Adaptive Prediction for Target Tracking Using Switching ARMA Models

David Miguel Belo Freire de Carvalho

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisors: Prof. Dr. João Manuel Lage de Miranda Lemos
Prof. Dr. Jorge dos Santos Salvador Marques

October 2018

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

This work counted with the support of a SPARSIS project research grant, funded by the Portuguese Foundation of Science and Technology (FCT) under contract PTDC/EEIPRO/0426/2014.

Abstract

This dissertation proposes methods that aim to describe the movement dynamics of a target in an image sequence and predict its future positions. The target can be a person, an animal, a car, or any other object of attention that one wishes to predict its trajectory.

The main idea is to treat each coordinate of a trajectory as an univariate time series, that can be approximated by a model. Fitting an ARMA or AR model to a time series requires the series to be stationary. This objective can be achieved by differencing non-stationary series one or more times, or through seasonal differentiation. After achieving stationarity, it is assumed that the resulting series follows an ARMA model, that allows the series to describe itself by a linear combination of its previous values, hence, allowing to further predict it and consequently, to predict the next steps of the trajectory. These models are also called ARIMA and SARIMA.

In an adaptive prediction framework, this dissertation uses techniques from supervised multiple models. In this realm, for a dataset of trajectories, different banks of models are created. When analyzing a new target, a multiple models architecture recursively chooses the best models to predict the target trajectory, thus allowing to adapt to the preceding movement.

The evaluation of the proposed methods involve splitting the a dataset into a training set, a validation set and a testing set. The models that best represent the training set are learned through a learning algorithm. Then, the parameters of the multiple models architecture are optimized with the validation set. Finally, the trajectories from the testing set are analyzed.

Multiple experiments were performed with synthetic and real world data. A dataset from the SPARSIS project is thoroughly analyzed and baseline for trajectory prediction in that set is establish.

Keywords: Time-Series; Prediction; ARMA; Multiple Models

Resumo

Neste trabalho são propostos métodos que visam descrever as dinâmicas do movimento de um alvo numa sequência de imagens e prever a sua trajetória futura. Este alvo, pode ser uma pessoa, um animal, um carro ou qualquer outro objeto em que haja interesse em prever o seu movimento.

A ideia principal, é tratar cada coordenada de uma trajetória como uma série temporal, que pode ser aproximada por um modelo. De forma a fazer esta aproximação de uma série temporal, é necessário que a série seja estacionária. Isto pode ser feito com recurso à diferenciação da série temporal, uma ou mais vezes, ou então com recurso a uma diferenciação sazonal. Tendo uma série estacionária, pressupõe-se que esta segue um modelo ARMA, o que significa que a própria série pode ser descrita por uma combinação linear de valores passados. Desta forma, consegue-se prever os seus valores futuros e consequentemente, prever os próximos pontos da trajetória. Este processo é também denominado de modelo ARIMA ou SARIMA para uma diferenciação sazonal.

Esta dissertação usa técnicas de múltiplos modelos supervisionados, tendo em conta uma predição adaptativa. Nesta conjuntura, é necessário a criação de bancos de modelos para cada conjunto de dados (*dataset*) de trajetórias diferente. Quando é analisado um novo alvo, é utilizado uma arquitetura de modelos múltiplos que recursivamente escolhe os melhores modelos para prever futuras posições do alvo. Deste modo, a predição é adaptada ao movimento até então descrito.

Para avaliar os métodos propostos, separamos o (*dataset*) em dados de treino (*training set*), dados de validação (*validation set*) e dados de teste (*testing set*). Os modelos que melhor representam o *training set* são obtidos pelo meio dum algoritmo de aprendizagem. Seguidamente, o *validation set* é utilizado para otimizar os parâmetros da arquitetura de modelos múltiplos. Por fim, as trajectórias do *testing set* são analisadas.

Foram realizadas várias experiências com dados sintéticos e reais. Um conjunto de dados obtido pelo projeto SPARSIS é cuidadosamente analisado e é definido um patamar base para a predição de trajetórias nesse *dataset*.

Palavras-Chave: Séries Temporais; Predição; ARMA; Múltiplos Modelos

Contents

List of Tables

List of Figures

List of Acronyms

AIC	Akaike's Information Criterion
AR	Auto Regressive
ARIMA	Auto Regressive Integrated Moving Average
ARMA	Auto Regressive Moving Average
ASNSC	Augmented Semi Nonnegative Sparse Coding
CNN	Convolutional Neural Network
FF	Far field
LPF	Low Pass Filter
LSTARIMA	Localized Space-Time ARIMA
LSTM	Long Short-Term Memory
MA	Moving Average
PPE	Predicted Position Error
RNN	Recurrent Neural Network
RPPE	Recursive Predicted Position Error
SARIMA	Seasonal Auto Regressive Integrated Moving Average
SDD	Stanford Drone Dataset
SR	Short range
TSE	Trajectory Set Error
VAR	Vector Auto Regressive
VARMA	Vector Auto Regressive Moving Average

Chapter 1

Introduction

1.1 Motivation

Trajectory prediction is an indispensable resource to understand different patterns and trends in our world. The knowledge provided by the understanding of these patterns becomes particularly relevant when it comes to the creation of models that are used to predict future actions and distinguish typical behaviors from atypical ones.

Researchers study these patterns in many different fields, for a variety of purposes. In zoology, wild animal behavior is studied through the analysis of free-ranging animals trajectories. By analysing of elk and deer trajectories [1] shows that realizations of the animals paths can be generated through a stochastic differential equation-based model, supported by the corresponding equations of physics. Weather analysts also use trajectory data and complex models to predict the weather. A spatial-temporal forecasting methodology is used in [2] to model and predict weather factors such as air temperature, wind speeds and solar radiation. Human mobility and surveillance also have been the focus of extensive study. Multiple motion models vector fields are used in [3] and applied in surveillance tasks for human activity classification. The Stanford Drone Dataset (SDD) is developed by [4] and used to learn the “unspoken” social etiquette between two targets, using that characterization to improve forecasts models and multi-target tracking. Human trajectories and intents are predicted in [5] through the identification of attractive or repulsive “fields” of public space in unobserved parts of videos. This dissertation falls into this latter category of surveillance.

Video tracking is a broad subject with many applications [6]. In particular, surveillance is very useful in real work applications. To name a few: monitoring the behavior of human drivers to improve autonomous vehicle development [7]; predicting ships and planes anomalies by analyzing their past trajectories [8], [9]; help improve security by detecting anomalous behaviors in crowded environments [10].

The purpose of this dissertation is to restore extensively studied methods of time-series predictions and apply them to trajectory prediction. The models used in these predictions are variations of the Auto Regressive Moving Average (ARMA) model, namely the ARIMA and the SARIMA model. These models were thoroughly studied and popularized by Box and Jenkins [11]. The ARIMA model became a widely used prediction model in a wide range of areas, including economics. As a result, it gained reputation as ”the standard of time series prediction” [12].

1.2 State of the art

The work on surveillance covers two different settings [3] : Short range (SR) and Long range or Far field (FF). In the SR setting, the camera is close to the subject and thus a great number of features can be extracted. Conversely, FF settings have the camera far away to the subject. As a result it becomes difficult to have detailed information of the subject and under these conditions only the past trajectory of the object can be recorded. In the following paragraphs we can see how recent studies in FF surveillance use different approaches to accurately predict future trajectories.

Regarding machine learning techniques, [13] proposes a Recurrent Neural Network (RNN) Long Short-Term Memory (LSTM) model that take into account the elements of a scene that can influence the trajectory of a target. In addition, a novel deep attention-based model is proposed in [14] that uses a Convolutional Neural Network (CNN) for feature extraction, a combination of two attention mechanisms and finally a LSTM network to achieve state of the art accuracy in target predictions when using the SDD dataset. Although these approaches are very accurate in their predictions, they lack in parameters that can describe the trajectory.

In other approaches, such as motion vector models, sparse techniques are used to learn motion models in a specify scene, in order to describe and predict trajectories. A novel method based on the ASNSC formulation is proposed in [15]. It intents to learn motion models from one environment, enabling to make predictions in other unseen ones. Additionaly, another study regarding sparse techniques [16] combines sparse motion fields together with a space-varying probabilistic switching mechanism, which leads to a simpler way of interpreting motion fields that have a meaningful description of the data.

This dissertation follows a different approach, applying traditional time series methods for trajectory analysis and prediction. Trajectory coordinates are treated as independent time series that are modeled by an ARIMA model or a SARIMA model. Previous work was already develop with similar approaches. In [17] several algorithms to predict long term human movement trajectories are used, one of which is Auto Regressive (AR) models. Although AR models were not found to be the ”winning” approach, they proved to be fast enough for an online application. Our work differs from this application, based on the fact that the predictions made are very short-term. Furthermore, not only AR models are considered, but all kinds of ARIMA models are. In [18] a novel spatial-time series model Traj-ARIMA based on the extension of the ARIMA model is presented. The main focus in this study is vehicle speed forecasting on a network-constrained trajectory data. Some results show poor accuracy when compared to measured data, which can be explained by the use of univariate models and model estimation for the whole trajectory. Although univariate models are used in this dissertation, an online approach is used together with multiple models to predict and adapt to the next steps of the trajectory. In a recent work [12] proposes an novel model that was based on a more complex variation of the ARIMA model, the Localized Space-Time ARIMA (LSTARIMA). Their method proved effective in predicting future traffic flow, but it was slightly less accurate than the LSTARIMA model. However, the model has lower computational complexity than its counterpart LSTARIMA, so it can proven to be useful in real-time prediction application purposes.

The recent advances in computation power and the exponential increase of trajectory data have provided us with a great number of training datasets. The SDD [4] is a large scale dataset that collects images and videos of various moving targets that move within the Stanford university campus. This dataset captures trajectories of pedestrians, cyclists, skateboarders, cars, buses, and golf carts. The SPARSIS project framework consists of many original publications and datasets that focus on characterizing human activities in video signals and detect abnormal behaviors. One of the datasets [19] contributed by the SPARSIS project captures the trajectories of various pedestrians going up and down a flight of stairs in Instituto Superior Técnico. The SPARSIS dataset is used to test the algorithms developed in this dissertation.

1.3 Original Contributions

This dissertation original contributions are as follows:

1. A multiple model architecture is presented, tested and several experiments are performed in order to regulate its parameters.
2. An original MATLAB function library (toolbox) was developed throughout the course of this dissertation, containing all algorithms used in this work. This includes: an ARMA time-series generator; a general ARMA predictor; several functions to add and remove non-stationarity and seasonal patterns form time series; a learning algorithm that automatically generates a bank of the best ARMA models for any dataset.
3. The SPARSIS dataset is thoroughly tested and a baseline for trajectory prediction is established.

1.4 Problem Formulation

The core of the problem addressed in this dissertation can be briefly summarized in the following sentence: given observations of a trajectory past coordinates as a sequence of two-dimensional position measurements $(x_1(k),\ x_2(k))$, sampled at a fixed time interval (Δt), compute an estimate of the future position of that trajectory, after m $(m \geq 1)$ steps $(\hat{x}_1(k+m|k),\ \hat{x}_2((k+m|k))$. A simple example is shown in the figure below.

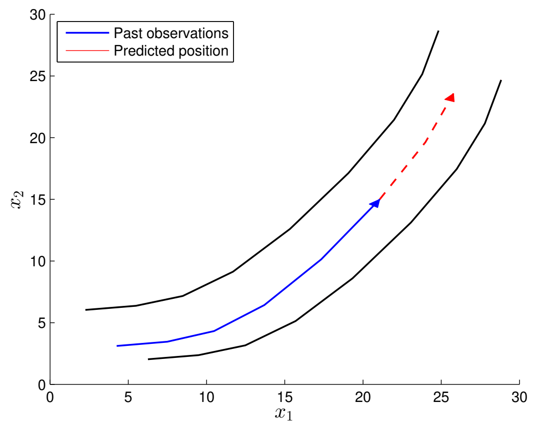


Figure 1.1: A trajectory prediction problem

The approach followed in this dissertation to the problem mentioned above is to handle each trajectory coordinate as a time series. As a result, the problem is simplified to multiple time series prediction problems. However, to solve a time series prediction problem in a computational tractable way with a least squares criterion, the series must become stationary. This condition is achieved through differentiation, or seasonal differentiation. The prediction problem is mathematically solved through a least squares criterion, which is further explained in Chapter 3. Its solution allows to compute an estimate of the future values for the time series as a linear combination of certain terms and coefficients. After having computed the forecast of the stationary series, the inverse process (integration) is executed in order to predict the original time series. Finally, an estimate of the future position of the trajectory is obtained. It is remarked that, according to the approach proposed it does not minimize the mean square error.

Even though the prediction problem has a straightforward linear solution, new problems arise regarding which coefficients values to use. To deal with this issue, a multiple model architecture was develop in Chapter 4. Since future positions are estimated online, the multiple model architecture allows to choose the best coefficients to use at each point in time, thus adapting the predicted target trajectory to the preceding movement. Because of this adaptive behavior, better results can be achieved in case the target changes the pattern of its trajectories.

Although only two dimensions are required to describe a 2D trajectory, this problem and its respective solution could be generalized for n-dimensions. The viability of the solution relies on the fact that each coordinate is predicted independently, thus the solution to the problem would be the same.

1.5 Dissertation Outline

This document is organized as follows. Chapter 2 provides a review of the models used in this dissertation and explains how they are used to generate synthetic trajectories. Then, the solution for the prediction problem is presented in Chapter 3, as well as some examples of single coordinate predictions. In Chapter 4 the multiple model architecture is presented along with the necessary algorithms to estimate the models parameters. Chapter 5 is dedicated to the presentation of results obtained with synthetic and real data. Finally, Chapter 6, draws conclusions on the experimental results, and some considerations about possible future work is presented.

Chapter 2

Models

2.1 Understanding the models

This section provides a review on some mathematical background knowledge regarding the models used in this dissertation, as well as some nomenclature required to fully understand and further replicate this work.

2.1.1 Discrete-time Gaussian white noise

A stochastic process represents a system of multiple random variables evolving through time. More specifically, the discrete-time white noise is a stochastic process consisting in a sequence of independent and identically distributed (i.i.d.) random variables. This means that each variable is mutually independent from others and all variables have the same probability distribution.

Let $e(k)$ be a signal describing a discrete-time Gaussian white noise. Consequently, the probability distribution of the random variables is a Gaussian (or Normal) distribution, with mean of $\mu = 0$ and variance σ^2,

$$e(k) \sim \mathcal{N}(\mu = 0,\, \sigma^2)\,, \forall\, k \in Z\,.$$

2.1.2 Shift operators

Let $y(k)$ be a random discrete-time signal of finite length K. Let the forward shift operator q^n be defined as

$$y(k)\, q^n = y(k+n)\ ,$$

and the backwards shift operator q^{-n} defined as

$$y(k)\, q^{-n} = y(k-n)\ .$$

2.1.3 Reciprocal polynomial

Let $\alpha(q)$ be a generic polynomial of order n

$$\alpha(q) = q^n + \alpha_1\, q^{n-1} + \cdots + \alpha_n\ ,$$

and his reciprocal polynomial be

$$\alpha^*(q^{-1}) = 1 + \alpha_1 \, q^{-1} + \cdots + \alpha_n \, q^{-n} \ ,$$

hence,

$$\alpha(q) = \alpha^*(q^{-1}) \, q^n \ .$$

2.1.4 Stationary models (AR, MA and ARMA)

Time series can be considered as stochastic processes [11]. These processes can be represented by models that take into account an intrinsic feature of a time series, that is, the dependence between adjacent observations.

Stationary models assume that the process remains in statistical equilibrium, thus the probabilistic properties do not change over time. In practice, this means that the process has a fixed mean and constant variance. Three of these stationary models are considered in this subsection.

The Auto Regressive (AR) model, described by

$$y(k) = -a_1 \, y(k-1) - \cdots - a_{n_a} \, y(k-n_a) + e(k) \ , \tag{2.1}$$

is a model in which the present value $y(k)$ is a linear sum of n_a past values of itself, plus an error. The Moving Average (MA) model, described by

$$y(k) = e(k) + c_1 \, e(k-1) + \cdots + c_{n_c} \, e(k-n_c) \ , \tag{2.2}$$

is a model in which the present value $y(k)$ is a linear sum of the error and its n_c previous values, assuming a zero mean. The Auto Regressive Moving Average (ARMA) model is a combination of these two previous models and it can described by

$$y(k) = -a_1 \, y(k-1) - \cdots - a_{n_a} \, y(k-n_a) + e(k) + c_1 \, e(k-1) + \cdots + c_{n_c} \, e(k-n_c) \ . \tag{2.3}$$

Using the shift operator q, one can rewrite the previous expression as

$$y(k) \, (\, 1 + a_1 \, q^{-1} + \cdots + a_{n_a} \, q^{-n_a} \,) = e(k) \, (\, 1 + c_1 \, q^{-1} + \cdots + c_{n_c} \, q^{-n_c} \,) \ . \tag{2.4}$$

2.1.5 Non-stationary models: Extending ARMA to ARIMA and SARIMA

Time series can exhibit non-stationary behavior, particularly when the mean varies over time. Nevertheless, such series can be transformed using methods of differentiation, in order to achieve a stationary series. To model a series with this nature, we look to integrate a stationary process in order to represent the non-stationary behavior of the series. This integration can be expressed via the shift operator

$$\frac{1}{(1-q^{-1})^d} \ ,$$

in which d is the order of the integration. It is by using this method in cascade with the ARMA model that we represent the Auto Regressive Integrated Moving Average (ARIMA) model.

A specific type of non-stationary behavior can be analyzed when there is a repeating pattern within the series. To achieve a stationary series, a seasonal differentiation is performed, taking the length of the pattern into account. To model this series a different kind of integration is used that can be described as

$$\frac{1}{1-q^{-s}} \, ,$$

in which s is the number of time steps for one seasonal period, or simply the length of the pattern. It is by using this method in cascade with the ARIMA model that we represent the Seasonal Auto Regressive Integrated Moving Average (SARIMA) model.

2.2 Simulating synthetic trajectory data

In this section we will be studying the mechanisms used to simulate ARMA, ARIMA and SARIMA processes. It is through these mechanisms that synthetic trajectory data is generated and ultimately used to describe the dynamics of a moving target trajectory.

2.2.1 Overview of the simulating mechanism

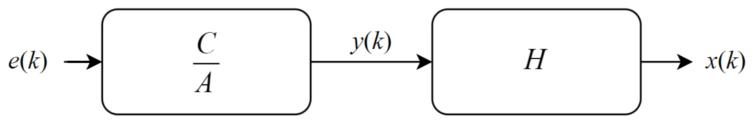


Figure 2.1: The model used for simulation

The diagram displayed in figure 2.1 essentially describes the model used to generate one of the trajectory dimensions.

It begins by generating $e(k)$, a white noise signal with zero mean and variance σ^2. This signal passes through a stable linear system C/A, resulting in $y(k)$, a stationary signal simulating an ARMA process. Finally, $y(k)$ goes through another system that we call H, with the purpose of making the output $x(k)$, a non-stationary signal, simulating either ARIMA or SARIMA processes.

Using this procedure we simulate a non-stationary process, describing a coordinate of a random trajectory. Finally, it is the combination of several outputs of this mechanism that generates the synthetic, or simulated, trajectory data.

2.2.2 The linear system and its relationship to ARMA processes

Various stationary stochastic processes can be generated by using white noise as an input of a linear system. Consider the linear system C/A which is a quotient between two polynomials, $C(q)$ and $A(q)$, of order n_c and n_a respectively.

$$y(k) = \frac{C(q)}{A(q)}\, e(k) \, , \tag{2.5}$$

expanding the previous equation (2.5)

$$y(k) = \frac{q^{n_c} + c_1\, q^{n_c-1} + \cdots + q_{n_c-1}\, q + c_{n_c}}{q^{n_a} + a_1\, q^{n_a-1} + \cdots + a_{n_a-1}\, q + a_{n_a}}\, e(k) \, . \tag{2.6}$$

This system can be represented in terms of the Z-transform. Taking the equations (2.5) and (2.6) one gets the following equations

$$Y(z) = \frac{C(z)}{A(z)}\, E(z)\ , \tag{2.7}$$

$$Y(z) = \frac{z^{n_c} + c_1\, z^{n_c-1} + \cdots + z_{n_c-1}\, z + c_{n_c}}{z^{n_a} + a_1\, z^{n_a-1} + \cdots + z_{n_a-1}\, z + a_{n_a}}\, E(z)\ . \tag{2.8}$$

Consider now, the reciprocal polynomials of $C(q)$ and $A(q)$,

$$C^*(q^{-1}) = 1 + c_1\, q^{-1} + \cdots + c_{n_c}\, q^{-n_c}\ ,$$

$$A^*(q^{-1}) = 1 + a_1\, q^{-1} + \cdots + a_{n_a}\, q^{-n_a}\ .$$

Using the previous reciprocal polynomials, we can rewrite the equation describing the ARMA process (2.4) as

$$y(k)\, A^*(q^{-1}) = e(k)\, C^*(q^{-1}) \Leftrightarrow y(k) = \frac{C^*(q^{-1})}{A^*(q^{-1})}\, e(k)\ . \tag{2.9}$$

Taking into account equations (2.5) and (2.9), one can write their relationship as

$$y(k) = \frac{C(q)}{A(q)}\, e(k) \Leftrightarrow y(k) = \frac{C^*(q^{-1})\, q^{n_c}}{A^*(q^{-1})\, q^{n_a}}\, e(k) \Leftrightarrow y(k) = q^{-n}\, \frac{C^*(q^{-1})}{A^*(q^{-1})}\, e(k) \tag{2.10}$$

with $n = n_a - n_c$, assuming a causal system, when $n \geq 0$. Therefore, we can write this last equation (2.10) in terms of the Z-transform

$$Y(z) = \frac{C(z)}{A(z)}\, E(z) \Leftrightarrow Y(z) = z^{-n}\, \frac{C^*(z^{-1})}{A^*(z^{-1})}\, E(z)\ . \tag{2.11}$$

2.2.3 The system that simulates non-stationarity

The previously discussed linear system C/A can only generate stationary signals. Hence, in order to generate processes that are not stationary, the system H is used in cascade with the linear system, corresponding to a convolution in the time domain or a multiplication in the Z domain. The output of the system H is the signal $x(k)$, or in terms of the Z-transform $\mathcal{Z}\{x(k)\} = X(z)$,

$$X(z) = H(z)\ \cdot Y(z)\ . \tag{2.12}$$

The $H(z)$ system provides some sort of integration of the stationary signal $y(k)$. The main two formats this system can take are based on the ARIMA and the SARIMA models. This will be seen in detail in the following sub-subsections:

2.2.3.1 Simulating an ARIMA process: The "double integrator"

In order to simulate an ARIMA process, of integration order d, the system H can be written as

$$H(z) = \frac{1}{(1 - z^{-1})^d}\ . \tag{2.13}$$

When $d = 2$ the system becomes particularly useful to generate synthetic trajectory data

$$H(z) = \frac{1}{(1 - z^{-1})^2}\ . \tag{2.14}$$

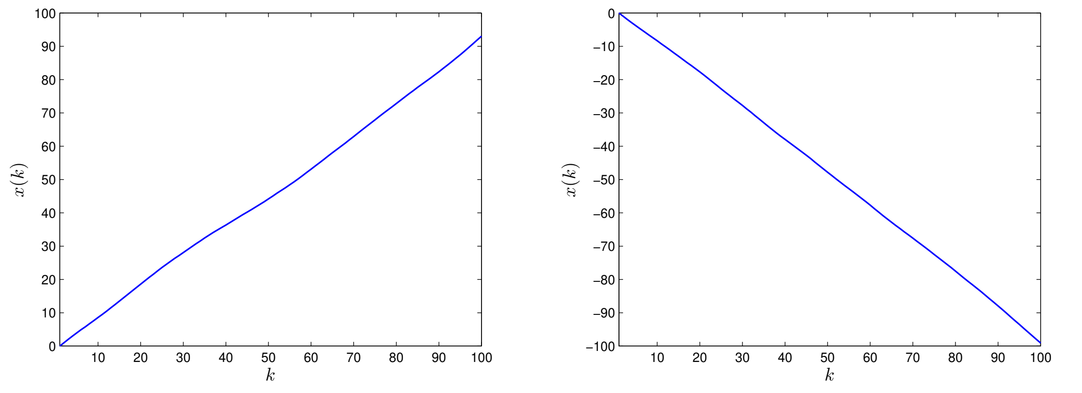
The particularly case of the system above describes a signal that exhibits a steadily increase, or decrease, over time, thus, it can be used to describe linear trajectories. We call this system the "double integrator", because it is a multiplication of two simple integrators. By replacing the previous equation (2.14) it in equation (2.12) we get

$$X(z) = H(z) \cdot Y(z) \Leftrightarrow X(z) = \frac{1}{(1 - z^{-1})^2} \cdot Y(z) \Leftrightarrow X(z) = \frac{1}{1 - 2z^{-1} + z^{-2}} \cdot Y(z) \ , \tag{2.15}$$

and we can further compute the inverse Z-transform to obtain the time domain equation

$$\mathcal{Z}^{-1}\{X(z) = \frac{1}{1 - 2z^{-1} + z^{-2}} \cdot Y(z)\} \Leftrightarrow x(k) = 2x(k-1) - x(k-2) + y(k) \ . \tag{2.16}$$

This system has two initial conditions ($x(k = 1); \ x(k = 2)$). It is by adjusting its values that we can determine the increase, or the decrease of the signal $x(k)$.



(a) Increasing initial conditions: $x(1) = 0; \ x(2) = 1$ (b) Decreasing initial conditions: $x(1) = 0; \ x(2) = -1$

Figure 2.2: Examples of the signal $x(k)$ using the "double integrator" system

Both examples in figure 2.2 have 2 initial conditions. In order to achieve these results the input signal $y(k)$ was generated with a Gaussian noise $e(k)$ with a small variance $\sigma^2 = 0,05$. The variance of the signal $e(k)$, must be much lower than the difference between the two initial conditions $\sigma^2 << x(2) - x(1)$ by at least one order of magnitude. This is relevant in order to properly define the trajectory we wish to simulate.

2.2.3.2 Simulating a SARIMA process: The "repeater"

Consider the system H written as

$$H(z) = \frac{1}{1 - z^{-s}} \ , \tag{2.17}$$

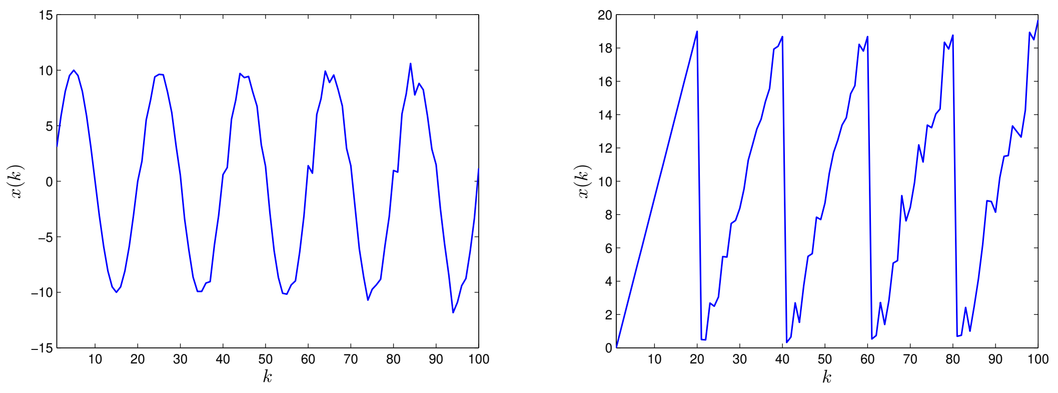
in which s is the number of time steps for one seasonal period. In other to simulate a SARIMA process, this system has to be in cascade with the system described in equation 2.13. Although the system 2.17 can only provide a seasonal integration, it is very useful by its own to describe circular trajectories. Thus, only SARIMA processes with integration order 0 are simulated. By replacing 2.17 in equation 2.12 we get

$$X(z) = H(z) \cdot Y(z) \Leftrightarrow X(z) = \frac{1}{1 - z^{-m}} \cdot Y(z) \ , \tag{2.18}$$

further computing the inverse Z-transform, we obtain the time domain equation

$$\mathcal{Z}^{-1}\{X(z) = \frac{1}{1 - z^{-s}} \cdot Y(z)\} \Leftrightarrow x(k) = x(k - s) + y(k) \ . \tag{2.19}$$

When $s = 1$ the system corresponds to a simple integrator with a pole in $z = 1$, when $s > 1$, there are s poles on the unit circle. This results in having a periodic signal $x(k)$ at the output. Thus, it is a good system to describe circular trajectories. We call this system the “repeater”, because it generates a repeating pattern that mimics the initial conditions $(x(k), k = \{1, \dots, s\})$.



(a) Initial conditions of a sinusoidal function

(b) Ascending initial conditions

Figure 2.3: Examples of the output signal when using the repeater with different initial conditions

Figure 2.3 shows two different examples of $x(k)$, each with 20 different initial conditions. In order to achieve these results the input signal $y(k)$ was generated with a Gaussian noise $e(k)$ with a variance $\sigma^2 = 1$.

2.2.4 Examples of trajectories

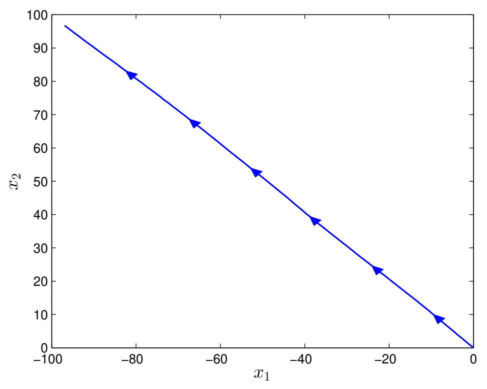
The final output of the mechanism, $x(k)$, represents one coordinate of a trajectory. In order to simulate a trajectory in two dimensions, two outputs must be generated for each trajectory. Some examples of trajectories that can be generated using these mechanisms are explained and displayed below.

2.2.4.1 Straight lines

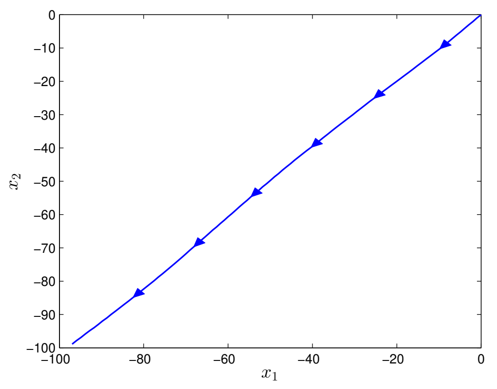
In order to generate straight line trajectories we use the “double integrator” system (as in figure 2.2) for each coordinate. We can generate lines in any direction by using different initial conditions in the system. Some examples are shown in figure 2.4.

2.2.4.2 Circumferences and ellipses

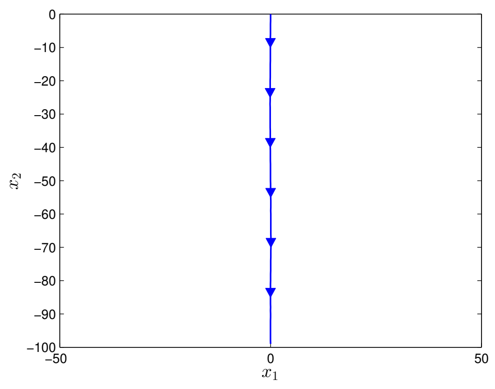
In order to generate circular trajectories we use the “repeater” system (as in figure 2.3(a)) for each coordinate. We can generate circular or elliptical trajectories by changing the initial conditions in each system. In order to simulate noisy trajectories, we can also increase the variance σ^2 of the Gaussian noise $e(k)$ in each system. Examples are shown in figure 2.5.



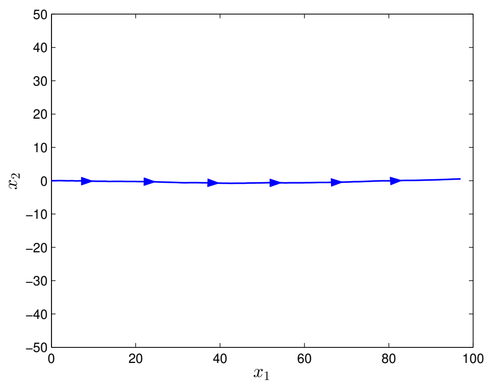
(a) Northwest direction



(b) Southwest direction

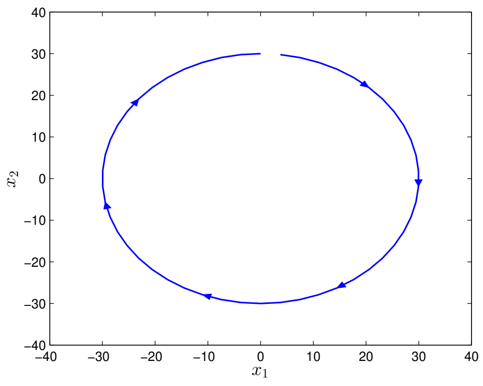


(c) South direction

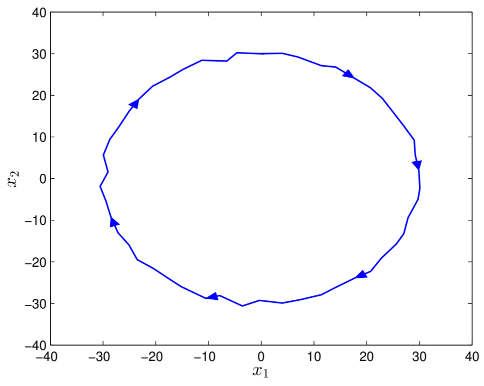


(d) East direction

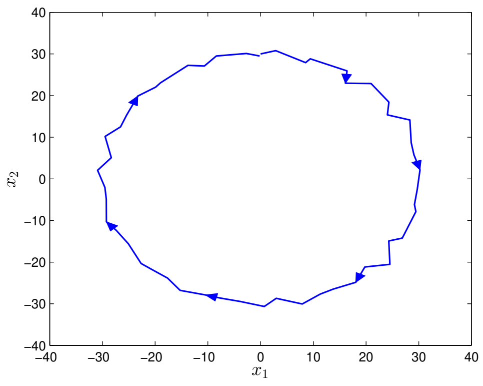
Figure 2.4: Examples of straight line trajectories



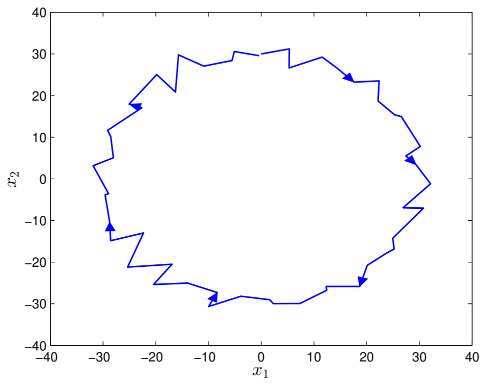
(a) Initial conditions of circular trajectory



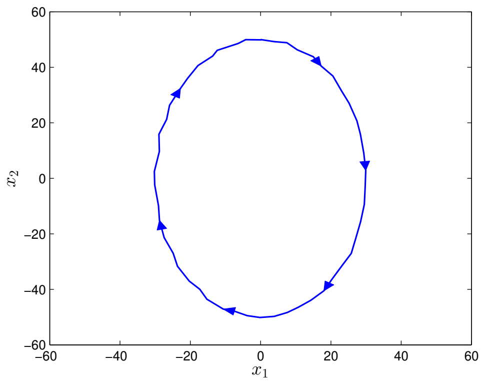
(b) Circular trajectory ($\sigma^2 = 0,5$)



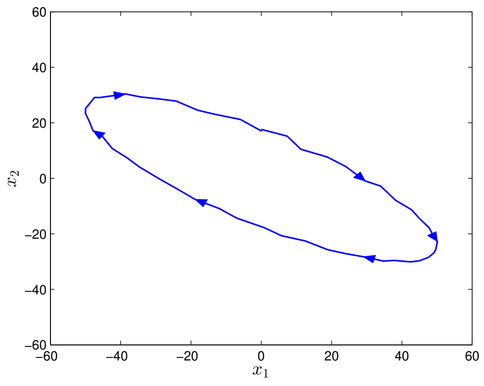
(c) Circular trajectory ($\sigma^2 = 1$)



(d) Circular trajectory ($\sigma^2 = 2$)



(e) Elliptical trajectory ($\sigma^2 = 0,5$)

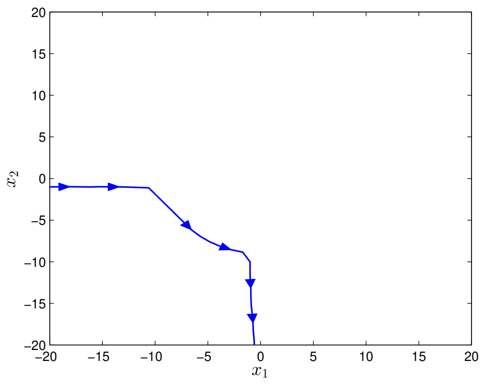


(f) A different elliptical trajectory ($\sigma^2 = 0,5$)

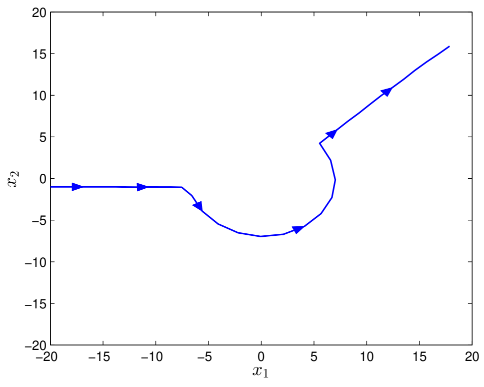
Figure 2.5: Examples of circular and elliptical trajectories

2.2.4.3 Trajectory commutation

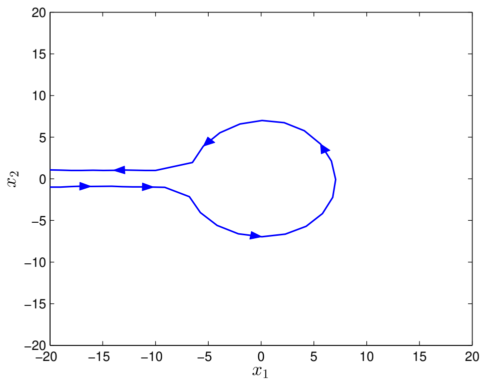
In order to simulate more complex movement, various line trajectories and circular trajectories are first generated. The different parts of the trajectories are then linked to simulate a target switching its movement. Some examples are shown in figure 2.6.



(a) First example



(b) Second example



(c) Third example

Figure 2.6: Examples of some more complex trajectories

Chapter 3

Prediction

3.1 Overview of the prediction procedure

The diagram in figure 3.1 displays the procedure used to estimate a prediction of a trajectory coordinate. In this section, an overview of this procedure is provided.

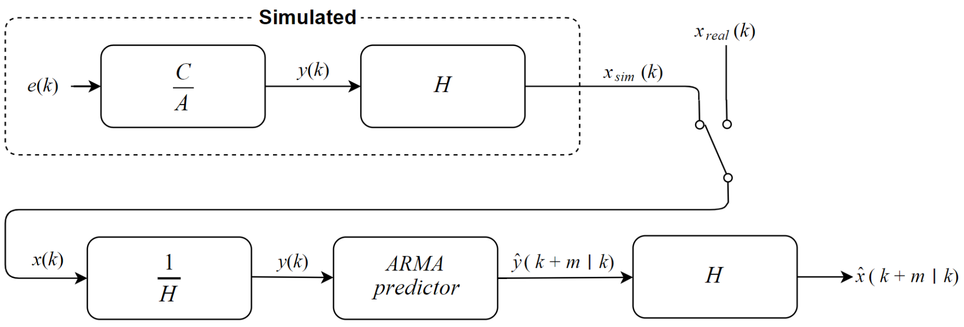


Figure 3.1: The prediction procedure

The procedure mentioned relies on observations of the signal $x(k)$, which can be simulated by the model described in Chapter 2 ($x_{sim}(k)$), or can be obtained from a trajectory in a real dataset ($x_{real}(k)$). Since the signal $x(k)$ is non-stationary, it is passed through the inverse of the H system, $1/H$, to obtain the stationary signal $y(k)$. Assuming $y(k)$ follows an ARMA process, an ARMA predictor is used to mathematically predict an estimate of $y(k+m)$, where m is the amount of time steps in the future we want to predict. This estimate is represented as $\hat{y}(k+m|k)$. The last system H is used in order to "regain" the signal non-stationarity. Finally, the future value of the trajectory coordinate is estimated and it is represented as $\hat{x}(k+m|k)$.

3.2 The system that removes the non-stationarity

The system H described in Chapter 2 section 2.2 uses some form of integration in order to simulate non-stationary behavior. In this section, we explain how to use the different forms of the inverse of the system H to transform the non-stationary signal $x(k)$ to the stationary signal $y(k)$, by means of differentiation. The equation of the system $1/H$ can be described in terms of the Z-transform by the following equation

$$Y(z) = \frac{1}{H(z)} \cdot X(z) \ . \tag{3.1}$$

3.2.1 The inverse of the “double integrator”

The “double integrator” system described in equation (2.14) models ARIMA processes of integration order $d = 2$. The inverse of this system is written as

$$\frac{1}{H(z)} = (1 - z^{-1})^2 \ . \tag{3.2}$$

By replacing the previous equation (3.2) in equation (3.1) we get

$$Y(z) = \frac{1}{H(z)} \cdot X(z) \Leftrightarrow Y(z) = (1 - z^{-1})^2 \cdot X(z) \Leftrightarrow Y(z) = (1 - 2z^{-1} + z^{-2}) \cdot X(z) \ , \tag{3.3}$$

and we can further compute the inverse Z-transform to obtain the time domain equation

$$\mathcal{Z}^{-1}\{Y(z) = (1 - 2z^{-1} + z^{-2}) \cdot X(z)\} \Leftrightarrow y(k) = x(k) - 2x(k-1) + x(k-2) \ . \tag{3.4}$$

3.2.2 The inverse of the “repeater”

The “repeater” system described in equation (2.17) models SARIMA processes of integration of order $d = 0$. The inverse of this system is written as

$$\frac{1}{H(z)} = 1 - z^{-s} \ . \tag{3.5}$$

By replacing the previous equation (3.5) in equation (3.1) we get

$$Y(z) = \frac{1}{H(z)} \cdot X(z) \Leftrightarrow Y(z) = (1 - z^{-s}) \cdot X(z) \ , \tag{3.6}$$

and to obtain the time domain equation, we use the inverse Z-transform

$$\mathcal{Z}^{-1}\{Y(z) = (1 - z^{-s}) \cdot X(z)\} \Leftrightarrow y(k) = x(k) - x(k-s) \ . \tag{3.7}$$

3.3 The ARMA predictor

3.3.1 The prediction problem

Let $y(k+m)$ be the future value of $y(k)$ after m ($m \geq 1$) steps, and let $\hat{y}(k+m|k)$ be the estimate of $y(k+m)$ given the observations until the instant k, O^k. The prediction problem consists in determining $\hat{y}(k+m|k)$ that minimizes the variance of the steady state error of the prediction

$$\underset{\hat{y}(k+m|k)}{\text{minimize}} \quad E[(y(k+m) - \hat{y}(k+m|k))^2 | O^k] \ . \tag{3.8}$$

3.3.2 Solution of the prediction problem

Going back to the equation (2.9), describing an ARMA process, we can write

$$y(k+m) = \frac{C^*}{A^*} \, e(k+m) \ . \tag{3.9}$$

Considering the long division between the reciprocal polynomials C^* and A^* (where A^* is a non-zero polynomial), we get

$$C^*(z^{-1}) = A^*(z^{-1}) * F_m^*(z^{-1}) + z^{-m}\, G_m^*(z^{-1}) \Leftrightarrow \frac{C^*(z^{-1})}{A^*(z^{-1})} = F_m^*(z^{-1}) + z^{-m}\, \frac{G_m^*(z^{-1})}{A^*(z^{-1})} \;, \tag{3.10}$$

in which F_m^* is the quotient polynomial of order $m-1$

$$F_m^*(z^{-1}) = 1 + f_1\, z^{-1} + f_2\, z^{-2} + \dots + f_{m-1}\, z^{-(m-1)} \;, \tag{3.11}$$

and $z^{-m}\, G_m^*(z^{-1})$ is the remainder polynomial, where $z^{-m}\, \frac{G_m^*(z^{-1})}{A^*(z^{-1})}$ is defined as

$$z^{-m}\, \frac{G_m^*(z^{-1})}{A^*(z^{-1})} = f_m\, z^{-m} + f_{m+1}\, z^{-(m+1)} + \dots \;. \tag{3.12}$$

Replacing (3.10) in equation (3.9), we get

$$y(k+m) = F_m^*(z^{-1})\, e(k+m) + \frac{G_m^*(z^{-1})}{A^*(z^{-1})}\, e(k) \;, \tag{3.13}$$

that enables a distinct separation of the equation in two parts: The part that depends only on future error values

$$F_m^*(z^{-1})\, e(k+m) = e(k+m) + f_1\, e(k+m-1) + f_2\, e(k+m-2) + \dots + f_{m-1}\, e(k+1) \;, \tag{3.14}$$

and the part that depends only on the innovations until the present instant k

$$\frac{G_m^*(z^{-1})}{A^*(z^{-1})}\, e(k) = f_m\, e(k) + f_{m+1}\, e(k-1) + \dots \;, \tag{3.15}$$

thus separating it in two uncorrelated parts.

In order to obtain the expression for the optimal predictor, consider now the equation for the prediction problem (3.8) and replace with the equation (3.13)

$$\underset{\hat{y}(k+m|k)}{\text{minimize}} \quad E\left[\left(\frac{G_m^*(z^{-1})}{A^*(z^{-1})}\, e(k) - \hat{y}(k+m|k) + F_m^*(z^{-1})\, e(k+m)\right)^2 |O^k\right] \;, \tag{3.16}$$

In order to further simplify the equations let A be

$$A = \frac{G_m^*(z^{-1})}{A^*(z^{-1})}\, e(k) - \hat{y}(k+m|k) \;,$$

and B equal to

$$B = F_m^*(z^{-1})\, e(k+m) \;.$$

The equation (3.16) simplifies to minimize the following expression

$$E[(A+B)^2|O^k] \;,$$

expanding the square and further simplifying the expression we get

$$E[A^2 + 2\,A\,B + B^2|O^k] \Leftrightarrow E[A^2|O^k] + E[B^2|O^k] + 2\,E[A\,B|O^k] \;.$$

Parts A and B are uncorrelated as explained above, (3.14) and (3.15), and each part depends on the error signal $e(k)$ that by definition of white noise has zero mean, making the average of the product of the two parts zero. The expression is simplified to

$$E[A^2|O^k] + E[B^2|O^k] \ . \tag{3.17}$$

The expected value of a term that depends on prior observations, when given those observations is the term itself

$$E[A^2|O^k] = A^2 \ ,$$

and the expected value of a term that depends on future observations, is the same when given prior observations,

$$E[B^2|O^k] = E[B^2] \ ,$$

further simplifying the expression (3.17) to

$$A^2 + E[B^2] \ .$$

Replacing A and B in the expression

$$\underset{\hat{y}(k+m|k)}{\text{minimize}} \quad \left(\frac{G_m^*(z^{-1})}{A^*(z^{-1})}\, e(k) - \hat{y}(k+m|k) \right)^2 + E[(F_m^*(z^{-1})\, e(k+m))^2] \ , \tag{3.18}$$

the minimum value is achieved when the left squared term is zero, since it is the only that depends on $\hat{y}(k+m|k)$. Therefore the optimal predictor equation will be

$$\hat{y}(k+m|k) = \frac{G_m^*(z^{-1})}{A^*(z^{-1})}\, e(k) \ . \tag{3.19}$$

Going back to equation (2.10) we can rewrite the previous equation in a way that depends on $y(k)$

$$\hat{y}(k+m|k) = \frac{G_m^*(z^{-1})}{C^*(z^{-1})}\, y(k) \ . \tag{3.20}$$

3.3.3 Variance of the prediction error

Considering the variance of the steady state error of the prediction, or simply the variance of the prediction error, allow us to have a measure of the quality of the prediction. The expression was obtained when finding the optimal predictor, that is the minimum value of the expression in (3.18)

$$E[(F_m^*(z^{-1})\, e(k+m))^2] \ .$$

Considering the equation (3.14)

$$E[(F_m^*(z^{-1})\, e(k+m))^2] = E[e(k+m) + f_1\, e(k+m-1) + f_2\, e(k+m-2) + \cdots + f_{m-1}\, e(k+1)] \ , \tag{3.21}$$

and considering that the error signal $e(k)$ has mean zero and $E[e^2(k)] = \sigma^2$, the variance of the prediction error is

$$\sigma_{error}^2 = (\, 1 + f_1^2 + f_2^2 + \cdots + f_{m-1}^2)\, \sigma^2 \ , \tag{3.22}$$

and the standard deviation of the prediction error is given by

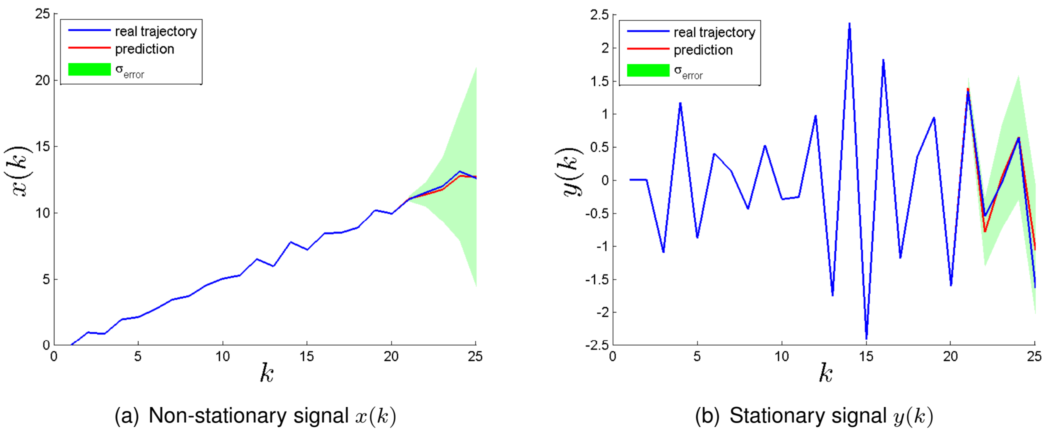
$$\sigma_{error} = \sqrt{(\, 1 + f_1^2 + f_2^2 + \cdots + f_{m-1}^2)\, \sigma^2} \ . \tag{3.23}$$

3.4 Prediction example

In this section we aim to exemplify the prediction procedure displayed in figure 3.1 with the two different forms of the H system. The objective is to make predictions of the signal $x(k)$ a number m of steps in the future. We first simulate $x(k)$ with a length of $K+m$. Then, we feed the prediction procedure the simulated signal with a length of K, predicting m steps ahead, allowing us to compare the predicted and the real signal. The stationary signal $y(k)$ and the non-stationary signal $x(k)$ are displayed in order to give a clear understanding of the procedure. In both examples, we generate the same ARMA process with length $K = 20$ and predict $m = 5$ steps ahead. The results are shown in the subsections below.

3.4.1 Predicting ARIMA processes

In order to test the prediction of an ARIMA process, we simulate it using the "integrator" form of the system H. The process was generated with a Gaussian noise $e(k)$ with a variance of $\sigma^2 = 0,2$ and initial conditions $x(1) = 0;\ x(2) = 1$.

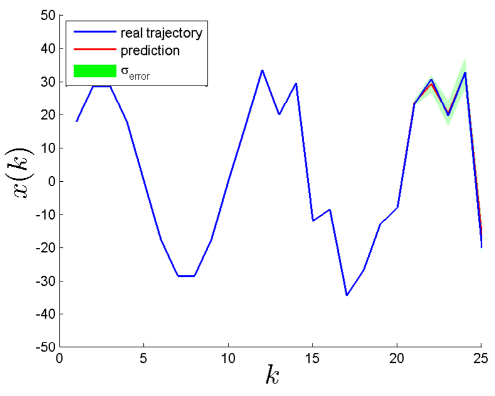


(a) Non-stationary signal $x(k)$ (b) Stationary signal $y(k)$

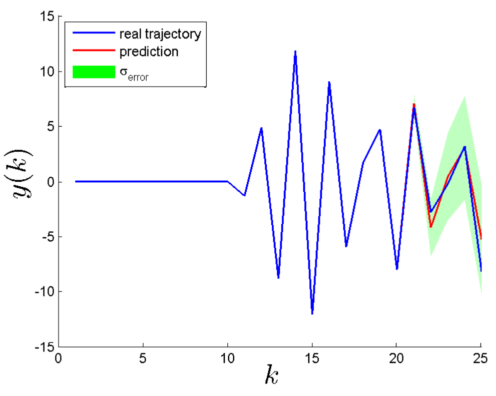
Figure 3.2: 5 step prediction of an ARIMA process

3.4.2 Predicting SARIMA processes

In order to test the prediction of a SARIMA process, we simulate it using the "repeater" form of the system H. The process was generated with a Gaussian noise $e(k)$ with a variance of $\sigma^2 = 1$ and 10 initial conditions $x(k), k = \{1, \ldots, 10\}$ that described a sinusoidal function.



(a) Non-stationary signal $x(k)$



(b) Stationary signal $y(k)$

Figure 3.3: 5 step prediction of a SARIMA process

Chapter 4

Model Switching

4.1 Multiple models architecture

This dissertation uses techniques from supervised multiple models, in order to predict trajectories at each instant (online). To this end, we propose the multiple models architecture, represented in figures 4.1 and 4.2.

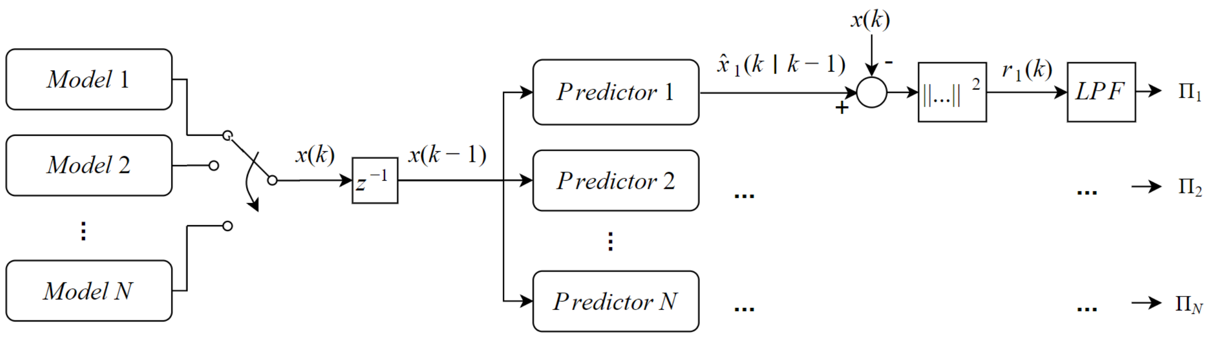


Figure 4.1: Multiple models architecture

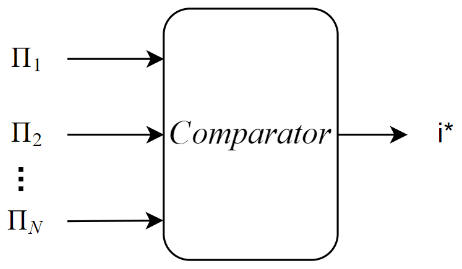


Figure 4.2: Choosing the model index i^* by comparing the score performances

A bank of N models is created *a priori* assuming that trajectory coordinates will follow one of the models. For each different model a predictor is formulated. We refer to a predictor as the entire prediction procedure (described in figure 3.1). The output of a model, the $x(k)$ signal, is lagged resulting in $x(k-1)$. Then, a one-step prediction is estimated to the lagged signal $x(k-1)$, obtaining $\hat{x}_1(k|k-1)$.

The squared error between the signal real value $x(k)$ and its prediction $\hat{x}_1(k|k-1)$ is computed, resulting in an error corresponding to each model $r_i(k)$, $i = 1, \ldots, N$. The error value is then used to compute a performance score Π_i to each model by a first order Low Pass Filter (LPF) according to

$$\Pi_i(k) = \alpha\, \Pi_i(k-1) + (1-\alpha)\, r_i(k)\ , \tag{4.1}$$

or by a second order LPF according to

$$\Pi_i(k) = 2\alpha\, \Pi_i(k-1) - \alpha^2\, \Pi_i(k-2) + (1-\alpha)^2\, r_i(k)\ , \tag{4.2}$$

where the LPF pole, $\alpha, (0 \leq \alpha \leq 1)$ is a parameter to select between 0 and 1.

Finally, a comparator selects the model index i^* with the lowest score value Π_i. The model index i^* is used to choose the corresponding model $Model_{i^*}$, as it is the best model to predict the future value, $\hat{x}_1(k+1|k)$.

4.2 Architecture parameters regulation

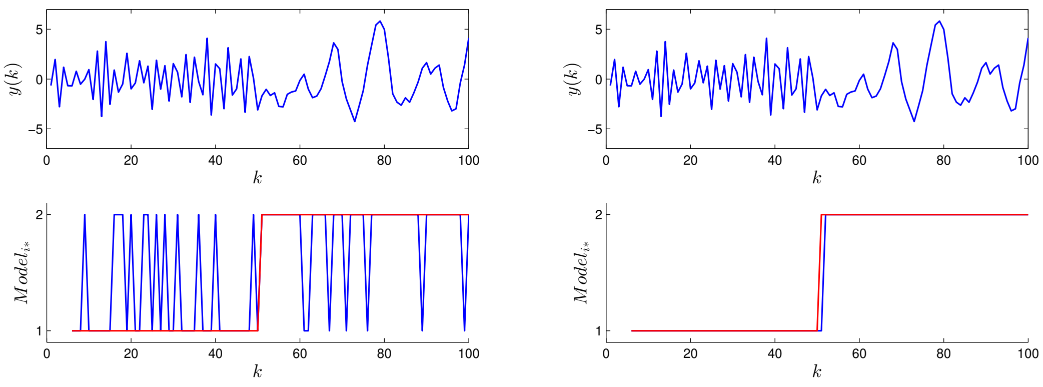
In this section we study the influence of several parameters of the multiple models architecture. Various examples of model identification are shown to justify the regulation of these parameters.

The models used to create the bank of models in the following experiments are stationary ARMA processes with different sets of coefficients. As it will be seen in Chapter 5 when working with real datasets, the non-stationarity of the processes becomes irrelevant, as trajectory coordinates are always differentiated twice. Therefore, we only consider stationary processes when regulating the architecture parameters.

In the following subsections, a bank of N models is used to simulate a $y(k)$ signal with length $K \times N$, where K is the length of a single model simulation. The models chosen at each instant are presented in blue and the models used to simulate the signal are presented in red. The percentage of correctly choosing the model used in the signal simulation is computed as a measure of accuracy

4.2.1 A simple example regarding the use of the LPF

In this experiment, the signal $y(k)$ has a bank with $N = 2$ different models, each simulated with a length of $K = 50$.



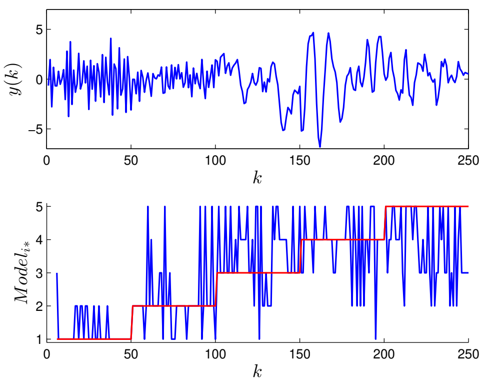
(a) Without LPF (b) With 1st order LPF

Figure 4.3: Influence of the LPF in model identification

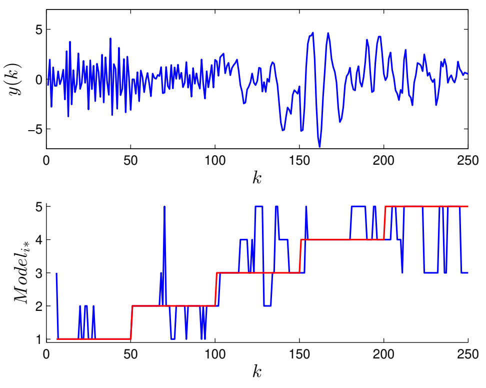
As it can be seen in figure 4.3, the LPF removes the high frequency commutation between the models. There is an initial period in which no model is selected, as the algorithm requires a certain number of values of the signal $y(k)$ to be able to predict it. In this simple example, a 98.95% accuracy was achieved using a first order LPF with $\alpha = 0.9$.

4.2.2 Influence of LPF pole

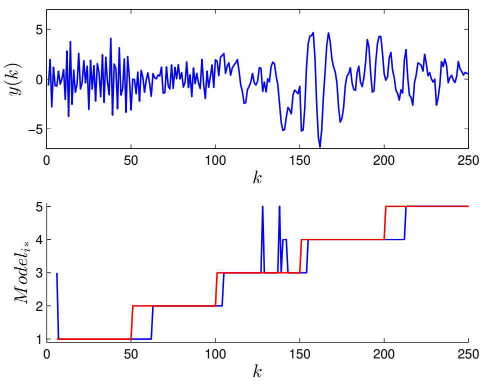
In this subsection, multiple experiments are performed with different values of α in a first order LPF. The signal $y(k)$ was simulated with a bank of $N = 5$ different models, each with a length of $K = 50$.



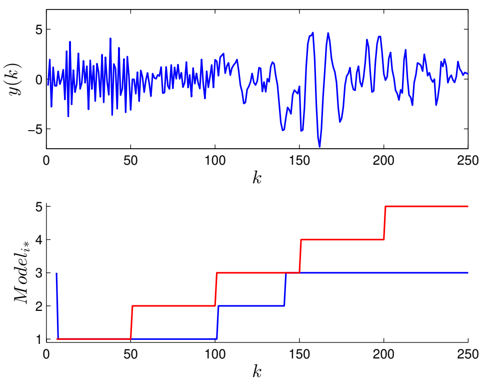
(a) $\alpha = 0.2$



(b) $\alpha = 0.7$



(c) $\alpha = 0.92$



(d) $\alpha = 0.99$

Figure 4.4: Various model switching examples with different values of α in a first order LPF

Figure 4.4 proves that the value of α clearly influences the accuracy of model identification. To better understand this influence, we compute the accuracy of model identification for α between 0 and 1 with an interval step of 0.01, $\{0, 0.01, 0.02, \ldots, 0.98, 0.99, 1\}$. This relation is represented in figure 4.5.

By analyzing 4.5 we see that the accuracy improves until a certain value, which is $\alpha = 0.92$, then drops suddenly as α approaches 1. This means there is an optimal value of α that maximizes the model identification accuracy, which we represent by α_o. The maximum achieved accuracy value was 84.49%

4.2.3 Influence of LPF order

In this subsection, we test the second order LPF in model identification. The bank used to simulate the signal $y(k)$ is the same as the one used in the previous subsection 4.2.2.

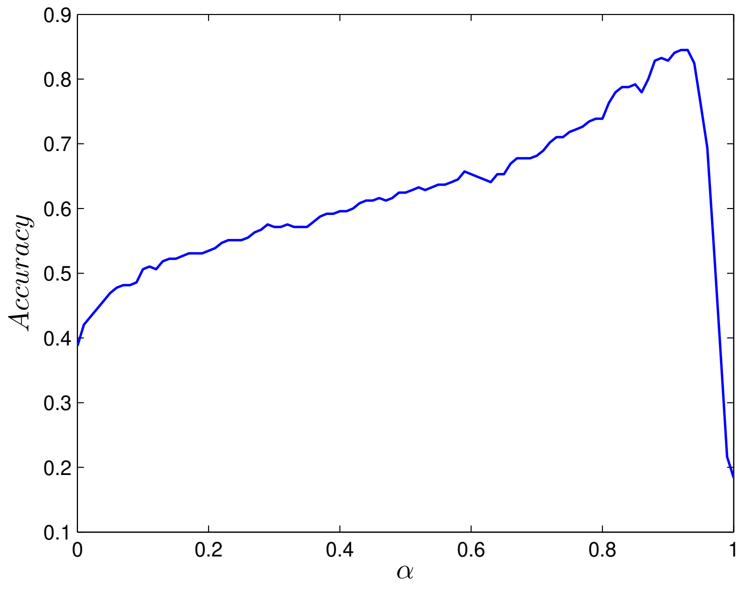


Figure 4.5: Influence of the parameter alpha in the first order LPF

In order to compare the first order LPF with the second order LPF the accuracy of model identification is computed for α values between 0 and 1 with an interval step of 0.01. The results are represented in figure 4.6.

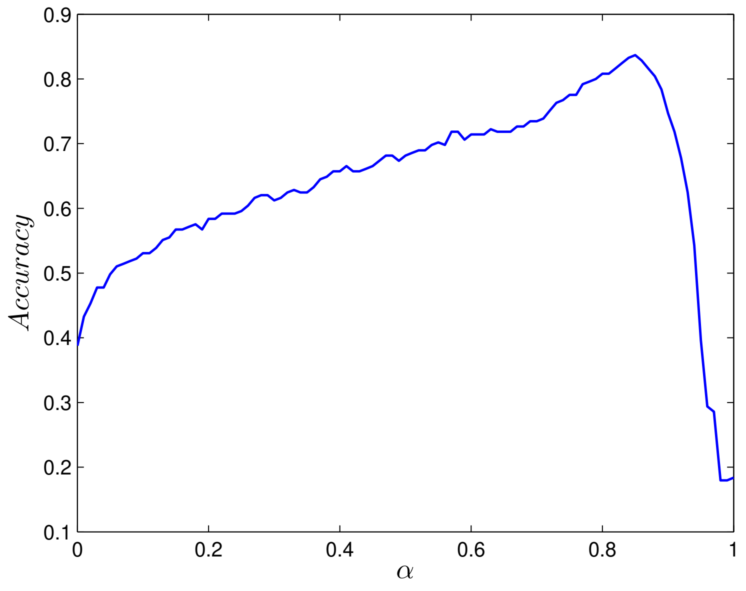


Figure 4.6: Influence of the parameter alpha in the second order LPF

Looking at figure 4.6 we can perceive a similar behavior to 4.5. The accuracy improves until a certain value and then drops for when it approaches 1. The maximum achieved accuracy value was 83.67% at $\alpha_o = 0.85$. Despite the fact that the maximum achieved accuracy using a first order LPF was a bit higher ($\approx 0.01\%$) than using a second order LPF, it is important to consider both orders when dealing with real data.

4.3 Creating the bank of models

To automate the creation of banks of models, an algorithm was developed to learn the best models in a set of trajectories (Training set). In this section we describe the algorithm and provide some background knowledge required to understand its procedure.

4.3.1 Akaike's Information Criterion

The Akaike's Information Criterion (AIC) [20] is a well known criteria used to compare different statistical models based on information theory. It provides a measure of the model quality by estimating the "amount" of lost information in a given model. Thus, the model with the lowest score is said to be the best model among the set. The chosen model will theoretically neither underfit or overfit the data.

4.3.2 Describing the algorithm

Given a set of trajectories we wish to create a bank of models for each coordinate. To automate this process the algorithm 1 was developed.

```
Data: Training set
Result: A bank of models for each coordinate
$number\_of\_coordinates \leftarrow$ Number of coordinates in a trajectory of the Training set
$number\_of\_trajectories \leftarrow$ Number of trajectories in the Training set
$a\_max \leftarrow$ Define maximum AR order
$c\_max \leftarrow$ Define maximum MA order
for $coordinate \leftarrow 1$ to $number\_of\_coordinates$ do
    for $trajectory \leftarrow 1$ to $number\_of\_trajectories$ do
        $process \leftarrow$ Get process of the corresponding $coordinate$ of the $trajectory$
        $stationary\_process \leftarrow$ Remove non-stationarity from $process$
        for $p \leftarrow 0$ to $a\_max$ do
            for $q \leftarrow 0$ to $c\_max$ do
                $model \leftarrow$ Learn the ARMA($p$,$q$) model from the $stationary\_process$
                $aic\_score \leftarrow$ Compute $model$ AIC score
            end
        end
        $best\_model \leftarrow$ Save the $model$ which scored the lowest $aic\_score$, in each $trajectory$
    end
    $models\_bank \leftarrow$ Save a bank consisting of the $best\_model$'s, of each $coordinate$
end
```

Algorithm 1: Pseudo-code of the algorithm that creates a bank of models for each coordinate

The pseudo-code written above describes the procedure through which a bank a models is created for each coordinate. The main idea is to model each trajectory coordinate to various ARMA processes, using the AIC to choose the best model available. The algorithm uses two pre-built matlab functions from the System Identification Toolbox: The *armax()* function, which allows to estimate model parameters by minimizing quadratic prediction error criterion, given the order of the AR component and the MA component; The *aic()* function, which allows to compute an AIC score, based on a given model.

Learning the models through the combination of these two functions as described in the algorithm 1 is computationally heavy, therefore not suitable to recursively estimate model coefficients online. This is why we create a bank of models offline for each coordinate, allowing to use the multiple models architecture described in this Chapter to choose the best model online.

Chapter 5

Experimental results

5.1 Metrics

In this section we explain the metrics that help us measure the quality of the prediction methods used.

5.1.1 Predicted Position Error

As described in Chapter 4, we choose the models based on their performance regarding a one-step coordinate prediction. To measure the performance of predicting future trajectory positions, we have to consider all coordinates. To that end, we compute the Euclidean distance between the one-step predicted position $(\hat{x}_1(k), \hat{x}_2(k))$, and its real value $(x_1(k), x_2(k))$. Let this be called Predicted Position Error (PPE) and be defined as

$$PPE(k) = \sqrt{(x_1(k) - \hat{x}_1(k))^2 + (x_2(k) - \hat{x}_2(k))^2}. \tag{5.1}$$

5.1.2 Recursive Predicted Position Error

To measure the error of recursively predicting an entire trajectory (of length K), we consider a mean of all PPE computed across the length of that trajectory. Let this be called Recursive Predicted Position Error (RPPE) and be defined as

$$RPPE = \frac{1}{K} \sum_{k=1}^{K} PPE(k). \tag{5.2}$$

5.1.3 Trajectory Set Error

When given a set of N trajectories, we want to be able to score that set, based on the quality of the predictions made to each trajectory. Thus, we consider a mean of each trajectory RPPE. Let this be called Trajectory Set Error (TSE) and be defined as

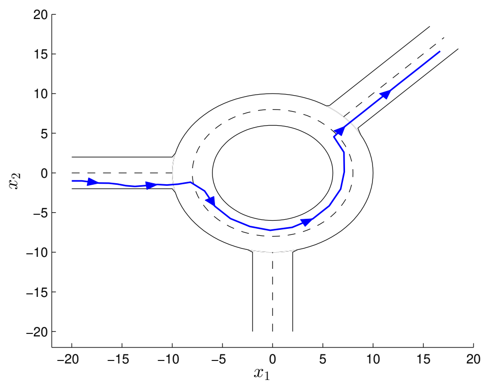
$$TSE = \frac{1}{N} \sum_{1}^{N} RPPE. \tag{5.3}$$

5.2 Synthetic data

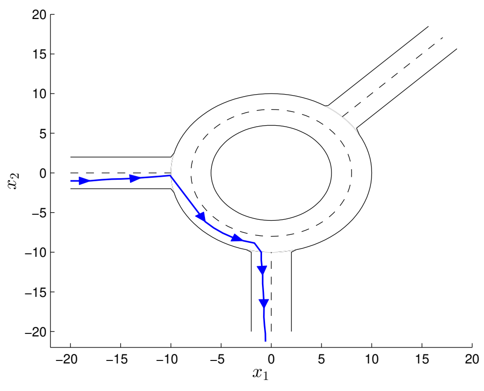
In this section we consider synthetic data to measure the performance of the proposed methods in the dissertation. For this reason, a dataset of trajectories is developed and some experiments are performed. The main objective in testing with synthetic data is to prove the viability of the proposed prediction methods in real applications.

5.2.1 Dataset

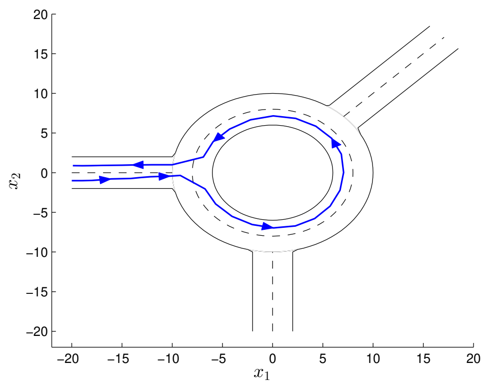
In order to test the methods proposed in this dissertation, a small dataset with synthetic trajectories was developed. In this set there are 3 different trajectories. Each one mimics a target (car, hypothetically) entering a roundabout and exiting at one of possible exits. To generate these trajectories we use the same methods as in Chapter 2 section 2.2. The dataset is represented in figure 5.1.



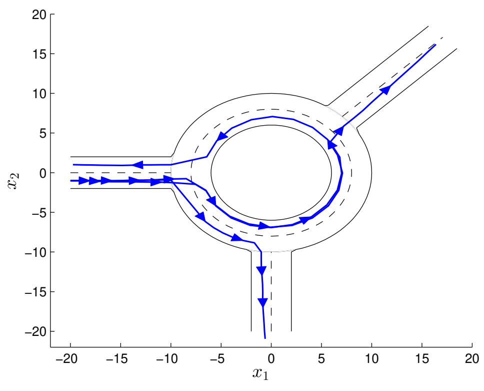
(a) Target exiting at the second exit



(b) Target exiting at the first exit



(c) Target exiting at the third exit



(d) All trajectories

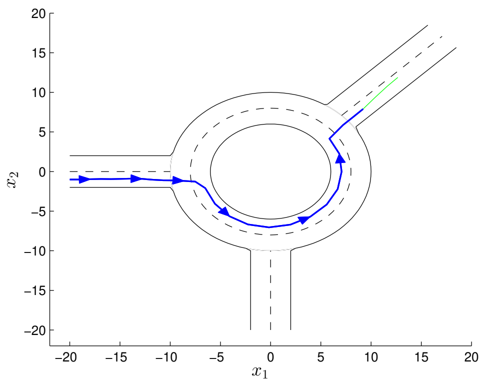
Figure 5.1: Synthetic dataset trajectories

The trajectories are composed of a linear motion (target entering or leaving the roundabout) and a circular motion (target going around the roundabout). Each of these motions is generated with a different pair of models (as seen in subsection 2.2.4) : $Motion_1$ - circular motion, generated with two “repeater” systems; $Motion_2$ - linear motion, generated with two “double integrator” systems. Hence, we refer to motion as the pair of models that supported its generation.

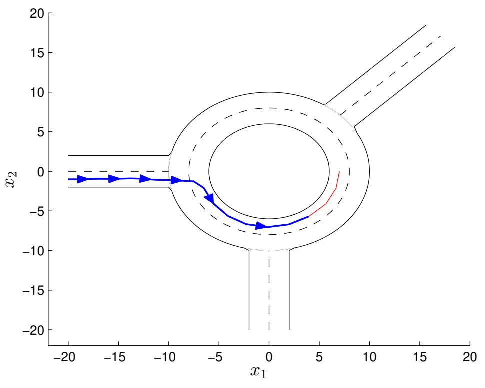
5.2.2 Experiment

The experiment conducted involves recursively predict the next future positions of a trajectory, by choosing the most appropriate models at that instant. Since there is only a pair of models to describe each motion, we can focus on choosing the correct motion at each instant.

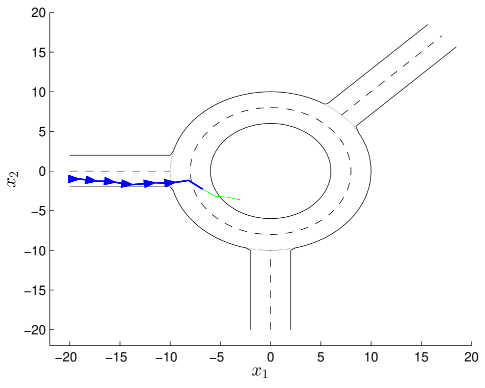
Consider predicting the trajectory represented in figure 5.1(a). To illustrate the predicted trajectory at each point in time, a 3-step prediction is made at each instant and its projected ahead of the trajectory, as represented in figure 5.2. The different colors of the prediction indicates the chosen motio:, the red line corresponds to $Motion_1$ and the green line corresponds to $Motion_2$.



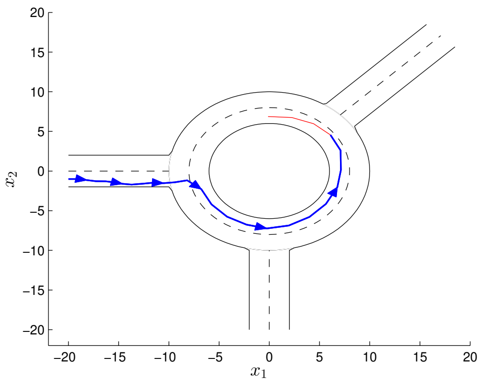
(a) Correct prediction using $Motion_2$



(b) Correct prediction using $Motion_1$



(c) Incorrect prediction using $Motion_2$



(d) Incorrect prediction using $Motion_1$

Figure 5.2: A 3-step prediction projection at different instants

The figure 5.2 shows a 3-step prediction of the chosen motion at different points in time. In both situations 5.2(a) and 5.2(b) the trajectory is correctly predicted. However, in situations 5.2(c) and 5.2(d) the trajectory is incorrectly predicted. Specifically, in the situation 5.2(c) the chosen motion is not identified correctly, therefore the prediction is incorrect. In 5.2(d) although the current motion is correctly identified, the target will switch motions in the next step, therefore predicted positions are incorrect.

In order to better understand the motion identification and the PPE regarding each motion across the length of the trajectory, we recursively predict one-step ahead positions and show the results in figure 5.3. The motion used for the generation is plotted in red, and the identified motion is plotted in blue.

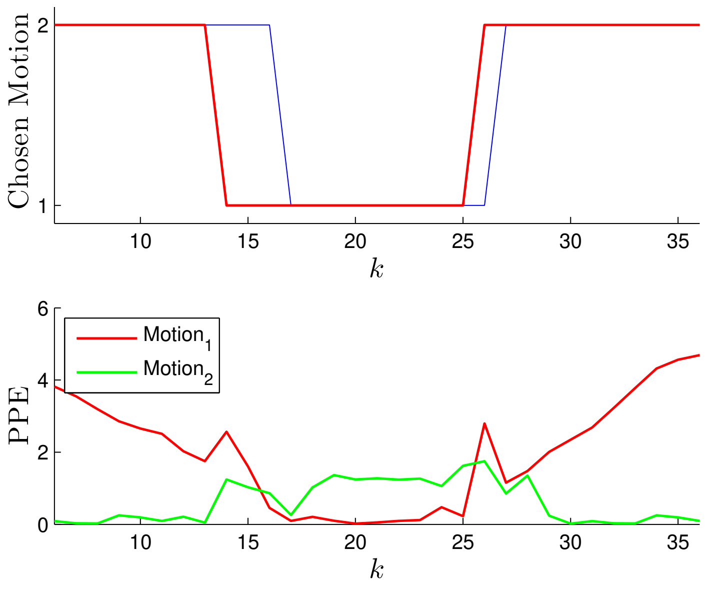


Figure 5.3: Motion identification and PPE regarding each motion across the length of the trajectory

5.2.3 Results

Through the analysis of the figure 5.3, we can see that the motion identification behavior is similar to the one seen in Chapter 4 subsection 4.2.1, which indicates an accurate identification. Regarding the quality of the prediction, we see that the PPE error is almost close to zero for the corresponding selected motion, except when k is close to 14 or close to 26. These particular instants occur when the target changes its motion. It is important to highlight that after a couple of iterations, the algorithm adapts to the corresponding model and the PPE error goes back to close to zero.

The same experiment was performed to the other two trajectories and the results are displayed in the table below.

Trajectory	5.1(a)	5.1(b)	5.1(c)
RPPE	0.3456	0.6289	0.4144
Accuracy	86.67%	89.66%	88.89%

Table 5.1: Results of recursive prediction using the synthetic dataset

Considering the RPPE results in table 5.1 we can compute the dataset TSE to be equal to 0.4630. If there was no error when predicting future positions, this value would be zero. We can also compute the average accuracy in motion identification to be $\approx 88\%$.

5.3 Real data: The SPARSIS dataset

In this section we thoroughly analyze the SPARSIS dataset and use it to test the proposed prediction methods and algorithms in this dissertation.

5.3.1 Dataset

The SPARSIS dataset captures the trajectories of various pedestrians going up and down a flight of stairs in Instituto Superior Técnico. The dataset has 47 trajectories and it is represented in figure 5.4.

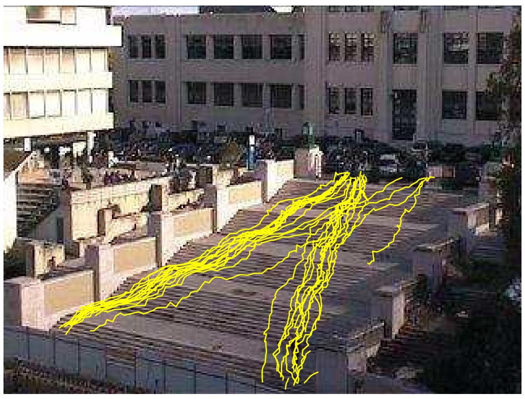


Figure 5.4: SPARSIS dataset

A homography correction of the dataset is proposed in [19], based on two different views of the stairs. This corrected dataset is preprocessed and then used in the experiments. The corrected dataset is represented in figure 5.5.

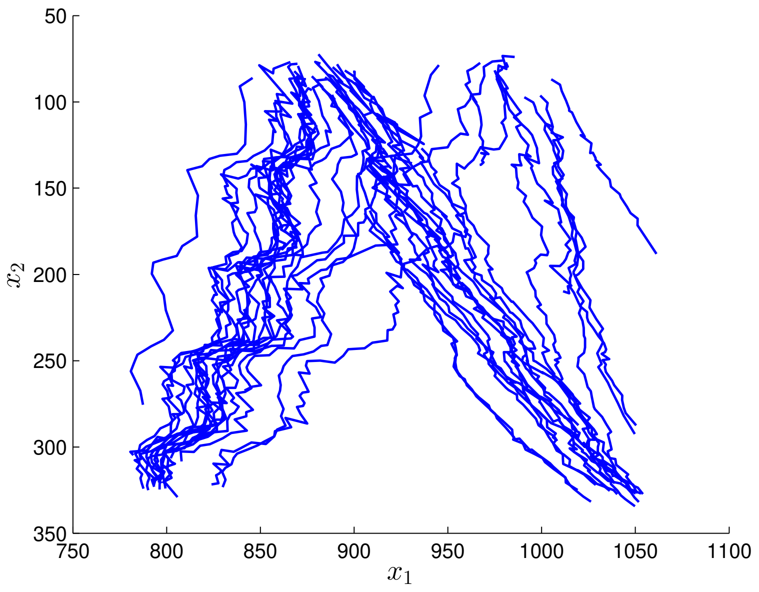


Figure 5.5: SPARSIS dataset after homography correction

We assume that every motion within this dataset is linear, as there are no circular motions in the trajectories. As such, every trajectory coordinate is differentiated twice using the inverse of the “double integrator” system, as explained in Chapther 2 section 3.2.1.

5.3.1.1 Data preprocessing

The first step when considering a new dataset is data preprocessing. We consider that small trajectories detain low information about other trajectories in the dataset, thus, they are not good candidates from which to learn the models. Therefore, to remove these outliers, every trajectory that has a length lower or equal to a certain threshold is removed. We found the threshold of 10 to be acceptable, hence, we removed most small trajectories without losing significant data. Therefore, the dataset used will consist in 44 trajectories.

5.3.1.2 Data separation

The dataset is divided in to three separate sets: Training set ($\approx 70\%$ of the dataset), Validation set ($\approx 10\%$ of the dataset), Testing set ($\approx 20\%$ of the dataset).

The idea is to use the Training set and the Validation set to estimate the best models and parameters offline, and use that information to recursively predict trajectories in the Testing set.

Training set

The Training set comprises the majority of the dataset ($\approx 70\%$). We use this data to learn the models the best represent the Training set, and therefore to learn an estimate of the models the best represent the entire dataset. The models are learned through the algorithm described in Chapter 4 section 4.3. This results in a bank of models for each coordinate.

Validation set

The Validation set is used to estimate the best parameters for the multiple models architecture. The procedure is based in the problem described in Chapter 4 subsection 4.2.2 to regulate the α parameter.

The objective is to use the banks of models created with the Training set to find α_o, which is the value of α that minimizes the TSE error in the Validation set.

Testing set

Finally, the Testing set is uses the the banks of models created and the value of α_o, to preform recursive predictions and the overall TSE error is computed, giving a measure of the performance of the overall algorithms and procedures.

5.3.2 Experiments

5.3.2.1 First experiment: a single trajectory example

In this first experiment we use a single trajectory taken from the dataset, and predict it with an overfitted model for each coordinate. This allows us to focus only on the recursive prediction of the trajectory.

Consider the following trajectory taken from the dataset, with length of $K = 66$ represented in red in figure 5.6. In order to analyze the recursive prediction, we stop the algorithm procedure at different instants to display its one-step prediction in figure 5.7. Looking at figure 5.7, in each subfigure we can see the one-step predicted position (red), the real trajectory (blue), and the standard deviation of the prediction error (green). The standard deviation of the prediction error gives us an area in which there is a high accuracy ($\approx 80\%$ in this specify trajectory) of delimiting the real position.

Further analyzing figure 5.7(a), we can plot the respective predictions of each coordinate as well as the prediction of the computed stationary signal, similarly as in Chapter 3 subsection 3.4.1. The result is represented in figure 5.8. We can see how, in fact, the algorithm recursively predicts positions using the methods proposed in this dissertation.

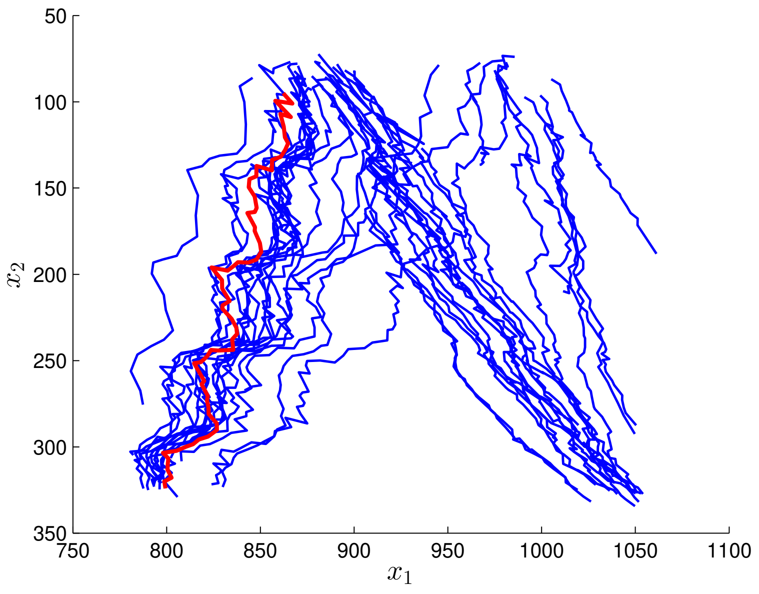
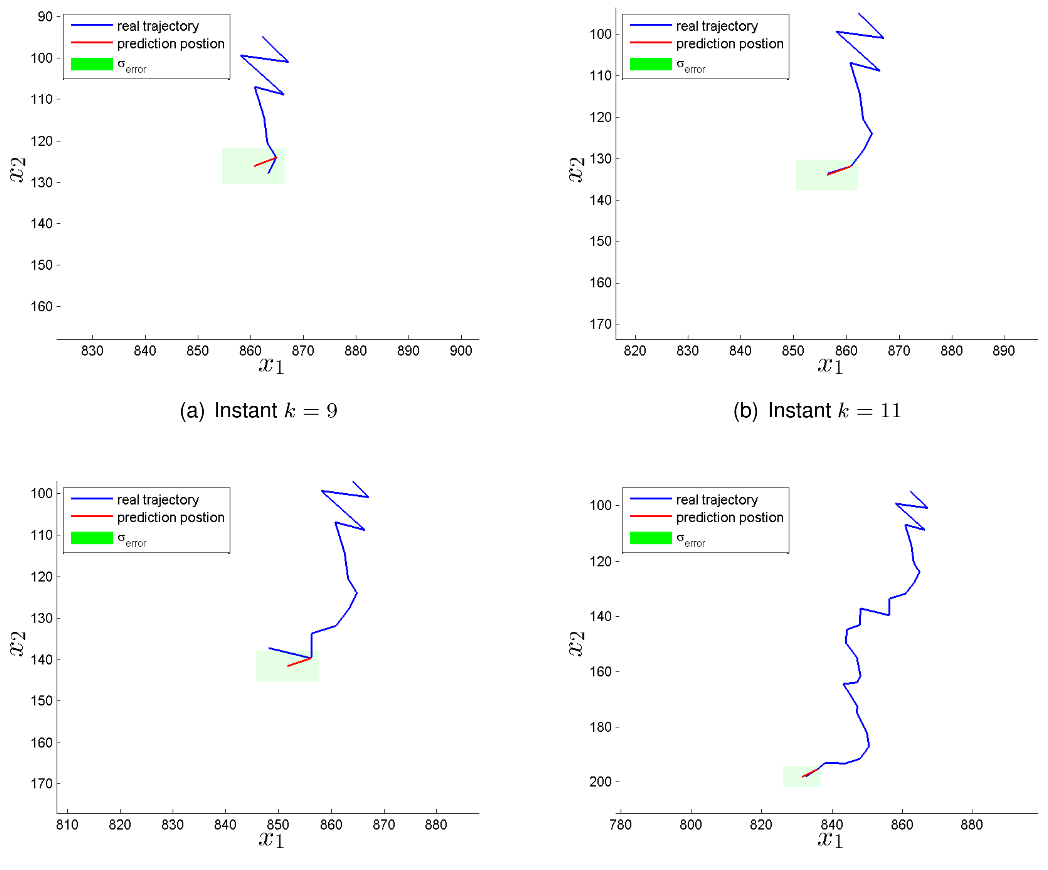


Figure 5.6: Trajectory considered



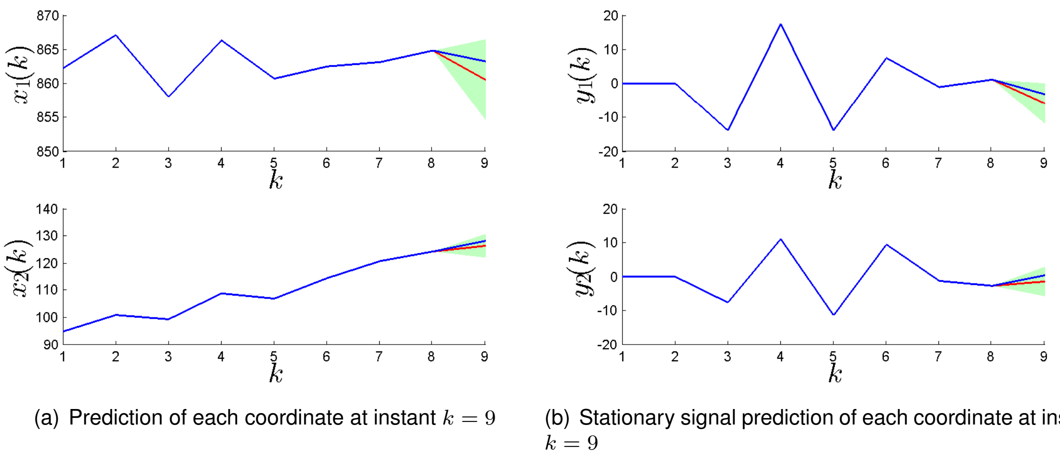
(a) Instant $k = 9$

(b) Instant $k = 11$

(c) Instant $k = 13$

(d) Instant $k = 29$

Figure 5.7: Prediction procedure at different instants


(a) Prediction of each coordinate at instant $k = 9$

(b) Stationary signal prediction of each coordinate at instant $k = 9$

Figure 5.8: The nonstationary signal $x(k)$ and the stationary signal $y(k)$ of each coordinate

5.3.2.2 Second experiment: considering the entire dataset

In this experiment we consider the entire dataset, therefore we go through the process of dataset separation (as explained in sub-subsection 5.3.1.2): learning the banks of models, estimating the multiple models parameter α, and finally computing the TSE of the Testing data.

We start by dividing the dataset in to the three separate sets: Training set, Validation set and Testing set. Then, we define the maximum AR order, and maximum MA order for the learning algorithm, resulting in two banks of 28 models. We consider different maximum orders of AR and MA. The banks of models created with the Training set are used to find α_o, which is the value of α that minimizes the TSE error in the Validation set. We also consider different LPF orders in this experiment. Finally, the Testing set preforms recursive predictions of the trajectories and the overall TSE error is computed, giving a measure of the performance of the overall algorithms and procedures.

A cross validation with multiple rounds is performed, in which the trajectories of the dataset are randomly shuffled before the dataset separation procedure.

5.3.3 Results

5.3.3.1 First experiment

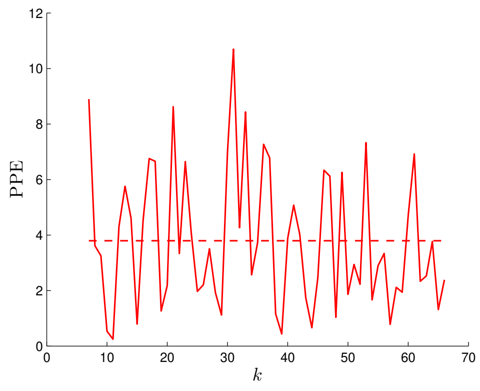
After recursively predict each position of the trajectory and estimating the standard deviation of the prediction error, we can analyze the PPE and the σ_{error} across the trajectory, represented in figure 5.9(a).

The total RPPE was 3.7955, represent in figure 5.9(a) as a dotted line. In regard to the standard deviation of the prediction error, the average of the σ_{error} across the trajectory coordinate x_1 was 5.4713, as for the x_2 coordinate it was 3.7263. We see that the σ_{error} declines in both coordinates as more observations are available to the algorithm.

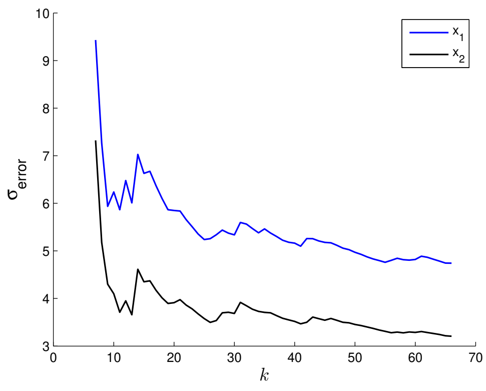
5.3.3.2 Second experiment

Learning the multiple models parameter

Consider two banks of models learned with both maximum MA and AR orders equal to 3. We focus on the problem of learning the multiple models parameter α_o. To find the optimal parameter, using



(a) Computed PPE across the trajectory

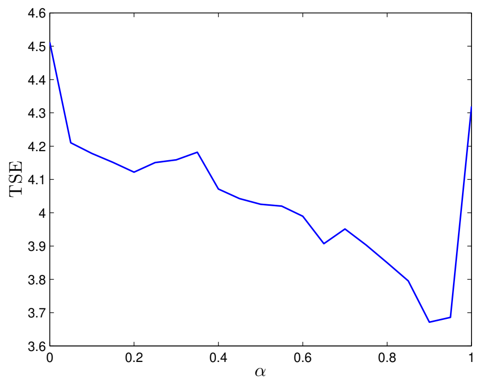


(b) Estimated σ_{error} across the trajectory for each coordinate

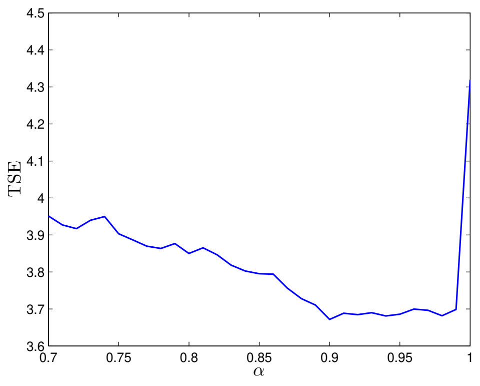
Figure 5.9: PPE and σ_{error} across the trajectory studied

the first order LPF, we compute the TSE of the validation set for α between 0 and 1 with an interval step of 0.05, $\{0, 0.05, 0.1, \ldots, 0.90, 0.95, 1\}$. After detecting a minimum, we zoom in $\alpha = \{0.7; \ldots; 1\}$ and make the same computation for α values between 0.7 and 1 with a finer interval step of 0.01, $\{0.7, 0.71, 0.72, \ldots, 0.98, 0.99, 1\}$. The results are presented in figure 5.10. The same procedure is used for the second order LPF and the results are presented in figure 5.11.

The α_o found using the first order LPF was 0.9, as for the second order LPF was 0.81.

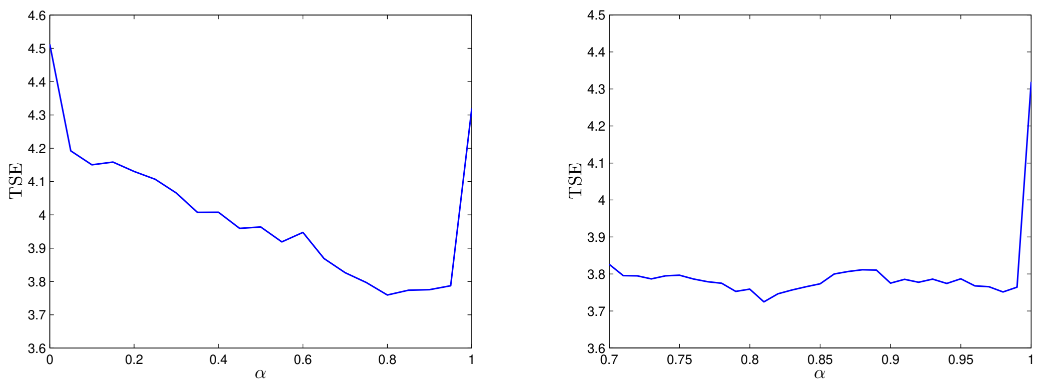


(a) Finding α_o



(b) Zoom in $\alpha = \{0.7; \ldots; 1\}$

Figure 5.10: Finding α_o with the first order LPF in the multiple models architecture



(a) Finding α_o

(b) Zoom in $\alpha = \{0.7; \ldots; 1\}$

Figure 5.11: Finding α_o with the second order LPF in the multiple models architecture

Cross validation

As previously explained in the experiment, we define different maximum MA and AR orders for the learning algorithm, and different LPF orders in the multiple models architecture, in order to test the overall TSE of the testing data. In total, six rounds are performed in order to correctly test the overall procedure. The results are displayed in the following tables (leftmost column being the LPF order).

		Maximum order (AR,MA)		
		(1,1)	(2,2)	(3,3)
1^{st}_{order}	α_o	0.99	0.88	0.9
	TSE	3.4414	3.5522	3.3460
2^{nd}_{order}	α_o	0.86	0.94	0.81
	TSE	3.4332	3.5118	3.5503

Table 5.2: Round 1 results

		Maximum order (AR,MA)		
		(1,1)	(2,2)	(3,3)
1^{st}_{order}	α_o	0.95	0.97	0.98
	TSE	3.8791	3.677	3.9581
2^{nd}_{order}	α_o	0.98	0.91	0.98
	TSE	3.8772	3.8406	3.9210

Table 5.3: Round 2 results

		Maximum order (AR,MA)		
		(1,1)	(2,2)	(3,3)
1^{st}_{order}	α_o	1	1	0.98
	TSE	3.7639	3.7639	3.8464
2^{nd}_{order}	α_o	1	1	0.97
	TSE	3.7639	3.7639	3.9346

Table 5.4: Round 3 results

		Maximum order (AR,MA)		
		(1,1)	(2,2)	(3,3)
1^{st}_{order}	α_o	0.96	0.92	0.91
	TSE	3.3760	3.4599	3.5154
2^{nd}_{order}	α_o	0.75	0.97	0.64
	TSE	3.3974	3.4378	3.6018

Table 5.5: Round 4 results

		Maximum order (AR,MA)		
		(1,1)	(2,2)	(3,3)
1^{st}_{order}	α_o	0.96	1	0.98
	TSE	3.3034	3.6677	3.4539
2^{nd}_{order}	α_o	0.93	0.96	0.98
	TSE	3.2726	3.4145	3.4336

Table 5.6: Round 5 results

		Maximum order (AR,MA)		
		(1,1)	(2,2)	(3,3)
1^{st}_{order}	α_o	0.93	0.85	0.81
	TSE	3.9642	4.0603	4.1286
2^{nd}_{order}	α_o	0.94	0.91	0.96
	TSE	3.8525	3.9617	4.0086

Table 5.7: Round 6 results

Looking at the cross validation average results presented in the table 5.8 below, we see that the lowest TSE is achieved when both maximum AR and MA orders are equal to 1 and the second order LPF is used. Futhermore, we observe that the first order LPF α_o has a low variation across the maximum AR and MA orders, when compared the same case in a second order LPF. Finally, we see that the TSE

		Maximum order (AR,MA)		
		(1,1)	(2,2)	(3,3)
1^{st}_{order}	α_o	0.97	0.94	0.93
	TSE	3.6213	3.6968	3.7081
2^{nd}_{order}	α_o	0.91	0.95	0.89
	TSE	3.5995	3.6551	3.7417

Table 5.8: Cross validation average results

increases as higher order models are available to the training algorithm.

The results in the table 5.8 establish a baseline for one-step trajectory prediction to the SPARSIS dataset.

Chapter 6

Conclusion

The purpose of this dissertation was to restore extensively studied methods of time-series predictions and apply them to trajectory prediction. We used ARMA, ARIMA and SARIMA models to describe and predict each trajectory coordinate. In addition, a multiple models architecture was used to recursively predict future positions, given a bank of models for each coordinate. This procedure allows to predict unseen trajectories and adapt to changes in behavior. In order to test the proposed methods, we conducted several experiments using synthetic data and real data.

The SPARSIS dataset was thoroughly analyzed and evaluated. We divided the dataset into three sets, each one for different purposes. A Training set is used to create the banks of models that best represent the set. A Validation set regulates the multiple models architecture. The Testing set is used to test the overall prediction procedure, while also providing an evaluation on how the learned models generalize to correctly describe the dataset.

Regarding the achieved results using real data, we observed that a higher maximum order of models actually increases the overall error. This indicates that the higher order models do not generalize when the objective is to minimize a one-step prediction of the dataset. Thus, lower order models should be used for short terms predictions. We also found that, when using the proposed multiple models architecture, the SPARSIS dataset one-step prediction overall error minimizes with a second order LPF with a pole close to 0.91. Furthermore, we have provided a baseline for trajectory prediction using the SPARSIS dataset.

As for future work, it would be interesting to use the proposed methods in other real datasets. One particularly interesting dataset is the SDD, which has a great number of trajectories of various moving targets such as pedestrians, cyclists, and cars. It would be curious to see if different sets of models correspond to different targets, thus, extending this work application to target classification. Also, we could compare our work results with the existing trajectory prediction techniques.

Finally, in order to achieve better results, we should consider multivariate time-series models. Since in this dissertation the predictions are made to each coordinate independently, we lose information regarding spatial correlations. Thus, a extension of this work could be made to cover Vector Auto Regressive (VAR) and Vector Auto Regressive Moving Average (VARMA) models.

Bibliography

[1] D. R. Brillinger, H. K. Preisler, A. A. Ager, and J. G. Kie, “An exploratory data analysis (EDA) of the paths of moving animals,” *Journal of Statistical Planning and Inference*, vol. 122, no. 1-2, pp. 43–63, 2004.

[2] S. Pravilovic, M. Bilancia, A. Appice, and D. Malerba, “Using multiple time series analysis for geosensor data forecasting,” *Information Sciences*, vol. 380, no. 0, pp. 31–52, 2017.

[3] J. C. Nascimento, J. S. Marques, and J. M. Lemos, “Modeling and classifying human activities from trajectories using a class of space-varying parametric motion fields.,” *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society*, vol. 22, no. c, pp. 1–13, 2013.

[4] A. Robicquet, A. Sadeghian, A. Alahi, and S. Savarese, “Learning social etiquette: Human trajectory understanding in crowded scenes,” *European Conference on Computer Vision (ECCV)*, vol. 9912 LNCS, pp. 549–565, 2016.

[5] D. Xie, T. Shu, S. Todorovic, and S. C. Zhu, “Learning and Inferring ’Dark Matter’ and Predicting Human Intents and Trajectories in Videos,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 7, pp. 1639–1652, 2018.

[6] E. Maggio and A. Cavallaro, *Video Tracking: theory and parctice*. Wiley, 2011.

[7] L. Fridman, D. E. Brown, M. Glazer, W. Angell, S. Dodd, B. Jenik, J. Terwilliger, J. Kindelsberger, L. Ding, S. Seaman, H. Abraham, A. Mehler, A. Sipperley, A. Pettinato, B. Seppelt, L. Angell, B. Mehler, and B. Reimer, “MIT Autonomous Vehicle Technology Study: Large-Scale Deep Learning Based Analysis of Driver Behavior and Interaction with Automation,” vol. 6, pp. 1–17, 2017.

[8] B. Ristic, B. L. Scala, M. Morelande, and N. Gordon, “Statistical analysis of motion patterns in AIS Data: Anomaly detection and motion prediction,” *2008 11th International Conference on Information Fusion*, no. January 2008, pp. 40–46, 2008.

[9] C. Gong and D. McNally, “A Methodology for Automated Trajectory Prediction Analysis,” *AIAA Guidance, Navigation, and Control Conference and Exhibit*, pp. 1–14, 2004.

[10] W. Li, V. Mahadevan, and N. Vasconcelos, “Anomaly detection and localization in crowded scenes,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 1, pp. 18–32, 2014.

[11] G. E. P. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time Series Analysis: Forecasting & Control*. Wiley, 5th ed., 2015.

[12] J. Chen, D. Li, G. Zhang, and X. Zhang, “Localized Space-Time Autoregressive Parameters Estimation for Traffic Flow Prediction in Urban Road Networks,” *Applied Sciences*, vol. 8, no. 2, p. 277, 2018.

[13] F. Bartoli, G. Lisanti, L. Ballan, and A. Del Bimbo, “Context-Aware Trajectory Prediction,” *arxiv.org/abs/1705.02503*, 2017.

[14] A. Sadeghian, F. Legros, M. Voisin, R. Vesel, A. Alahi, and S. Savarese, “CAR-Net: Clairvoyant Attentive Recurrent Network,” *ECCV 2018*, pp. 151–167, 2018.

[15] G. Habibi, N. Jaipuria, and J. P. How, “Context-Aware Pedestrian Motion Prediction In Urban Intersections,” *arxiv.org/abs/1806.09453*, 2018.

[16] J. C. Nascimento and J. S. Marques, “A Sparse Approach to Pedestrian Trajectory Modeling Using Multiple Motion Fields,” *IEEE Int. Conf. Image Processing, Beijing*, pp. 2538–2542, 2017.

[17] S. Hellbach, J. P. Eggert, E. K, and H.-m. Gross, “Time Series Analysis for Long Term Prediction of Human Movement Trajectories,” *Processing*, no. Iconip 2008, pp. 567–574, 2009.

[18] Z. Yan, “Traj-ARIMA,” *Proceedings of the Second International Workshop on Computational Transportation Science - IWCTS ’10*, no. I, p. 11, 2010.

[19] J. S. Marques, M. Barão, and J. M. Lemos, “Alignment of velocity fields for video surveillance,” *Pattern Recognition Letters*, vol. 33, no. 12, pp. 1632–1637, 2012.

[20] H. Akaike, “Information theory and an extension of the maximum likelihood principle,” *International symposium on information theory*, pp. 267–281, 1973.