Probabilistic Dynamical Models
for the Analysis of Affective Facial Expressions

Relatore: Prof. Giuseppe Boccignone
Correlatore: Dott. Giuliano Grossi

Tesi di Laurea di:
Matteo Bodini
Matricola: 901336

Anno Accademico 2017-2018
Acknowledgments

Before showing the work that I have done in the last six months, I must especially thank all the people that contributed to this: I would like to start from my supervisor, Professor Giuseppe Boccignone and Giuliano Grossi. They proposed to me to start this thesis and it was a big pleasure to work with them. They have been always and widely available anytime I needed it and they taught me many things, starting from their courses that I have attended, up to now as I’m working with them. I want to thank also all the members of PhuSe Lab for the same reason. They helped me a lot in this work, especially Vittorio Cuculo for the Benchmark part that you will read in Chapter 5.

Other, special, thanks go also to Professor Ottavio D’Antona and Mauro Torelli. I have worked with them in the last months on combinatorial analysis after I have attended the course of Teoria dei Grafi. They can’t imagine how many things they taught me about everything. In particular Professor Ottavio D’Antona introduced me to the research field of combinatorial analysis, he helped me in writing my first scientific paper and he let me to attend to my first conference, COMBINATORICS 2018. The style in which I have written this work would not be the same without him. Lots of thanks also go to Mauro, as also him influenced me with his precision and his working method. It is a pleasure to work with him.

Finally, I want to thank my family and my friends that supported me in this months and in particular Laura, that is always present in everything I have to face.
Abstract

Understanding affective signals from others is crucial for both human-human and human-agent interaction. In this work we approach to the analysis of emotions displayed in the course of a face-to-face interaction between an expresser (typically a human) and an observer (agent, either virtual or real). Different signals can be considered such as physiological signals, speech and video information. We take account in particular of the latter and we focus on one of the fundamental problems of the analysis of affective facial expressions: this is the developing of an internal visuomotor mapping and his predictive use in the interpretation of perceived facial cues from an observer during a dyadic interaction with another agent.

Therefore, the aim of this work is the study and the development of an interaction system, which is able to detect the emotion felt by a human expresser via internal mirroring performed by an observer (e.g. a robot or an avatar). The problem will be handled in a biologically inspired vein drawing on a mirroring approach rather than a classic pattern recognition pipeline (feature extraction followed by classification/regression), thus differentiating from the vast majority of approaches proposed in literature, for an example see [33]. At the heart of this approach there is the mirroring of the expresser’s modalities (in particular facial expression) by the observer, which relies on visuomotor and visceromotor representation rather than on the only visual expertise which can be provided by standard computer vision techniques. This participates in building a deep understanding of the perceived emotions [1].
Introduction

A large body of evidence [57, 25] shows that alongside the sensory information concerning others’ social stimuli — actions, in a wide sense —, one’s own motor and visceromotor representations of those stimuli are enacted. Humans mirror gestures, postures, emotions, speech of other perceived humans, at least neurally, and sometimes bodily and behaviorally. These mirroring processes ground the capability of own reproduction of the action in question “as if” a similar action was performed or a similar emotion experienced. It has been argued that such simulation-based mechanism is likely to play a crucial role in individual cognition and social interaction [57, 25].

Under such rationale, we intend to investigate and develop a model where the interaction between the expresser and the observer is regulated by the sensorimotor mediation. In a nutshell, at the earliest processing stages, the observer acquires sensory information from the expresser. Secondly, an “action perception” stage mediates between the early sensory perception of the expresser and the emotion system activity. This stage is conceived as a mirroring mechanism based on a probabilistic affect space.

Based on these considerations, assume a face-to-face interaction between two agents, say the expresser and the observer, that share a common model of both action perception and production. The “inputs” to the observer’s model are provided by the facial display of the expresser. We intend to investigate and develop the modelling of a suitable visuomotor mapping of perceived facial cues to observer’s internal somatic motor space. Such mapping is at the heart of the affective perception of facial expressions.

The rationale behind this study is that facial expressions are facial actions and are likely to draw on simulation mechanisms underlying action perception in general [70]. Indeed, affective facial expression perception is unlikely to rely on a traditional pattern recognition pipeline (feature extraction followed by classification) [70, 1, 11]. As witnessed by a large body of neurobiological and psychological literature, e.g., [70, 1, 11, 49], in the course of a dyadic engagement, the observer’s visual system, while perceiving expresser’s facial display, interacts with an extended system, which involves the emotion system. Interaction is regulated by the mediation of a visuomotor component, which is used to couple visual cues to the observer’s own motor representation, namely the internal motor states of the somatomotor state-space. The latter, on one side, works close together with core affect components [70], which in turn activate visceromotor responses accounting for physiological signal variations. On the other side, it allows the matching between the visual description of observed facial actions and the sensory predictions of imitative motor plans [10, 11, 49, 29, 63].

The problem of laying down a comprehensive computational model for affective facial expression perception grounded in simulation mechanisms has been addressed in [8]. In that case, however, the visuomotor mapping problem was solved under simplifying assumptions.
to afford a maximum-likelihood solution [8, 14]. Here we focus on a more general solution, a Bayesian filtering approach in the form of a particle filter whose dynamics is constrained by a continuous latent space of motor actions. Not much effort has been spent in such direction. Particle filters [17, 3, 12] are powerful tools for Bayesian state estimation in non-linear systems. The key idea of particle filters is to approximate a posterior distribution over unknown state variables by a set of particles, drawn from this distribution. Particle filtering has been widely used for object tracking [68] and provides a flexible and effective tool for integrating dynamic facial cue tracking in a Bayesian framework, e.g. [7]. PF has been jointly used with a 3D facial model for expression tracking account for multi-class dynamics conditioned on the expression [16]. However, in that case the auto-regressive dynamics was conditioned on a discrete emotion label, with one model per expression. Here, more generally, we condition on a latent continuous action space, which can also include trajectories corresponding to discrete emotions as specific instances, but is not limited to such instances. Though not addressing the issue of motor simulation, Fan et. al. [23] exploit the motor control sequence - derived from a 3D shape model as the observation input to a Kalman filter. The authors are mostly concerned with the classification of basic emotions, rather than building a continuous latent space of actions akin to support visuomotor learning.

Hence, the perception-action cycle mechanism governing the interaction can be formulated in a probabilistic setting having at its core a continuous dynamic latent affect space representation. Such modelling choice, for instance can rely on Gaussian Process Latent Variable Models (GP-LVM).

Latent Variable Models (LVMs) [5, 47] are a kind of pattern extraction methods. A latent variable model generally refers to a statistical model that relates a set of variables (so-called manifest variables) to a set of latent variables under the assumption that the responses on the manifest variables are controlled by the latent variables. Furthermore, we can provide latent variables with various meanings for specific tasks. In Dimension Reduction, we assume that the latent variables are the low dimensional representations of high dimensional samples. In clustering, the latent variables can be defined to represent the clustering membership of samples. This flexible definition of latent variables has made LVMs widely be used in many machine learning tasks. Gaussian Process Variable Models (GPLVMs) [39], as a large class of LVMs, have been explored and applied in many machine scenarios.

GP-LVM are suitable to learn a latent space in an unsupervised mode by handling signals provided. A similar approach could be based on autoencoder-like architectures. But, beyond the specific architectural choices, the crucial feature of the model should be the capability of handling a continuous representation of the affective episode.

Such approach presents several advantages:

- It is a unified model, in the sense that it is based on an analysis-by-synthesis approach, namely, it is capable of handling both affect detection and affect expression within the same framework.

- By exploiting internal simulation mechanisms, it should require only limited examples at the learning stage a cogent issue in human — agent interaction systems —, thus being easily adaptable to specific populations and scenarios.
Particular emphasis will be placed on the latter modality, and we plan to consider facial signals as it is clear, in fact, that they convey a lot of emotional information.

At the design level, we will investigate on the recent research trend that addresses the use of Gaussian Process Latent Variable Models (GP-LVM) together with Bayesian approaches. On the other hand, we exploit PGMs, where recognition algorithms can be seen as deterministic approximation to message passing-based inference.
# Contents

## Acknowledgments

### Abstract

## Introduction

## List of Figures

## List of Tables

1. **The analysis of facial expressions**
   1.1 Emotions and facial expressions: the main guidelines .......................... 1
   1.1.1 Anger ........................................................................ 4
   1.1.2 Disgust ..................................................................... 4
   1.1.3 Happiness ................................................................... 4
   1.1.4 Sadness ..................................................................... 4
   1.1.5 Surprise ...................................................................... 5
   1.1.6 Contempt .................................................................... 5
   1.1.7 Fear .......................................................................... 5
   1.2 The study of micro expressions ................................................. 6
   1.3 FACS method: Facial Action Coding System ......................... 7

2. **Gaussian Process Latent Variable Model**
   2.1 A general introduction .......................................................... 11
   2.2 Principal Component Analysis .............................................. 12
   2.3 Multivariate Gaussian Distribution ...................................... 13
   2.4 A Probabilistic Model for PCA ........................................... 15
   2.4.1 The Principal Subspace of PPCA ................................. 16
   2.5 Gaussian Process Latent Variable Model .............................. 17
   2.5.1 Dual Probabilistic PCA .............................................. 17
   2.5.2 Gaussian Processes .................................................... 18
   2.5.2.1 The use of Bayesian reasoning ............................... 19
   2.6 Summary ....................................................................... 20

3. **Particle Filters**
   3.1 Hidden Markov Models (HMM) ........................................... 23
   3.1.1 Inference in HMM ....................................................... 24
   3.2 Particle Filtering ................................................................ 27
   3.2.1 Monte Carlo Methods ............................................... 27
   3.3 Importance Sampling & Sequential Importance Sampling ..... 28
   3.4 Bootstrap (SIR) ............................................................... 30
   3.4.1 Sampling Impoverishment Problem .............................. 31
List of Figures

1.1 Seven images portraying Tim Roth, English actor and filmmaker, that plays the seven basic emotions. Aside of every picture are listed the main characteristics of every emotion. .................................................... 3

2.1 A LVM learns the mapping of input $x_i$ to output $y_i$ by encoding the relationship in matrix $\mathbf{W}$, which models the behaviour of the underlying function. 15

2.2 Plot of a Gaussian process prior (up) and posterior (down). The semi-transparent functions represent samples from the likelihood distribution (image from: https://pythonhosted.org/infpy/gps.html). ...................... 19

3.1 A schematic view of the SIR algorithm. ................................. 30

4.1 The probabilistic graphical model. We can see graphically the conditional independences of the random variables that help us to factorize the joint probability distribution. For sake of simplicity we have not reported $\theta$. It is important to note that the state $m_t$ influences the future state $m_{t+1}$ and the next point in the latent space $\mathbf{a}_{t+1}$. ........................................... 37

4.2 The subset of CANDIDE-3 model vertices. ............................... 42

4.3 The order of the subset of CANDIDE-3 model vertices. .................. 42

4.4 The 68 facial landmark indexes of Dlib. ................................. 43

4.5 Two examples of landmarks estimation. ................................... 43

5.1 A frame from a video contained in the RAVDESS dataset. As you can see it is very clear and suitable to our tests. ................................. 49

5.2 The plot shows the ratio of processed frames for every procedure and for every scenario. We can see that Particle filters are able to propose a prediction for every frame. .................................................. 50

5.3 The plot shows the RMSE for every procedure and for every scenario. We can see that conditioned particle filter is better than the standard one three out of the five. ....................................................... 51

5.4 The plot shows the RMSE for every procedure and for every scenario, where transparency is proportional to the processed frames ratio. .............. 51
List of Tables

1.1 A table for the interpretation of emotions. . . . . . . . . . . . . . . . . . . 9
1 The analysis of facial expressions

In the introduction we have written about facial expressions and in particular of emotions, leaving the reader with his own idea of whatsoever they are. Intuitively, everyone has his own idea of what is an emotion and of what is a facial expression, but now, before we explain our model, it is fundamental that one has a clear idea of what do we precisely mean by facial expressions end in particular of what is an emotion. This is exactly the target of this preliminary chapter.

Paragraphs 1.1 and 1.2 will be focused on expressions and their analytical description. Finally, in paragraph 1.3 we will face the theme of the study of micro expressions, introducing a measurement system for facial movements: Facial Action Coding System (FACS).

1.1 Emotions and facial expressions: the main guidelines

Among the different modalities of non-verbal communication of one’s emotive state, such as gestures and pose, we can’t take into account gaze and face, that represent the main reason that draws people attention and interest. Ultimately, the face represents a privileged system for convey informations and, in particular, emotions.

Remarks about the appearance of emotions on the face are already noticeable in many Latin and Greek authors and after in many medieval authors, but the first really scientific work to which the contemporary authors refer is The Expression of Emotions in Man and Animals of Charles Darwin (1872). This work, is the first treaty on facial expressions: the objective of Darwin was to point out that expressions are coupled with a mental state, that not only includes emotions, but also sensations, behavior and personality traits. The main relevance of his work, in this area, it is the one of considering facial expression in "groups" among which there is a difference, and not as fixed and immutable categories.

The influence of Darwin has given rise to two different schools of thought: an ethological one, to be understood as the study of the animal behavior in the environment, and a psychological one. The studies from ethologists are focused mainly on the interaction level, while the studies of psychologists are more close on Darwin’s theories, in particular in 70’s under Paul Ekman. In modern times we can notice three main schools of thought.

The first is the one by Robert S. Woodworth (1938). He assumed that facial expression convey families of emotions, based on pleasure or unpleasantness, autonomous activation or relaxation, attention or rejection.

The second one was developed by Charles E. Osgood at al. (1966) who considered the
The analysis of facial expressions

exhibition of facial expression and the response of the observer to it. He gave also proofs of the cross-cultural generality of the meaning of the face.

The third one is represented by Nico Henri Frijda et al. (1989), that proposed a perception model based on information processing: who assess the facial expression, should describe not only the single emotion that he recognizes on the face, but also imagine the internal states of the person that exhibits the expression. Finally, he have also to take into account the changing in his the internal state, caused by the observed. Another important perspective concerning facial expression are the mechanisms underlying their production. In this regard, there are mainly two opposite theories: the global hypothesis and the dynamic hypothesis.

According to the global hypothesis from Fridlund et al. (1987) and Izard (1977), the expressive face configurations are universally shared, fixed and specific for every emotion. Alternatively, the dynamic theory provides for a cumulative and sequential process for every facial expression, that is the result of the progressive accumulation and dynamic integration of the single phases. According to this approach, facial expressions are temporary motor configurations, flexible and variable and able to adapt to different situations.

Another focus concerns the conveying of the meaning of facial expression divided into emotive prospective and communicative prospective. For Ekman and Izard, supporters of the emotive approach, the micro expressions, that are unintended facial expression lasting 1/25 seconds, have primarily an emotive value as they are an immediate, spontaneous and involuntary response of emotions. Looking facial micro expressions of a subject you can “read” the emotions that he is feeling, as discrete categories. From this idea it derives the cultural invariability that Ekman et al. (1971) have verified with subject from different cultures compared to expressions corresponding to seven basic emotions. However, this studies were criticized because of the methods and therefore for the data obtained. In any way, there are correspondences that gave rise to the minimum universal theory that say that there exist a minimum degree of similarity between cultures in the interpretation of facial micro expressions.

Beyond the theory that had been developed around facial expressions, everyone has agreed in supporting the universality of the expressive communications of emotions: “Many facial micro expressions exist throughout the world, in every race and culture.” (Frijda, 1989).

These are the conclusions that also Darwin had reached and, in more recent times Fridlund et al. (1987), according to which the expressions of anger, disgust, happiness, sadness, surprise, contempt and fear are universal.

Each of these emotions can be recognized in the face of everyone, no matter of ethnicity, culture, gender, religion. In the following we introduce a description of the main characteristics of each of the seven principal emotions.
1.1 Emotions and facial expressions: the main guidelines

Figure 1.1: Seven images portraying Tim Roth, English actor and filmmaker, that plays the seven basic emotions. Aside of every picture are listed the main characteristics of every emotion.
1 The analysis of facial expressions

1.1 Anger

There are many motivations that can make us angry: an example is something unexpected that hurts us, but also a sense of injustice or eventually a verbal or physical aggression. Naturally, it is expressed at many levels. Differently from other emotions, such as anger and surprise, it does not occur immediately, but tends to reach its peak stepwise. It can begin as a sense of irritation and leaving immediately, just perceived, or otherwise if it is not stopped it gets harder to control. In Fig. 1.1 are summarized the characteristics of the anger expression: when it manifests itself, it appears on the front, eyes and mouth.

As you can see looking at the front, the eyebrows are get lower, the eyebrows internal angles also get lower moving at the center forming two vertical lines in the middle. The upper and lower eyelid are tightened and the lips are pressed. The latter are pressed, unless one is speaking or shouting, in this case the mouth is opened, but still tightened.

1.2 Disgust

You feel disgust when you feel repulsion for something. It can bee an object, a smell, a taste or a thought, or sometimes only the memory of something for which you have felt disgust. In all of this situations, the first reaction is of moving away or get rid of what causes disgust.

As it is shown in Fig. 1.1, an expression of disgust manifests itself with the upper lid raised, cheeks rise up and cause the raising of the lower eyelids. The eyebrows get lower with the upper eyelids, while the nose wrinkle.

1.3 Happiness

Happiness represents the emotion that everyone want to feel as often as possible. It is the most pleasant emotion because when you feel it you feel good, indeed one prefers to hang with happy people instead of who aren’t. Unlike fear, one memorizes happy situations to replicate them or to make them happen again.

An happy face, like the one presented in Fig. 1.1 shows an expression where the mouth moves toward the external part of the face and it moves upward. Cheeks rise up and cause the raising of the lower eyelids and the wrinkles that form around eyes. Finally, a line is formed starting from the nose until the mouth angles.

1.4 Sadness

When you lose something you feel sadness: it can be the losing of someone or something to which we were attached. It can happen after the separation from a friend, a beloved person, the loosing of a job or your own health. Sadness is an emotion that allow us to overcome this losses and to communicate to others that we need to be comforted.
1.1 Emotions and facial expressions: the main guidelines

The face of a sad person, in Fig. 1.1, shows that the internal angles of eyebrows rise up toward the center and the upper eyelids rise up in the internal angles. The mouth angles are down and when sadness intensifies lips tremble and the lower lip extends forward.

1.1.5 Surprise

When you have to face something unexpected, the first sensation that you feel it the one of surprise. It can happen when you meet someone that you don’t see for a long time, or when you receive an information that you wouldn’t expect or in lots of circumstances that have in common the fact that they happen without any notice. This emotion is special because it is instantaneous: it is a short term emotion, a fraction of a second that represents the time in which the brain develop this new information. When it has finished, it will appear the next emotion, that depends on the motivation of this surprise. For example it can be fear if suddenly there is a car that cuts us off. Finally, surprise is not a positive or negative emotion. It does not give joy or confusion: it is useful to perceive and develop a new information. It will be the next emotion that will be positive or negative.

In Fig. 1.1 you can see that in the part of the front, the eyebrows are up and form a curve upwards. Often, as a consequence of this movement, horizontal lines form that cross all the front. The eyes are peeled and this allow the brain to acquire the most of the informations about a new situation. No tension appears on the upper and lower eyelid. As a consequence of surprise, the mouth opens with a downward movement of the jaw that separates dental arches.

1.1.6 Contempt

The contempt is an emotion felt only against people. You don’t feel contempt for objects or smells, instead you feel disgust. It manifests itself when you are facing situations that you think are immoral and you feel a sense of superiority over those did the action. It can happen when you see a man mistreating a woman or if you see a man that mistreat an animal. You feel contempt also when your advice or you opinion are not taken into account.

In Fig. 1.1 you can see a typical expression of contempt. Within the basic emotions, the contempt is the only one that is asymmetrical as it is only on one part of the face. It appears only in the bottom of the face where the mouth tends towards the external part with an angle curved upwards.

1.1.7 Fear

Fear is an emotion from which anyone would give up, not only for the feelings that which involves, but for the causes that trigger it. The priority of the brain is to survive and for achieve this objective, it needs to get away from all the situations that can endanger it, from all that can harm physical, moral and psychological damage. For achieve this objective,
1 The analysis of facial expressions

fear is “activated”: a negative and unpleasant sensation that causes an immediate physical reaction of escape to create distance from the danger. Furthermore, it permits to the brain to remember the danger, combining it with the sensation of fear so that it will avoid it in the future.

In Fig. 1.1, the eyebrows are up and positioned almost horizontally (in contrast to the surprise where they are hold upwards), the internal angles moves towards the center and on the front horizontal lines appears only in the middle (in contrast to the surprise where the horizontal lines cross all the front). Eyes opens because eyebrows and eyelids rise up, but we don’t have the same “wide open” effect like in surprise, as they are tense and upper eyelids tend to rise up. Mouth can be opened like in surprise, but it is not relaxed: it is tense in the lops that tends toward the external part.

1.2 The study of micro expressions

When you look someone that express an emotion, the decoding of his expressiveness seems not to be a significant problem for the most of the people. The conveyed information in them is a part of interactions and social relationships and its elaborations does not represent a problem. On the other hand, researchers sometimes disagree on the interpretation of the informations conveyed from the face and on the most suitable method for study them. The main problem lies in the recognition or not, in the existence or not and in the nature of a behavioral intention or of an affective state inside face expressions. To answer to different questions regarding the existing connections between facial micro expressions and personality traits, the emotive experience and communicative processes, from 60’s have been studied several techniques for the recognition and the analysis of facial expressions.

The techniques that focus on the face, as the main element in the expression of emotions, are divided in judgment and measurement studies.

The latter are focused mainly in the recognition of movements, that is to say the aspects that don’t take into account what that want to be said with facial behavior. They are based on the recognition of visible behavioral facial units. They are different from judgment methods as, despite they need the judgment of an observer, the assessments are purely descriptive and not interpretative. These studies are further divided into two groups: the ones based on theoretical reflection and the ones based on the muscular anatomy of the face.

The first ones, identify the movement of facial combinations associated with particular emotions, the “universal” ones, but they don’t allow the measurement the intensity of the behavior and they don’t explain to action different from the default ones, as they are a prototype of determined emotive states. To this category belong the Facial Affect Scoring Technique of P. Ekman et al. (1971), the Maximally Discriminative Facial Movement Coding System of Izard (1979) and the System for Identifying Affect Expression by Holistic Judgment of Izard and Dougherty (1980).

Considering the studies about muscular anatomy, the first work was presented by Hjortsjö (1969). Learning to voluntarily move the facial muscles, Hjortsjö developed a precise description of the change resulting from every facial muscular movement and established a
1.3 FACS method: Facial Action Coding System

Ekman and Friesen developed the Facial Action Coding System (1978) as an observation and classification system of every visible facial movements, also the minimum ones, with respect to their anatomical components that are the building elements of facial exhibitions. Among the various system of analysis and measurement of facial expressions studied so far, the FACS is the most understandable, complete and versatile. In this system, the face is considered as a multidimensional response system, flexible and specific. The face conveys informations through four signal classes:

- **static signals**: they represent almost permanent traits of the face, as its structure and the subcutaneous tissue that shape the face;
- **slow signals**: they are composed by the changes that happens on the face over time and are represented by wrinkles and in the changing of the skin;
- **artificial signals**: they are represented from the traits of the face determined by artificial elements, such as glasses, cosmetic products, surgery etc.;
- **fast signals**: they are variations caused by neuromuscular activity and determine the actual facial expressions.

The sum of the four classes of signals, determine the physiognomy of a face and individually communicate many messages. To study emotions we analyze fast signals that change eyes, eyebrows, mouth and lips shape.

To give a specific index for every type of movement and expression, the FACS take into account 44 fundamental units named “Action Units” by Ekman and Friesen (in the following we will name them AU), that can give rise to 7000 possible combinations. Totally, 58 different types of movements have been classified, while other are not associated to any particular emotion.

For every AU in the FACS manual are explained:

- in section A: the changes visible on the face, caused by that movement;
- in section B: informations about perform that specific movement, with the description of any difficulties and useful solution to overcome them;
The analysis of facial expressions

• in section C: how to register the intensity of that AU.

The FACS is a purely descriptive observation system, so not subject to give immediately an interpretative meaning to the facial mimicry. To achieve this aim, the authors referred only to the analysis of the anatomic and human face movements. As each of this movements is the result of the action of facial muscles, the FACS take into account the way in which every muscle acts in modifying visibly the configuration of the face itself. Furthermore, Ekman and Friesen, referred to the Duchenne technique, that consists in the stimulation different muscle fibers with electrodes to determine the kinesics effects externally visible.

The main limitation of FACS, is the fact of taking into account only the visible changes of human face and not the not visible ones. It is an explicit choice of Ekman for consider only what can be effectively used in the non verbal communication. In FACS indeed, are not considered the static characteristics of the face, such as makeup, hairstyle and facial expression as blush and sweating. For the authors, in this situation, should be involved other research methodologies. To precisely describe every AU, Ekman and Friesen had taken into account the body language in his movement and not only in the final static configurations.

A relevant aspect of this methodology is the possibility of detect the levels of intensity of every muscular activity. The authors have created an objective tool for the categorization of facial movements, that can be used according to the researches that one want to explore and that will be include its application.

The intensity levels are five and are:

- A: weak presence of action;
- B: light evidence;
- C: clear signals of movement;
- D: intense signals of movement;
- E: maximum intensity of the movement.

This intensity scale has not regular intervals: levels A, B and E have a small amplitude, while level C and D include the most of facial movements. In an analysis of a facial movement, the letter coupled with a given level of recognized intensity is put aside the number of relative AU. For instance, it is possible to detect the difference of intensity for the AU which is associated with the raise of the lower eyelid due to the action of the orbicular muscle of the eye in two images assessed by the same authors respectively with intensity B and E.

With the creation of this coding system, the authors created also a manual published in 1992 for the researchers who want to do research in the analysis of the analysis of facial expressions. A FACS analyst indeed, in addition to know the AU, has also to be able to select an observed expression, dividing it in the single unities that underline the facial movement. He has to be able to score intensity, assess the presence and the absence of
1.3 FACS method: Facial Action Coding System

Table 1.1: A table for the interpretation of emotions.

<table>
<thead>
<tr>
<th>Emotion</th>
<th>AU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Happiness</td>
<td>6+12</td>
</tr>
<tr>
<td>Sadness</td>
<td>1+4+15</td>
</tr>
<tr>
<td>Surprise</td>
<td>1+2+5B+26</td>
</tr>
<tr>
<td>Fear</td>
<td>1+2+4+5+20+26</td>
</tr>
<tr>
<td>Anger</td>
<td>4+5+7+23</td>
</tr>
<tr>
<td>Disgust</td>
<td>9+15+16</td>
</tr>
<tr>
<td>Contempt</td>
<td>6+R12A+R14A</td>
</tr>
</tbody>
</table>

every muscular action and also the presence of bilateral asymmetries, identified by letters L or R (Left or Right side of the face).

The score units, that are the list of the AU involved in a facial expression with the respective intensity scores, are purely descriptive: they not interfere with the interpretation of emotions and can be converted by a software that uses an interpretation dictionary, as the one in Table 1.1.

The advantages offered by the use of FACS as system of analysis of facial expressions are therefore methodological: it is a system based on the muscular anatomy of mimic muscles, as such it is more precise and allow to identify and detect every possible movement with the exact number of muscles that contract or relax to execute it.

The attribution of the movements to AU, instead of the single muscular action, makes this coding more clear, as every AU identifies a single facial movement, while a muscle can underline more movements, so a facial action can be obtained by the action of more than a muscle. The main drawback stays in the difficulty of learn the techniques and the time necessary for its use.
2 Gaussian Process Latent Variable Model

In the previous chapter we gave a short introduction to the reader of what is an emotion and what is a facial expression. Furthermore, we described the FACS system, the most used tool for the measurement and analysis of facial expressions. Finally, we described the Cohn-Kanade Database, that will be the emotional database that we are going to use when we will instantiate our model. Now we formally describe the tool that we will use for building our internal visuomotor mapping which we will use in a predictive manner to interpret the perceived facial cues from an observer. In particular, we will exploit a dimensionality reduction tool, contained in the class of Latent Variable Models (LVM): we will establish the technical foundations required for understanding the Gaussian Process Latent Variable Model (GP-LVM), a probabilistic nonlinear dimensionality reduction model, following [15].

The chapter is organized as follows: after an introduction to the problem, we briefly recall Principal Component Analysis (PCA) and we review its probabilistic equivalent that contributes to the derivation of GP-LVM. We will end by summarizing the main advantages of using GP-LVM.

2.1 A general introduction

When we resort to machine learning methodologies to face a particular problem, we have often to deal with high dimensional data. Many common problems can be considered: in a classification problem we want to determine the class of some new observations through a classifier learned from a set of training data; in clustering, the goal is to group a set of data so that those in the same group, a cluster, are more similar (considering some notion of similarity) to each other than to data in other groups [4, 30]. Considering in particular the last task, we can notice that the purpose of its use is to understand the inherent low dimensional structure of a given data set. Either way, the main goal of machine learning techniques is to learn the low dimensional underlying patterns from those set of, eventually high dimensional, data [4].

In the literature, we can find various machine learning methods to face the above problems [37, 4, 30, 26]. Beyond these, Latent Variable Models (LVM), [4, 5, 46], have been used in image recognition [53], information retrieval [9] and speech recognition [13]. In principle, a LVM refers to a statistical model that relates a set of variables, the so-called manifest variables, to a set of latent variables under the assumption that the responses on the manifest variables are controlled by latent variables. A remarkable feature of these models
lies in the fact that we can provide different meanings to latent variables, depending on a specific task. This flexible definition has made LVM largely be used in several machine learning tasks. In the clustering problem, latent variables can represent the clustering membership of samples [44]. In the main topic of this review, dimensionality reduction, we assume that latent variables are the low dimensional representations of high dimensional samples.

Moving beyond, many machine learning models can be considered as special cases or variants of LVM, e.g. neural networks [27] and Principal Component Analysis [46]. Among these, Gaussian Process Latent Variable Models (GP-LVM) represent a large class of LVM and have been applied in many machine learning contexts. Such models can be seen as a mix of LVM and Bayesian non parametric Gaussian Process (GP). A GP is a probabilistic model used in regression, classification and clustering [55, 71]. At first sight, we can consider these GP-Based models to be a set of LVM where each observed variable is the sum of the corresponding latent variable and noise. The difference from other LVM is that latent variables can be considered as functional variables which are the noise free form of observed variables. In LVM, the aim is to learn latent variables or the underlying pattern of data. GP-LVM try to infer the target variable of new sample by integrating out the latent variables. This stands for the major difference between the latter and other LVM. For the purpose of infer latent variables, GP-LVM assumes that the functional variables are generated by GP from some low dimensional latent variables. It is these that we should infer from data. Hence, we can learn latent variables by integrating out the functional variables and maximizing the log marginal likelihood. Even if GP-LVM was first proposed for dimensionality reduction, it has been widely used in many scenarios, such as Facial Expression Recognition [22] and Human pose estimation [20].

LVM have been widely studied and extended. Here we are going to present a review and an analysis especially of one of them, GP-LVM, focusing on the problem of dimensionality reduction. In the next paragraphs we describe the foundations necessary to understand the model. We start by recalling the idea of PCA and the definition of the multivariate gaussian distribution. It is important to revise the latter because we are going to make calculations on it many times. Then, we introduce a probabilistic version of PCA from which [38] derive GP-LVM. We will end this chapter summing up which are the main advantages of using GP-LVM.

### 2.2 Principal Component Analysis

If we consider the set of multivariate analysis methods, Principal Component Analysis (PCA) is a statistical technique used to perform a dimensionality reduction which is linear. It was originally introduced by [52], and independently developed by [28], who presented the standard algebraic derivation of PCA. For such reason it is also known as Hotelling Transform.

To recall the idea of PCA, consider a set of observable variables that we can measure. It is common that the collected data show that some variables change together: this relation is formally measured as the covariance. PCA searches for an orthogonal set of principal components that retain maximal variance. A principal component can be seen...
as a combination of the observed variables. If two observed variables strongly covary linearly, then it makes sense to describe the data with a single variable instead. The more linearly observed variables covary, the less information we lose from choosing a smaller set of principal components, thus effectively reducing dimensionality.

Let us now give a more formal definition of principal component. Given a set of \( n \) observed \( d \)-dimensional data represented as a design matrix, \( X = [x_1, ..., x_n]^T \), the \( q \) principal components \( w_j, j \in \{1, ..., q\} \), are the orthonormal axes with maximal variance retained. The first principal component is a linear function \( \alpha_1^T X \) that retains most variance of \( X \), where \( \alpha_1 = [\alpha_{11}, \alpha_{12}, ..., \alpha_{1n}] \) is a vector of \( n \) constants such that [31, p.4]:

\[
\alpha_1^T X = \alpha_{11} x_1 + \alpha_{12} x_2 + ... + \alpha_{1n} x_n = \sum_{i=1}^{n} a_{1i} x_i.
\]

The following principal components are found by looking for a linear function that is orthogonal to the selected principal components and retain maximum variance.

PCA can be performed by singular value decomposition (SVD) of design matrix \( X \) [31, pp.44-46]. SVD is a factorization by which we can write the design matrix as

\[
X = U \Sigma V^T,
\]

where given \( r = r(X) \) denotes the rank of \( X \), then \( U \in \mathbb{R}^{n \times r} \) is a matrix of orthonormal columns that are the left singular vectors, \( \Sigma \in \mathbb{R}^{r \times r} \) is a diagonal matrix of the singular values of \( X \), and \( V \in \mathbb{R}^{d \times r} \) is a matrix of orthonormal columns that are the right singular vectors.

A limitation of standard PCA is the lack of a probabilistic solution. One way to formulate a probabilistic model is to introduce the notion of noise among observed variables. It is often the case that we assume the noise to follow a Gaussian distribution. In the remainder of this review the Gaussian distribution will take a central role, therefore we will proceed to describe its characteristics in detail.

### 2.3 Multivariate Gaussian Distribution

The Gaussian (normal) distribution is a reasonable prior assumption for data that is subject to the central limit theorem, which states that as the sample size of a population tends to infinity, the distribution becomes normally distributed [4, p.78]. A random variable \( X \) that is normally distributed with mean \( \mu \) and variance \( \sigma^2 \) is denoted as

\[
X \sim \mathcal{N}(\mu, \sigma^2).
\]

The Gaussian density for a single variable \( y \) is expressed as [4, p.78]:

\[
\mathcal{N}(y|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp \left( -\frac{(y - \mu)^2}{2\sigma^2} \right),
\]

\[
\mathcal{N}(y|\mu, \sigma^2) \equiv p(y|\mu, \sigma^2).
\]
For a $d$-dimensional vector $y$, the multivariate Gaussian distribution with mean vector $d$-dimensional $\mu$, $d \times d$ covariance matrix $\Sigma$, and the determinant of $\Sigma$ as $|\Sigma|$, is

$$
\mathcal{N}(y|\mu, \Sigma) = \frac{1}{(2\pi)^{\frac{d}{2}} \sqrt{|\Sigma|}} \exp\left(-\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right).
$$

The Gaussian distribution is closed under addition (2.1), scaling (2.2), and multiplication (2.3) - all of which yields a result that is also a Gaussian distribution [55, p.200]. These notable properties will be useful in later sections as they allow analytical integration of multivariate Gaussian distributions.

$$
\sum_{i=1}^{n} y_i \sim \mathcal{N}\left( \sum_{i=1}^{n} \mu_i, \sum_{i=1}^{n} \sigma_i^2 \right) \quad (2.1)
$$

$$
w y \sim \mathcal{N}(w\mu, w^2 \sigma^2) \quad (2.2)
$$

$$
\mathcal{N}(x|a, A) \mathcal{N}(x|b, B) = Z^{-1} \mathcal{N}(x|c, C), \quad (2.3)
$$

$$
Z^{-1} = \frac{1}{(2\pi)^{\frac{d}{2}} \sqrt{|A + B|}} \exp\left(-\frac{1}{2} (a - b)^T (A + B)^T (a - b) \right),
$$

$$
c = C(A^{-1} a + B^{-1} b),
$$

$$
C = (A^{-1} + B^{-1})^{-1}.
$$

Let $w \sim \mathcal{N}(\mu_1, \sigma_1^2)$ and $h \sim \mathcal{N}(\mu_2, \sigma_2^2)$ be jointly Gaussian distributed variables, if the variables are independent, then $p(w, h) = p(w)p(h)$. The joint probability density is thus,

$$
p(w, h) = \frac{1}{\sqrt{2\pi \sigma_1^2 \sigma_2^2}} \exp\left(-\frac{1}{2} \left( \frac{(w - \mu_1)^2}{\sigma_1^2} + \frac{(h - \mu_2)^2}{\sigma_2^2} \right) \right).
$$

In matrix form, the joint probability is

$$
p(w, h) = \frac{1}{2\pi \sqrt{\sigma_1^2 \sigma_2^2}} \exp\left(-\frac{1}{2} \left( \begin{bmatrix} w \\ h \end{bmatrix} - \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} \right)^T \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix}^{-1} \left( \begin{bmatrix} w \\ h \end{bmatrix} - \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} \right) \right).
$$

Assuming independence, the joint probability density for a $n$-dimensional vector $y$ is expressed as

$$
p(y) = \frac{1}{2\pi \sqrt{|D|}} \exp\left(-\frac{1}{2} (y - \mu)^T D^{-1} (y - \mu) \right), \quad (2.4)
$$

where $D \in \mathbb{R}^{n \times n}$ is the diagonal matrix of the variances [4, p.78].
2.4 A Probabilistic Model for PCA

Tipping and Bishop introduced a probabilistic framework for principal component analysis by constraining the noise distribution of a LVM [61].

A LVM transforms a set of \( n \) \( d \)-dimensional observed variables encoded as a design matrix, \( Y = [y_1, ..., y_n]^T \), to a set of \( n \) \( q \)-dimensional latent (unobserved) variables, \( X = [x_1, ..., x_n]^T \). Latent variables are parsimonious, it is generally the case that \( q \ll d \), explaining the original data with fewer variables. A notable latent variable model is that of factor analysis, one that assumes linearity in relation of the observed data set. For each observed data point, \( y_i \in Y, 1 \leq i \leq n \), there is an associated latent variable \( x_i \). The original data can be represented in terms of the corresponding latent variable as

\[
y_i = W x_i + \mu_i + \epsilon_i. \tag{2.5}
\]

The matrix \( W \) represents the linear relationship between the latent space with the data-space. Figure 2.1 shows how the input latent variable and the matrix that model the relation contributes to predicting an output observed variables. The parameter \( \mu_i \) allows for non-zero mean, and the \( \epsilon_i \) parameter represents noise within the model. We notice that standard PCA can be viewed as a variant of factor analysis where the noise parameter is not accounted for and furthermore, PCA reduce dimensionality without explaining the relation between the original data and its principal components. Regression is possible with a model that explains the relation between observed variables and latent variables.

**Figure 2.1:** A LVM learns the mapping of input \( x_i \) to output \( y_i \) by encoding the relationship in matrix \( W \), which models the behaviour of the underlying function.

The latent variable model developed by Tipping and Bishop performs principal component analysis by modelling the noise parameter of equation 2.5 as an isotropic, spherical Gaussian distribution. The noise values, \( \epsilon_i \in \mathbb{R}^{d \times 1} \), are sampled from a independent spherical Gaussian distribution

\[
\epsilon_i \sim N(0, \beta^{-1} I).
\]

The conditional probability distribution of a observed variables \( y_i \) given input variables \( x_i \) is thus Gaussian distributed as

\[
p(y_i|x_i) = N(W x_i + \mu_i, \beta^{-1} I).
\]

The prior of latent variables is assumed to be standard Gaussian with zero mean and unit covariance, defined as \( x_i \sim N(\mathbf{0}, I) \). The marginal distribution for the observed data \( y_i \) is obtained by integrating out the latent variables. From equation 2.4, an arbitrary rotation matrix \( R^T \) can be applied to the basis, forming the correlated Gaussian,

\[
p(y_i) = \frac{1}{2\pi \sqrt{|D|}} \exp \left( -\frac{1}{2} (R^T y_i - R^T \mu_i)^T D^{-1} (R^T y_i - R^T \mu_i) \right).
\]
This gives an eigenvalue decomposition of the inverse covariance matrix, and thus the covariance matrix,
\[ C^{-1} = RD^{-1}R^T, \]
\[ C = RDR^T. \]
As a consequence, we can derive that given \( x_i \sim \mathcal{N}(\mu_i, \beta^{-1}) \) and \( y_i = Wx_i \), then the distribution of the observed variables can be denoted as \( y_i \sim \mathcal{N}(W\mu_i, W\beta^{-1}W^T) \). Thus, with a prior as standard Gaussian, \( \mathcal{N}(0, I) \),
\[ Wx_i \sim \mathcal{N}(0, WW^T), \]
\[ y_i \sim \mathcal{N}(0, C), \]
where the observation covariance model is \( C = WW^T \beta^{-1}I \), with corresponding log-likelihood
\[ \mathcal{L} = \frac{n}{2} (d\ln(2\pi) + \ln|C| + \text{tr}(C^{-1}S)), \tag{2.6} \]
\[ S = \frac{1}{n} \sum_{i=1}^{n} (y_i - \mu)(y_i - \mu)^T. \]
We can write the likelihood for a data point as
\[ p(y_i|x_i, W, \beta) = \mathcal{N}(y_i|Wx_i, \beta^{-1}I). \tag{2.7} \]
Integrating over the latent variables gives the marginal likelihood,
\[ p(y_i|W, \beta) = \int p(y_i|x_i, W, \beta)p(x_i)dx_i. \]
As the prior of probabilistic PCA is modelled as a standard Gaussian distribution, \( p(x_i) = \mathcal{N}(x_i|0, I) \), marginalisation of the integral obtains the marginal likelihood of each data point as
\[ p(y_i|W, \beta) = \mathcal{N}(y_i|0, WW^T + \beta^{-1}I). \]
Assuming that the data points are independent, the likelihood of the full data set is the product of each marginal likelihood,
\[ p(Y|W, \beta) = \prod_{i=1}^{n} p(y_i|W, \beta). \]

### 2.4.1 The Principal Subspace of PPCA

Tipping and Bishop showed that all potential solutions for \( W \), the likelihood (2.6), is of the form [61]
\[ W = U_q(K_q - \sigma^2 I)^{1/2}R. \]
One particular case of interest is when the likelihood is maximised,
\[ W_{ML} = U_q LR, \tag{2.8} \]
\[ L = (\Lambda_q - \sigma^2 I)^{1/2} \]
2.5 Gaussian Process Latent Variable Model

The matrix $U_q$ contains the column vectors that are the principal eigenvectors, $A_q = [\lambda_1, ..., \lambda_q]$ represents the diagonal matrix of the corresponding eigenvalues, and $R$ represents an arbitrary orthogonal rotation matrix. Maximising the likelihood of $W$ by equation 2.8 on the latent variable model defined by equation 2.5 maps the latent space to the principal subspace of the observed data. Satisfying $W_{ML}$, the latent variable model, is effectively equivalent to standard principal component analysis.

2.5 Gaussian Process Latent Variable Model

The PCA formulation by Tipping and Bishop allows a probabilistic model, however, it assumes the relation of data is still linear. The GP-LVM is a nonlinear latent variable model derived from a dual of the probabilistic PCA by replacing the inner product kernel with Gaussian processes [38]. A nonlinear embedding is a more suitable model for capturing high dimensional data.

2.5.1 Dual Probabilistic PCA

The dual probabilistic PCA introduced by Lawrence allows for latent mappings to be non-linearised through the kernel trick by performing computations implicitly in the high dimensional space [55, p.8]. Dual probabilistic PCA marginalises the parameters, $W$, and optimises latent variables, $X$. It is the dual approach of the standard probabilistic PCA where the parameters are optimised and marginalises the latent variables.

First, a conjugate prior to the likelihood of probabilistic PCA (2.7) is taken to be a spherical Gaussian distribution,

$$p(W) = \prod_{i=1}^{d} \mathcal{N}(w_i | 0, I).$$

As marginalisation of both $W$ and $X$ is often intractable in practice, $W$ is selected for marginalisation as the conjugate prior is Gaussian distributed, thus, it can be integrated analytically. The marginalised likelihood of $W$ is

$$p(Y|X, \beta) = \prod_{i=1}^{d} p(y_{i,:}|X, \beta),$$

The $y_{i,:}$ parameter represents the $i^{th}$ column of $Y$, where

$$p(y_{i,:}|X, \beta) = \mathcal{N}(y_{i,:} | 0, XX^T + \beta^{-1} I).$$

(2.9)

The objective function is the log-likelihood

$$L = -\frac{dn}{2} \ln 2\pi - \frac{d}{2} \ln |K| - \frac{1}{2} tr(K^{-1}YY^T),$$

(2.10)

$$K = XX^T + \beta^{-1} I.$$

In the original paper, Lawrence found the gradients of the log-likelihood (2.10) with respect
to $X$ as

$$\frac{\sigma L}{\sigma X} = K^{-1}YY^TK^{-1}X - dK^{-1}X.$$ 

A stationary point where the gradients are zero is given by

$$\frac{1}{d}YY^TK^{-1}X = X.$$ 

The values for $X$ which maximise the likelihood are given by singular value decomposition of $X$,

$$X = U\Sigma V^T.$$ 

$U$ is a matrix whose orthonormal column vectors are the first eigenvectors of $YY^T$. $\Sigma$ is a diagonal matrix of singular values, whose $i$th element is $l_i = (\lambda_i - \frac{1}{d})^{-\frac{1}{2}}$, where $\lambda_i$ is the eigenvalue associated with the $i$th eigenvector $\frac{1}{d}YY^T$. $V$ is an arbitrary rotation matrix. Lawrence showed that the eigenvalue problem developed here is equivalent to the eigenvalue problem solved in probabilistic PCA, and thus, dual probabilistic PCA is also effectively equal to standard PCA when the likelihood is maximised. Dual probabilistic PCA assumes that the output dimensions are linear, independent, and identically distributed. Infringing upon these assumptions derive new probabilistic models.

### 2.5.2 Gaussian Processes

A Gaussian process (GP) is a non-parametric statistical model that is a distribution over functions. It treats each observed variable as an independent distribution. A prior probability distribution is an assumption of belief before taking into account of evidence. Observing the output of a continuous function provides information regarding its behaviour around that specific point. For a noiseless model, we can be certain that the input must intersect the point at an observed output, as shown in figure 2.2. An observation refines our belief to obtain a posterior. A model with noise can infer that the mapping is nearby the observation, thus treating each observed variable as an independent distribution allows us to take into account of the noise model for prediction.

Formally, a Gaussian process is the infinite generalisation of the Gaussian distribution, specified by a mean function $m(x)$ and covariance function $k(x, x')$ of a real process $f(x)$ [55, p.13],

$$f(x) \sim GP(m(x), k(x, x')),$$

where

$$m(x) = \mathbb{E}[f(x)],$$

$$k(x, x') = \mathbb{E}[(f(x) - m(x))(f(x') - m(x'))].$$

The mean function of a GP is generally assumed to be zero, unless stated otherwise. Polynomial regression models yield best results when the behaviour of the observed data resembles the selected polynomial function, but determining a suitable function is challenging. As a non-parametric model, Gaussian processes provides a probability distribution over a space of functions that associates a likelihood for each function, presenting the opportunity to sample various functions that encode the behaviour of observed data. GP regression models can be overfitted, but overfitting a space of functions is more lenient than overfitting a single function.
2.5 Gaussian Process Latent Variable Model

![Figure 2.2: Plot of a Gaussian process prior (up) and posterior (down). The semi-transparent functions represent samples from the likelihood distribution (image from: https://pythonhosted.org/infpy/gps.html).](image)

2.5.2.1 The use of Bayesian reasoning

The observed data \( y \) is assumed to be modelled by a function \( f(x) \) with input \( x \) and corrupted by noise, \( \epsilon \). Noise interference depends on the problem, a simple relationship is a model with additive noise,

\[
f(x) = x^T w \quad y = f(x) + \epsilon.
\]

Here, \( w \) represents the parameters that specify the behaviour of the function. Take \( X \) as the design matrix of the input values. Suppose that the noise on each observed variable is modelled by an independent Gaussian distribution, \( \epsilon \sim \mathcal{N}(0, \sigma_n^2) \), the joint likelihood is
the product of marginal likelihoods,
\[ p(y|X, w) = \prod_{i=1}^{n} p(y_i|x_i, w) = \mathcal{N}(X^T w, \sigma^2_n I). \]

In \textit{Bayesian reasoning}, we update our prior knowledge with observed evidence to obtain a posterior. This methodology is expressed in \textit{Bayes’ Theorem},
\[ \text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{marginal likelihood}}, \quad p(w|y, X) = \frac{p(y|X, w)p(w)}{p(y|X)}. \]

The marginal likelihood is given by
\[ p(y|X) = \int p(y|X, w)p(w)dw. \]

The posterior expresses what we know about the parameters using the likelihood and the prior. We can then use the posterior to make an informed prediction for test inputs.

The dual probabilistic PCA model uses a Gaussian process prior that is corrupted by Gaussian noise \[\epsilon \sim \mathcal{N}(0, \beta^{-1} I)\]. The covariance function (kernel) is thus,
\[ k(x_i, x_j) = x_i^T x_j + \beta^{-1} \delta_{ij}. \quad (2.11) \]

Parameters \(x_i\) and \(x_j\) are vectors from the space of inputs to the function and \(\sigma_{1ij}\) represents the Kronecker delta, defined by
\[ \delta_{ij} = \begin{cases} 1, & \text{if } i = j, \\ 0, & \text{if } i \neq j. \end{cases} \]

Taking inputs from matrix \(X\) and evaluating the covariance function at each observed variable gives the covariance matrix,
\[ K = XX^T + \beta^{-1} I. \]

The element at \(i_{th}\) row and \(j_{th}\) column of \(K\) is given by the prior distribution (2.11). Thus, the marginal likelihood of dual probabilistic PCA is a product of \(d\) independent Gaussian processes. The covariance function of a Gaussian process describes the properties of functions, such as variability. Learning in Gaussian processes is to determine hyperparameters of a covariance function that is suitable for the problem being modeled.

\section*{2.6 Summary}

We have explained that dimensionality reduction with LVM aspires to explain the underlying behavior of observed data. Dimensionality reduction was introduced through PCA and was formulated a probabilistic framework. We finally looked at the derivation of nonlinear latent variable models such as GP-LVM.
Now that we have established a foundation for probabilistic LVM, we will look at the advantages of use GP-LVM. First, most of the existing LVM are parametric models in which there is a strong assumption on the projection function or data distribution. Such a parametric construction form partly loses flexibility in modeling. Further, besides widely used in dimensionality reduction, GP-LVM can also be extended to other machine learning tasks due to its characteristics below. Firstly, its distribution-free assumption on prior of latent variables provides us a lot of opportunities to improve it. Secondly, we can also exert classical kernel methods for a further expansion of GP-LVM [43].

We finally remark that the optimization process of a GP-LVM made adopting a MAP approach, however, hides a big drawback as pointed out in [62]: sensitiveness to overfitting is caused by the fact that MAP is a point estimation and tends to maximize with respect to $X$ and the hyperparameters. For such reason, in [62] it is presented a variational Bayesian approach to marginalise the latent variables $X$, optimising the resulting lower bound on the marginal likelihood with respect to the hyperparameters.
3 Particle Filters

In the previous chapter we have described the main tool we will use for building our internal visuomotor mapping: the GP-LVM. Now we briefly describe the important class of approximate inference methods of Particle Filters (PF) [18, 19, 17, 59, 3], that will be crucial in the description of our model.

In order to describe such class of inference methods suppose, generally speaking, that we know some outside information about a system and that we can get observations which depend on the system state. If the actual state of the system is hidden from us, how can we find such state relying only on our observations? It turns out that we can infer the hidden state with a procedure called particle filtering process. This one can be performed in several ways, for such reason we talk about the class of PF.

To provide a simple example of use of PF, let us to think about a robot positioned in a room for which we know the floor plan, but we don’t know the position of our robot. However, if the robot is able to record some kind of measurements with a sensor, for example its distance to walls, we can use the saved data and our prior information about the floor plan to make predictions about where the robot could be positioned [59].

We will now formally describe PF, but before will precisely define the problem setting in the next section. Only once, we will derive PF algorithms.

3.1 Hidden Markov Models (HMM)

To model the inference problem we have described in the previous section we have to introduce the concept of non observable system and of measure on it. These two concepts can be conveniently and formally grouped with Hidden Markov Model (HMM) [54, 59]. This is defined as a Markov process in which states that cannot be directly observed, but with outputs that depend just on the hidden state of the system. If we let the random variable (RV) $X_i$ represent the $i$-th system state, and the RV $Y_i$ represent the $i$-th system output, then we have:

$$X_1 \sim \mu(x_1),$$

$$X_i|(X_{i-1} = x_{i-1}) \sim f(x_i|x_{i-1}),$$

$$Y_i|(X_i = x_i) \sim g(y_i|x_i).$$

where $X_1$ is sampled by the prior $\mu(x_1)$ and $f$ follows the Markov property, therefore the current state $x_i$ depends only on the previous state $x_{i-1}$. Also we remark that $g$ points out that the current observation $y_i$ depends on the already mentioned current state $x_i$. 


Let us now define with \( x_{1:t} \) the ordered \( t \)-sequence of hidden states \((x_1, \ldots, x_t)\) where \( x_i \) is the \( i \)-th hidden state. In the same way, we name \( y_{1:t} \) the ordered \( t \)-sequence of observations \((y_1, \ldots, y_t)\). Taking into account of this notation, we can directly derive two important probability distributions from our HMM setting [54]: the first one is the probability of the \( t \)-sequence of hidden states \( x_{1:t} \),

\[
p(x_{1:t}) = p(x_1)p(x_2| x_1)p(x_3| x_2, x_1) \ldots p(x_t| x_{t-1}, \ldots, x_1) \\
= p(x_1)p(x_2| x_1)p(x_3| x_2) \ldots p(x_t| x_{t-1}) \\
= \mu(x_1) \prod_{i=2}^{t} f(x_i|x_{i-1}),
\]

and the second one is the probability of the \( t \)-sequence of observations \( y_{1:t} \), conditioned on the hidden states sequence \( x_{1:t} \)

\[
p(y_{1:t}|x_{1:t}) = p(y_1|x_{1:t})p(y_2|x_{1:t}, y_1)(y_3|x_{1:t}, y_2, y_1) \ldots (y_t|x_{1:t}, y_{t-1}, \ldots, y_1) \\
= p(y_1|x_1)p(y_2|x_2) \ldots (y_t|x_t) \\
= \prod_{i=1}^{t} g(y_i|x_i).
\]

### 3.1.1 Inference in HMM

Now that we have stated the concepts of system and measure on it, we formally define the inference problem [54, 59]. Infer the state of the system means the possibility of modeling the distribution \( p(x_t|y_{1:t}) \) or the full conditional distribution \( p(x_{1:t}|y_{1:t}) \). Once we can model these distributions we infer the hidden state or the ordered \( t \)-sequence of hidden states computing the expected value of functions defined on these distributions. Hence we want to use the available observations to figure out \( p(x_t|y_{1:t}) \), that comes as the marginal of \( p(x_{1:t}|y_{1:t}) \):

\[
p(x_t|y_{1:t}) = \int p(x_{1:t}|y_{1:t}) dx_{1:t-1}.
\]

If we consider the marginal of \( p(x_{1:t}|y_{1:t}) \), we can refactor the full conditional distribution by means of our prior distributions (3.4) and (3.5). In this way we can write:

\[
p(x_{1:t}|y_{1:t}) = \frac{p(x_{1:t}, y_{1:t})}{p(y_{1:t})} \\
= \frac{p(x_{1:t})p(y_{1:t}|x_{1:t})}{p(y_{1:t})} \\
= \frac{p(x_{1:t})p(y_{1:t}|x_{1:t})}{\int p(x_{1:t}, y_{1:t}) dx_{1:t}} \\
= \frac{\mu(x_1) \prod_{i=2}^{t} f(x_1|x_{i-1}) \prod_{i=1}^{t} g(y_i|x_i)}{\int \mu(x_1) \prod_{i=2}^{t} f(x_1|x_{i-1}) \prod_{i=1}^{t} g(y_i|x_i) dx_{1:t}}.
\]
Hence,

\[ p(x_t | y_{1:t}) = \int \frac{\mu(x_1) \prod_{i=2}^t f(x_i | x_{i-1}) \prod_{i=1}^t g(y_i | x_i)}{\int \mu(x_1) \prod_{i=2}^t f(x_i | x_{i-1}) \prod_{i=1}^t g(y_i | x_i) dx_1:t-1} \, dx_{1:t-1}. \]  

(3.8)

Now we can make two points: Although the (3.8) seems to be really confusing, it is just the integration of the functions \( f \) and \( g \) that we already know. On the other hand, integrals such (3.8) are not always analytically tractable and we can’t always obtain \( p(x_t | y_{1:t}) \).

When the integrals in (3.8) are tractable we can compute them, but evaluate (3.8) for every time step is highly inefficient. A suitable option is to compute the inference recursively in two parts: these are generally called marginal prediction equation and marginal update equation [59]. Under such idea, we derive the prediction equation

\[ p(x_t | y_{1:t-1}) = \int f(x_t | x_{t-1}) p(x_{t-1} | y_{1:t-1}) \, dx_{t-1}. \]  

(3.9) Prediction Equation

If we integrate out \( x_{t-1} \), we have

\[
p(x_t | y_{1:t-1}) = \int p(x_{t-1} | y_{1:t-1}) \, dx_{t-1}
= \int p(x_t | x_{t-1}, y_{1:t-1}) p(x_{t-1} | y_{1:t-1}) \, dx_{t-1}
\]

and if we use of the Markov property we finally see that

\[ p(x_t | x_{t-1}, y_{1:t-1}) = p(x_t | x_{t-1}) = f(x_t | x_{t-1}). \]

Now we derive the Marginal Update Equation

\[ p(x_t | y_{1:t}) = \frac{p(x_t | y_{1:t-1}) p(y_t | x_t)}{p(y_t | y_{1:t-1})}. \]  

(3.10) Update Equation

The derivation is composed by two parts: First, we will show that (3.10) holds when \( p(x_t, y_{1:t}) = p(x_t, y_{1:t-1}) p(y_t | x_t) \) holds, after we will show that \( p(x_t, y_{1:t}) = p(x_t, y_{1:t-1}) p(y_t | x_t) \). We begin by applying the well known Bayes rule to the left side of (3.10):

\[
p(x_t, y_{1:t}) = \frac{p(x_t | y_{1:t-1}) p(y_t | x_t)}{p(y_t | y_{1:t-1})} p(y_{1:t})
= \frac{p(x_t, y_{1:t-1}) p(y_t | x_t)}{p(y_t | y_{1:t-1})} p(y_{1:t})
= \frac{p(x_t, y_{1:t}) p(y_t | x_t)}{p(y_{1:t})}
= p(x_t, y_{1:t}) p(y_t | x_t).
\]
We now show $p(x_t, y_{1:t}) = p(x_t, y_{1:t-1})p(y_t|x_t)$. Let’s start by the right-side:

$$p(x_t, y_{1:t-1})p(y_t|x_t) = p(x_t|y_{1:t-1})p(y_{1:t-1}) \frac{p(x_t, y_t)}{p(x_t)}$$

$$= p(x_t|y_{1:t-1})p(y_{1:t-1}) \frac{p(x_t|y_t)p(y_t)}{p(x_t)}.$$

We recall that we followed the basic probability rules:

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)},$$

with: $x = x_t, y = y_{1:t-1}, z = y_t$. We can now see that:

$$p(x_t|y_{1:t-1}, y_t) = \frac{p(y_{1:t-1})p(y_t)p(x_t|y_{1:t-1})p(x_t|y_t)}{p(y_{1:t-1}, y_t)}.$$  

Hence,

$$p(x_t|y_{1:t-1}) \frac{p(x_t|y_t)p(y_t)}{p(x_t)} = p(x_t|y_{1:t})p(y_{1:t})$$

$$= p(x_t, y_{1:t}).$$

We have seen how equations (3.9) and (3.10) provide us the marginal prediction and marginal update steps respectively. These can set the basis for an estimation algorithmic approach: by performing these two steps for each time $t$, we obtain $p(x_t|y_{1:t})$ for every $t$, from the initial state until the last state. This approach is much more computationally efficient as we can use only the information coming from the previous iteration, rather than performing a high dimensionality integration for every single time step. Unfortunately, the main drawback of this method lies in the fact that most of the times the integral in the prediction step is not analytically solvable, thus we can’t recursively calculate it [59]. Before going any further, let us mention a case where recursively calculation of the marginal prediction and marginal update steps is possible. Then in Subsection 3.2 we will introduce the particle filtering method, suitable when the marginal prediction and marginal update equations are not recursively calculable.

So, as it is usual, when we have to face a difficult problem, setting some constraints sometimes can make the problem easier: in our case we can require that $p(x_{t-1}|y_{1:t-1})$ is Gaussian, so that also $p(x_t|y_{1:t})$ is Gaussian and also we can set $f$ and $g$ to be linear functions. If such conditions hold, the above recursive procedure does get easier and in the literature it is known as Kalman Filter (KF) [34, 59]. Basically, KF rewrites everything with matrices as $f$ and $g$ are linear functions. Therefore, we can see $x_t$ and $y_t$ as a system of matrix equations, and also prove that (3.9) and (3.10) become Gaussian distributions. Finally, we can obtain a set of equations that give us the KF. We won’t show these and their derivations as it is outside the scope of this discussion and because we want to explore a more general approach.
3.2 Particle Filtering

In case the integral in the marginal prediction step is not analytically solvable, the recursive formulation we have proposed is intractable. What we mean is that, even if (3.9) and (3.10) seems to be recursive, they are not as the integral in (3.9) does not decompose. In these cases, we need to go back to (3.7) and simulate that distribution, that we can rewrite as:

\[ p(x_{1:t}|y_{1:t}) = \frac{p(x_{1:t})p(y_{1:t}|x_{1:t})}{\int p(x_{1:t}, y_{1:t})dx_{1:t}} = \frac{\mu(x_1) \prod_{i=2}^{t} f(x_i|x_{i-1}) \prod_{i=1}^{t} g(y_i|x_i)}{p(y_{1:t})} = p(x_{1:t-1}, y_{1:t-1}) \frac{f(x_t|x_{t-1})g(y_t|x_t)}{p(y_{1:t-1})} \propto p(x_{1:t-1}, y_{1:t-1})f(x_t|x_{t-1})g(y_t|x_t) \]

(3.11)

The above one is calculable since we suppose to know \( f \) and \( g \) and we know \( p(x_{1:t-1}, y_{1:t-1}) \) recursively. We don’t know \( p(y_t|y_{1:t-1}) \), but it is not a problem as we are only interested in the probability of a given sequence of states \( x_{1:t} \) compared to the probability of other states (for such reason we have written \( \propto \)). Finally, we have these prediction and update steps [59]:

Prediction step: \( p(x_{1:t}|y_{1:t-1}) = f(x_t|x_{t-1})p(x_{1:t-1}|y_{1:t-1}) \)

(3.12)

Update step: \( p(x_{1:t}|y_{1:t}) \propto g(y_t|x_t)p(x_{1:t}|y_{1:t-1}) \)

(3.13)

Let’s think about what we got. We wanted to find a distribution for \( p(x_t|y_{1:t}) \), then we found a solvable equation for \( p(x_{1:t}|y_{1:t}) \) and realized that our goal was just the marginalization of this. We integrated \( p(x_{1:t}|y_{1:t}) \) over \( x_{1:t-1} \) to obtain the marginal but the integral is recursively calculable only under some constraints. When \( p(x_{1:t}|y_{1:t}) \) does not satisfy such constraints no analytical solution can be found and thus we turn to a simulation method: the particle filtering. Hence, in the following section we will introduce a Monte Carlo method that will let us model \( p(x_{1:t}|y_{1:t}) \) by recursively simulating the distributions (3.12) and (3.13) for every time step.

3.2.1 Monte Carlo Methods

In general, not just in our case, we can proxy any distribution \( \pi(x_{1:n}) \) by sampling from \( \pi \) so that \( X^{(i)}_{1:n} \sim \pi(x_{1:n}) \) for \( i = 1, \ldots, N \), and assigning each sample a uniform probability of \( 1/N \) [17, 18]. For instance, if \( x_{1:n}^{(i)} = (x_1, \ldots, x_n) \) the probability that \( X_{1:n} = (x_1, \ldots, x_n) \) is \( 2/N \). We will name \( x_{1:n}^{(i)} \) as the \( i \)-th sample of the sequence \( x_{1:n} \). In the same way, we will name \( x_{n}^{(i)} \) as the \( i \)-th sample for the state \( x_n \). Therefore, we can write
3 Particle Filters

our model of $\pi$:

$$\hat{\pi}(x_{1:n}) = \begin{cases} 
1/N & \text{at } x_{1:n}^{(1)} \\
\vdots \\
1/N & \text{at } x_{1:n}^{(N)} 
\end{cases} \quad (3.14)$$

We can exploit the proposed framework as we can build approximations of (3.12) and (3.13) sampling from our transition distribution for each time step $x_t^{(i)} \sim f(x_t|x_{t-1})$ for $i = 1, \ldots, N$. Taking these samples, we proxy the transition distribution where each particle $i$ has probability of $1/N$. So far, we only consider the probability of $x_t$, given the previous state $x_{t-1}$:

$$\hat{f}(x_t|x_{t-1}) = \begin{cases} 
1/N & \text{at } x_t^{(1)} \\
\vdots \\
1/N & \text{at } x_t^{(N)} 
\end{cases} \quad (3.15)$$

Just like it is, this method does not sound very helpful as we are modeling a distribution from which we can sample from: therefore it seems reasonable that we already know analytically the distribution. In the next section we are going to see how this simulation technique will come into play.

3.3 Importance Sampling & Sequential Importance Sampling

Here we will analyze the method of particle filtering. In the following, we will deal with the full posterior $p(x_{1:t}|y_{1:t})$, but anyway simulating the latter is the same as simulating our target distribution: the marginal of the full posterior, $p(x_t|y_{1:t})$. The only noticeable difference lies in the fact that the full posterior will be $t$-dimensional while our target distribution will be just $t$-th dimensional.

**Importance Sampling** [18, 59, 19] (IS) is a technique that enable us to sample from a proposal distribution $q(x_{1:n})$, that we know (e.g. a Uniform or Gaussian distribution), and then assign a weight to the samples from this proposal distribution to match our target distribution $t(x_{1:n})$ from which we would like to sample from. At a glance, this may sound counter-intuitive (if can compute the weight between the target and proposal distributions, why shouldn’t we know the target?), but if we think the weight to be our previous function $g$, IS begins to be meaningful. For now, assume we can proxy the prediction step by assigning to each sample $x_{1:t}^{(i)}$ a probability (weight) $w_{1:t-1}^{(i)}$, then we can then proxy the update step as:

$$\hat{p}(x_{1:t}|y_{1:t}) = \begin{cases} 
w_{1:t-1}^{(1)} \frac{g(y_t|x_{1:t}^{(1)})}{\sum_{j=1}^{N} w_{1:t-1}^{(j)} g(y_t|x_{1:t}^{(j)})} & \text{at } x_{1:t}^{(1)} \\
\vdots \\
w_{1:t-1}^{(N)} \frac{g(y_t|x_{1:t}^{(N)})}{\sum_{j=1}^{N} w_{1:t-1}^{(j)} g(y_t|x_{1:t}^{(j)})} & \text{at } x_{1:t}^{(N)} 
\end{cases} \quad (3.16)$$

Now we propose our first particle filtering algorithm: **Sequential Importance Sampling** (SIS). This easily comes from the last concepts we have introduced. We have to proxy the prediction and update distributions and we have already written the prediction step. For
3.3 Importance Sampling & Sequential Importance Sampling

each time step \( t \) we estimate the update distribution as:

\[
\hat{\pi}(x_{1:t}|y_{1:t-1}) = \begin{cases} 
  w_{1:t-1}^{(1)} \text{ at } x_{1:t}^{(1)} \\
  \vdots \\
  w_{1:t-1}^{(N)} \text{ at } x_{1:t}^{(N)} 
\end{cases}
\]  

(3.17)

in which \( w_{1:t-1}^{(i)} \) represent the weights followed by a normalization, up to time \( t-1 \), defined as it follows:

\[
w_{1:t-1}^{(i)} = \frac{w_{1:t-2}^{(i)} g(y_{t-1}|x_{t-1}^{(i)})}{\sum_{j=1}^{N} w_{1:t-2}^{(j)} g(y_{t-1}|x_{t-1}^{(j)})}
\]  

(3.18)

In essence we simulate a particle \( i \) up to time \( t \), given the previous sequence \( x_{1:t-1}^{(i)} \), weighted by \( w_{1:t-1}^{(i)} \). We then sample the current state \( x_{t}^{(i)} \) from the transition distribution and add it to the current sequence \( x_{1:t}^{(i)} = (x_{1:t-1}^{(i)}, x_{t}^{(i)}) \). This ends the prediction step. For the update step, we multiply the probability of \( X_{1:t} = x_{1:t}^{(i)} \), written as \( p(x_{1:t}|y_{1:t-1}) = w_{1:t-1}^{(i)} \), by \( g(y_{t}|x_{t}) \) and finally normalize the weights so that they sum up to 1. We can summarize this procedure in the below algorithm:

1. Initialization:

   For \( i = 1, \ldots, N \):
   
   - Sample \( x_{1}^{(i)} \sim \mu(x_{1}) \)
   - Assign weights \( \tilde{w}_{1}^{(i)} = g(y_{1}|x_{1}^{(i)}) \)
   - Normalize weights \( w_{1}^{(i)} = \frac{\tilde{w}_{1}^{(i)}}{\sum_{i=1}^{n} \tilde{w}_{1}^{(i)}} \)

2. Importance Sampling:

   For \( t = 2, \ldots, T \):

   For \( i = 1, \ldots, N \):
   
   - Sample \( x_{t}^{(i)} \sim f(x_{t}|x_{t-1}^{(i)}) \)
   - Assign weights \( \tilde{w}_{t}^{(i)} = g(y_{t}|x_{t}^{(i)}) \)
   - Normalize weights \( w_{t}^{(i)} = \frac{\tilde{w}_{t}^{(i)}}{\sum_{i=1}^{n} \tilde{w}_{t}^{(i)}} \)

3. Return \( \{x_{T}^{(i)}, w_{T}^{(i)}\}_{i=1}^{N} \)

If we check in the literature, we see that SIS algorithm is hardly used because of the phenomenon of “degeneracy” [19, 3]. Without going into detail, we simply explain what do we mean: when we sample from a weighted distribution, it happens that some points
are coupled with a very low weight. In the next iterations it is likely that this weights become eventually negligible. Hence, we get a bunch of points having nearly most of the weight while the rest that have “degenerated”. This represents a problem because the distribution is now defined by few of the initial points, which is not enough to obtain a good representation of the target distribution.

Despite this remarkable problem, SIS lay the foundation for a widely used algorithm, of which we will discuss in the following lines.

### 3.4 Bootstrap (SIR)

We can try to reduce the degeneracy phenomenon by resampling. After every iteration $t > 0$ we can resample all the $N$ particles from the weighted distribution, defined in the previous time step. In this way, particles with low weights at $t$ will have a low probability of be kept to $t + 1$.

Down here we rewrite the SIS algorithm with the additional resampling step, called Bootstrap or Sequence Importance Resampling [59, 18, 3]. Finally, to clearly visualize the resampling step, check Figure 3.1: a picture is worth 1000 words.

![Figure 3.1: A schematic view of the SIR algorithm.](image-url)
3.4 Bootstrap (SIR)

1. Initialization:
   For $i = 1, \ldots, N$:
   
   Sample $x_1^{(i)} \sim \mu(x_1)$

   Assign weights $\tilde{w}_1^{(i)} = g(y_1|x_1^{(i)})$

   Normalize weights $w_1^{(i)} = \frac{\tilde{w}_1^{(i)}}{\sum_{i=1}^{n} \tilde{w}_1^{(i)}}$

2. Importance Sampling:
   For $t = 2, \ldots, T$:

   For $i = 1, \ldots, N$:

   Sample $x_t^{(i)} \sim f(x_t|x_{t-1}^{(i)})$

   Assign weights $\tilde{w}_t^{(i)} = g(y_t|x_t^{(i)})$

   Normalize weights $w_t^{(i)} = \frac{\tilde{w}_t^{(i)}}{\sum_{i=1}^{n} \tilde{w}_t^{(i)}}$

   Resample $\tilde{x}_t^{(i)}$ from $x_t^{(i)}$ according to the weight distribution with replacement.

3. Return $\{x_T^{(i)}, w_T^{(i)}\}_{i=1}^{N}$

3.4.1 Sampling Impoverishment Problem

While the strategy we have introduced addresses the degeneracy phenomenon, we have to set out that also resampling does present a problem which is called sampling impoverishment problem [18, 3]. This happens because the higher weighted points are more likely to be drawn many times. Rather than having some low weighted points scattered in the state-space — the case of SIS when we have degeneracy — we have lots of uniformly weighted points concentrated in areas where high weighted points initially occurred. To put it simply: degeneracy phenomenon leads to a diverse scattering of point masses with few ill defined peaks and sampling impoverishment leads only to well defined peaks. There exists methods that face both these problems but they require advanced techniques, that we won’t write about: SIR is generally good enough and is the mostly used IS algorithm [18, 19, 3].

Don’t count your chickens before they hatch!
3.5 Conclusion

To conclude, PF are a class of numerical method for proxy hidden states of HMM. We have seen that there are several approaches, but they generally are based on taking a some guesses, and then updating them considering some available information. This permit us to weight the guesses and constantly refine them over a large number of iterations.

Without anticipating too much, in the next chapter we are going to show how we have used the methods we have introduced, GPLVM and PF, to build our model.
4 The Model

In this chapter, at long last, we are ready to explain our model that will allow to perform the predictive tracking of facial expressions of an observed agent. The chapter is organized as follows: in the first part we theoretically introduce the model and after we propose a practical instantiation of it, defining which implementation choices can be done. This will take us to the final part of this work where we will benchmark the model, showing its performances.

To introduce our model, assume for the observer a latent facial action space spanned by the time-varying random variable (RV) \( a(t) \), suitable to condition the dynamics of control parameters for facial actions, say RV \( \Theta(t) \). The latter are used to tune, at any time \( t \), the observer’s internal somatomotor state of face dynamics, indexed by state variable \( w(t) \).

The internal motor model is used to predict observable facial features, along the course of interaction and to infer the facial dynamics by correcting model’s parameters with respect to the observation error. In a nutshell, the observer’s perception and prediction evolves according to the following stochastic state-space equations:

\[
\Theta(t) = f_\Theta(\Theta(t-1), a(t), \epsilon_\Theta(t)),
\]

\[
w(t) = f_w(w(t-1), \Theta(t), \epsilon_w(t)),
\]

\[
y(t) = f_y(w(t), \epsilon_y(t)),
\]

Equations above define a forward model from the action space to the generation of predictions of visual facial features \( y(t) \), via the somatomotor representation \( w(\Theta(t)) \). The inverse problem of defining a correspondence between the perceived features \( \hat{y}(t) \) and the internal motor parameters \( m \in \Theta(t) \), \( \hat{y}(t) \mapsto m(t) \), is named **visuomotor mapping** in robotics [48]. In this work, the inferred visual features \( \hat{y}(t) \) are facial landmarks; the somatomotor representation \( w(\Theta(t)) \) is detailed in the following.

4.1 Internal model

Most generally, assume a 3D deformable face model \( w(t) := w(\Theta(t)) \), being \( \Theta(t) \) the vector of all involved parameters. Namely, \( w = [w_1 \cdots w_N] \) is a vector of vertices \( w_i = [X_i, Y_i, Z_i]^T \), where \( i \) indexes the \( i \)-th vertex of the model. The evolution of the face model at time \( t \) is represented by the state vector \( w_i(t) = [X_i(t), Y_i(t), Z_i(t)]^T \). Under the face-to-face interaction assumption, the face model dynamics is that of a deformable (i.e., not rigid) body. In other terms the face model \( w_T \) can be generated from the standard mean model \( \bar{w} \) which is deformed by both individual biometric characteristics and the facial action performed at time \( t \).
The fundamental theorem of Helmoltz [66] establishes that the most general motion of a sufficiently small element of a general deformable body can be represented as a rotation, a deformation (extension, contraction) and a translation of its parts. At any time $t$, facial movement will locally move a 3D vertex $w_i$ to position $w'_i = w_i(t + \delta t)$ according to the law

$$w_i(t + \delta t) = w_i + \mathbf{d}w_i,$$  \hspace{1cm} (4.4)

where

$$\mathbf{d}w_i = \mathbf{R}_i w_i + \mathbf{D}_i w_i + \mathbf{t}_i.$$  \hspace{1cm} (4.5)

Here, $\mathbf{R}_i$ and $\mathbf{D}_i$ are local rotation and deformation matrices, respectively, and $\mathbf{t}_i$ is a translation vector. The deformation state, in our case, is related to facial expression dynamics. It is thus convenient to distinguish between face biometric deformations (static) and dynamic deformations due to facial gestures or expressions. Formally, deformations

$$\mathbf{dW}_i^S = [\mathbf{d}w_{i,1}^S, \cdots, \mathbf{d}w_{i,N_s}^S]$$

and

$$\mathbf{dW}_i^M = [\mathbf{d}w_{i,1}^M, \cdots, \mathbf{d}w_{i,N_m}^M]$$

are constant $3 \times N_s$ and $3 \times N_m$ matrices, respectively, where each vector $\mathbf{d}w_{i,j}^S$ and $\mathbf{d}w_{i,k}^M$ represents the single Shape Unit (SU) and Action Unit (AU) deformation at vertex $i$, respectively. The columns of $\mathbf{dW}_i^S$ are vectors of control point displacements due to biometric traits of the individual (mouth width, eye distance, etc.). The columns of $\mathbf{dW}_i^M$ encode vectors of point displacements due to gestures. Ideally, when dealing with expressions, each such vector should correspond to AUs related to Ekman’s FACS (Facial Action Coding System [21]); these describe the change in face geometry when the corresponding AU is enabled due to the motor activation of facial muscles. Cogently, the effect of the SUs and AUs is controlled via the shape and motor/action parameter vectors

$$\mathbf{s}_I = [s_1, \cdots, s_{N_s}]^T,$$

$$\mathbf{m}_I = [m_1, \cdots, m_{N_m}]^T.$$

Hence, the deformation term on the r.h.s of Eq.4.5 is written as

$$\mathbf{D}_i w_i = \mathbf{dW}_i^S \mathbf{s}_I + \mathbf{dW}_i^M \mathbf{m}_I.$$  \hspace{1cm} (4.6)

Note that $\mathbf{d}w_i$ has to be small, though, finite movements can always be obtained from infinitesimal ones. Also, Helmholzt’s theorem summarised by Eq. 4.5 holds for point dependent rotation, deformation and translation. However, face motion can be considered as global rigid motion, constrained by cranial pose dynamics, plus a local nonrigid motion originated by facial muscle-based deformations. In other terms, $\mathbf{R}_i = \mathbf{R}$ and $\mathbf{t}_i = \mathbf{t}, \forall i$, can be assumed as global motion operators, $\theta = (\mathbf{R}, \mathbf{t})$ equally applying to all vertices, without loss of generality [42].

For what concerns $\mathbf{R}$, a 3D rotation can be represented by vector $\mathbf{\omega} = [\omega_x, \omega_y, \omega_z]^T$. The exponential twist representation [24, 50] considers the rotation as the infinite limit of $k$ rotations

$$\mathbf{R}(\mathbf{\omega}) = \lim_{k \to \infty} \left( \mathbf{I} + \frac{1}{k} [\mathbf{\omega}]_\times \right)^k = \exp([\mathbf{\omega}]_\times),$$  \hspace{1cm} (4.7)

where $[\mathbf{\omega}]_\times$ denotes the $3 \times 3$ cross product (skew-symmetric) matrix

$$[\mathbf{\omega}]_\times = \begin{bmatrix}
0 & -\omega_z & \omega_y \\
\omega_z & 0 & -\omega_x \\
-\omega_y & \omega_x & 0
\end{bmatrix}.$$
such that $[\omega]_\times v = \omega \times v$, $v \in \mathbb{R}^3$ being a vector to which rotation is applied, and $I$ is the $3 \times 3$ identity matrix.

By expanding the matrix exponential as a Taylor series, $\exp([\omega]_\times) = I + [\omega]_\times + [\omega]_\times^2 + \cdots$ and using the first order approximation, we can write small incremental rotations as $R(\omega) \approx I + [\omega]_\times$. Note that, under such approximation one can conveniently represent $R$ as

$$R \approx I + \omega_x G_x + \omega_y G_y + \omega_z G_z$$  \hspace{1cm} (4.8)

where $G_x, G_y, G_z$ are Lie algebra matrices or $SO_3$ generators.

Thus, at any time $t$, facial movement will locally move a 3D vertex $w_i$ to position $w_i(t + \delta t) = w_i(t) + dW_i$ (for unitary time step $\delta t = 1$, without loss of generality) according to the law:

$$w_i(t + 1) = w_i(t) + R(t)w_i(t) + dW_i^S s_{l} + dW_i^M m(t) + t(t),$$  \hspace{1cm} (4.9)

where $R$ and $t$ represent the rotation matrix and the translation vector, respectively, that is the global rigid motion constrained by cranial pose dynamics.

Equation 4.9 is the internal motor state equation. This allows model-based prediction of the dynamics of the observed face as in the visual space via the projection $T(w)$ of the 3D vertices on the 2D image coordinate system. Given the small depth of the face [51], a weak perspective projection can be adopted thus

$$T(w_l) = s\Pi w_l = s \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} w_l$$  \hspace{1cm} (4.10)

where $s$ is the weak-perspective scale parameter, and $l$ indexes the $L$ vertices that are in correspondence with extracted facial landmarks. Equations 4.9 and 4.10 provide an explicit form for Eqs. 4.2 and 4.3, respectively; in particular, Eq. 4.9 is assumed to define a deterministic dynamics.

It has been shown [8] that the full parameter vector $\Theta = [s, \omega^T, s^T, m^T, t^T]^T$ can be roughly estimated via maximum likelihood. Indeed, under the assumption that $\epsilon_y$ represents Gaussian noise, the observation equation is

$$y_l = T(w_l) + \epsilon_y,$$  \hspace{1cm} (4.11)

and parameter estimation boils down to the negative log-likelihood minimisation problem, $(\hat{m}(t), \hat{\theta}(t)) = \arg \min_{\Theta} \frac{1}{2\sigma_{y}^2} \sum_{l=1}^{L} \|y_l - T(w_l)\|^2 + L \log(2\pi\sigma_{y}^2)$, which can be easily solved in closed matrix form. Also, note that the full parameter vector $\Theta$ needs to be estimated only at the onset of the interaction; in subsequent steps only $(m(t), \theta(t))$ are needed.

Though this can be assumed a reasonable solution, one might consider a more general approach under less restrictive conditions; in particular, it is worth spending such effort for estimating the most critical motor parameters $m$. 

\section{4.1 Internal model}
4.2 Constrained dynamics of motion parameters

To such end it is convenient to rewrite the stochastic state-space dynamics defined by Eqs. 4.1, 4.2 and 4.3 in the equivalent probabilistic form

\[
\theta(t) \sim p(\theta(t) \mid \theta(t-1)), \quad (4.12)
\]
\[
a(t) \sim p(a(t) \mid m(t-1)), \quad (4.13)
\]
\[
m(t) \sim p(m(t) \mid m(t-1), a(t)), \quad (4.14)
\]
\[
y(t) \sim p(y(t) \mid m(t), \theta(t)), \quad (4.15)
\]

Whilst Eq. 4.12 accounts for global motion parameters, Eqs 4.13, 4.14 and 4.15 are at the heart of the visuomotor mapping \( \hat{y}(t) \mapsto m(t) \).

Specifically, Eq. 4.13 states that trajectories \((a(1), a(2), \cdots, a(t))\) occurring in the probabilistic latent manifold of facial actions, are obtained by sampling at time \(t\) a state \(a(t)\) conditioned on the state of motor control parameters \(m(t-1)\) at previous time \(t-1\). Subsequently, \(a(t)\) is exploited to constrain the motor control dynamics \(m(t-1) \mapsto m(t)\) (Eq. 4.14). Eventually, facial landmark prediction is performed (Eq. 4.15).

Denote, for notational simplicity, \(y_{1:t} = (y(1), y(2), \cdots, y(t))\), and analogously for \(a_{1:t}\) and \(m_{1:t}\); write \(z_{1:t} = \{a_{1:t}, m_{1:t}\}\). The visuomotor mapping entails the inverse problem of computing, at time \(t\) the posterior \(p(m_t \mid y_{1:t}, a_{1:t})\), that is inferring, at time \(t\), the motor parameters \(m_t\) given the sequence \(y_{1:t}\) of observed facial features, and optimal action controls \(a_{1:t}\)

This result is in principle achievable by marginalising over the joint posterior \(p(z_{1:t} \mid y_{1:t})\). The latter, in a Bayesian setting, can be written in recursive form as

\[
p(z_{1:t} \mid y_{1:t}) = \frac{p(y_{1:t})}{p(y_{1:t-1})} \frac{p(z_{1:t}, y_{1:t-1})}{p(z_{1:t-1}, y_{1:t-1})} = \frac{p(y_{1:t})}{p(y_{1:t-1})} \frac{p(z_{1:t}) p(z_{1:t-1}, y_{1:t-1})}{p(z_{1:t-1}) p(z_{1:t-1}, y_{1:t-1})} = \frac{p(y_{1:t})}{p(y_{1:t-1})} \frac{p(z_{1:t}) p(z_{1:t-1}) p(a_{1:t})}{p(y_{1:t-1})}, \quad (4.16)
\]

where we have used the d-separation properties in the PGM shown in Fig. 4.1, under Markov assumptions. In general Eq. 4.16 cannot be solved in closed form, thus we resort to particle filter approximation:

\[
p(z_{1:t} \mid y_{1:t}) \approx \sum_{n=1}^{N} \rho_{t}(n) \delta_{\hat{z}_{1:t}^{(n)}}(z_{1:t}). \quad (4.17)
\]

where \(\rho_{t}(n)\) is the normalised weight of sample \(n\) at time \(t\) and \(\delta(\cdot)\) is the Dirac delta function.
In Eq. 4.20 the guidance of action \( \mathbf{a}_t \) so that the weight update in Eq. 4.19 boils down to

\[
\mathbf{a}_t \text{ influences the future state } \mathbf{m}_{t+1} \text{ and the next point in the latent space } \mathbf{a}_{t+1}.
\]

Figure 4.1: The probabilistic graphical model. We can see graphically the conditional independences of the random variables that help us to factorize the joint probability distribution. For sake of simplicity we have not reported \( \mathbf{\theta} \). It is important to note that the state \( \mathbf{m}_t \) influences the future state \( \mathbf{m}_{t+1} \) and the next point in the latent space \( \mathbf{a}_{t+1} \).

To such end we introduce the proposal densities

\[
q(\mathbf{z}_{1:t} \mid \mathbf{y}_{1:t}) = q(\mathbf{z}_t \mid \mathbf{z}_{1:t-1} \mathbf{y}_{1:t}) q(\mathbf{z}_{1:t-1} \mid \mathbf{y}_{1:t-1}),
\]

hence the unnormalised weights can be computed as

\[
\tilde{\nu}^{(n)}_t \propto \frac{p(\mathbf{y}_t \mid \mathbf{m}_t^{(n)}) p(\mathbf{m}_t^{(n)} \mid \mathbf{m}_{t-1}^{(n)}, \mathbf{a}_t^{(n)}) p(\mathbf{a}_t^{(n)} \mid \mathbf{m}_{t-1}^{(n)}) \tilde{\nu}^{(n)}_{t-1}}{q(\mathbf{z}_t^{(n)} \mid \mathbf{z}_{1:t-1}^{(n)}, \mathbf{y}_{1:t})}
\]

where \( \tilde{\nu}^{(n)}_{t-1} = \frac{p(\mathbf{m}_{t-1}^{(n)}, \mathbf{a}_{t-1}^{(n)} \mid \mathbf{y}_{1:t-1})}{q(\mathbf{z}_{t-1}^{(n)} \mid \mathbf{y}_{1:t-1})} \), and \( q(\mathbf{z}_t^{(n)} \mid \mathbf{z}_{1:t-1}^{(n)}, \mathbf{y}_{1:t}) = q(\mathbf{z}_t^{(n)} \mid \mathbf{z}_{t-1}^{(n)}, \mathbf{y}_{1:t}) \) is assumed, so to keep the most recent part of the trajectory and observation sequence, rather than the whole history.

Following the generate and test approach, we can sample from the prior:

\[
q(\mathbf{z}_t^{(n)} \mid \mathbf{z}_{t-1}^{(n)}, \mathbf{y}_{1:t}) = p(\mathbf{m}_t^{(n)} \mid \mathbf{m}_{t-1}^{(n)}, \mathbf{a}_t^{(n)}) p(\mathbf{a}_t^{(n)} \mid \mathbf{m}_{t-1}^{(n)}),
\]

so that the weight update in Eq. 4.19 boils down to

\[
\tilde{\nu}^{(n)}_t \propto p(\mathbf{y}_t \mid \mathbf{m}_t^{(n)}) \tilde{\nu}^{(n)}_{t-1},
\]

followed by weight normalisation \( \nu^{(n)}_t = \frac{\tilde{\nu}^{(n)}_t}{\sum_{j=1}^N \tilde{\nu}^{(n)}_j} \).

In Eq. 4.20 \( p(\mathbf{m}_t^{(n)} \mid \mathbf{m}_{t-1}^{(n)}, \mathbf{a}_t^{(n)}) \) models the dynamics of motor control parameters under the guidance of action \( \mathbf{a}_t \) spanning the latent low dimensional manifold; \( p(\mathbf{a}_t^{(n)} \mid \mathbf{m}_{t-1}^{(n)}) \) constrains the choice of current action \( \mathbf{a}_t \) based on the state of motor control parameters at previous time \( t - 1 \). The overall results is that action trajectories in the latent space, both at learning and inference stage, are constrained by the dynamics of the process.

In this work the action latent space is shaped in the form of a Back-Constrained Gaussian Process Latent Variable Model (BC-GPLVM [40]). The BC-GPLVM builds on the
4 The Model

GPLVM to provide a non linear mapping between the motor control parameters and the low dimensional manifold as \( m = f(a) + \epsilon \). The \( i \)th function \( f_i \) is a Gaussian process \( f_i \sim \mathcal{GP}(f \mid 0, k(a, a')) \) where \( k \) is a kernel function, e.g., the RBF kernel \( k(a, a') = m \exp \left( -\frac{\gamma}{2} ||a - a'||^2 \right) + m_0 \), \( m \), \( \gamma \) and \( m_0 \) being the kernel parameters. The noise term \( \epsilon \) denotes Gaussian noise \( \epsilon \sim \mathcal{N}(\epsilon \mid 0, \sigma_n^2 I) \) with variance \( \sigma_n^2 \) and \( I \) is the identity matrix \( D \times D \). Learning correspond to finding the parameters and points in latent space that maximises the logarithm of the likelihood function \( \ln p(m \mid a) = -\frac{1}{2} \text{tr}(m^T \mathbf{K}^{-1} m) \), where \( \mathbf{K} = \mathbf{K} + \sigma_n^2 I \) and \( \mathbf{K} = [k_{nm}] \) is the kernel matrix with \( k_{nm} = k(a_n, a_m) \). The objective function is optimised with gradient based techniques, e.g. the scaled conjugate gradient method.

The rationale behind the BC-GPLVM is to define the latent point \( a \) as a smooth mapping of \( m, a = g(m) \), for example the linear mapping,

\[
g_i(m) = \sum_{t=1}^{T} a_{ti} k_b(m, m_t^{'}) + b_i
\]

where \( g_i \) denotes the \( i \)th component of \( a \) and \( a_{ti}, b_i \) are parameters. The kernel \( k_m \) denotes the RBF kernel defined in the space motor parameters

\[
k_b(m, m') = m \exp \left( -\frac{\gamma}{2} ||m - m'||^2 \right) + m_0.
\]

In BC-GPLVM we incorporate this mapping into the objective function and then optimize with respect to the parameters \( a_{ti}, b_i \) instead of the latent space points. Eventually,

\[
p(a^{(n)}_i \mid m^{(n)}_{t-1}) = \mathcal{N}(a^{(n)}_i \mid g(m^{(n)}_{t-1}), \Sigma),
\]

where \( \Sigma \) is a diagonal covariance matrix.

The BC-GPLVM allows to easily calculate the predictive distribution for the new \( m_{\text{new}} \) and \( a_{\text{new}} \) \cite{Williams98} as \( p(m_{\text{new}} \mid a_{\text{new}}, m, a) = \mathcal{N}(m_{\text{new}} \mid \mu_{\text{new}}, \sigma_{\text{new}}^2 I) \), with predictive mean and variance \( \mu_{\text{new}} = m^T \mathbf{K}^{-1} \mathbf{k}, \sigma_{\text{new}}^2 = \mathbf{k}^T (\mathbf{K} + \sigma_n^2 I)^{-1} \mathbf{k} \). Here \( \mathbf{k} \) denotes a column vector containing the values of kernel function \( k(x, a_{\text{new}}, a_{\text{new}}) \) for \( n = 1, \ldots, N \).

This result is useful for dealing with the term \( p(m^{(n)}_t \mid m^{(n)}_{t-1}, a^{(n)}_t) \) in the prior given by Eq.4.20. We approximate such term as

\[
p(m^{(n)}_t \mid m^{(n)}_{t-1}, a^{(n)}_t) \propto p(m^{(n)}_t \mid m^{(n)}_{t-1}) p(m^{(n)}_t \mid a^{(n)}_t).
\]

The first component is assumed to account for a Gaussian random walk of motor parameters,

\[
p(m^{(n)}_t \mid m^{(n)}_{t-1}) = \mathcal{N}(m^{(n)}_t \mid m^{(n)}_{t-1}, \Phi),
\]

with diagonal covariance matrix \( \Phi \). The second term is the predictive distribution

\[
p(m^{(n)}_t \mid a^{(n)}_t) = \mathcal{N}(m^{(n)}_t \mid m^T \mathbf{K}^{-1} \mathbf{k}, \Lambda).
\]

with diagonal covariance matrix \( \Lambda \).
4.3 A model instantiation

Next we define the likelihood function $p(y_t | m^{(n)}_t)$. To such end we take into account the error (distance) $d_n = y_t - T(w_t(m^{(n)}_t))$, for each particle, and define the likelihood to be an half normal standard distribution

$$p(y_t | m^{(n)}_t) \sqrt{\frac{2}{\pi}} \exp \left(- \frac{[d_n]^2}{2}\right).$$

The rationale behind this choice is to give high likelihood to distances close to zero and low likelihood to higher distances. Low error means that sampled points are near to the observed landmarks, thus they are retained as high quality particles.

Eventually, it is worth remarking that to prevent the typical degeneration phenomenon we add a resampling step, as widely described in [19].

4.3 A model instantiation

Here we describe a real instantiation of the tracking procedure we have proposed in the previous paragraphs in order to see which results we have obtained in the benchmark chapter.

For the motor space representation $w$ and its deformations we exploit the 3D face model CANDIDE-3 [2], which is a 3D wireframe model of approximately 113 vertices $w_i$ and 184 triangles, that easily fits our needs. Indeed, CANDIDE-3 directly accounts for encoding the matrices of Shape Unit (SU) and Action Unit Vector (AUV) deformations parameters at vertices ($dW^S_i$ and $dW^M_i$) together with related control parameters $s_i$ and $m_i$, respectively. AUVs determines a change in face geometry and implement a subset of the Ekman’s Action Units of FACS [21].

For the training step of BC-GPLVM model we simply selected a subset of samples from the well known Cohn-Kanade database, Kanade et al. [35]. We have seen in the first Chapter that the database includes 486 sequences from 97 posers. Each sequence begins with a neutral expression and proceeds to a peak expression to which is given an emotion label. The emotion label refers to what expression was requested rather than what may actually have been performed.

Hence, in this section we propose a deterministic tracking procedure implemented through the Gauss-Newton non-linear least squares technique [6] and we compare this to our conditioned particle filtering technique. We also propose a particle filtering without conditioning (represented by Fig. 4.1 where the higher level is removed) to completely assess our model. Before viewing these three approaches we will also explain necessary preprocessing we have done on CANDIDE-3 model and which facial landmark prediction method we have used.

4.3.1 CANDIDE-3 model preprocessing

Here we explain how he have preprocessed the wireframe file, defining the CANDIDE-3 model (v3.1.6), available at: http://www.icg.isy.liu.se/candide/candide3.wfm.
In the previous paragraphs we have explained that we measure the likelihood of our predicted CANDIDE-3 shape looking at the distance from landmarks. Usually, the number of CANDIDE-3 vertices differ from the number of landmark extracted from a landmark predictor, so we have to define a mapping between CANDIDE-3 vertices and the extracted landmark to compare them. For such reason we selected a subset $c$ containing the vertices of CANDIDE-3 we are considering:

$$c = \{29, 28, 30, 32, 10, 65, 63, 61, 62, \text{ (chin and jaw)} \}
\begin{align*}
  &20, 23, 101, 109, 95, 103, \text{ (left eye)} \\
  &53, 56, 102, 110, 96, 104, \text{ (right eye)} \\
  &31, 64, 89, 88, 85, 8, 86, 79, 7, 80, 87, 40, \text{ (mouth)} \\
  &59, 26, 111, 112, 5, 6, 94, \text{ (nose)} \\
  &48, 49, 50, \text{ (right eyebrow)} \\
  &15, 16, 17 \} \text{ (left eyebrow)}
\end{align*}$$

We can see which vertices we are considering and their order in the Fig. 4.2 and 4.3.

Now we have to think to the landmarks: they are extracted from a given a video source. Every frame is analyzed and if there is a face in the current frame we extract a set of facial landmarks. There are a variety of facial landmark detectors, and therefore various implementations of them. We used the landmark detector implemented in Dlib\(^1\), that is an implementation of the method of Kazemi and Sullivan [36]. In short, their method is based on:

- A training set composed by faces on which landmarks are labelled. Images are labeled by hand, specifying the coordinates.
- Priors. Specifically, a probability given on distance between two input pixels.

Starting from these, they train an ensemble of regression trees to estimate the landmarks positions from pixel intensities. No feature extraction is performed. The final result is a landmark detector capable of detecting facial landmarks in real-time with high reliability predictions.

We use the pre-trained detector that comes with Dlib. With that we are able to predict the position of 68 points that map to the structure of the face. The original indexes of the 68 points are reported in FIG. 4.4 and in FIG. 4.5 there are two prediction examples. We specify that in our work we prefer to count indexes from 0. We also specify that the Dlib pre-trained detector is trained on the iBUG 300-W dataset [58] where all images contained are manually labelled with 68 points. It is possible also to use other datasets i.e. HELEN dataset [41] that is based on a 194 point model.

We finally define the set $y$ containing all the ordered Dlib landmarks indexes.

\(^1\)Dlib is available at https://github.com/davisking/Dlib.
4.3 A model instantiation

\[ y = \{16, 14, 11, 9, 8, 7, 5, 2, 0, \text{ (chin and jaw)} \]
\[ 45, 42, 46, 47, 44, 43, \text{ (left eye)} \]
\[ 36, 39, 41, 40, 37, 38, \text{ (right eye)} \]
\[ 54, 48, 60, 64, 55, 57, 59, 53, 51, 49, 62, 66, \text{ (mouth)} \]
\[ 31, 35, 34, 32, 30, 33, 28, \text{ (nose)} \]
\[ 17, 19, 21, \text{ (right eyebrow)} \]
\[ 26, 24, 22 \} \text{ (left eyebrow)} \]

The order of Dlib landmarks is given according to the iBUG 300-W face landmarks dataset. After defining this matching, we have parsed the wireframe file taking a subset of the Action Units Vectors and the Shape Units. In particular, we consider only the Shape Units for which we have enough landmarks for estimation. Here the list:

- AUV0 Upper lip raiser (AU10)
- AUV11 Jaw drop (AU26/27)
- AUV2 Lip stretcher (AU20)
- AUV3 Brow lowerer (AU4)
- AUV14 Lip corner depressor (AU13/15)
- AUV5 Outer brow raiser (AU2)
- AUV6 Eyes closed (AU42/43/44/45)
- AUV7 Lid tightener (AU7)
- AUV8 Nose wrinkler (AU9)
- AUV9 Lip presser (AU23/24)
- AUV10 Upper lid raiser (AU5)
- Eyebrows vertical position
- Eyes vertical position
- Eyes, width
- Eyes, height
- Eye separation distance
4 The Model

Figure 4.2: The subset of CANDIDE-3 model vertices.

Figure 4.3: The order of the subset of CANDIDE-3 model vertices.
4.3 A model instantiation

Figure 4.4: The 68 facial landmark indexes of Dlib.

Figure 4.5: Two examples of landmarks estimation.
4 The Model

- Cheeks z
- Nose vertical position
- Mouth vertical position
- Mouth width
- Eyes vertical difference
- Chin width

Finally, to finish the preprocessing, we have rotated CANDIDE-3 reference frame by $180^\circ$ along x axis, as we suppose to use a pinhole camera model.

4.3.2 Gauss-Newton baseline tracker

Now we are ready to show how to fit CANDIDE-3 model through the Gauss-Newton method. We use this instead of the standard Newton’s method because it has the advantage that second derivatives, which are challenging to compute, are not required [6].

Before using Gauss-Newton method we have to define a projection model. Let us to define scale: As we did before, we name the vector containing the subset of CANDIDE-3 vertices $c$ and in the same way the vector $y$. We also name respectively $\bar{c}$ and $\bar{y}$ their mean along rows. We compute the initial estimation parameters starting to subtract the mean from $c$ and $y$:

$$
c_{\text{centered}} = c - \bar{c},
$$

$$
y_{\text{centered}} = y - \bar{y}.
$$

If we set $c'_{\text{centered}} = c$ where we remove last column, scale $a$ is computed as $a = \frac{\|y_{\text{centered}}\|}{\|c'_{\text{centered}}\|}$ and we finally obtain traslation as $t = \bar{y} - \bar{c}'$. We can now define the projection of CANDIDE-3 model on the camera space through the equation

$$
c_p = aP\left(c_0 + \sum_{i=1}^{n} w_i \cdot S_i\right) + t, \quad (4.25)
$$

where

- $c_p$: projected CANDIDE-3 model
- $a$: scale
- $P$: first two rows of the rotation matrix
- $c_0$: standard CANDIDE-3 model
4.3 A model instantiation

- \( w_i \): weights of AUVs and SUs
- \( S_i \): AUVs and SUs
- \( n \): number of AUVs and SUs
- \( t \): translation vector

Now we have to define Gauss-Newton method. We first need a residual function that computes residual: it is a function in the form

\[
r(m) = y - f(x, m),
\]

where \( r \) is the residual function and \( x = c, f \) is the projection function and \( m \) are the estimated model parameters. Now we have to set a numeric implementation of the Jacobian matrix \( J_f(m) \) of the vector residual function \( r(m) \). We can define it as

\[
[J_f(m)]_{ij} = \frac{\partial r_i(m)}{\partial m_j} = -\frac{\partial f(x_i, m)}{\partial m_j},
\]

where \( J_f(x) \) has a number of rows equals to the number of CANDIDE-3 vertices and a number of columns equals to the size of model parameters \( m \).

We can now perform Gauss-Newton method. Our aim is to iteratively find the values of the model parameters \( m \) which minimize the square of the residual function defined above. The method starts with an initial guess \( m_0 \) for the minimum. We can set \( m_0 \) as \( a = \|y_{centered}\| \) and translation as \( t = y - c' \) considering the first frame, set rotation vector \( r \), set to \([0, 0, 0] \) and the weights of Action Units and Shape Units, all set to 0. Now we can proceed by by the iteration

\[
m_{s+1} = m_s + \left( J_f(m_s)\mathbf{T} J_f(m_s) \right)^{-1} J_f(m_s)\mathbf{T} r(m_s).
\]

Here we have express Gauss-Newton method in terms of the Jacobian \( J_f(m_s) \) of the projection function \( f \). Iterations stops when the difference of cost between consecutive iterations is less than a defined \( \epsilon \) value or after a max defined number of iteration, for every video frame coming from the considered video source.

4.3.3 Auto regressive particle filter tracker

In this section we show how to fit CANDIDE-3 model through a particle filtering technique, which state equation is an auto regressive model.

Before beginning the tracking procedure we have to build a prior distribution \( p(m_0) \). A way to build such prior distribution is to take the initial frame of the video we are considering and to run the Gauss-Newton tracker, explained in the previous section, on that frame. In this way we obtain the optimized parameters \( m_0^* \) and if we add Gaussian noise we can
4 The Model

We build a prior distribution as

\[ p(m_0) = m_0^* + \mathcal{N}(0, \Sigma_{\text{prior}}). \]

We simply estimate \( \Sigma_{\text{prior}} \) by calculating the covariance between the parameters \( m_{0:T} \) extracted by the first \( T \) frames of the given video.

After we extract \( m_0^* \) from the first frame we extract \( N \) number of particles \( \{m_i(0)\}_{i=0}^N \) from the Gaussian prior distribution \( p(m_0) = \mathcal{N}(m_0, m_0^*, \Sigma_{\text{prior}}) \) and we initially assign to every particle the same weight \( 1/N \). So we have an initial particle set

\[ M_0 = \{m_0^{(i)}, w_0^{(i)}\}_{i=0}^N = \{m_0^{(i)}, 1/N\}_{i=0}^N. \]

Before the beginning of the tracking procedure we recall how we build our likelihood function that gives the likelihood of a sampled particle. Ad before, we name \( y \) the subset of comparable landmarks and in the same way for CANDIDE-3 model \( c \). We also define the projection function \( g(m) \) which takes only the comparable CANDIDE-3 points \( c \).

Considering these, we can define the likelihood of the particle \( m_t^{(i)} \) by calculating \( d_i = \|g(m_t^{(i)}) - y_t\| \). We scale the distance \( d_i \) in the interval \([0, 2]\) as

\[ d_i = a(b - a)\frac{d_i - \text{min}(D)}{\text{max}(D) - \text{min}(D)} \quad \text{with } a = 0 \text{ and } b = 2. \]

Finally, we calculate the weight \( w_t^{(i)} \) as the probability of draw the distance \( d_i \) from a half normal standard distribution \( Y \), with \( Y = |X| \) and \( X \sim \mathcal{N}(0, 1) \):

\[ w_t^{(i)} = f_Y(d_i; 1) = \sqrt{\frac{2}{\pi}} \exp \left( -\frac{[d_i]^2}{2} \right). \]

Other long tailed distribution can be used for this purpose. We specify that we have scaled the distances as the most of the probability density of the half Normal standard distribution is in the interval \([0, 2]\). In this way, we can assure a good weight variance.

Now we perform the filtering procedure with a particle filter. First, we have to perform a resampling on the initial particle set \( M_0 \). We perform a stratified resample: this aims to make selections relatively uniformly across the particles. It divides the cumulative sum of the weights into \( N \) equal divisions, and then selects one particle randomly from each division. This guarantees that each sample is between 0 and \( 2/N \) apart. We also implemented other resampling strategies, for more details see [3]. We set as state equation the following auto regressive model

\[ m_t^{(i)} = \sum_{i=1}^W \alpha_i m_{t-i} + \nu_t \quad \text{with} \quad \nu_t \sim \mathcal{N}(0, \Sigma_{\text{process}}). \]

We set a window size of \( W \) and we learn the weights \( \alpha_i \) with \( i = 1, \ldots, W \) with a maximum likelihood approach, see [32]. For the first \( W-1 \) frames we perform a simply random walk

\[ m_t^{(i)} = m_{t-1}^{(i)} + \nu_t \quad \text{with} \quad \nu_t \sim \mathcal{N}(0, \Sigma_{\text{process}}). \]

After the last step, we compute the weights for all the drawn particles. Next, we normalize
them as \( w_t^{(i)} = \frac{w_t^{(i)}}{\sum_{n=1}^{N} w_{t}^{(n)}} \). Finally, we compute the next state expected value as

\[
\mathbb{E}[m_t] = \sum_{n=1}^{N} w_t^{(n)} m_t^{(n)}
\]

and we save the next particle set \( \mathcal{M}_t = \{m_t^{(i)}, w_t^{(i)}\} \). We can resume this procedure in the below Algorithm 1.

### Algorithm 1: PARTICLE-FILTER(\( \{y_t\}_{t \in 1:N}, p(m_0), J, \Sigma, W \))

**Input:** Landmarks sequence \( \{y_t\}_{t \in 1:N} \), prior \( p(m_0) \), number of particle \( J \), process noise covariance \( \Sigma \) and window size \( W \).

**Output:** State vectors \( m_t \) for \( t = 1, 2, \ldots, N \).

**Initialize** the particle set \( \mathcal{M}_0 = \{m_0^{(j)}, 1/J\}_{j=1}^{J} \) where \( m_0^{(j)} \sim p(m_0) \).

**Add** \( m_0 \) to the list \( L \).

for \( t = 1, 2, \ldots, N \) do

- **Resample** \( \mathcal{M}_{t-1} \) and obtain the new sample set \( \mathcal{M}_{t-1} = \{m_{t-1}^{(j)}, w_{t-1}^{(j)}\}_{j=1}^{J} \).

  for \( j = 1, 2, \ldots, J \) do

  - **Save** in \( L \) the size of \( L \).

     if \( l < W \) then

     - **Predict** the sample \( m_t^{(j)} \) from \( m_{t-1}^{(j)} \) as

       \[
       m_t^{(j)} = m_{t-1}^{(j)} + U_t \quad \text{with} \quad U_t \sim \mathcal{N}(0, \Sigma).
       \]

     else

     - **Learn** through a maximum likelihood approach the weights \( \alpha_i \) with \( 1 \leq i \leq W \), from vectors \( m_{t-W}^{(j)}, \ldots, m_{t-1}^{(j)} \), saved in the list \( L \).

     - **Predict** the sample \( m_t^{(j)} \) as

       \[
       m_t^{(j)} = \sum_{i=1}^{W} \alpha_i m_{t-1}^{(i)} + U_t \quad \text{with} \quad U_t \sim \mathcal{N}(0, \Sigma).
       \]

  - **Compute** the difference \( d_j = \|g(m_t^{(j)}) - y_t\| \).

  - **Scale** \( d_j \) in the interval \([0, 2]\):

    \[
    d_t = a(b - a) \frac{d_t - \min(D)}{\max(D) - \min(D)} \quad \text{with} \quad a = 0 \text{ e } b = 2.
    \]

  - **Compute** the new weight \( w_t^{(j)} \) as the probability of draw the distance \( d_j \) from a half Normal standard \( Y \), with \( Y = |X| \) and \( X \sim \mathcal{N}(0, 1) \):

    \[
    w_t^{(j)} = f_Y(d_t; 1) = \frac{\sqrt{2}}{\sqrt{\pi}} \exp\left(-\frac{d_t^2}{2}\right).
    \]

  - **Normalize** the weights:

    for \( j = 1, 2, \ldots, J \) do

    \[
    w_t^{(j)} = \frac{w_t^{(j)}}{\sum_{i=1}^{J} w_t^{(i)}}
    \]

  - **Obtain** the state estimate \( m_t = \sum_{i=1}^{J} w_t^{(i)} m_t^{(i)} \).

  - **Save** the new particle set \( \mathcal{M}_t = \{m_t^{(j)}, w_t^{(j)}\} \).

  - **Add** \( m_t \) to \( L \).

We finally specify that we set \( \Sigma_{\text{process}} \) as the covariance matrix of the reconstruction error estimated by the autoregressive model. Another choice is to manually set a suitable
covariance matrix. A way to estimate such matrix can be the calculation of the covariance between the differences $m_t - m_{t-1}$ for the parameters $m_0:T$ extracted by the first $T$ number of considered frames of the given video.

### 4.3.4 Auto regressive conditioned particle filter tracker

Now we introduce the conditioning with our learned latent space. As we said, before running the tracking procedure we train a BC-GPLVM model using a selected subset of samples from the Cohn-Kanade database. We specify that there are several ways to practically integrate the information from a learned model in a context like the one we described. Here we propose a way that naturally derives from the PGM we proposed in Fig 4.1.

To integrate the learned model we modify the weight update step: now we simply compute the new weight $w_t^{(j)}$ as the probability of draw the distance $d_j$ by landmarks from a half Normal standard $Y$, with $Y = |X|$ and $X \sim N(0, 1)$ as

$$w_t^{(j)} = f_Y(d_j; 1) = \frac{\sqrt{2}}{\sqrt{\pi}} \exp \left( -\frac{[d_j]^2}{2} \right)$$

and we simply normalize weights as

$$w_t^{(j)} = \frac{w_t^{(j)}}{\sum_{i=1}^{J} w_t^{(i)}}.$$

We modify the model as we know that we can map a high dimensionality motor parameter in latent learned space through

$$p(a_t^{(j)} | m_t^{(j)}) = N(a_t^{(j)} | g(m_t^{(j)} - 1), \Sigma).$$

Hence, we map all of our particles in the latent space and we observe for each of them the probability of having the associated high dimensionality point. We know that probability as we can compute it with the predictive distribution

$$l_t^{(j)} = p(m_t^{(j)} | a_t^{(j)}) = N(m_t^{(j)} | \mathbf{m}_t^{T} \mathbf{K}^{-1} \mathbf{a}_t^{(j)}, \Lambda).$$

Now we integrate the learned informations by computing and normalizing the new weights:

$$w_t^{(j)} = \frac{w_t^{(j)} \cdot l_t^{(j)}}{\sum_{i=1}^{J} w_t^{(i)} \cdot l_t^{(i)}}.$$

In this ways we perform a correction on the tracking procedure as particles are forced to stay in the learned manifold my BC-GPLVM. When high dimensionality motor parameters go out of the manifold their weights are corrected as the associated $l_t^{(j)}$ is low.

In the next, and almost final chapter, we will see that this conditioning is valuable with some useful benchmarks.
5 Benchmarks

In this chapter we propose a validation of our technique, by proposing some benchmarks. We underline that in the previous works that faced this problem, like [16], the authors haven’t proposed any benchmarks. Here we propose some suitable tests as we think that it is important to show that the model properly works. For testing, we rely on the Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS) [45].

Construction and validation of the RAVDESS is described in [45]. It contains 7356 files. Each file was rated 10 times on emotional validity, intensity, and genuineness. Ratings were provided by 247 individuals who were characteristic of untrained adult research participants from North America. A further set of 72 participants provided test-retest data. High levels of emotional validity, interrater reliability, and test-retest intrarater reliability were reported. We used this dataset as it is clear and simple to obtain. A frame from a video contained in the dataset is visible in Fig. 5.1.

First, for benchmarking our procedure we need a ground truth: as it was not available, we decided to build our ground truth extracting landmarks from any video contained in the dataset with the Dlib landmark predictor. After, we computed some metrics comparing the landmarks predicted from Gauss-Newton method and Particle Filtering methods with that predicted landmarks. In particular, we computer the Root Mean Square Error (RMSE) and the ratio of processed frames by all the methods.

Figure 5.1: A frame from a video contained in the RAVDESS dataset. As you can see it is very clear and suitable to our tests.
Figure 5.2: The plot shows the ratio of processed frames for every procedure and for every scenario. We can see that Particle filters are able to propose a prediction for every frame.

Furthermore, in order to test the predictive capability of our procedure we also decided to introduce some occlusions and distortions in the video. We decided to include these:

- Black ball: during the video we show a black ball that cross the video through a diagonal movement. The ball appears for the whole video and its diameter is a quarter of the greater dimension of the video.

- missing frames: we simulate to miss some frames. In our tests we suppose that we lose 5 frames every 10 frames.

- downsampling: we proportionally downsample the video from the beginning to the end.

- noise: we add fixed Gaussian noise to the video.

Now we are ready to see the results we have obtain. We start from the plot shown in Fig. 5.2. In this plot we show the ratio of processed frames through the different procedures. Here we can see the limit of a deterministic technique like Gauss-Newton method: it is not able to propose a prediction for every frame, as it completely relies on the landmarks extracted from Dlib. Since they are not available for every frame, when they are not available this baseline method is not able to propose a prediction.

Now that we have shown the advantage of Particle Filters techniques, we show that our conditioned particle filter seems to be better than the standard one. We show this calculating the RMSE through all the scenarios we have proposed. You can see the associated plot in Fig. 5.3. The standard particle filter seems to be better in the black ball and missing frames scenario. On the other side the conditioned one is better when we downsample...
and we add noise and, in particular, is better in the normal condition without adding any artificial distortion.

Summing up the situation, you can look at Fig. 5.4. We conclude that Gauss-Newton obtains the best RMSE, but it is able to process many less frames than particle filtering techniques. The latter can process every frame and produce a prediction and we see that our conditioned particle filter is better than the standard one three out of the five. A good tracking method that we propose and that follows this thesis can be the integration of Gauss-Newton technique with particle filtering techniques, in particular with the conditioned one. We can use Gauss-Newton when measurements are available and conditioned particle filter when they are not available, obtaining an high reliable tracker. Furthermore, we can also redesign our predictive scheme, and in the last section we propose our final idea.

Figure 5.3: The plot shows the RMSE for every procedure and for every scenario. We can see that conditioned particle filter is better than the standard one three out of the five.

Figure 5.4: The plot shows the RMSE for every procedure and for every scenario, where transparency is proportional to the processed frames ratio.
6 Future work

We have seen that the core of the project was the design of a unified central dynamic affect space representation, in the shape of a dynamic probabilistic latent space. In the previous chapter, we have seen that GP-LVM has proven to be a suitable option, as also witnessed in [65]. However, even in their deep, multilevel version, GP-LVM have the intrinsic limitation of lacking explicit dynamics. We tried to introduce a dynamics in the motor space parameters through an autoregressive model and we used GP-LVM as a regularizer, but, at the most general level, we argue that it will be convenient to rethink the whole model in terms of a hierarchy of stochastic processes, each process being defined over a latent variable state-space.

Let us now give a more precise shape to the stochastic view of the theoretical model and to related implications. On a measurable state-space \((\Omega, \mathcal{A}, \mathbb{P})\) the following are given:

- a family of probability measures \(\mathcal{M} = \{\mathbb{P}_\theta, \theta \in \Theta\}\) depending on a parameter \(\theta\), with density \(\mathbb{P}_\theta\);
- a pair of stochastic processes \(X = \{X_t, 0 \leq t \leq T\}\) and \(Y = \{Y_t, 0 \leq t \leq T\}\) taking values in \(\mathbb{R}^x\) and \(\mathbb{R}^y\), respectively.

Suppose \(X\), under \(\mathbb{P}_\theta\), is a Markov process with an infinitesimal generator, then we can write the state-space equations of a dynamical stochastic system in the following form of Ito SDE (to be interpreted as an Ito stochastic integral):

\[
\begin{align*}
\frac{dX_t}{dt} &= f(X_t, C_t) dt + D^{1/2} dW_t, \\
\frac{dY_t}{dt} &= g(X_t, C_t) dt + R^{1/2} dV_t,
\end{align*}
\]

where \(W = \{W_t, 0 \leq t \leq T\}\) and \(V = \{V_t, 0 \leq t \leq T\}\) are respectively independent standard processes (e.g. Wiener), of the same dimension of \(X\) and \(Y\) respectively. \(D, R\) are diffusion coefficients. The latter could be in general a function of the states, i.e. \(D = D(X_t), R = R(X_t)\). The variable \(C\) also is defined for wider generality as the stochastic processes \(C = \{X_t, 0 \leq t \leq T\}\), though in specific cases can be deterministic, stochastic, or both. It represents the system control, which is also variously referred to in the literature as input, cause or source. This can be shaped in many ways, for example as a function of both \(X\) and \(Y\) (e.g. to introduce feedback) or an exogenous input (e.g., the labelling sequence provided along a supervised learning stage). Also, we denote \(f\) and \(g\) the generic (vector or scalar valued) nonlinear, potentially time-varying functions, i.e. mappings of the kind \(T \times L^2(\Omega, \mathcal{A}, \mathbb{P}) \mapsto L^2(\Omega, \mathcal{A}, \mathbb{P})\) to a (Lebegue square-integrable) Hilbert space \(L^2(\Omega, \mathcal{A}, \mathbb{P})\) with finite second-order moments. In fact, Eqs. 6.1 and 6.2 can be easily recognised as diffusion processes, \(f\) and \(g\) being their respective drifts [64]. We can think of this processes as the limit of the discrete-time processes.
Equations 6.3 and 6.4 are known as the Euler-Maruyama approximation of Eqs. 6.1 and 6.2.

Assume that \( \Omega \) is the canonical state-space \( \Gamma ([0, T]; \mathbb{R}^x + \mathbb{R}^y) \), in which case \( X \) and \( Y \) are the canonical processes on \( \Gamma ([0, T]; \mathbb{R}^x) \) and \( \Gamma ([0, T]; \mathbb{R}^y) \), respectively, and \( P_\theta \) is the probability law of \( (X, Y) \). In such case \( X \) is the state process, which is not directly observed; rather, the information about its evolution is obtained through the noisy observed process \( Y \). Then, Eqs. 6.1 and 6.2 define a generalised input-output state-space system (SSM) where the states \( X \) mediate the influence of the input on the output and endow the system with memory. The state and observation perturbations or fluctuations are provided by noise terms \( \epsilon_X, \epsilon_Y \), which can be defined via the stochastic integrals \( W_t = \int_0^t \epsilon_X s \, ds \), \( V_t = \int_0^t \epsilon_Y s \, ds \). In the case of \( W, V \) being Wiener processes, \( \epsilon_X, \epsilon_Y \) represent Gaussian additive noise, and have the same dimension of \( X, Y \), respectively. If errors are iid Gaussian random variables, then the specific scaling of the white noise with \( \Delta t \) gives rise to the nondifferentiable trajectories of sample paths characteristic for a diffusion process. The classic input-output SSM can be recovered from Eqs. 6.1 and 6.2, under the independence assumption (\( Y_t \perp Y_{t-1} \mid X_t \)):

\[
\begin{align*}
    dX_t &= f(X_t, C_t) dt + D^{1/2} dW_t, \\
    Y_t &= g(X_t, C_t) + R^{1/2} \epsilon_Y, \quad (6.5)
\end{align*}
\]

Obviously, considering Eqs. 6.3 and 6.4, under the conditional independence assumption (\( Y_{t+\Delta t} \perp Y_t \mid X_{t+\Delta t} \)), then \( Y_t \) only depends on \( X_t, C_t \) and we recover the discrete time input-output SSM. Equations 6.1 and 6.2 above formalise the generative process: when the dynamics unfolds, the process \( Y \) generates a \( \sigma \)-algebra.

The framework defined via Eqs. 6.1 and 6.2 ensures that the probability measures in \( \mathcal{M} \) are mutually absolutely continuous. Then, the connection of the stochastic process to the Bayesian setting can be made via the Radon-Nikodym Theorem [60]. If we let \( \theta_0 \) be the reference set of parameter and write \( P_{\theta_0} \) as \( P_0 \) (prior measure, with associated prior density \( P_{\theta_0} \)), the Radon-Nikodym derivative of \( P_\theta \) with respect to \( P_0 \) provides the complete data likelihood

\[
\frac{dP_\theta}{dP_0} = \frac{P_\theta (Y_t \mid C_t)}{P_0 (Y_0:T \mid C_0:T)}. \quad (6.7)
\]

Let \( P_0^Y \) denote the restriction of \( P_0 \) to the \( \sigma \) algebra generated by process \( Y \), then the likelihood function for estimating the parameters \( \theta \) on the basis of a given observation path \( Y = \{Y_t, 0 \leq t \leq T\} \) can be expressed as [67]

\[
\mathcal{L}(\theta \mid Y_{0:T}) = E_0 \left[ \frac{dP_\theta^Y}{dP_0} \mid Y_{0:T}, C_0:T \right], \quad (6.8)
\]

where \( E_0 \) denotes the expectation under \( P_0 \). This provides the necessary link to the
Bayesian view of the dynamic stochastic process, which is in turn instantiated in terms of a Probabilistic Graphical Model. Thus a research line we will pursue is under which conditions it is possible to generalise the input-output SSM (Eqs. 6.5, 6.6) into a hierarchical form spanning on $l = 1, \cdots, L$ levels:

\[
\begin{align*}
\dot{C}_t^{(L)} &= g(C_t^{(L+1)}) + \epsilon_C^{(L+1)}, \\
\vdots & \quad \vdots \\
\dot{X}_t^{(l)} &= f(X_t^{(l)}, C_t^{(l)}) + \epsilon_X^{(l)}, \\
\dot{C}_t^{(l-1)} &= g(X_t^{(l)}, C_t^{(l)}) + \epsilon_C^{(l)}, \\
\vdots & \quad \vdots \\
\dot{X}_t^{(1)} &= f(X_t^{(1)}, C_t^{(1)}) + \epsilon_X^{(1)}, \\
\dot{Y}_t &= g(X_t, C_t) + \epsilon_Y,
\end{align*}
\] (6.9)

Note that in the hierarchical form the controls $C_t^{(L)}, C_t^{(L-1)}, \cdots, C_t^{(1)}$ are used to link levels: $\dot{C}_t^{(l)}$ constrains as a top-down signal either $\dot{C}_t^{(l-1)} \dot{X}_t^{(l-1)}$; at the same time $\dot{C}_t^{(l-1)}$ accounts for the emission of $\dot{X}_t^{(l-1)}$. The hidden states $X_t^{(L)}, X_t^{(L-1)}, \cdots, X_t^{(1)}$ provide the necessary dynamics over time. This is in particular true when we assume time conditional independencies between controls, i.e., $(C_t^{(l)} \perp C_t^{(l-1)})$, and between observations, $(Y_t \perp Y_{t-1} | X_t)$. The conditional independence of the fluctuations at different hierarchical levels means that the hierarchy has a Markov property over levels. Thus in probabilistic terms we can write

\[
P_\theta(X_t, C_t) = P_\theta(C_t) \prod_{l=1}^{L-1} P_\theta(X_t^{(l)}, | C_t^{(l)}) P_\theta(C_t^{(l)}) | \dot{X}_t^{(l+1)}, \dot{C}_t^{(l+1)}),
\] (6.10)

where the prior on controls $P_\theta(C_t)$ is restricted to the uppermost level $L$.

This provides substance to our surmise that the layers of the dynamic input-state-output models can be seen as form of hierarchical predictive coding [69]. Taking stock of the above discussion, the model we will investigate, can also be viewed in the perspective of a hierarchical predictive scheme.
Bibliography


[39] Neil D Lawrence. Gaussian process latent variable models for visualisation of high


