, tracking, and reasoning with relations..

Harville, M. (2002). Stereo person tracking with adaptive plan-view statistical templates. *Image and Vision Computing*, *22*, 127–142.

Heitz, G., & Koller, D. (2008). Learning spatial context: Using stuff to find things. In *ECCV (1)*, pp. 30–43.

Hongeng, S., Nevatia, R., & Brémond, F. (2004). Video-based event recognition: activity representation and probabilistic recognition methods. *Computer Vision and Image Understanding*, *96*(2), 129–162.

Intille, S. S., & Bobick, A. F. (1999). Visual recognition of multi-agent action using binary temporal relations. In *CVPR*, pp. 1056–.

Isard, M., & Blake, A. (1998). A mixed-state condensation tracker with automatic model-switching. In *ICCV*, pp. 107–112.

Isard, M., & MacCormick, J. (2001). Bramble: A bayesian multiple-blob tracker. In *ICCV*, pp. 34–41.

Ivanov, Y. A., & Bobick, A. F. (2000). Recognition of visual activities and interactions by stochastic parsing. *IEEE Trans. Pattern Anal. Mach. Intell.*, *22*(8), 852–872.

Jaeger, M. (1997). Relational bayesian networks. In *UAI*, pp. 266–273.

Jansen, R., Yu, H., Greenbaum, D., Kluger, Y., Krogan, N. J., Chung, S., Emili, A., Snyder, M., Greenblatt, J. F., & Gerstein, M. (2003). A bayesian networks approach for predicting protein-protein interactions from genomic data.. *Science*, *302*(5644), 449–453.

Kersting, K., & Raedt, L. D. (2000). Bayesian logic programs. In *ILP Work-in-progress reports*.

Kersting, K., & Raiko, T. (2005). "say em" for selecting probabilistic models for logical sequences. In *UAI*, pp. 300–307.

Khan, Z., Balch, T. R., & Dellaert, F. (2004). An mcmc-based particle filter for tracking multiple interacting targets. In *ECCV (4)*, pp. 279–290.

Koller, D. (1999). Probabilistic relational models. In *ILP*, pp. 3–13.

Laptev, I., Caputo, B., Schüldt, C., & Lindeberg, T. (2007). Local velocity-adapted motion events for spatio-temporal recognition. *Computer Vision and Image Understanding*, *108*(3), 207–229.

Liu, J. S., & Chen, R. (1998). Sequential monte carlo methods for dynamic systems. *Journal of the American Statistical Association*, *93*, 1032–1044.

Luong, Q.-T., Weber, J., Koller, D., & Malik, J. (1995). An integrated stereo-based approach to automatic vehicle guidance. In *ICCV*, pp. 52–57.

MacCormick, J., & Blake, A. (2000). A probabilistic exclusion principle for tracking multiple objects. *International Journal of Computer Vision*, *39*(1), 57–71.

McKenna, S. J., & Nait-Charif, H. (2007). Tracking human motion using auxiliary particle filters and iterated likelihood weighting. *Image Vision Comput.*, *25*(6), 852–862.

McKenna, S., Jabri, S., Duric, Z., & Wechsler, H. (2000). Tracking interacting people. In *4th Int. Conf. on Automatic Face and Gesture Recognition*, pp. 384–353, Grenoble, France.

Migliore, D. A., Matteucci, M., & Naccari, M. (2006). A revaluation of frame difference in fast and robust motion detection. In *VSSN '06: Proceedings of the 4th ACM international workshop on Video surveillance and sensor networks*, pp. 215–218, New York, NY, USA. ACM.

Milch, B. (2006). *Probabilistic Models with Unknown Objects.* Ph.D. thesis, Computer Science Division, University of California, Berkeley.

Minker, J., & Seipel, D. (2002). Disjunctive logic programming: A survey and assessment. In *Computational Logic: Logic Programming and Beyond*, pp. 472–511.

Monnet, A., Mittal, A., Paragios, N., & Ramesh, V. (2003). Background modeling and subtraction of dynamic scenes. In *ICCV*, pp. 1305–1312.

Moore, D. J., & Essa, I. A. (2002). Recognizing multitasked activities from video using stochastic context-free grammar. In *AAAI/IAAI*, pp. 770–776.

Murphy, K. (2002). *Dynamic Bayesian Networks: Representation, Inference and Learning.* Ph.D. thesis, Computer Science Division, University of California, Berkeley.

Needham, C. J., & Boyle, R. D. (2001). Tracking multiple sport players through occlusion, congestion and scale..

Okuma, K., Taleghani, A., de Freitas, N., Little, J. J., & Lowe, D. G. (2004). A boosted particle filter: Multitarget detection and tracking. In *ECCV (1)*, pp. 28–39.

Pearl, J. (1986). Probabilistic reasoning using graphs. In *IPMU*, pp. 200–202.

Rasmussen, C., & Hager, G. D. (2001). Probabilistic data association methods for tracking complex visual objects. *IEEE Trans. Pattern Anal. Mach. Intell.*, *23*(6), 560–576.

R.Cucchiara, M., & A.Prati (2003). Detecting moving objects, ghost, and shadows in video streams.. pp. 1337 – 1342. IEEE transactions on pattern analysis and machine Intelligence.

R.Cutler, & L.Davis (1998). View-based detection.. pp. 495–500, Brisbone, Australia. Proceedings Fourteenth International Conference on Pattern Recognition.

Reid, D. (1979). An algorithm for tracking multiple targets. *Automatic Control, IEEE Transactions on*, *24*(6), 843–854.

Russell, S., & Norvig, P. (2003). *Artificial Intelligence: A Modern Approach* (2nd edition edition). Prentice-Hall, Englewood Cliffs, NJ.

Russell, S. J., & Norvig, P. (2002). *Artificial Intelligence: A Modern Approach* (Second edition). Prentice Hall.

Ryoo, M. S., & Aggarwal, J. K. (2006). Recognition of composite human activities through context-free grammar based representation. In *CVPR (2)*, pp. 1709–1718.

Sedgewick, R. (2001). *Algorithms in c, part 5: graph algorithms, third edition*. Addison-Wesley Professional.

Spagnolo, P., Leo, M., D'Orazio, T., Caroppo, A., & Martiriggiano, T. (2006). An energy-based background modelling algorithm for motion detection. In *ICINCO-RA*, pp. 378–383.

Stauffer, C., & Grimson, W. E. L. (1999). Adaptive background mixture models for real-time tracking. In *CVPR*, pp. 2246–2252.

Studeny, M., & Bouckaert, R. R. (1998). On chain graph models for description of conditional independence structures. *Ann. Statist*, *26*, 1434–1495.

Thrun, S., Burgard, W., & Fox, D. (2005). *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press.

Tovinkere, V., & Qian, R. J. (2001). Detecting semantic events in soccer games: Towards a complete solution. In *ICME*.

Tran, S. D., & Davis, L. S. (2008). Event modeling and recognition using markov logic networks. In *ECCV (2)*, pp. 610–623.

wu Liao, X., Wan, T., & Li, Y. (2008). A bayesian network model under group decision making for evaluating it outsourcing risk. *Risk Management and Engineering Management, 2008 International Conference on*, *0*, 559–564.

Yan Ke, R. S., & Hebert, M. (2007). Event detection in crowded videos. In *IEEE International Conference on Computer Vision*.

Yoshinari, K., & Michihito, M. (1996). A human motion estimation method using 3-successive video frames. In *Proc. of Int.Conf. on Virtual Systems and Multimedia (GIFU)*, pp. 135–140.

Zhao, T., Nevatia, R., & Wu, B. (2008). Segmentation and tracking of multiple humans in crowded environments. *IEEE Trans. Pattern Anal. Mach. Intell.*, *30*(7), 1198–1211.

Acknowledgements

Prima di partire per un lungo viaggio
devi portare con te la voglia di non tornare piu'.

Irene Grandi from the song “Prima di partire per un lungo viaggio”.

Molti (maligni!) penseranno che la citazione si riferisce al mio non voler mai piu' tornare in Bicocca ... Ebbene no! Il dottorato e' stato un viaggio e come e' giusto, in quanto viaggio, l'ho percorso “gustandomelo”.

Il dottorato e' stata un'esperienza molto costruttiva e la crescita personale e di ricercatore che ne ho derivato non e' stata tutta merito mio ma, soprattutto, delle persone che mi hanno indirizzato e accompagnato in questo cammino e che ora mi piacerebbe ringraziare una per una.

Prima tra tutte la mia relatrice la professoressa Messina (“La Prof”). Ha saputo guidarmi ma anche consigliarmi sulle strade da prendere. Ha saputo farmi ragionare e non l'ho sempre subito ascoltata ma con il tempo ho scoperto che ha sempre ragione!

“A braccetto” con La Prof non puo' mancare il professor Archetti. Sempre presente anche se spesso non fisicamente. Tante delle idee presenti in questa tesi sono nate dalla discussione con lui, da sue proposte - a dire il vero spesso un po' strampalate- e dalla sua voglia di “provare”. Sua sorella, prof, l'ha mai ringraziata in una tesi?

Nel professor Sorrenti ho sempre trovato un confidente e un amico, ma anche una persona sul cui giudizio potevo contare e della cui opinione potevo fidarmi. Grazie!

I tre anni di dottorato sono stati tre anni pieni di amici incontrati lungo questo cammino: Dario, Yuri (che ha consegnato la mia domanda di tesi, grazie!) Simone, Sergio, i due Daniele (che poi ho ritrovato a Toronto, Sara (Un amico e' per sempre!), Elena, Daniele (che per primo mi ha spiegato particle filter), Axel (che ora iniziera' questo ”viaggio”), Paolo, Davide e tanti altri con i quali ho condiviso tanti momenti di scherzi, di pausa, di chiacchera e anche a volte di ..., grazie a tutti ragazzi!

Ma piu' di tutti devo ringraziare le mie due colleghe e compagne di (s)ventura: Elisabetta e Ilaria. Ne abbiamo passate tante insieme, lavorativamente parlando e non solo, ma i miei ringraziamenti sono soprattutto per questo ultimo anno e mezzo in cui, se pur lontana, mi avete sempre e comunque fatto sentire una di voi. Grazie!

Il MIND (ex LIFE) e' stato un bellissimo gruppo nel quale lavorare (e per ognuno degli studenti passati di li' conservo un ricordo particolare) ma c'e' un altro gruppo che mi ha accolta e fatta sentire a casa in quest ultimo periodo: COGS alla UofT. Per farmi capire anche da loro cambio lingua per un attimo.

At UofT I found a great professor, Craig, who has been always (almost) available for suggestions about which was the right thing to do and who introduced me to my Co-advisor, David, the long meetings with whom made a lot for this Thesis. But I also found a lot of colleagues and friends always there when I need something. I remember the long lunches with Laurent, that made me feeling like at home, the small talks with Kevin who support me a lot and the strange conversations with Tyler. But I also keep in my heart the first encouragement received at UofT by Darius (”You have your application and it is interesting enough!”) who is having a baby girl!

Ma io fino a qui non sarei arrivata senza l’educazione e l’esempio che ho ricevuto dai miei genitori (e so che ora mia madre iniziera’ a piangere): mi hanno sempre dato la possibilita’ di perseguire i miei sogni anche se questo ha voluto dire per loro ”stringere la cinghia” o vedermi andare dall’altra parte del mondo. Meno male che c’e’ sempre stato il mio fratellino a tener loro compagnia, grazie Davide (non solo per questo).

Sono certa che la mia nonna conservera’ copia di questa tesi nel salotto di casa sua. Grazie nonnina, per i peperoni, i broccoletti ma anche per l’orgoglio con il quale sempre mi guardi.

Visto che abbiamo fatto trenta facciamo trentuno, e lasciatemi ringraziare tutti gli amici di Milano che mi hanno sempre accolta a braccia aperte ogni volta che sono tornata in Italia e per tutti loro Vera, una cara amica, che probabilmente dovra’ stampare questa tesi, grazie!

Ho dedicato a lui la prima riga che ho scritto di questa tesi, voglio dedicargli anche l’ultima. Ho trovato in te un amico e un confidente prima che un marito. Sai benissimo che senza di te nulla in questi tre anni sarebbe stato possibile e ti ringrazio, per tutte le opportunita’, gli sproni, gli aiuti e tutto l’amore che ci hai messo. E, non ultimo, per i sogni che ora perseguiamo insieme ... Cavolo! ora sto piangendo anche io ....