

Estimation de mouvement des obstacles mobiles: une approche statistique

Dizan Alejandro Vasquez Govea

▶ **To cite this version:**

Dizan Alejandro Vasquez Govea. Estimation de mouvement des obstacles mobiles: une approche statistique. [Travaux universitaires] 2003. inria-00182078

HAL Id: inria-00182078
https://inria.hal.science/inria-00182078

Submitted on 7 Nov 2007

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Insititut National Polytechnique de Grenoble

Diplome d'Études Approfondies
Spécialité: Imagerie, Vision et Robotique

Estimation de Mouvement des Obstacles Mobiles: Un Approche Statistique

(Motion Estimation for Mobile Obstacles: A Statistical Approach)

Dizan Vasquez

Date de Soutenance: 11 Septembre 2003

Encadrant: Dr. Thierry Fraichard

Projet préparé au sein du projet CYBERMOVE,
au Laboratoire GRAphique, VIsion, Robotique
à Inria Rhône-Alpes
ZIRST, 655 av. de l'Europe
38330 Montbonnot St. Martin
FRANCE

Contents

1 Introduction **1**
- 1.1 Problem Overview . . . 1
- 1.2 Objectives . . . 2
- 1.3 Contributions . . . 2
- 1.4 Structure of the Document . . . 3

2 Related Works **4**
- 2.1 Motion and Motion Models . . . 4
- 2.2 Generative Motion Models . . . 5
  - 2.2.1 Parametrical Models . . . 5
  - 2.2.2 Random Models . . . 6
  - 2.2.3 Intentional Models . . . 6
  - 2.2.4 Estimation . . . 6
  - 2.2.5 Related approaches . . . 7
  - 2.2.6 Analysis . . . 7
- 2.3 Descriptive Motion Models . . . 8
  - 2.3.1 Grid Models . . . 9
  - 2.3.2 Cluster based Models . . . 11
  - 2.3.3 Other Models . . . 14
- 2.4 Data Clustering . . . 14
  - 2.4.1 Definition of the problem . . . 14
  - 2.4.2 Similarity Measures . . . 15
  - 2.4.3 Clustering Techniques . . . 16
  - 2.4.4 Hierarchical Clustering . . . 16
  - 2.4.5 Partitional Clustering . . . 18
  - 2.4.6 Fuzzy Clustering . . . 20

3 Proposed Approach **21**
- 3.1 Learning Motion Patterns . . . 22
  - 3.1.1 Similarity Measure . . . 22
  - 3.1.2 Clustering Algorithm . . . 22
  - 3.1.3 Calculating Class Statistical Parameters . . . 25
- 3.2 Estimating Trajectories . . . 26
  - 3.2.1 Partial Distance . . . 26
  - 3.2.2 Calculating Likelihood . . . 26

3.2.3 Cluster Probability 26
3.3 Analysis 27

4 Experimental Results **28**
4.1 Generation of the Training Set 28
4.2 Estimation of K and clustering 29
4.3 Generation of the Test Set 30
4.4 Benchmarking 31
4.5 Graphical Representation of Estimation 33
4.6 Analysis 33

5 Conclusions **37**
5.1 Summary and Contributions 37
5.2 Perspectives and Future Work 38

List of Figures

2.1 Raw trajectories of people in an office environment 9
2.2 Statistical Grid for example data 10
2.3 Resulting Clusters for example data 13
2.4 Clustering Techniques 16
2.5 Example Clusters (taken from [Jain 99]) 17
2.6 Corresponding Dendogram (taken from [Jain 99]) 17
2.7 Distance Between Clusters 18

4.1 The INRIA entry Hall 29
4.2 Examples of generated clusters for the three algorithms. 31
4.3 Benchmark Results 32
4.4 Trajectory Estimation for Different Instants 33
4.5 Results by Number of Clusters 34
4.6 Clustering times 35
4.7 The Camera View 36
4.8 Some Clusters obtained using real data 36

List of Tables

4.1 Clustering Results 30

Abstract

The main objective of this work is to search for a motion estimation technique for vehicles and pedestrians having the following properties:

a) It should produce estimations with a long Time Horizon.

b) It should be as general as possible and work with many different kinds of objects.

c) It should be fast enough to give estimations in real time.

We propose a motion estimation technique based on pairwise clustering which.verifies the required properties.

We have implemented and tested our approach, comparing it with a technique that we consider to represent the state of the art in clustering techniques. In order to perform the comparison, we propose a benchmark that can be used to test other motion estimation techniques.

Keywords: Motion Estimation, Pairwise Clustering, Trajectory Clustering, Trajectory Prediction

Chapter 1

Introduction

1.1 Problem Overview

In order to survive, most animals and every intelligent being have to be able to navigate purposefully within the environment they inhabit. One of the greatest challenges of navigating in a real environment is to deal with the obstacles populating it. This challenge is further complicated by the fact that many of those obstacles are not static: they are involved in activities and their situation evolves over time. In order to successfully interact with moving obstacles, it is necessary to predict their movements.

Motion modeling and estimation is a research area which has applications in many different fields, ranging from video surveillance [Koller 94] to robot navigation [Kyriakopoulos 92]. Many techniques have been proposed in the literature, however, most of the existing proposals share an important drawback: the estimations they provide are only sound during a short time interval. We say that these techniques have a short *Time Horizon*.

If we can find techniques having a longer Time Horizon, we would have more trustful navigation schedules. Hence, it is very important to develop motion models for obstacles that allow to estimate trajectories as far as possible in the future and with a maximum of certainty. This problem, however, sets significant difficulties: a dynamic environment is often populated by many kinds of moving obstacles (cars, pedestrians, industrial robots, etc.) having very different motions resulting from a wide variety of factors (kinematic constraints, dynamic constraints, intentionality, etc.) In order to estimate their motion, it would be necessary to have a model which takes into account the features of each kind of obstacle. The problem with such a model is that it would be utterly complex. The preferred alternative is to have many different models, one for each kind of obstacle, unfortunately, the development of those specialized models implies previous domain knowledge, and its reusability is severely limited due to its

specificity.

In this context, we would like to have motion estimation techniques having a good compromise between generality and complexity so that they can be applied to a wide spectrum of situations while still having long Time Horizons. We believe in the possibility of such a technique, and propose in this document an approach that, we hope, constitutes a step towards it.

1.2 Objectives

This work aims at finding a motion estimation technique that can be implemented in an actual project: ParkView. The goal of ParkView is to develop autonomous robot vehicles able to evolve in a parking lot. In order to attain this autonomy, a mechanism is needed that allows to predict motion of obstacles which populate the car's environment, notably other cars and pedestrians. Given the characteristics of the project, the resulting technique should verify the following properties:

1. It should produce estimations with a long Time Horizon.
2. It should be as general as possible and work with many different kinds of objects.
3. It should be fast enough to give estimations in real time.

1.3 Contributions

As a result of our work we present the following contributions:

- We propose a cluster-based motion estimation technique which verifies the properties mentioned in section 1.2. With this technique:
  - We propose a dissimilarity metric that allows the use of pairwise clustering algorithms. To our knowledge, this is a novel approach to trajectory clustering.
  - We propose a way to calculate the number of clusters to be used in the clustering solution.
  - We present an algorithm to calculate the mean value of a cluster of trajectories and then use this mean value in order to calculate the corresponding standard deviation.
  - We propose a way to estimate the likelihood that a partially observed trajectory belongs to a given cluster.

- We implemented and tested our approach using simulated data.
- Finally, we compare our results against those of a technique [Bennewitz 02] that we consider as representing the state of the art in this kind of estimators. In order to perform this comparison we propose a benchmark that may be useful for evaluating other techniques.

1.4 Structure of the Document

The introduction explained in a loose way some of the reasons behind the interest on motion estimation. It also defined the main goal of our work, as well as three required properties of the resulting technique. Finally, it succinctly describes our contributions. The remainder of this document is organized as follows.

Chapter 2 The second chapter reviews basic notation and concepts which are used throughout the document. A classification of existing motion models in two categories (Descriptive and Generative) is proposed based on the underlying representation. Both categories are briefly explained and discussed. Finally, given that our approach strongly relies on data clustering, an overview of this domain is given.

Chapter 3 In the third chapter, we explain the details of the approach we propose. We describe the learning stage, which is based on pairwise clustering by means of a measure of dissimilarity between trajectories. We propose a way of estimating the number K of clusters of a clustering instance. We explain how we calculate statistical parameters for each cluster and how we use these values in order to estimate motion. Finally, we discuss the theoretical advantages and drawbacks of our approach.

Chapter 4 In the fourth chapter, we present an overview of the tests we have performed. We describe the dataset generation process. After that, we comment the results we obtained using the different clustering algorithms on the dataset. Finally, we explain the benchmarking process and discuss the results we obtained.

Chapter 5 The last chapter summarizes the main issues of this work, gives directions to future work and discusses extensions and improvements of the models and algorithms presented.

Chapter 2

Related Works

Motion has always intrigued humanity. Zeno's famous paradoxes testify the fascination exerted by motion in humanity since the ancient times. In all these years, a huge body of work has been developed around the study of moving objects. This chapter will present an overview of the techniques that have been developed in order to model and predict an object's motion; It will also provide the reader with a general discussion of data clustering, which constitutes the hearth of the approach we propose.

2.1 Motion and Motion Models

Before going any further, it is important to define motion, what it is? To us, and throughout this document, motion is defined as the change in position and / or orientation of the parts of an object that takes place as time passes.

At any given moment t an object is said to be in a certain *configuration*, which is a mathematical specification of the position and orientation of the different parts of the object with respect to a fixed frame. For example, in the case of a single rigid object which is able to move on a plane, a configuration q is specified using a vector $q = [x, y, \theta]$. We call *configuration space* (represented by the symbol $\mathbb{C}$) the space defined by all the possible configurations for an object in the absence of obstacles, so that every configuration is represented by a point in $\mathbb{C}$. In the case of a rigid object on the plane, the configuration space is 3-dimensional.

In general, there are two ways of modeling motion: The first one, often called *descriptive*, or *phenomenological* models, represents a particular motion instance as a succession of configurations in a way very similar to a series of photos. Usually, a descriptive model is represented as list of configurations $q = \{q_0, q_1, ..., q_T\}$ where each $q_t \in \mathbb{C}$ represents the configuration of the object in the instant t and T represents the total duration of the motion.

The other modeling approach tries to model motion as a function of a given number of parameters, thus, the same model can be used to represent multiple motion instances. A very important factor in this modeling approach is the introduction of time t as a parameter; this allows to model the evolution of dynamic systems and, by changing the value of t, predicting future configurations, or estimating configurations previous to the first one that was observed. Due to this capability of representing "unseen" configurations, these models are called *generative*. A generative model is a function $q(t)$ of time $q : [0, T] \rightarrow \mathbb{C}$ which outputs the corresponding configuration $q(t)$ for a given instant t.

The following two sections provide a deeper, but far from exhaustive, description of both modeling approaches.

2.2 Generative Motion Models

Generative motion models are based on physical laws. They work by calculating acceleration $\ddot{q}(t)$ as a function of time and then integrating once to calculate velocity $\dot{q}(t)$ and twice to get the actual configuration $q(t)$ the whole process is outlined in the following equations:

$$\dot{q}(t) = \dot{q}(0) + \int_0^T \ddot{q}(\tau)d\tau \tag{2.1}$$

$$q(t) = q(0) + \int_0^T \dot{q}(\tau)d\tau \tag{2.2}$$

What distinguishes different Generative Motion Models is the way they define $\ddot{q}$. In the rest of his section, we define a classification of generative motion models based on [Zhu 90] A particular approach can correspond to just one category or a combination of them.

2.2.1 Parametrical Models

In parametrical models acceleration is considered to follow a known form:

$$\ddot{q}(t) = \varphi_p(t) \tag{2.3}$$

Where $\varphi_p(t)$ is a function and p is the constant parameter vector for φ. For example, if $\varphi_p(t)$ is a first degree polynomial of the form $\varphi_p(t) = c_1 t + c_2 = 0$ then $p = [c_1, c_2]$ So, if we know the form of φ and the parameters p, as well as the velocity $\dot{q}(0)$ and configuration $q(0)$ in a moment $t = 0$, we can calculate the velocity and

configuration after a given interval T. applying expressions 2.1 and 2.2. An example of a parametrical model is [Kalman 60] where acceleration is supposed to be constant.

2.2.2 Random Models

Random Models assume that changes in acceleration are determined by a given probability distribution, such as Gaussian, Uniform or Poisson.

$$\ddot{q}(t) = \omega(t) \tag{2.4}$$

Where $\omega(t)$ is function which generates a vector using a given probability distribution. An example can be seen in [Strehl 98] where the probability distribution for $\ddot{q}$ is iteratively calculated and $\omega(t)$ is generated taking the acceleration with maximum probability.

2.2.3 Intentional Models

In intentional models, objects move along a scheduled path, usually attempting to accomplish one or more goals. The obstacle can react to the environment and even perform collision avoidance. For this model, we characterize acceleration as:

$$\ddot{q}(t) = e(t) \tag{2.5}$$

Where $e(t)$ is the acceleration chosen by the obstacle's intentional strategy. We can wonder about the nature of the difference between expression 2.4 and 2.5 which have identical form. If we take a closer look, they are very different: $\omega(t)$ is just a vector (depending on nothing but the underlying probability distribution and its parameters), while $e(t)$ is usually a very complex function strongly dependent on the actual configuration of the world, the internal state of the object and the navigation strategy itself. Moreover, in real life, access to the particular motion strategy (and thus to the form of e) is frequently impossible. And estimation is a very difficult task even when the motion strategy is known. As a matter of fact, motion planning can be viewed as an example of an intentional model (see [Latombe 91]).

2.2.4 Estimation

Theoretically, if we have an appropriate motion model and if we know the value of all its parameters as well as its actual velocity and configuration, then future configuration prediction is straightforward. Unfortunately, in most real-life cases, some or all of those parameters are unknown and have to be estimated. The same applies to velocity and configuration.

The process of estimation uses a set of observations, gathered through some kind of measuring device or sensor, in order to approximate the values of the unknown variables for the given motion model.

2.2.5 Related approaches

An estimation technique is constituted of a motion model, as well as state and parameter estimation processes. The vast majority of approaches found in literature are based in Physics Laws (uniform accelerated motion) as motion models and some variation of Kalman Filter or Extended Kalman Filter estimation processes.

The Kalman Filter [Kalman 60], is an optimal state estimator. Its origin lies in the problem of estimating satellite and aircraft trajectories using radar data. In this problem, configuration (x, y, z) and velocity $(\dot{x}, \dot{y}, \dot{z})$ should be estimated from the observation of two angles: site and azimuth (θ, φ). Future state prediction can be performed using the estimated values of configuration and velocity.

A dynamic programming approach to estimation is proposed in [Larson 66], the paper shows that the technique can be reduced to a Kalman Filter. A Kalman Filter improved using Montecarlo Simulation is applied in [Chang 77] to robot navigation.

More recently, recursive formulations of the Extended Kalman Filter have been proposed by [Blostein 95]. A two-layer Extended Kalman Filter is applied to tracking moving targets in [Kawase 98].

Other approaches for state and parameter estimation for generative models have been used. The application of fuzzy logic and neural networks to state estimation is studied in [Patt 98]. Local avoidance using an optimization function is presented in [Chien 89]. Iterated wavelet decomposition is introduced in [Leduc 98]. [Knudsen 92] proposes the use of a Fourier Transform to find parameters using information in the frequency domain.

2.2.6 Analysis

Generative models are founded on Physics, thus, one of their main strengths is that they can be explained using well known general principles. Their predictions are very precise, given a complete knowledge of the underlying expressions and good present state estimation. Moreover, they take into account kinematic and dynamic constraints so they always produce estimations which are physically attainable.

These models suffer however from an important weakness in predicting motion for obstacles engaged in intentional behaviors: these behaviors produce motion that is composed of many patterns. For example, let us imagine a car moving in a city: while it is in a street having no intersections, its motion can be predicted using simple motion

models. But it is very difficult to find an generative model capable of predicting its behavior when it arrives to a crossroads. This situation leads to short Time Horizon prediction capability.

Another problem with generative models is that they assume that motion can be predicted using only the current state. This assumption does not always hold. Let us get back to our example: if we only know that the car is in a crossroad, we don't have a way to predict if it is going to do a left-turn. However, if we know that he has turned left on the last three crossroads, we can assume that the probability of another left-turn is very low because, otherwise, it will arrive to the same place. In this sense, we say that most generative approaches work as a first order Markov Model.

One advantage of these models is that, their predictions are still valid (for a short amount of time) when applied to unexpected or atypical trajectories.

2.3 Descriptive Motion Models

Descriptive motion models describe a particular motion instance at successive instants. Approaches based on this kind of models depart from a fundamental assumption:

"For a given area, motion can be described in terms of typical motion patterns: objects interact with the environment in well established ways that can be observed consistently".

This assumption has two very important implications: Motion Patterns depend on the work space, and they can be observed consistently. This leads to a general approach to estimation using descriptive models or "patterns" as they are frequently called in the literature: Observe the workspace in order to find the representative motion patterns, and then, use these representative patterns to explain observed motion. Hence, most descriptive motion estimation techniques have three components:

1. **Data Capture**. This component is responsible for converting sensor input (usually coming from a set of cameras or laser range finders) into numeric information representing the observed trajectories. Output data is in the form of a series of trajectories D, often called *training data*, where individual trajectories d^i are represented by consecutive observations $d_i = \{d_i^1, d_i^t, ..., d_i^{T_i}\}$ where T_i is the total number of samples for the trajectory, $1 \leqslant t \leqslant T_i$ orientation is not used in most approaches, and time information can be also omitted if d_i is sampled at regular intervals or if we are only interested in the trajectories' shape.

2. **Learning Algorithm**. The learning algorithm uses the output of the capture system to generate a data structure of statistical information which is particular to each technique.

3. **Estimation Algorithm**. The estimation process tries to match an observed partial trajectory $x_{partial}$ with the data structure and then uses the result to estimate future motion. The details depend on the particular technique being used.

Data capture is basically the same regardless of the estimation technique. What really distinguishes each approach are the learning and the estimation algorithms.

In the following sections, we are going to describe two kinds of techniques that cover most of the approaches in the area: grid models and cluster based models. In order to simplify the explanation, we will use a fictitious data set consisting of trajectories of people moving in a simplified office environment (fig. 2.1). In order to increase readability, similar trajectories are supposed to go in the same direction. We are going to represent each trajectory d_i by a vector of configurations $d_i^t = \{x_i^t, y_i^t\}$ sampled at regular intervals.

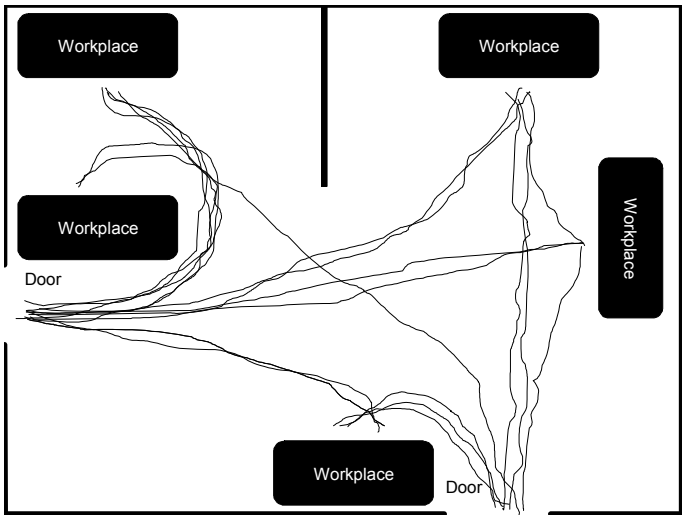


Figure 2.1: Raw trajectories of people in an office environment

2.3.1 Grid Models

This term groups a number of proposals loosely based on Occupancy Grids [Elfes 89]. The main idea is to divide the working space using a tessellation (usually rectangular cells) and then, to calculate the probability for each of the resulting cells of being occupied. We are going to explain the technique proposed in [Kruse 96] which is representative of this kind of approach.

Learning Algorithm

The workspace is subdivided into rectangular cells c. For each of these cells we are going to calculate probabilities regarding both presence and direction of motion. We compute the probability of an object moving in one of eight directions $P_\alpha(c)$ where $\alpha \in \{0, \frac{1}{4}\pi, ..., \frac{7}{4}\pi\}$. We compute also the probability of an object resting in the same grid for a while $P_{partly}(c)$. Having calculated that, the occupancy probability for the cell $P_{occ}(c)$ is given by the probability of resting in the same place plus the probability of moving on each of the eight directions: $P_{occ}(c) = P_{partly}(c) + \sum_\alpha P_\alpha(c)$.

To calculate $P_\alpha(c)$ and $P_{partly}(c)$ we proceed as follows: we observe the environment during a given time T_o at fixed intervals Δt. For each cycle we update the values for all the occupied cells by adding $\frac{\Delta t}{T_o}$ to $P_{partly}(c)$ or one of the $P_\alpha(c)$ (depending on the object's motion) and then calculating $P_{occ}(c)$.

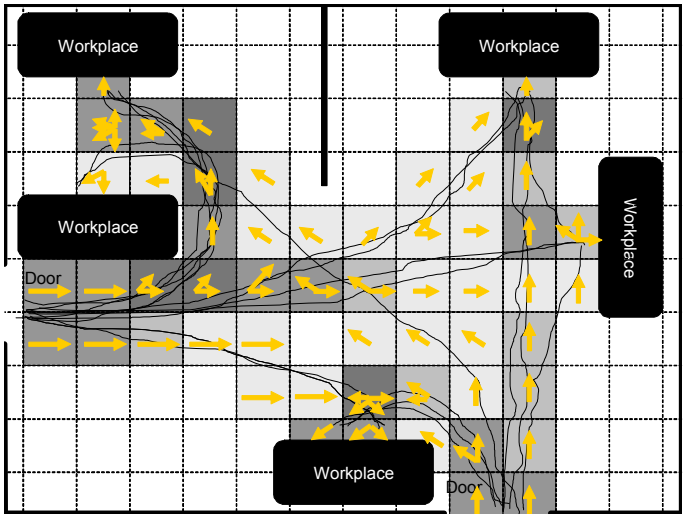


Figure 2.2: Statistical Grid for example data

A representation of the resulting statistical grid for the example data is shown in fig. 2.2. Darker cells have a higher $P_{occ}(c)$ than lighter ones. $P_\alpha(c)$ is represented by arrows.

Estimation Algorithm

In the proposal being explained, there is no explicit estimation process. The statistical grid is used to compute a collision probability field (cp-field) which is directly used in motion planning to calculate the path with minimum collision probability using, a technique similar to that of gradient descent commonly used for potential fields.

Related Approaches

There exist many approaches which resembles the one presented here: In [Tadokoro 95], the statistical grid is manually introduced by a human and probabilities of occupation of neighboring cells in the future are computed using Markov Chains to produce near future estimations. The work of [Tanaka 02] use very similar mechanics but it discretizes the space using a set of "feature points" instead of a grid.

Analysis

One important characteristic of most grid-based models is that they don't produce "real" descriptive motion models: we obtain only a decomposition of the space with no explicit information about the paths themselves. This can be considered a drawback, nonetheless, grid approaches are useful in the sense that they provide information about regions and this information can be used even in the absence of sensorial input. This kind of models are useful, for example for motion preplanning; where we look for a general schedule to be followed before knowing actual environment configuration and we want to avoid crowded regions.

The main advantage of grid models is its simplicity. These approaches are also capable of producing estimates of unobserved trajectories.

2.3.2 Cluster based Models

This family of approaches tries to group full trajectories in *clusters* which correspond to typical patterns. The main difficulty in this approach is that, although there is an extensive number of clustering algorithms (see [Kaufman 89] and [Jain 99]), most of them operate on vector spaces or need some kind of similarity measure and is not always easy to reformulate them in order to perform trajectory clustering. Here, we are going to discuss the approach followed by [Bennewitz 02].

Learning Algorithm

Both patterns and trajectories are represented as sequences of fixed length T. As actual trajectory lengths may vary, they are normalized as follows: We choose T as the length of the longest trajectory. Trajectories of length $T' < T$ are extended to length T by appending the final configuration for $T - T'$ times.

The approach assumes that an obstacle is engaged in one of K motion patterns. Each pattern, denoted by θ_k with $1 \leq k \leq K$ is represented by T probability distributions $P(d_i^t \mid \theta_k^t)$. Those distributions represent the probability that an obstacle is at configuration d_i^t at time t given that it is engaged in motion pattern θ_k.

Calculating the likelihood of a trajectory under the k-th motion model is straightforward:

$$P(d_i \mid \theta_k) = \prod_{t=1}^{T} P(d_i^t \mid \theta_i^t) \tag{2.6}$$

In theory, to perform the clustering for a given training dataset D we only need to apply the expression 2.6 to calculate the likelihood of each one of the trajectories in D under each motion model and then assigning them to the most likely one. Unfortunately, the parameters of the K models are unknown, hence, a way to estimate them is needed: Let's assume that all the $P(d \mid \theta_k^t)$ functions can be represented as multidimensional gaussians having a fixed global known standard deviation σ, so that the only unknown parameters of our θ_k models are the means of those gaussians $\theta_k = \{\mu_k^1, ..., \mu_k^T\}$. The approach aims to find a maximum likelihood hypothesis $h = \{\theta_1, ..., \theta_K\}$ which maximizes $P(D \mid h)$. A trajectory d_i in D is assumed to belong to a single cluster; this fact is represented by a set of correspondence variables $c_i = \{c_i^1, ..., c_i^K\}$ for each d_i where $c_i^k = 1$ if d_i belongs to cluster k and $c_i^k = 0$ otherwise.

The values of c_i are not included in the training data D, they are hidden variables. It seems adequate to use the EM (Expectation-Maximization) algorithm which is a widely used approach to learning hidden variables. We are going to briefly describe the algorithm, without going into details. We refer the interested reader to [Mitchell 97], for an excellent introduction to the subject, or [McLachlan 97] for a comprehensive resource. EM is an iterative algorithm that searches for a maximum likelihood hypothesis by repeating two steps:

1. **Expectation**. Calculate the expected value $E[c_i^k]$ of each hidden variable c_i^k, assuming the current hypothesis h.

2. **Maximization**. Calculate a new maximum likelihood hypothesis $h' = \{\theta'_1, ..., \theta'_K\}$, assuming that the value taken by each hidden variable c_i^k is its expected value $E[c_i^k]$ calculated in step 1. Then replace the hypothesis $h = h'$ and iterate.

In figure 2.3 we can see a representation of the resulting clusters for the example data. The black lines are the values of μ, and the gray areas represent σ.

Estimation Algorithm

The estimation algorithm is simple: Expression 2.6 is applied to all K parameters θ, replacing T with the length of the partial trajectory being predicted. Then, the motion model having maximal probability is selected and output as an estimation.

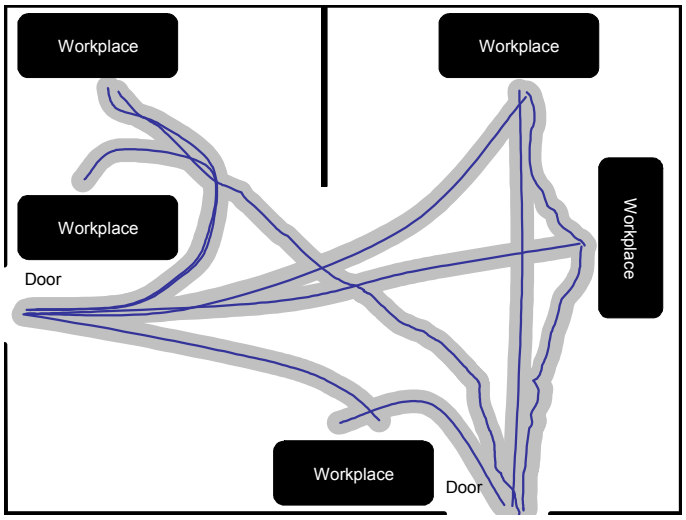


Figure 2.3: Resulting Clusters for example data

One advantage of using EM is that the mean values found in the final h are at the same time a representation of the cluster itself, which can be directly used to represent the predicted trajectory.

Related Approaches

An alternative proposal [Gaffney 99] approximates trajectories using polynomials and then applying EM to find the polynomial coefficients. The same paper proposes also the use of EM to estimate trajectories using series of locally weighted functions.

Another interesting approach is presented in [Kruse 97]: they suggest representing the trajectories by joined line segments obtained through recursive subdivision. Clustering is then performed on the segments instead of the whole trajectories. The resulting clusters are used for motion preplanning and to choose between reactive avoidance behaviors.

Analysis

In general, cluster based models are able to produce long-term, accurate predictions of trajectories that correspond to typical patterns. Even when these techniques are not able to uniquely determine the corresponding motion pattern, they are able to produce short-term estimates based on closest matching patterns. The main drawback of these techniques it that they fail to predict trajectories that don't correspond to known motion patterns, this severely reduces their applicability to non structured environments where atypical trajectories are more likely to appear. On the other

hand, as they impose very few restrictions on the form of the trajectories, they are able to predict very complicated patterns. This makes them very useful in structured environments, where we can find many common, but very complex trajectories.

One of the greatest strenghts of clustering based approaches is that they take into account not only the present state, but all information regarding past behavior, in this sense, they act as Markov Models of order $T_{observed}$ which is the number of configurations in the observed trajectory.

2.3.3 Other Models

Other techniques are somewhat difficult to classify. In [Boyd 00], space is modeled as a network of interconnected areas, and probabilities of transition between areas are calculated, this technique does not produce trajectory estimations but is useful to improve motion planning in highly structured environments. A technique wich classifies trajectories using Kohonen Networks, but does not produce estimates is introduced in [Owens 00], It has been applied video surveillance to classify trajectories as "suspicious" or "dangerous", for example.

2.4 Data Clustering

We are going to discuss data clustering in this section because it is the basis for our proposed approach. Data Clustering aims to find structure in raw data. The general idea is to partition an entire data set into meaningful groups called clusters. This suits well to our goal of finding general motion patterns in raw trajectory data coming from sensors. In this chapter, we present an overview of the field as a background for our proposal.

2.4.1 Definition of the problem

There are many definitions of what data clustering is, here we present a formal definition adapted from [Fung 01]:

Definition 1 *Let $D \in M^{N \times T}$ be a set of data items, known as training data. D represents a set of N data items d_i in $\mathbb{R}^T$ where each component of d_i is called a feature and $\mathbb{R}^T$ is called the feature space. The goal is to partition D into K groups $C_k \in C$ where C_k is a particular group and C is the set of all the possible groups, such that data items that belong to the same group are more similar to each other than to data items in different groups. Each of the K groups is called a cluster. The result of the algorithm is an injective mapping $D \longmapsto C$ of data items d_i to clusters C_k. We call this process Data Clustering.*

There is a vast collection of clustering algorithms in the literature. Unfortunately, there is no technique that is universally applicable to all kind of problems, this is due to the fact that clustering algorithms often contain implicit assumptions about cluster shape and follow different criteria [Jain 99]. Moreover, most algorithms need to know in advance the number K of clusters into which the data is going to be classified. The problem of calculating K remains an open problem despite the existence of several proposals [Fraley].

2.4.2 Similarity Measures

Similarity is a key concept of data clustering, and most techniques need some measure of similarity in order to work. This measure can be an explicit function of the data being analyzed, but it can also come from more subjective sources (i.e. human answers to a survey) The choice of such a measure is a critical aspect of a clustering solution. Due to the spatial nature of our problem, in this section we will focus on distance measures for patterns whose feature space is continuous

In general, for continuous feature spaces, the complementary concept of *dissimilarity* is used by means of a *distance measure* defined in the feature space.

Definition 2 *Given a Feature space S, a distance or metric $\delta : S \times S \longmapsto \mathbb{R}$ is a function such that for all $x, y, z \in S$, all the following properties are verified:*

(i) $\delta(x, x) = 0$;

(ii) $\delta(x, y) > 0, x \neq y$;

(iii) $\delta(x, y) = \delta(y, x)$;

(iv) $\delta(x, z) \leq \delta(x, y) + \delta(y, z)$

Most clustering techniques can work with a *submetric* which doesn't have to verify all of the properties.

One of the most used metrics is the Euclidean distance:

$$\delta(d_i, d_j) = (\sum_{t=1}^{T}(d_i^t - d_j^t)^2)^{1/2} = \|d_i - d_j\|_2$$

Which is a special case of the Minkowski Metric

$$\delta(d_i, d_j) = (\sum_{t=1}^{T}|d_i^t - d_j^t|^p)^{1/p} = \|d_i - d_j\|_p$$

Minkowski Metrics presents some drawbacks, like the tendency of the largest-scaled features to dominate and distortion due to correlation among features. In order to alleviate this situation, a number of other metrics have been proposed like Mahalanobis Distance, Hausdorff Distance and many others [Mendelson 75].

Clustering algorithms can work in one of two ways (and sometimes in both of them): the first mode is called *model based clustering* and, as its name suggest, it is based on the manipulation of cluster representations or models which are particular to the data being clustered. The second mechanism is known as *pairwise clustering* and it works using a table of dissimilarity values, consisting on the $N(N-1)/2$ pairwise distance values for the N patterns in the training set. Pairwise clustering does not calculate a representation of the clusters being created so, if such a representation is needed, a mechanism should be proposed to provide it.

2.4.3 Clustering Techniques

In this section, we briefly explain a taxonomy of clustering techniques depicted in figure 2.4 and adapted from [Fung 01].and [Jain 99]. All the techniques can work both in model based and pairwise modes, with the exception of partitional generative methods, which are inherently model based.

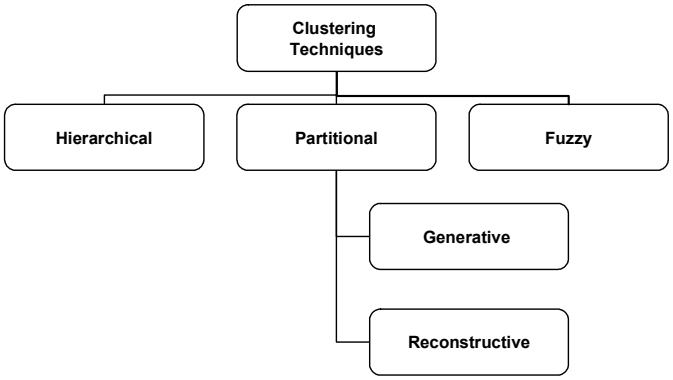


Figure 2.4: Clustering Techniques

2.4.4 Hierarchical Clustering

The idea of hierarchical clustering is to produce a recursive representation of the groups. The output is a *dendogram* which is a tree-like structure whose root node

is a single cluster grouping all the observations and whose branches correspond to splittings of the parent clusters. Leafs are N singletons corresponding to all the data items in the training data set D.

An interesting property of dendograms is that they can be broken at different similarity levels yielding to different clusterings of the data. In figures 2.5 and 2.6) we can see the resulting clusters for a particular dissimilarity level chosen in the corresponding dendogram.

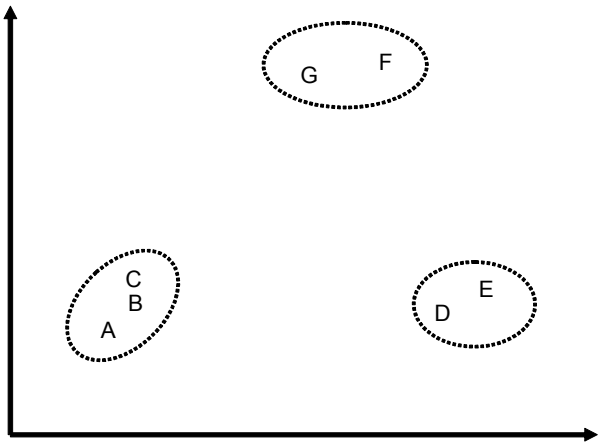


Figure 2.5: Example Clusters (taken from [Jain 99])

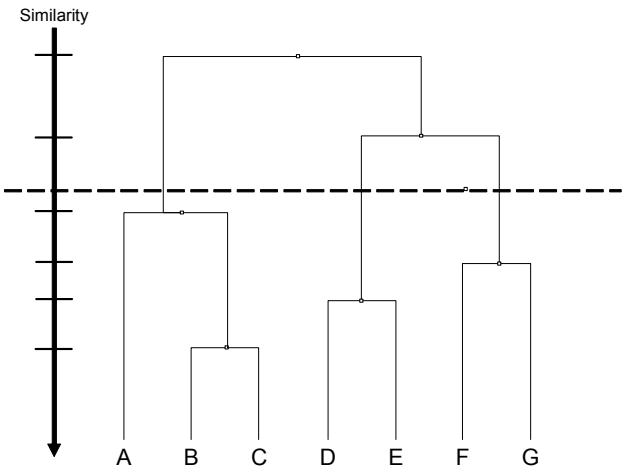


Figure 2.6: Corresponding Dendogram (taken from [Jain 99])

Hierarchical algorithms are called divisive if they start from a single cluster and then subsequently partition it. They are called agglomerative if they start from N clusters corresponding to each one of the training samples in D and then start to form fewer groups. Divisive and Agglomerative strategies correspond to top-down and bottom-up construction of the dendogram, respectively.

An agglomerative algorithm starts, as we said, with N clusters, one for each data item d_i Then, in the first step, it joins the two closest clusters. As both clusters consist of only one data item, this means that the algorithm joins the pair of data items having smallest dissimilarity, leaving us with $N-1$ clusters. This process is repeated until we have only one cluster. However, this calls for a definition of distance between clusters having more than one item. In fact is the definition of this intercluster distance what distinguishes different clustering algorithms [Kaufman 89]: Three of the most common measures, are nearest neighbor (single link), farthest neighbor (complete link) and average distance (see figure 2.7).

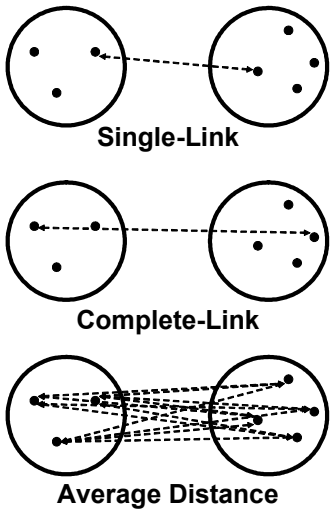


Figure 2.7: Distance Between Clusters

2.4.5 Partitional Clustering

Partitional techniques output a single partition of training data instead of a hierarchical clustering structure. The number of clusters K in the resulting partition is usually passed as a parameter to the clustering algorithm, the determination of K is a key decision and a difficult problem itself. Partitional algorithms usually work by optimizing a criterion function defined either locally or globally respect to the training data D. Partitional techniques can be classified as *Generative* or *Reconstructive* [Fung 01] (do

not confuse with Generative Motion Models).

Generative Algorithms

Generative algorithms assume that noisy data has been generated by K qualitatively similar, stochastic processes. [Fung 01] reformulates the problem as follows:

Consider a data set $D = \{d_1, ..., d_m\}$ consisting of observations from a set of K unknown distributions $C_1, ..., C_K$. If the density of an observation d_i with respect to C_{ki} is given by $f_k(d_i \mid \theta)$ for some unknown set of parameters θ. The probability that d_i belongs to distribution C_k is denoted by c_i^k. Since each observation is assumed to belong to just one distribution, we have the constraint $\sum_{k=1}^{K} c_i^k = 1$. The goal of generative algorithms is to find the parameters θ and c that maximize the likelihood or, for analytical purposes, the log-likelihood:

$$L(\theta, c) = \sum_{i=1}^{N} \ln(\sum_{k=1}^{K} f_k(d_i \mid \theta) c_i^k)$$

The Estimation-Maximization Algorithm ([McLachlan 97]) is often used to optimize this function.

One advantage exclusive of generative algorithms is that they calculate cluster representation and cluster assignation simultaneously.

Reconstructive Algorithms

This approach partitions the data into clusters by minimizing a cost function. The resulting assignment problem is a deterministic NP-hard combinatorial optimization problem [Buhmann 00]. The essential difference between the various types of reconstructive algorithms lies in the techniques used to model and minimize the cost function. Deterministic and stochastic approaches have been proposed. The classical example of deterministic algorithms is the classic K-means ([MacQueen 67] and [Hartigan 78]) algorithm, which minimizes the within-groups sum of squares:

$$Cost(c) = \sum_{k=1}^{K} \sum_{i=1}^{\|C_k\|} \sum_{j=1, j \neq i}^{\|C_k\|} \left(c_k^i - c_k^j \right)^2$$

Where c_k^i represents the i-th member of cluster k. And c is a particular clustering.

An often used stochastic technique is Simulated Annealing [Dubes 89] A pairwise clustering algorithm based in Deterministic Annealing is proposed in [Hofmann 97].

2.4.6 Fuzzy Clustering

While in the other clustering approaches one object belongs to one and only one cluster, fuzzy clustering associates each pattern with every cluster using a membership function [Zadeh 65]. In other words, the output of such algorithm is a clustering, but not a partition. This kind of clustering produces a matrix U having $N \times K$ elements. [Jain 99] shows an iterated approach using the following fuzzy estimator:

$$E^2(D,U) = \sum_{i=1}^{N} \sum_{k=1}^{K} u_i^j \, \|d_i - c_k\|^2$$

Where u_j^i is an element of U and c_k is the k-th fuzzy cluster center.

Chapter 3

Proposed Approach

We propose an approach that aims at verifying the objectives fixed in section 1.2, let's recall them: We want a technique able of predicting motion of cars and pedestrians. This technique should verify the following properties:

1. It should produce estimations with a long Time Horizon.
2. It should be as general as possible and work with many different kinds of objects.
3. It should be fast enough to give estimations in real time.

We have chosen to work on an estimation technique based on clustering. The rationale behind this decision obeys to the first two objectives: After reviewing existing techniques, we have found that clustering based techniques provide the longest Time Horizon for their predictions; at the same time, almost no assumptions are made about motion characteristics, so that this kind of techniques can be applied in a general fashion.

Our method is similar in many ways to the approach described in subsection 2.3.2 that we are going to call EM estimator or EME. The main difference is that our approach is based on a dissimilarity metric which allows the use of pairwise clustering algorithms, so we decided to call it Pairwise Estimator or PWE.

Our method consists of two components (see subsection 2.3):

1. **Learning.** The goal of learning is to classify a training set of trajectories into K classes, corresponding to motion patterns. In our approach, the number and characteristics of the classes are not known *a priori* and they are discovered during the learning process.
2. **Estimation**. Our estimation process calculates, for a given observed fraction of trajectory $o_{partial}$, its likelihood under each one of the K classes. The estimation

is the mean value of the class with maximum likelihood. Alternatively, we can present different possibilities by returning the means of all the classes having likelihood greater or equal than a given threshold.

The details of the approach are presented in the following sections.

3.1 Learning Motion Patterns

In order to discover the typical motion patterns of obstacles we will use a dissimilarity measure $\delta(d_1, d_2)$ to construct a dissimilarity table which can be fed into any pairwise clustering algorithm.

3.1.1 Similarity Measure

As our problem lies in the space-time domain, we consider an extension of the euclidean distance (see section 2.4.2). In order to apply our extension, we should introduce some conventions. A trajectory d_i of duration T_i can be viewed as a continuous, piecewise defined function $d_i(t)$ of time, where $d_i(0)$ represents the beginning of the motion and $d_i(T_i)$ is the end of the motion. This function can be derived from descriptive models assuming that any two subsequent configurations $d_i^t = \{x_i^t, y_i^t\}$ and $d_i^{t+1}\{x_i^{t+1}, y_i^{t+1}\}$ are joined by a straight line $y = k_1 x + k_0$and calculating the values of k_0 and k_1 for each of the T_i pieces. As a convention, we will say that $d_i(t) = d_i(T)$ for all $t > T_i$ and that the function is not defined for $t < 0$. The dissimilarity measure is:

$$\delta(d_i, d_j) = \left(\frac{1}{\max(T_i, T_j)} \int_{t=0}^{\max(T_1,T_2)} (d_i(t) - d_j(t))^2 \, dt \right)^{1/2} \tag{3.1}$$

Which is the average euclidean distance between the two functions. We have chosen the average because we want our measure to be independent of the length of the trajectories being compared.

3.1.2 Clustering Algorithm

The most important problem for running any clustering algorithm that uses K as a parameter is to estimate the value of K. EME solves this problem using the following method: it starts with one cluster and runs the algorithm. Whenever the EM has converged to a local maximum, it looks for trajectories with low data likelihood under the estimated models; when one such trajectory is found, a new model component is introduced that is initialized to that very trajectory. It also performs a redundancy

check for existing models: if global likelihood is not significantly reduced by eliminating a model, then the model is reduced. After performing both tests, the EM algorithm is run again. The entire process is repeated until there is no change in the number of models.

Our approach follows a different technique that we will proceed to explain.

Calculating the Number of Clusters

In order to estimate the value of K we have decided to use a clustering algorithm which does not need to know this value in advance. In particular, we are going to use a Complete-Link hierarchical agglomerative algorithm (see subsection 2.4.4). Normally, this algorithm produces a structure providing different clusterings at different dissimilarity levels. In our case, we are going to output just one clustering, corresponding to a threshold chosen *a priori*. The complete link algorithm has the advantage of producing compact clusters and it guarantees that the dissimilarity between any two members of the same cluster is smaller or equal than the chosen threshold.

Complete-Link or CL is a pairwise clustering algorithm, so its input is a matrix Δ of dissimilarities between clusters calculated using the expression 3.1. The matrix is initialized assuming that each individual trajectory d_i belongs to a different cluster. The algorithm is presented below.

Algorithm 3 *Agglomerative Complete-Link Clustering*

WHILE $min(\Delta) <$ *threshold DO*
    merge clusters corresponding to $min(\Delta)$*;*
    delete rows and columns in Δ *for the merged clusters;*
    add entries in Δ *for the new cluster;*
END

The creation of the new entries depends on the definition of distance between clusters. This definition is in fact what distinguishes different agglomerative algorithms. For complete-link, the distance between clusters is the distance between the farthest neighboring trajectories (figure 2.7).

Clustering with Deterministic Annealing [Hofmann 97]

Once we have an estimation of the value of K we can apply a pairwise clustering algorithm of our choice. An obvious question is immediately raised here: why do I want to cluster my data using another algorithm when I have already a clustering found by complete-link? The answer is that complete-link is based in a local objective function

(minimal dissimilarity) and this local character frequently lends to "unnatural" clustering of data. Other approaches, like Deterministic Annealing try to optimize a global function and their results are often more satisfactory. At the end, we are not sure that the added clustering step will increase the performance of our estimation approach in a relevant way, so, in a later stage, we are going to evaluate the performance of Complete-Link vs. Deterministic Annealing.

Deterministic Annealing Clustering is based on statistical physics. It works with two sets of variables: 1) $M_{i\nu}$, which has a value of 1 if sample i belongs to cluster ν, and 0 otherwise; and 2) $\varepsilon_{i\nu}$ which represents the average influence of exerted by all $M_{k\nu}, k \neq i$ on the assignment $M_{i\nu}$ also known as mean-field. Both sets of variables are interdependent and their values can be iteratively calculated in a way very similar to Estimation-Maximization. The variables are related by the two following expressions:

$$\langle M_{i\nu} \rangle = \frac{e^{-\varepsilon^*_{i\nu}/\Upsilon}}{\sum_{\alpha=1}^{K} e^{-\varepsilon^*_{i\alpha}/\Upsilon}} \tag{3.2}$$

and

$$\varepsilon^*_{i\nu} = \frac{1}{\sum_{j=1, j\neq i}^{N} \langle M_{j\nu} \rangle} \sum_{k=1}^{N} \langle M_{k\nu} \rangle \left(\delta(d_i, d_k) - \frac{1}{2\sum_{j=1, j\neq i}^{N}} \sum_{j=1}^{N} \langle M_{j\nu} \rangle \, \delta(d_j, d_k) \right) \tag{3.3}$$

As it name suggests, deterministic annealing uses a control value or temperature Υ in a manner very similar to simulated annealing. It tracks solutions from high to low temperatures, where gradually more and more details of the original objective function appear. The complete resulting algorithm, as given in [Hofmann 97] is:

Algorithm 4 *Deterministic Annealing Clustering*

INITIALIZE $\varepsilon^{*0}_{i\nu}$ *and* $\langle M_{i\nu} \rangle^0$ *RANDOMLY;*
temperature $\Upsilon \leftarrow \Upsilon_0$*;*
WHILE $\Upsilon > \Upsilon_{final}$
  $s \leftarrow 0$*;*
  REPEAT
    E-Like Step: estimate $\langle M_{i\nu} \rangle^{s+1}$ *as a function of* $\varepsilon^{*s}_{i\nu}$*;*
    M-Like Step: calculate $\varepsilon^{*s+1}_{i\nu}$ *for given* $\langle M_{i\nu} \rangle^{s+1}$*;*
    $s \leftarrow s + 1$*;*
  UNTIL all ($\langle M_{i\nu} \rangle^s$, $\varepsilon^{*s}_{i\nu}$) *satisfy 3.3;*
$\Upsilon \leftarrow \eta\Upsilon; \langle M_{i\nu} \rangle^0 \leftarrow \langle M_{i\nu} \rangle^s; \varepsilon^{*0}_{i\nu} = \varepsilon^{*s}_{i\nu}$*;*

3.1.3 Calculating Class Statistical Parameters

As we use only distance information for our clustering process, no cluster representation is calculated for the resulting clusters. In this section, we propose a way to calculate the mean trajectory for each cluster and then the standard deviation for each cluster based on this mean value.

Calculating the Mean Trajectory

Let C_k be a cluster having N_k trajectory functions $d_i(t) \mid 1 \leq i \leq N_k, d_i(t) \in C_k$, then we define the mean value of C_k as a function

$$\mu_k(t) = \frac{1}{N_k} \sum_{i=1}^{N_k} d_i(t) \tag{3.4}$$

This expression is defined for $t \in \mathbb{R}^+$. As the t can take any positive real value, we need a way to calculate it without having to go through all the infinite possible values of t. In order to do that, we are going to exploit the fact that all trajectory functions $d_i(t)$ are piecewise first-degree polynomials. Due to this fact, it suffices to calculate $\mu(t)$ for values of t corresponding to the extreme points of each "piece" d_i, the intermediate values are implicitly represented by the straight line equation. This leads naturally to the linear time-sweep algorithm which is presented below:

Algorithm 5 *Time-Sweep Mean Trajectory Calculation*

events $= \varnothing$*;*
FOR every d_i *in* C_k *DO*
    add all start and end times for pieces in d_i *to events;*
sort$(events, ascending)$*;*
$\mu_k = \varnothing$*;*
FOR every t *in events DO*
    add $\mu_k(t)$ *to* μ_k*;*

In the preceding algorithm $\mu_k(t)$ represents a continuous function while μ_k represents a descriptive model consisting on a discrete enumeration of configurations.

Calculating the Standard Deviation

Calculation of σ for cluster C_k is straightforward using the following expression:

$$\sigma_k = \left(\frac{1}{N_k} \sum_{i=1}^{N_k} \delta(d_i, \mu_k)^2 \right)^{1/2} \tag{3.5}$$

3.2 Estimating Trajectories

In order to estimate trajectories, we calculate the likelihood of a partially observed trajectory $o_{partial}$ under each one of the classes. To do that, we model classes as gaussian sources with mean value and standard deviation as calculated in the previous section.

3.2.1 Partial Distance

As we are dealing with partial trajectories, we need to modify expression 3.1 to account for this. The modification consists in measuring the distances respect to the duration of the partial trajectory:

$$\delta_{partial}(o_{partial}, d_i) = \left(\frac{1}{T_{partial}} \int_{t=0}^{T_{partial}} \left(o_{partial}(t) - d_i(t)\right)^2 dt \right)^{1/2} \tag{3.6}$$

Where $\delta_{partial}$, $o_{partial}$ and $T_{partial}$ are the partial distance, partially observed trajectory, and duration of the partial trajectory, respectively.

3.2.2 Calculating Likelihood

With the partial distance defined in expression 3.6, we can directly estimate likelihood of $o_{partial}$ under each cluster C_k.

$$P(o_{partial} \mid C_k) = \frac{1}{\sqrt{2\pi\sigma_k}} e^{-\frac{1}{2\sigma^2}\delta_{partial}\left(o_{partial}, \mu_k\right)^2} \tag{3.7}$$

Once we have calculated the likelihood, we can choose to estimate the trajectory using the mean value of the cluster with maximal likelihood, or to present the different possibilities having likelihood greater than a given threshold, for example.

3.2.3 Cluster Probability

An additional concept that can be introduced in our approach in latter works is the probability of occurrence of a cluster with respect to the other clusters. It allows us to make estimations in the absence of any sensorial input, answering questions like: I know that there is an object in the environment but I can't see it, what's the most likely trajectory for it? This probability can be approximated using the following expression:

$$P(C_k) \approx \frac{N_k}{N_D}$$

Where N_k is the number of trajectories in cluster k and N_D is the total number of trajectories in the data set D.

3.3 Analysis

As we said at the start of this chapter, we consider that Cluster Models have the advantage of providing longer Time Horizons because they produce estimates consisting in complete trajectories. Cluster Models made almost no assumptions about object's motion and this yields to generality. On the other hand, Cluster Models have the important problem of not being able to predict trajectories that have not been observed.

Our approach is similar in form to EME, but it is based on pairwise clustering. In this sense, it is more general, because it allows choosing between many existing pairwise clustering algorithms while EME is tied to Expectation-Maximization. In addition, we consider that our approach presents the following advantages when compared to EME:

Independency of Motion Representation EME imposes three conditions on the training set: trajectories must be represented using a descriptive model, trajectories should be evenly sampled over time and all must have the same length. In contrast, PWE accept both generative and descriptive motion representations, samples can be uneven and length can be different.

Lower computational complexity of the learning algorithm As we have said, the main difference between EME and PWE is that in PWE, we can choose a clustering algorithm. This allows us to use fast and simple computational algorithms as hierarchical clustering. As for EME, it is tied to EM, and it operates on very high dimensional vectors, this leads to great time and memory complexity and difficults the application on large data sets.

Another practical problem of EME is that likelihood is computed as a multiplication of probabilities for every component of a model: as probabilities are always <= 1, this can lead to numerical precision problems when the length of the trajectories exceeds a certain limit.

A drawback of our approach, when compared to EME, is that it requires and extra step to calculate cluster representations, while EME produces them directly as a part of its clustering algorithm. This drawback, however, is not very important, because the extra step is performed offline.

Chapter 4

Experimental Results

We have implemented our approach and tested it with simulated data. The algorithm proposed in [Bennewitz 02] was also implemented in order to compare it with our approach.

Java 1.4 was used as implementation language and tests were run in a PC having an 1.3 GHz Athlon microprocessor and 256 MB of RAM, the operating system was Windows XP.

The whole experimental process consisted of five steps.

1. Generation of a Training Set of 1000 trajectories using a trajectory simulator.
2. Calculation of K using the complete-link algorithm. Clustering using deterministic annealing and Estimation-Maximization algorithms.
3. Generation of a Test set of 500 trajectories using a trajectory simulator.
4. Benchmark the training set against the three different clustering algorithms and evaluate results using a performance metric.

Details on each one of the steps are provided in the following sections.

4.1 Generation of the Training Set

To test our approach, we used a simulator of the INRIA entry hall environment. In order to simulate trajectories, we defined a number of control points corresponding to important features of the environment; some of them are shown in figure 4.1.

We defined 32 trajectory prototypes consisting of sequences of control points to be traversed, the number of control points on each prototype ranged from 2 to 6. Trajectories were generated as follows:

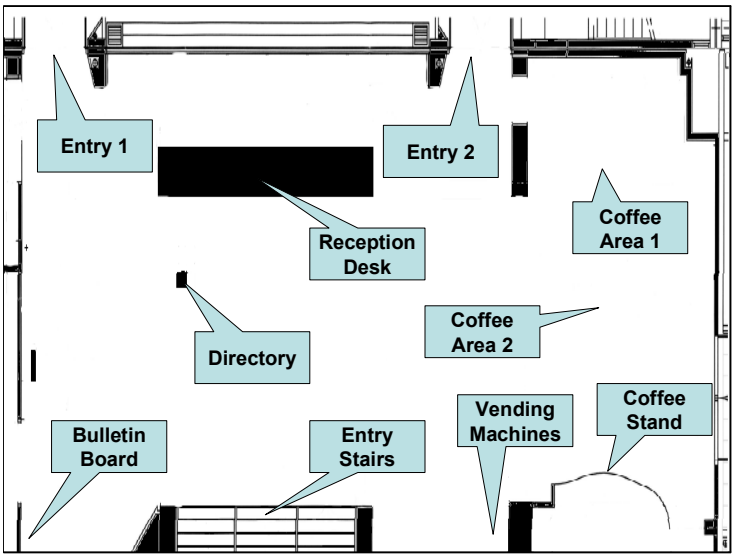


Figure 4.1: The INRIA entry Hall

1. Actual points corresponding to each of the control points are generated using a bidimensional gaussian distribution with mean values corresponding to the center of the control point and σ equal to 20 cm.

2. Motion was simulated advancing in fixed steps of 10 cm from the last control point in the direction of the next one, with a standard deviation of 0.3 rad. The next control point is considered to be reached when distance to it is less or equal than 10 cm.

3. Step 2 is repeated until the last control point is reached.

The 1000 trajectories in the training set D where generated choosing trajectory prototypes at random according to a given, not uniform probability distribution. The purpose of using such distribution is to simulate the fact that there are certain kinds of trajectories that occur more often that others.

4.2 Estimation of K and clustering

The complete link (CL) clustering algorithm was run against the data set using three different distance thresholds: 20, 30 and 40 cm. The resulting clusterings where saved for benchmarking. The obtained values of K (59, 101 and 205, respectively) were fed into the Deterministic Annealing (DA) algorithm and the output clusterings were saved. It worths noting that DA sometimes produces clusters which have no assigned

elements. In our case, we simply ignored those clusters, hence, the actual numbers of clusters were 59, 99 and 138, respectively. For EME, we started with one cluster and used values of σ equal to 20, 15 and 10 cm, these values produced 36, 53 and 133 clusters, respectively. Results of this step are presented in table 4.1.

The values we have chosen for $maxDistance$ and σ worth some discussion. We used three different sets of values, because we know that these values affect the number of clusters that we are going to obtain, and we are interested in studying the effect of the number of clusters on estimation performance. We started by choosing the values for $maxDistance$ knowing that they have a clear physical interpretation: for a given cluster C_k the maximal distance between two trajectories appertaining to the cluster will not exceed $maxDistance$. As for the values of the parameter σ for EM, we have chosen, to base them on the values of $maxDistance$ using the expression $\sigma = \frac{maxDistance}{2}$.

Algorithm	**Parameter**	**Resulting Clusters**
CL	$maxDistance = 40\,\text{cm}$	59
CL	$maxDistance = 30\,\text{cm}$	101
CL	$maxDistance = 20\,\text{cm}$	205
DA	$K = 59$	59
DA	$K = 101$	99
DA	$K = 205$	138
EM	$\sigma = 20\,\text{cm}$	36
EM	$\sigma = 15\,\text{cm}$	53
EM	$\sigma = 10\,\text{cm}$	133

Table 4.1: Clustering Results

All three algorithm produced clusters which seemed reasonable when inspected visually, an example can be seen in figure 4.2 where similar clusters are displayed for the three algorithms. In each image, orange trajectories correspond to original trajectories in the data set, and the blue one corresponds to the cluster mean value.

4.3 Generation of the Test Set

The test set χ of 500 trajectories was generated in an analogous way to the one used for generating training data. The same parameters and trajectory prototypes were used.

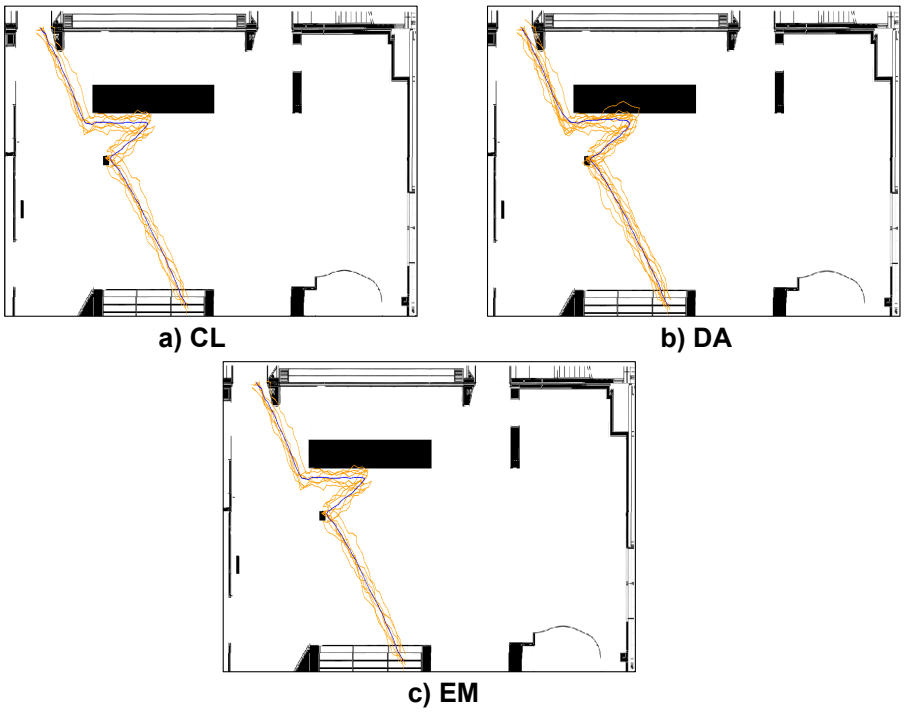


Figure 4.2: Examples of generated clusters for the three algorithms.

4.4 Benchmarking

In order to compare the different approaches and clustering algorithms, we defined a simple performance metric. Its inputs are a given clustering C, the test set χ, and the *percentage* in length of each trajectory in the test set that will be used to find an estimate.

The metric is calculated using the following algorithm, where N_χ is the total number of trajectories in the test set.

Algorithm 6 *Performance metric*

Function PerformanceMetric($\chi, C, percentage$*)*

$result \leftarrow 0$;

FOR each trajectory χ_i *in the test set* χ *DO*

calculate $\chi_i^{percentage}$;

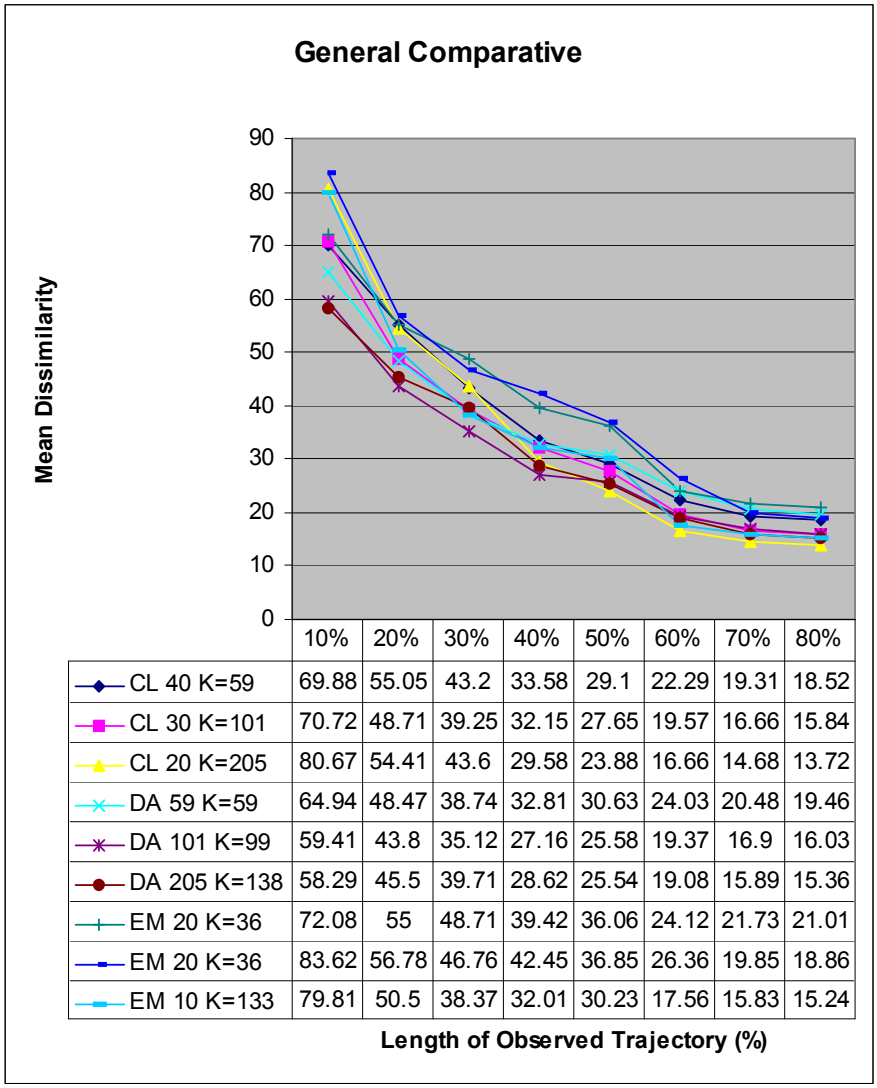
find the cluster C_k *with max likelihood for* $\chi_i^{percentage}$;

$result \leftarrow result + \delta(\chi_i, \mu_k)$;

END FOR

$result \leftarrow result/N_\chi$;

The metric is basically the average dissimilarity between the real trajectories in the test set and the estimations obtained using a given percentage of those trajectories. Lower values are better then higher ones. Tests where performed against the 9 clusterings listed in table 4.1. General results are shown in figure 4.3, there we can see, for example, that the clustering produced using the EM algorithm with $\sigma = 20$ and having $K = 36$, scored a value of 48.71 when estimations where calculated using 30% of the length of the observed (test set) trajectories.



	10%	20%	30%	40%	50%	60%	70%	80%
CL 40 K=59	69.88	55.05	43.2	33.58	29.1	22.29	19.31	18.52
CL 30 K=101	70.72	48.71	39.25	32.15	27.65	19.57	16.66	15.84
CL 20 K=205	80.67	54.41	43.6	29.58	23.88	16.66	14.68	13.72
DA 59 K=59	64.94	48.47	38.74	32.81	30.63	24.03	20.48	19.46
DA 101 K=99	59.41	43.8	35.12	27.16	25.58	19.37	16.9	16.03
DA 205 K=138	58.29	45.5	39.71	28.62	25.54	19.08	15.89	15.36
EM 20 K=36	72.08	55	48.71	39.42	36.06	24.12	21.73	21.01
EM 20 K=36	83.62	56.78	46.76	42.45	36.85	26.36	19.85	18.86
EM 10 K=133	79.81	50.5	38.37	32.01	30.23	17.56	15.83	15.24

Figure 4.3: Benchmark Results

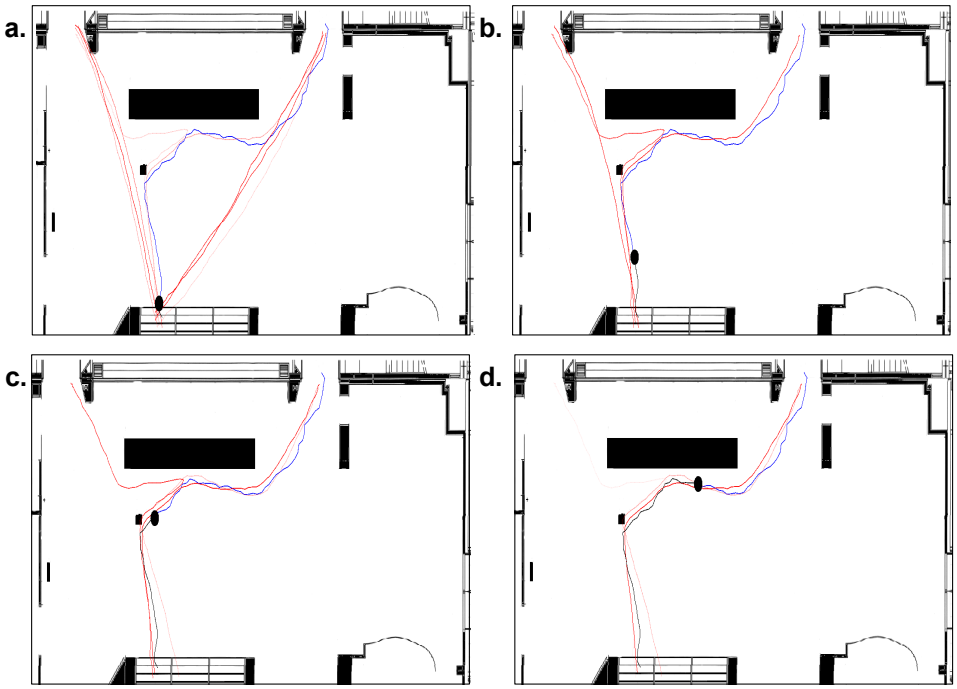


Figure 4.4: Trajectory Estimation for Different Instants

4.5 Graphical Representation of Estimation

We have implemented a GUI that displays results of the estimation process as it evolves over time (figure 4.4). In order to do this, likelihood is calculated for all clusters using expression 3.7. The GUI displays the average values of the clusters having a likelihood greater than a given threshold; most likely estimates are shown in dark red, less likely ones are lighter. The object's position is represented by a black circle, the observed trajectory is shown with a black line. Future motion is depicted in blue.

As can be seen in (figure 4.4), even if motion estimates change over time, near future motion is accurately estimated, except for the initial configuration, which can not be very discriminating because not much information is available.

4.6 Analysis

When observed using the GUI, predictions based on all the 9 clusterings seem to be "reasonable" and it is difficult to make an evaluation using this empirical data. In contrast, numerical results presented in figure 4.3 shows very interesting results. In the first place, we can see that quality of estimations seems to converge as the length

of observed trajectory grows, which seems reasonable: we expect precision to increase as more information is used in the estimation process, and, at the same time, we know that final precision will be limited by the number of available clusters.

We can see also that, in general, DA performs better than the two other algorithms (see figure 4.5) especially for values of observed length smaller than 40%. We can interpret this as saying that DA seems to produce clusterings that represent real motion more accurately than the other two algorithms.

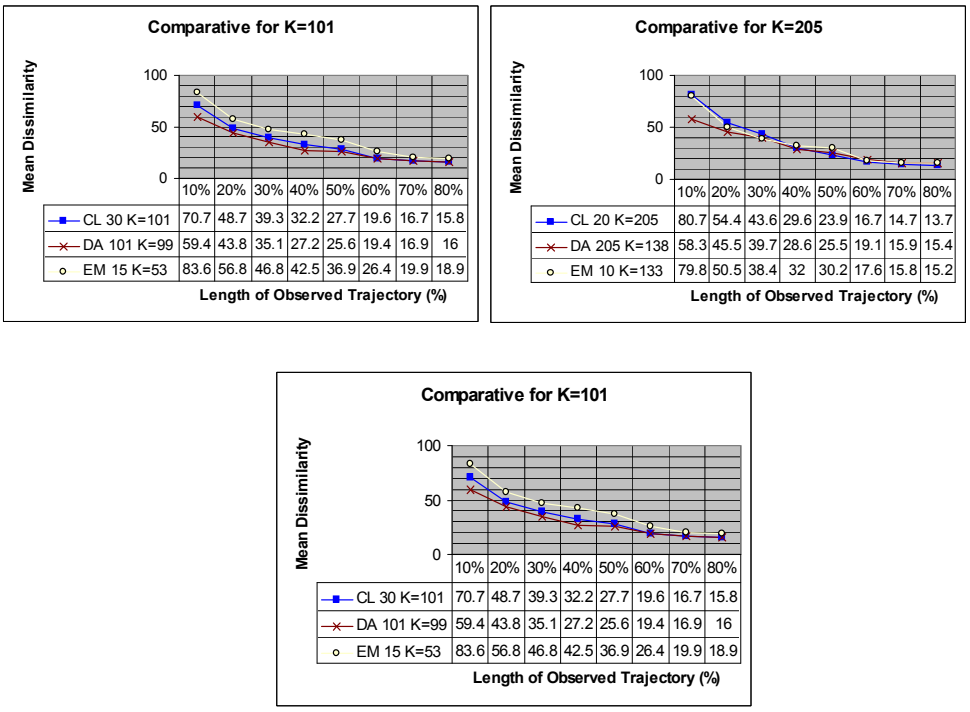


Figure 4.5: Results by Number of Clusters

In figure 4.6, we present the processing times for the three clusterings performed for each algorithm in ascending order with respect to the number of clusters. CL is the best absolute performer; in addition we can see that clustering time is practically independent of the number of clusters. As a matter of fact, most of the time was consumed calculating the dissimilarity table. Next in performance is Expectation-Maximization, whose longer time was around half an hour for 133 clusters. Finally, we can see that DA is much slower than the other two algorithms. Thus, the precision of DA seems to have a cost, however, it worths noting that clustering is an off-line process which is only run once.

Both DA and CL share an important drawback when compared to EM, the dis-

similarity table uses memory proportional to the square of the size of the training set, this limits the number of samples that can be actually used in learning.

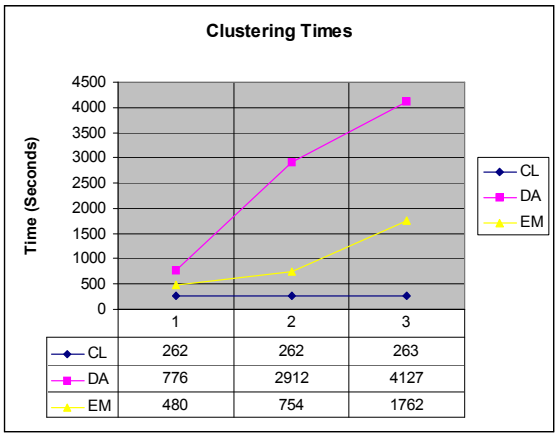


Figure 4.6: Clustering times

We can conclude that, according to the tests performed using our benchmarking criterion, PWE performs better than EME in producing long term estimations of the motion of objects, and that its only drawback is that the number of samples in the training set is limited by the amount of memory available.

As for our third goal, the possibility of using our approach in real time, we should say that in our tests the maximum time needed for estimation is slightly smaller than one second for a trajectory having 120 points. This, is not acceptable for use in real time applications, however, a drastic improvement can be obtained by incrementally calculating likelihoods for each new observation using the result obtained for the last observation. With this simple optimization, the maximal estimation time would be reduced to about 1/100 of a second which is suitable for most real time applications dealing with humans and cars.

To conclude this chapter, we are going to present some early results obtained using real data. The trajectories were captured using a vision system developed by the team PRIMA of the INRIA using a camera placed in the entry hall of the laboratory (figure 4.7). Trajectory coordinates where transformed from the image to the hall's plan by correcting the distortion and then applying an homography.

Some of the resulting clusters can be seen in figure 4.8. As we have said, these are only early results: we are still working in issues like filtering out artifacts that appear due to light reflecting on the floor. In any case, we expect to solve this issues in the short term in order to test our estimation technique both in the entry hall and in the lab's parking lot.



Figure 4.7: The Camera View

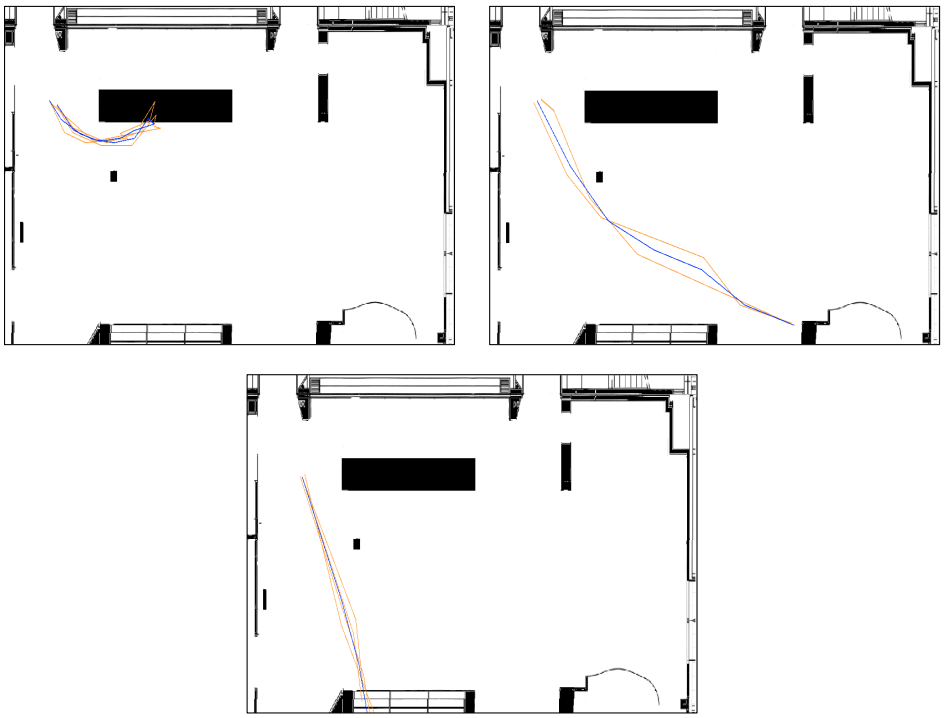


Figure 4.8: Some Clusters obtained using real data

Chapter 5

Conclusions

The work presented in this document deals with the problem of predicting future motion of objects using knowledge about past and present motion. We are particularly interested in motion prediction for pedestrians and vehicles.

5.1 Summary and Contributions

We have presented an introduction to the field and proposed a classification of existing techniques based in motion representation. We have reviewed the two resulting categories: Generative and Descriptive Models. We also have outlined data clustering, which is the basis for our approach.

We have proposed and implemented a motion estimation technique called Pairwise Estimation (PWE) based on clustering. As most estimation techniques based on descriptive models, our approach is defined by two processes: learning and estimation:

1. For the learning stage, we propose a distance metric that can be used to perform trajectory clustering using any existing pairwise clustering algorithm. According to our knowledge, this is a novel approach to trajectory clustering. We also describe how to use the Complete-Link clustering algorithm in order to calculate the number of clusters. We propose a simple algorithm to calculate the mean value for a cluster of trajectories. This mean trajectory is then used to calculate the standard deviation for each cluster.

2. For the estimation stage, we propose a simple way to calculate the likelihood that a partially observed trajectory corresponds to a certain cluster. This likelihood, can be calculated for each cluster, obtaining in this way a probabilistic representation of motion. We can also simply select the most likely pattern obtaining in this way a deterministic prediction.

We implemented our approach and compared its performance with another technique (EME [Bennewitz 02]) which we consider to represent the state of the art in cluster based estimators. To compare the two techniques, we used a performance metric that we propose as a general benchmark for techniques which estimate complete trajectories. In our tests, PWE performed better than EME.

Based on tests, we can say that the objectives that we have fixed at the start of the project have been fulfilled: .Our technique produces long-term estimations of the motion of objects; the approach is applicable to a wide variety of obstacles; and it can produce estimates in real-time. However, we must highlight the fact that these tests were run on simulated data. We still have to validate our results using real data, acquired through a vision system.

5.2 Perspectives and Future Work

As mentioned in the last section, the next step is the implementation of our technique on a real system. In particular, we are going to implement and test our technique in two different environments: The INRIA entry hall and the parking lot. Data will be acquired through a vision system developed by team PRIMA which does both object identification and trajectory tracking for multiple objects.

Besides this practical step, there are some extensions to our approach that could yield interesting results:

Use of Cluster Probability Cluster probability (section 3.2.3) could be used to improve trajectory estimation under the condition that it is not uniformly distributed. An extension could consider this probability as a part of the estimation process

Weighted Estimations In our work we have calculated estimates using two criteria: a) all the means of the clusters having likelihood greater than a given threshold, and b) the mean of the cluster with the greatest likelihood. A third estimation criterion could be a weighted average of all the models based on its likelihood. In theory, such estimation would be able to predict "unknown" trajectories which are a linear combination of known patterns.

Dynamic Environments Many environments are dynamical in the sense that its configuration can change over time. An example of this is a corridor having doors which can be open or closed. We think that it is possible to extend our approach into a framework which considers these dynamic features of the environment.

We would like to conclude this work by recalling the fact that motion prediction of objects following an intentional or "intelligent" behavior is a very important and extremely challenging problem. The approach we present in this document relies in the structured nature of the spaces being observed and is not able to predict trajectories that have not been observed. It is limited also by the fact that it ignores dynamic environments and interaction between objects. We think, however, that our work is a step in the direction of taking into account the complex processes which lie behind intentional motion.

Bibliography

[Bennewitz 02] Maren Bennewitz, Wolfram Burgard & Sebastian Thrun. *Learning Motion Patterns of Persons for Mobile Service Robots.* In Proceedings of the 2002 IEEE ICRA, pages 3601–3606, 2002.

[Blostein 95] Steven D. Blostein & Robert M. Chann. *The Use of Image Plane Velocity Measurements in Recursive 3-D Motion Estimation from a Monocular Image Sequence.* pages 797–801, 1995.

[Boyd 00] Jeffrey E. Boyd, Jean Meloche & Yehuda Vardi. *Statistical Tracking in Video Traffic Surveillance.* 2000.

[Buhmann 00] Joachim M. Buhmann. *Learning and Data Clustering*, 2000.

[Chang 77] C.B. Chang, R.H. Whiting, L. Youens & M. Athans. *Application of the Fixed-Interval Smoother to Maneuvering Trajectory Estimation.* IEEE Transactions on Automatic Control, vol. 5, pages 876–879, 1977.

[Chien 89] Y.P. Chien & A.J. Koivu. *Visual Feedback to Predict Obstacle Motion for on-Line Collision-Free Trajectory Planning of Cylindrical Robots.* IEEE/RSJ International Workshop on Intellingent Robots and Systems, pages 612–618, 1989.

[Dubes 89] R.W. Dubes & R.C. Dubes. *Experiments in Projection and Clustering by Simulated Annealing.* Pattern Recognition, vol. 22, pages 213–220, 1989.

[Elfes 89] A. Elfes. *Using Occupancy Grids for Mobile Robot Perception and Navigation.* Computer, vol. 22, pages 46–57, 1989.

[Fraley] C. Fraley & A.E. Raferty. *How Many Clusters? Which Clustering Method? Answers Via Model-Based Cluster Analysis.* Rapport technique 329, University of Washington.

[Fung 01] Glenn Fung. *A Comprehensive Overview of Clustering Algorithms*, 2001.

[Gaffney 99] Scott Gaffney & Padhraic Smyth. *Trajectory Clustering with Mixtures of Regression Models*. Rapport technique 99-15, University of Californa, Irvine, 1999.

[Hartigan 78] J.A. Hartigan & M.A. Wong. *A K-Means Clustering Algorithm*. Applied Statistics, vol. 28, pages 100–108, 1978.

[Hofmann 97] Thomas Hofmann & Joachim M. Buhmann. *Pairwise Data Clustering by Deterministic Annealing*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 19, pages 1–14, 1997.

[Jain 99] A.K. Jain, M.N. Murty & P.J. Flynn. *Data Clustering: A Review*. ACM Computing Surveys, vol. 31, pages 265–322, September 1999.

[Kalman 60] R.E. Kalman & R.S. Bucy. *New Results in Linear Filtering and Prediction Theory*. Journal of Basic Engineering ASME Transactions, vol. 83, pages 95–107, 1960.

[Kaufman 89] Leonard Kaufman & Peter J. Rousseeuw. Finding groups in data: An introduction to cluster analysis. Wiley Series In Probability And Mathematical Statistics. John Wiley and Sons, Inc., 1989.

[Kawase 98] Tetsuya Kawase, Hideshi Tsurunosono, Naoki Ehara & Iwao Sasao. *Two-Stage Kalman Estimator Using Advanced Circular Prediction for Maneuvering Target Tracking*. pages 2453–2456, 1998.

[Knudsen 92] Knud Steven Knudsen & Leonard T. Bruton. *Moving Object Nonlinear Trajectory Estimation in the Transform/Spatiotemporal Mixed Domain*. pages 2481–2484, 1992.

[Koller 94] D. Koller, J. Weber, T. Huang, J. Malik, G. Ogasawarea, B. Rao & S. Russell. *Towards Robust Automatic Traffic Scene Analysis in Real-Time*. In Proceedings of the 33rd Conference on Decision and Control, pages 3776–3781, December 1994.

[Kruse 96] E. Kruse, R. Gutsche & F.M. Wahl. *Estimation of Collision Probabilities in Dynamic Environments for Path Planning with Minimum Collision Probability*. In Proceedings of IROS, pages 1288–1295, 1996.

[Kruse 97] E. Kruse, R. Gusche & F. M. Wahl. *Acquisition of Statistical Motion Patterns in Dynamic Environments and their Application to Mobile Robot Motion Planning.* pages 713–717, 1997.

[Kyriakopoulos 92] K.J. Kyriakopoulos & G.N. Saridis. *An Integrated Collision Prediction and Avoidance Scheme for Mobile Robots in Non-Stationary Environments.* In Proceedings of the International Conference on Robotics and Automation, pages 194–199, May 1992.

[Larson 66] R.E. Larson & J. Peschon. *A Dynamic Programming Approach to Trajectory Estimation.* IEEE Transactions on Automatic Control, pages 537–540, 1966.

[Latombe 91] Jean-Claude Latombe. Robot motion planning. Kluwer Academic Publishers, Norwell, MA, 1991.

[Leduc 98] Jean-Pierre Leduc, Jon Corbelt, Mingqi Kong, Victor Wiekerhauser & Bijoy Ghosh. *Accelerated Spatio-Temporal Wavelet Transforms: and Iterative Trajectory Estimation.* pages 2781–2784, 1998.

[MacQueen 67] J. MacQueen. *Some Methods for Classification and Analysis of Multivariate Observations.* In L.M. Le Cam & J. Neyman, editeurs, Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability, volume 1, pages 281–297. University of California Press, 1967.

[McLachlan 97] Geoffrey J. McLachlan & Thriyamakam Krishnan. The EM algorithm and extensions. John Wiley and Sons, Inc., 1997.

[Mendelson 75] B. Mendelson. Introduction to topology. Allyn and Bacon, 1975.

[Mitchell 97] T.M. Mitchell. Machine learning. Mc Graw-Hill, 1997.

[Owens 00] Jonathan Owens & Andrew Hunter. *Application of the Self-Organising Map to Trajectory Classification.* pages 1–7, 2000.

[Patt 98] Ivan Patt, Wing Ching, Lio Yonghzhi, Leonard Chin & Dinesh Mital. *Neuro-Fuzzy Techniques for Airborne Target Tracking.* pages 251–257, 1998.

[Strehl 98] Alexander Strehl & J.K. Aggarwal. *A New Bayesian Relaxation Framework for the Estimation and Segmentation of Multiple Motions*, 1998.

[Tadokoro 95] Satoshi Tadokoro, Masaki Hayashi, Yasuhiro Manabe, Yoshihiro Nakami & Toshi Takamori. *Motion Planner of Mobile Robots Which Avoid Moving Human Obstacles on the Basis of Stochastic Prediction.* pages 3286–3291, 1995.

[Tanaka 02] Kanji Tanaka. *Detecting Collision-Free Paths by Observing Walking People.* In Proceedings of the 2002 IEEE/RSJ IROS, pages 55–60, 2002.

[Zadeh 65] L.A. Zadeh. *Fuzzy Sets.* Information Control, vol. 8, pages 338–353, 1965.

[Zhu 90] Qiuming Zhu. *A Stochastic Algorithm for Obstacle Motion Prediction in Visual Guidance of Robot Motion.* pages 216–219, 1990.