University of Nottingham
Computer Science
Computer Vision Laboratory

Continuous Regression

A Functional Regression approach to Real-time Facial Landmark Tracking

Enrique Sánchez-Lozano

Submitted in part fulfilment of the requirements for the degree of
Doctor of Philosophy in Computer Science of the University of Nottingham,
June 2017

Abstract

Facial Landmark Tracking (Face Tracking) is a key step for many Face Analysis systems, such as Face Recognition, Facial Expression Recognition, or Age and Gender Recognition, among others. The goal of Facial Landmark Tracking is to locate a sparse set of points defining a facial shape in a video sequence. These typically include the mouth, the eyes, the contour, or the nose tip. The state of the art method for Face Tracking builds on Cascaded Regression, in which a set of linear regressors are used in a cascaded fashion, each receiving as input the output of the previous one, subsequently reducing the error with respect to the target locations.

Despite its impressive results, Cascaded Regression suffers from several drawbacks, which are basically caused by the theoretical and practical implications of using Linear Regression. Under the context of Face Alignment, Linear Regression is used to predict shape displacements from image features through a linear mapping. This linear mapping is learnt through the typical least-squares problem, in which a set of random perturbations is given. This means that, each time a new regressor is to be trained, Cascaded Regression needs to generate perturbations and apply the sampling again. Moreover, existing solutions are not capable of incorporating incremental learning in real time. It is well-known that person-specific models perform better than generic ones, and thus the possibility of personalising generic models whilst tracking is ongoing is a desired property, yet to be addressed.

This thesis proposes Continuous Regression, a Functional Regression solution to the least-squares problem, resulting in the first real-time incremental face tracker. Briefly speaking, Continuous Regression approximates the samples by an estimation based on a first-order Taylor expansion yielding a closed-form solution for the *infinite* set of shape displacements. This way, it is possible to model the space of shape displacements as a continuum, without the need of using complex bases. Further, this thesis introduces a novel measure that allows Continuous Regression to be extended to spaces of correlated variables. This novel solution is incorporated into the Cascaded Regression framework, and its computational benefits for training under different configurations are shown. Then, it presents an approach for incremental learning within Cascaded Regression, and shows its complexity allows for real-time implementation. To the best of my knowledge, this is the first incremental face tracker that is shown to operate in real-time. The tracker is tested in an extensive benchmark, attaining state of the art results, thanks to the incremental learning capabilities.

Acknowledgements

I would like to start this by thanking my advisors Dr. Michel Valstar and Dr. Yorgos Tzimiropoulos. The final outcome of this thesis wouldn't be possible without your help and support. Both of you have spent lots of time teaching me valuable things, about writing, about research, about the way things should be done... but not only this, you also spent a valuable amount of time in teaching me how to handle my feelings. Although this "lecture" is still underway, I think we have made some progress on this. Thanks for your support in the hardest parts of this period. It definitely made me not give up and complete this piece of work.

Thanks to Dr. José Luis Alba-Castro. You always have the time to chat and discuss about everything, and I have to recognise that you were like my second father at the School of Telecommunications (given that he was also there!), and somehow I have to be thankful for all your help. Also, my special thanks to Dr. Daniel González-Jiménez, my first advisor, who always was the lucid person to say what was correct for all of us. I would like to also thank Dr. Fernando De la Torre, who lit my spark in theoretical research, with an application to real-life problems. Thanks for the opportunity you brought to me to have an amazing time at Carnegie Mellon, and live an unforgettable experience.

Thanks to all the Computer Vision Laboratory people. The Lab is probably one of the best environments I have found to work at. It is a pleasure to work with people who are driven by the aim to succeed in academic research, but collaborative and willing to help and spend their time helping other peers. Besides, I have to thank my colleagues for making the Lab a place in which one can balance work and life, so that one can succeed in both, and we can always enjoy a couple of pints and have fun when we feel stuck. It is impossible for me to name all of you, but I want to highlight those that stood my changeable mood day by day: the B86 Lab people: Aaron, Adrian, Dottie and Themos. This was a nice crew. Also, thanks to Brais, who was really helpful in key moments, I appreciate your help and advice. Thanks to Joy for her priceless support on language correctness. I cannot forget about Paula, who was a really good lab mate, and now is still a good friend.

Thanks to all those friends that were, are, and hope will be, there time to time to enjoy a beer. Patri, Jose, David, Ernesto, and all SM Color friends, who always bring me a warm welcome in the basketball court. Thanks to Miguel, it is always a pleasure to catch up with you.

A special thanks to my family: Pili, Quique, Marta, Alberto, Susa, Pilar and Manolo. Good and unforgettable times. There are no words to express my gratitude.

And finally, thanks to Marta Dominguez, who knows me and stays there day and night no matter what happens. Any word here would mean nothing compared to everything you gave to me all this time. This acknowledgment has to be extended to Lluvia and Nika as well. You will not read this, but you three are the most important thing that ever happened to me.

Dedication

Ao meu pai, Dr. Enrique Sánchez Sánchez (8-Xullo-1955, 19-Decembro-2015).

Thanks for your time, all the things you taught me, and your eagerness to make me want to be a good person, no matter any academic title or position. I still feel I am learning from you.

A mi madre, María Del Pilar Lozano Escolar (30-Septiembre-1957).

Por tu santa paciencia.

‘Túzaro!’

ESS

Contents

List of Tables

List of Figures

Table of Symbols

$\mathbf{E}_a$	$\triangleq$	a-dimensional Identity matrix
$\mathbf{1}_a$	$\triangleq$	a-dimensional vector with all elements set to 1
$\mathbf{s}$	$\triangleq$	Shape
$\mathbf{p}$	$\triangleq$	Shape parameters
$\mathbf{s}^*$, $\mathbf{p}^*$	$\triangleq$	Ground-truth shape / shape parameters
$\mathbf{s}_0$	$\triangleq$	Mean shape
$\mathbf{x}$, $\mathbf{x}^*$	$\triangleq$	Image features / Features extracted at the ground-truth positions
f, $f(\mathbf{s})$, $f(\mathbf{p})$	$\triangleq$	Image features extracting function
D	$\triangleq$	Dimensionality of the feature vector
d	$\triangleq$	Dimensionality of the feature vector after dimensionality reduction
f'	$\triangleq$	Derivative of feature extraction function f
$\mathbf{J}$, $\mathbf{J}^*$	$\triangleq$	Jacobian of image features / Jacobian of image features evaluated at the ground-truth positions
$\mathbf{B}_a$	$\triangleq$	Matrix of appearance bases
$\mathbf{B}_s$	$\triangleq$	Matrix of shape bases
N	$\triangleq$	Number of training images
K	$\triangleq$	Number of perturbations per image
$\boldsymbol{\mu}$	$\triangleq$	Mean vector of shape (parameters) displacements
$\boldsymbol{\Sigma}$	$\triangleq$	Covariance matrix of shape (parameters) displacements
$p(.)$	$\triangleq$	Probability distribution (pdf)
$\mathbf{M}$	$\triangleq$	$[\boldsymbol{\mu}, \boldsymbol{\Sigma} + \boldsymbol{\mu}\boldsymbol{\mu}^T]$
$\mathbf{B}$	$\triangleq$	$\begin{pmatrix} 1 & \boldsymbol{\mu}^T \\ \boldsymbol{\mu} & \boldsymbol{\Sigma}+\boldsymbol{\mu}\boldsymbol{\mu}^T \end{pmatrix}$
$\mathbf{D}_i^*$	$\triangleq$	$[\mathbf{x}_i^*, \mathbf{J}_i^*]$ Ground-truth data for image i
$\bar{\mathbf{D}}^*$	$\triangleq$	$[\mathbf{D}_1^*, \ldots, \mathbf{D}_N^*]$

Chapter 1

Introduction

Recent years have seen the field of Human-Computer Interaction, or HCI, growing fast, up to a level in which it is natural for us to interact with machines, even though the interaction itself may not yet be natural. The time where only experts were capable of using machines is over; now we are all[1] surrounded by computers, and users are now in charge. It is thus a fact that we are embracing a great challenge in making machines easily accessible and handled by everyone. New methods other than the classical mouse and keyboard have been developed to improve our interaction with novel devices. For instance, fingertips emerged as the natural tool to interact with recent mobile devices, and keyboards are being used less. Nowadays, cameras are embedded in even unimaginably small devices, and mobile phones have become the new de-facto portable cameras, which are now easily accessible and low-priced. Among all new methods, or tools, to interact with machines, one that is attracting a lot of attention is the face.

Faces are a main signal for humans when interacting with others. Our faces reveal important information about us. They reveal who we are (*identity*), our age and gender (i.e., our *demographic group*), and even our ethnicity. Faces also reveal our *emotional state*, such as whether we are angry, happy, disgusted, sad or surprised. It has been shown that facial expressions are universal with respect to emotional responses (Ekman & Oster 1979). Besides this, our

[1]Better said: we wish we were, though there is still a huge challenge in making machines, along with other resources, accessible to **all**. See for example the problem of the "digital divide".

face displays some of our *social traits*, such as our personality. We can generally tell whether a specific face belongs to an extrovert person, or whether it belongs to an easy-going person. Recently, new datasets and challenges have emerged aimed at using Computer Vision to effectively predict these traits (Ponce-Lopez et al. 2016). Finally, our faces reveal certain *social signals* (Pentland 2007), such as our willingness to buy a specific product, as well as our comfort when interacting with other people.

It is thus a research target to determine how humans behave by analysing their faces, and how faces can be modelled by means of analysing their spatio-temporal appearance variations. Even further, people demand easier ways to control devices: we want to unlock our mobile phones using our faces, or have a camera take a picture whenever we show a smile. All of these face-computer interactions are typically included in different mainstream research fields: face recognition, age and gender estimation, facial expression recognition and social traits. Face recognition, as well as age and gender estimation attempt to model identity, subject to appearance and aging factors, as well as age and gender themselves. On the other hand, facial expression recognition and social trait classification typically aims to analyse the spatio-temporal variation of faces, in terms of their expressions, and how we perceive and classify them. These topics are now of increasing demand in cross disciplinary research fields, such as entertainment, health, education or marketing. Typically, these cross disciplinary fields are also categorised within the novel field of "Affective Computing", which gathers Computer Vision, Psychology, Medicine and Neuromarketing into a field of research that ultimately attempts to make machines interact with users in an affective way; that is to say, based upon their emotional responses.

Regarding **entertainment** applications using faces, we barely need words to describe the success of Apps such as *Snapchat* (`https://www.snapchat.com/`), or *Masquerade* (`http://msqrd.me/`). In the field of **marketing**, we find many companies now dedicated to video analytics, such as *Visage Technologies* (`https://visagetechnologies.com/`), *Releyeble* (`http://www.releyeble.com/`), or *CrowdEmotion* (`http://www.crowdemotion.co.uk/`). All these start-ups build mainly upon analysing users' faces, to provide the customer with certain metrics that might be used to promote some products over others. These companies offer an enclosed

Figure 1.1: Sample pictures taken from `http://www.crowdemotion.co.uk/` (*upper-left*) and `http://www.releyeble.com/` (*upper-right*). Both companies offer an automated analysis of users emotions, so that further marketing and advertising can be done upon potential interest. The *bottom* image shows the case studies offered by **Visage Technologies**. As can be seen, the applications range is huge.

product capable of giving the customer all the behaviour metrics taken from the users. This is becoming increasingly important, because it allows machines to analyse huge amounts of data automatically and almost in real time, thus helping retailers focus on maximising their benefits over the market space. A sample picture is shown in Figure 1.1, where the products are clearly focused on improving market success.

But, even though marketing and entertainment appear to be the most visible fields of application, we cannot leave aside health and education. The automated analysis of faces is helping **health** in many ways. To name few: the automatic assessment of chronic pain (Aung et al. 2015, Egede et al. 2017), the detection of depression and concealing depression (Solomon et al. 2015), distinguishing between Autism Spectrum Disorder and Attention Deficit Hyperactiv-

ity Disorder (Jaiswal et al. 2017), estimating the gestational age of newborns (Torres Torres et al. 2017), or the automated support of children with autism syndrome with "affective" video games (Gordon et al. 2014). In general, one can discern a number of medical conditions that alter expressive behaviour. Such conditions have been coined 'Behaviomedical' (Valstar 2014), with the science of monitoring or treating such conditions coined 'Behaviomedics'. Similarly, **education** can benefit from newer advances in face analysis. The field of "Affective Tutoring Systems" (Ammar et al. 2010) has recently proven to improve how students engage in certain subjects they might feel are difficult, being driven by an automated system that can show content based on their emotional responses.

All of this illustrates the wide field of applications in which analysing faces plays an important role. But, what does "analysing faces" mean for a Computer Vision scientist? Certainly, it can be seen as a black box receiving an image or a video, and returning a class, where class can mean, e.g., whether a face is smiling or not, or whether it belongs to an specific person or not. The black box needs to process the image, by first locating the face and extracting some image descriptors, and then predicts the target class. In this whole process, the first step is crucial for the performance of any subsequent step. That is to say, precisely locating the face is very important. The more accurate the location of the face, the better the classification task will be, since any further analysis can be done in a more semantically meaningful way. For example, a small patch of pixels is more semantically meaningful if it is known that the appearance pertains to the mouth and is centred on a lip corner point, rather than being a specific block in a regular grid returned by a face detector. The simplest way of locating a face is by detecting a bounding box around the face; a problem that is known as "face detection", which has been widely studied within the Computer Vision community. However, relying only on such a vague location will lead to extracting image information in a poor way. That is to say, one would better analyse the face once a semantic meaning has been assigned to each of its elements: if we know that a specific region of the face is a mouth, we can further process it given that contextual knowledge. For instance, let us assume that we want to detect when someone smiles, using the geometric deformation of the mouth. If we *just* detect the face and align it with respect to the bounding box, we might not know where the mouth is. Thus,

sometimes we will extract information from the moustache, the chin, or anywhere else but the mouth. Hence, localising the face more accurately would improve the performance of the smile detector.

The problem of locating a set of points within a face, such as the mouth, the eyes, or the contour, is known as **Face Alignment**, or **Facial Point Localisation**, when it refers to static images, and as **Face Tracking** when it refers to video sequences. Typically, Face Alignment starts from an average shape placed within the bounding box given by a face detector, whereas Face Tracking takes advantage of the fact that faces move smoothly (given a sufficiently high framerate), to start from the points estimated at the previous frame. Thus, despite being trained in a similar way, they differ in some important aspects. In any case, Face Tracking algorithms adapt existing techniques from Face Alignment methods, and most of the methods to date exploit different techniques for Face Alignment, leaving aside the real-time capabilities that are crucial for a tracking system. The contributions of this thesis are thus in the field of Face Tracking: the system resulting from the application of the techniques presented herein is the first face tracking system capable of working in real-time whilst performing incremental learning. We will later see what incremental learning is and why it is so important. The set of points consisting of those target locations is known as *shape*. Figure 1.2 depicts what we refer to as a set of points, or *shape*. Thus, we can state that an accurate localisation of the points belonging to a shape is a key step for most face analysis systems. However, despite it being the first step (the second if we consider face detection as the first, separated from the face alignment step), for all the applications described above, it is still an open research topic in Computer Vision. The development of an accurate tracking or localisation system, with real-time capabilities, is still underway.

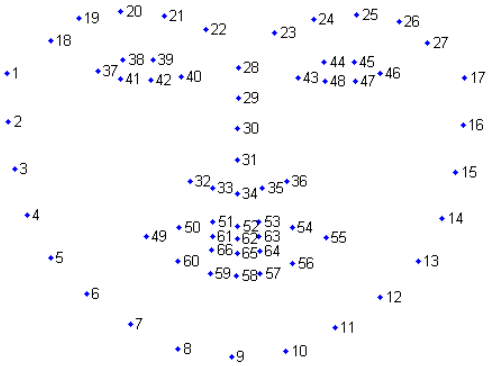

Figure 1.2: The set of points belonging to a *shape*. As can be seen, these points are sorted so that each point will always index a specific key location

We can perhaps discern two breakthroughs in the field of Face Alignment. The first one can be traced back to the early 00's, in which Cootes et al. (2001) proposed Active Appearance Models

(AAMs). AAMs are simply a statistical way to describe deformable objects (faces in our case), by allowing the points describing them (the shapes), and the corresponding appearances (the pixels), to vary under a small number of degrees of freedom, known as *parameters*. This way, AAMs provide a linear basis of an object's representation, in which any object instance can be represented by a linear combination of the basis. These bases capture the way objects vary by means of both shape and appearance. In other words, AAMs represent faces by a small set of parameters, encoding how shapes and appearances vary. We can therefore generate as many instances of the models (i.e., faces) as we want, just by changing the values of this small set of parameters. The face alignment problem was presented in Cootes et al. (2001) as the problem of finding the parameters that best describe an input face. These parameters are found by minimising a reconstruction error, named *residual*, measuring how much an instance of the model looks like the target face. We can say that the AAMs sparked the field of Face Alignment; the work of Cootes et al. is among the most cited works within all disciplines of Computer Science ($>$ 12000 citations in its 3 main papers Cootes et al. (1995, 1998, 2001)). We will briefly review the AAMs work in Chapter 3.

The AAMs influenced face alignment research in the past decade, especially after the landmark paper of Matthews & Baker (2004), who revisited many features of the original paper, proposing novel optimisation procedures for them. More specifically, they proposed an Inverse Compositional Framework (IC), which became the state of the art until the end of the 00's. However, despite the success of both AAMs and the IC, they were limited in some ways: they were hindered by a lack of generalisation, and, most importantly, were slow, and thus not suitable for a wide range of applications. The generalisation problem of AAMs and IC comes from the fact that they are *generative*: the localisation is based on the residual between the image and an instance of the model, meaning that, the less similar the face is with respect to the model, the harder it is for the model to fit it properly. For instance, a model generated from female faces will likely fail when fitting a male face. In that sense, the generalisation capabilities are said to be person-dependent.

Follow-up works attempted to overcome these two main limitations. Different methods were proposed: graphical models (Valstar et al. 2010), boosting (Liu 2011), and exhaustive con-

strained search (Saragih et al. 2011). These methods are typically classified as *discriminative*: they attempt to learn how to estimate the real locations (from here called the *ground-truth* locations), given the current information (i.e., the current shape estimate and the appearance). That is to say, they "discriminate", i.e., differentiate, a good localisation from a wrong one, and thus look for the location that would have the highest discriminative score. A simple approach for discriminative methods is using a Linear Regressor (Tresadern et al. 2010). The use of Linear Regression is simple: if we have a set of annotated images (known as *training set*), we can systematically displace the ground-truth of each of them, and extract some local descriptors (known as *features*). The regressor will be a map from these descriptors towards the displacement that has been applied to the ground-truth for the features to be taken, i.e., a Linear Regressor extracts some local information from a current estimate of where the points are, and then predicts where these points should be moved to be closer to the ground truth. However, we can not include *all* possible displacements from any region of the image and assume a single regressor will accurately learn from all of them. It is known that, the bigger the input variance (i.e., the range of input shapes), the lower the capacity of a regressor to work well. There is a trade-off between robustness and sensitivity of a regressor. In order to bypass this trade-off, Cao et al. (2012) adapted the Cascaded Regression approach of Dollár et al. (2010), to the space of shapes. However, the most successful form of Cascaded Regression is the Supervised Descent Method (SDM) (Xiong & De la Torre 2013). Briefly speaking, the SDM consists of a set of linear regressors in cascade, in which each regressor utilises the output shape of the previous one to predict a new shape, in a coarse-to-fine strategy. If the fitting process will always start with an average shape, we can train a regressor tasked with only rigid variations of that shape. Then, the second regressor will take the expected output and predict a new shape, and so forth. This way, the first level of the cascade will be highly robust at the cost of not being very precise, whereas subsequent levels will become gradually more sensitive, at the cost of retaining robustness. This novel approach revolutionised the state of the art in the field, bringing to the fore a fast and accurate method for landmark localisation, also suitable for tracking tasks. As of 2016, the SDM is considered a state-of-the-art method for face alignment and tracking.

However, the cascaded regression algorithm still suffers from three major limitations. First of all, the training, which is formulated by means of the common Least Squares problem, needs an exponential number of samples with respect to the dimensionality of the image descriptors in order for the models not to be biased. In general, the number of perturbations per image is small, and thus these will be biased. When creating a small number of samples, these hardly meet the distribution that has generated them. Second, it is still an open question how the training data, as well as their corresponding initialisations, affect performance in a specific scenario. In addition, training a cascaded regression model is computationally expensive. The training process is slow and requires data to be collected (i.e., sampled) each time a new model is to be trained for a different configuration. This way, training different models under different configurations is not feasible in many cases. Moreover, the computational cost of training a model grows exponentially with the number of cascade levels, the number of images used, and the number of perturbations generated per image. Finally, and more importantly, state of the art methods in cascaded regression are not capable of updating trained models in real time. It is well known that person-specific models will perform better than generic methods when the target person is known(Gross et al. 2005). However, training a person dependent model for each potential user is impossible in practice. Thus, a method capable of updating a tracker's model given the user's information, and in real-time, is a desired property for generic models. As we shall see, state-of-the-art methods are far too slow to be updated online when the tracking is ongoing.

Building upon the Cascaded Regression framework, this thesis analyses the three problems above, and ultimately presents a novel approach for the Linear Regression training, with an application to face tracking. It is important to distinguish the tasks of localisation in still images (detection) and in videos (tracking), since the former has no prior information and has to detect the facial landmarks from a given face detection, whereas the latter exploits temporal information. While the detection problem has been widely tackled in the literature, little attention has been paid to the tracking stage. There are two main reasons behind this: the first one is that existing methods were typically too slow, and thus impractical for the task of real-time video tracking. If we ignore the real-time requirement, processing each of the frames

can be done independently. The second one is the lack of annotated data for video sequences in challenging scenarios. The only extended benchmark that exists to date is the 300VW (Shen et al. 2015), which was released very recently.

Therefore, even though during this thesis the analysis of Linear Regression has led to two publications in the field of detection (Sánchez-Lozano et al. 2013, Sánchez-Lozano, Martinez & Valstar 2016), the work developed towards fulfilling this thesis focuses on tracking. The main contribution of this thesis is the development of a **Continuous Regression** approach for solving the Least Squares problem that is used in the training process for the tracking system. The Continuous Regression presented in this thesis builds upon, and further extends, a **Functional Regression** (Ramsay & Silverman 1997) approach. Functional Regression is a branch of statistics whose aim is modelling data assuming that observations are actually outcomes of continuous functions. That is to say, instead of having samples, we have continuous functions, and, thereby, what we observe is nothing but an outcome of such function. The study of Functional Regression has been widely explored in many fields of statistics. However, it has been difficult to extend it to the domain of images. When doing so, the main idea of considering samples as continuous functions is just to consider all the infinite samples taken in the surrounding points around the ground-truth data. However, existing approaches for Functional Regression are limited in what is referred to as the "sampling" assumption. We might argue that, as long as all samples are taken, the way each of them is taken is not important. Mathematically speaking, this assumption makes it hard to solve the Least Squares problem when the output data (i.e. the points) is, to some extent, correlated. For example, when perturbing the mouth, we can expect certain movements not to happen, e.g, the lower lip to appear above the upper lip. Clearly, the mouth moves as a whole. Existing functional regression approaches would assume that sampling each point of the mouth is independent with respect to other points belonging to it. In other words: it would assume that the dimensions are not correlated. In our application area of face tracking, this is clearly not the case.

The research presented in this thesis introduces two novel components to the Functional Regression approach for solving the Least Squares problem, that will help overcome the limitations listed above. More specifically, the *Continuous Regression* presented here: **1)** approximates the

input space by using the first-order Taylor expansion of the image features, and; **2)** introduces a "data term" tasked with encoding the correlation between target dimensions (the points). To the best of my knowledge, this is the first time that Functional Regression is solved using these two novel components, and the first time it is applied to the domain of images in this way. The solution presented in this thesis is finally introduced within the Cascaded Regression framework, resulting in a performance that is equivalent to that of sampling-based regression. The Continuous Regression presents some advantages with respect to the standard SDM: it can be updated in real-time, and it will enable the training of models, for different scenarios, without the need of sampling again, in a very fast training process. That is to say, the results of this thesis will lead to a fast and accurate incremental face tracking system. This new framework attains state-of-the art results in tracking, whilst working in real-time.

1.1 Contributions

Summarising, the contributions of this thesis are as follows:

- It presents a Functional Regression solution for the Least Squares problem, with an application to the imaging domain. The method is coined **Continuous Regression**. The closed-form solution presented herein relies on a first-order Taylor expansion of the image features and, to the best of my knowledge, this is the first attempt to solve the continuous least-squares problem with such basis, rather than with other complex basis typically used for that purpose.

- It presents a novel solution for the Continuous Regression in spaces of **correlated variables**, which is crucial when it comes to model facial point locations. More specifically, this thesis proposes to solve Continuous Regression over a *probability measure*, introducing a **"data term"** tasked with correlating dimensions. Instead of minimising the empirical loss function associated to the Least-Squares problem, the new solution *can be seen* as minimising the expected loss function. The final solution will confirm the assumption that the sampling distribution is not important when it comes to taking *all* samples within

a neighborhood, but will mathematically help dimensions to be correlated. Results will confirm the importance of that term, and thus the importance of proposing the Functional Regression solution upon the new measure.

- It presents a novel approach for **incremental learning**, i.e., the updating process. Unlike previous works, the tracking system presented in this thesis (coined **Incremental Cascaded Continuous Regression, iCCR**), is able to update a tracker's model in near real-time. Results will show that incremental learning is actually crucial to improve the localisation results, thus supporting the need for a system capable of doing it in real-time.

1.2 Outline

The thesis is organised as follows. Chapter 2 gives a detailed introduction to the Face Alignment and Tracking problem, depicting an overview of the whole system. Then, Chapter 3 presents a comprehensive literature review. The core of this thesis starts in Chapter 4, where the problem of Continuous Regression is presented. First, the Least-Squares problem is extended to the continuous domain using a standard Functional Regression form, and an approximation of the input space by a first-order Taylor expansion is given. The limitations of Continuous Regression using classical Functional Regression approaches are then studied in Chapter 5, and an alternative measure is utilised to allow the extension of Continuous Regression to spaces of correlated variables. Continuous Regression is used for the task of tracking, which differs from the task of detection in the way it is initialised (i.e., from the points of the previous frame). Afterwards, Chapter 6 derives an incremental learning approach under the context of Continuous Regression, and analyses its complexity, showing its capabilities for real-time implementation. The results clearly illustrate the benefit of incremental learning, especially in challenging scenarios. The final system is then presented and assessed in an annotated benchmark. Finally, Chapter 7 provides a detailed discussion and future work, briefly analysing the results achieved in this thesis.

1.3 Publications

The main research conducted in this thesis has been partially published in the following conferences and journals, sorted by date:

- Enrique Sánchez Lozano, Fernando De la Torre, and Daniel González-Jiménez.
  Continuous Regression for Non-rigid Image Alignment.
  ECCV 2012 - European Conference on Computer Vision.
  Sánchez-Lozano et al. (2012)

- Enrique Sánchez-Lozano, Brais Martinez, Georgios Tzimiropoulos, and Michel Valstar.
  Cascaded Continuous Regression for Real-time Incremental Face Tracking.
  ECCV 2016 - European Conference on Computer Vision.
  Sánchez-Lozano, Martinez, Tzimiropoulos & Valstar (2016)

Also, it is worth mentioning the following manuscript, which is now under review:

- Enrique Sánchez-Lozano, Georgios Tzimiropoulos, Brais Martinez, Fernando De la Torre, and Michel Valstar.
  A Functional Regression approach to Facial Landmark Tracking.
  Submitted for review at IEEE Trans. on Pattern Analysis and Machine Intelligence - Preprint available on arXiv.
  Sánchez-Lozano, Tzimiropoulos, Martinez, De la Torre & Valstar (2016)

Finally, despite not being part of this thesis, the following publications helped understand the problem of Linear Regression for Face Alignment, although the contributions are tangential to the work of this document.

- Enrique Sánchez-Lozano, Enrique Argones-Rúa, and Jose Luis Alba-Castro.
  Blockwise Linear Regression for Face Alignment.
  BMVC 2013 - British Machine Vision Conference.
  Sánchez-Lozano et al. (2013)

- Enrique Sánchez-Lozano, Brais Martinez, and Michel Valstar.

  Cascaded Regression with Sparsified Feature Covariance Matrix for Facial Landmark Detection.

  PRL 2016 - Pattern Recognition Letters **73**, pp. 19-26

  Sánchez-Lozano, Martinez & Valstar (2016)

Chapter 2

Background Theory

This Chapter introduces the reader to the problem of Face Alignment and Tracking. First of all, Section 2.1 introduces the problem that motivates the research conducted towards fulfilling this thesis. Then, Section 2.2 introduces certain notation that will be followed throughout the thesis; Sections 2.3 and 2.4 introduce the *Shape Model* and *Appearance Model*, which are the main components involved in the problem of Face Alignment. Finally, Section 2.5 presents a brief classification of existing methods that have been typically used in the literature towards solving the problem of Facial Point Localisation.

2.1 Problem definition

Facial Landmark Detection, or Face Alignment, aims to locate a set of n specific points on either images (a problem known as detection) or videos (known as tracking). In this thesis, n will be typically set up to 66 points, corresponding to those shown in Figure 1.2 (Chapter 1). The set of points to be located is called a *shape*. Mathematically speaking, a 2D shape is a vector describing the location of the x and y coordinates of its n points. That is to say, we define a shape as $\mathbf{s} = \{x_i, y_i\}_{i=1...n} \in \mathbb{R}^{2n}$. Throughout this thesis, the notation convention that will be followed will be to represent shapes as $2n$ dimensional column vectors, in which the x coordinates are located first. That is to say, a shape $\mathbf{s}$ will be represented as $\mathbf{s} = (x_1, \ldots, x_n, y_1, \ldots, y_n)^T$.

Thereby, the problem of Face Alignment consists of locating these points in an image, or a video, in which there is a face. Without loss of generality, and for the task of detection, we will assume that a Face Detection process has been successfully carried out, and thus we have a bounding box within which we assume the face is contained. Figure 2.1 depicts an example of the output of a Face Detection system. Many built-in systems can be used for that task, such as the Viola-Jones algorithm (Viola & Jones 2004), the open source dlib `dlib.net`, or a part-based model (Orozco et al. 2015). The whole process is depicted in Figure 2.1. In both the point detection and tracking tasks, the localisation process starts with an initial guess of where the shape might be. In the case of detection, the initial guess is an average shape, like the one shown in Figure 2.2 (**Left**). During tracking, we can expect that at sufficiently high frame rates the shape variations between consecutive frames will not be drastically high, i.e., faces will move smoothly from frame to frame, and thus a better initialisation would be the shape estimated for the previous frame. In both cases, we can treat the problem as a supervised learning process, in which there exists an available *training set*, consisting of a set of images (containing faces), that have been manually annotated. In most cases, a *model* is learnt from available data, and the localisation process consists of *fitting* the model onto a new image, so that the points lie where we expect them to.

2.2 Notation

Throughout this thesis, the following notation will apply, unless explicitly stated otherwise. Bold uppercase letters represent matrices ($\mathbf{B}$). Bold lowercase letters denote column vectors ($\mathbf{b}$). Non-bold letters represent scalar variables (b), or functions (f). The L2 norm of a vector will be represented as $\|\mathbf{b}\|_2 = \sqrt{\mathbf{b}^T\mathbf{b}}$. The matrix norm (also known as Frobenius norm) is represented as $\|\mathbf{B}\|_F = \sqrt{tr(\mathbf{B}\mathbf{B}^T)} = \sqrt{\sum_{i,j} b_{ij}^2}$, where b_{ij} is the i,j entry of matrix $\mathbf{B}$, and $tr(\mathbf{B}) = \sum_i b_{ii}$ is the trace of a square matrix. The weighted norm is represented as $\|\mathbf{b}\|_\mathbf{W} = \sqrt{\mathbf{b}^T\mathbf{W}\mathbf{b}}$. The k-th dimensional identity matrix will be represented as $\mathbf{E}_k$. Note that $\|\mathbf{b}\|_2 = \|\mathbf{b}\|_\mathbf{E}$. The k-th dimensional vector with all its elements set to 1 will be denoted as $\mathbf{1}_k$. The composition of two functions $f(x)$ and $g(x)$ is denoted as $f \circ g = f(g(x))$. The

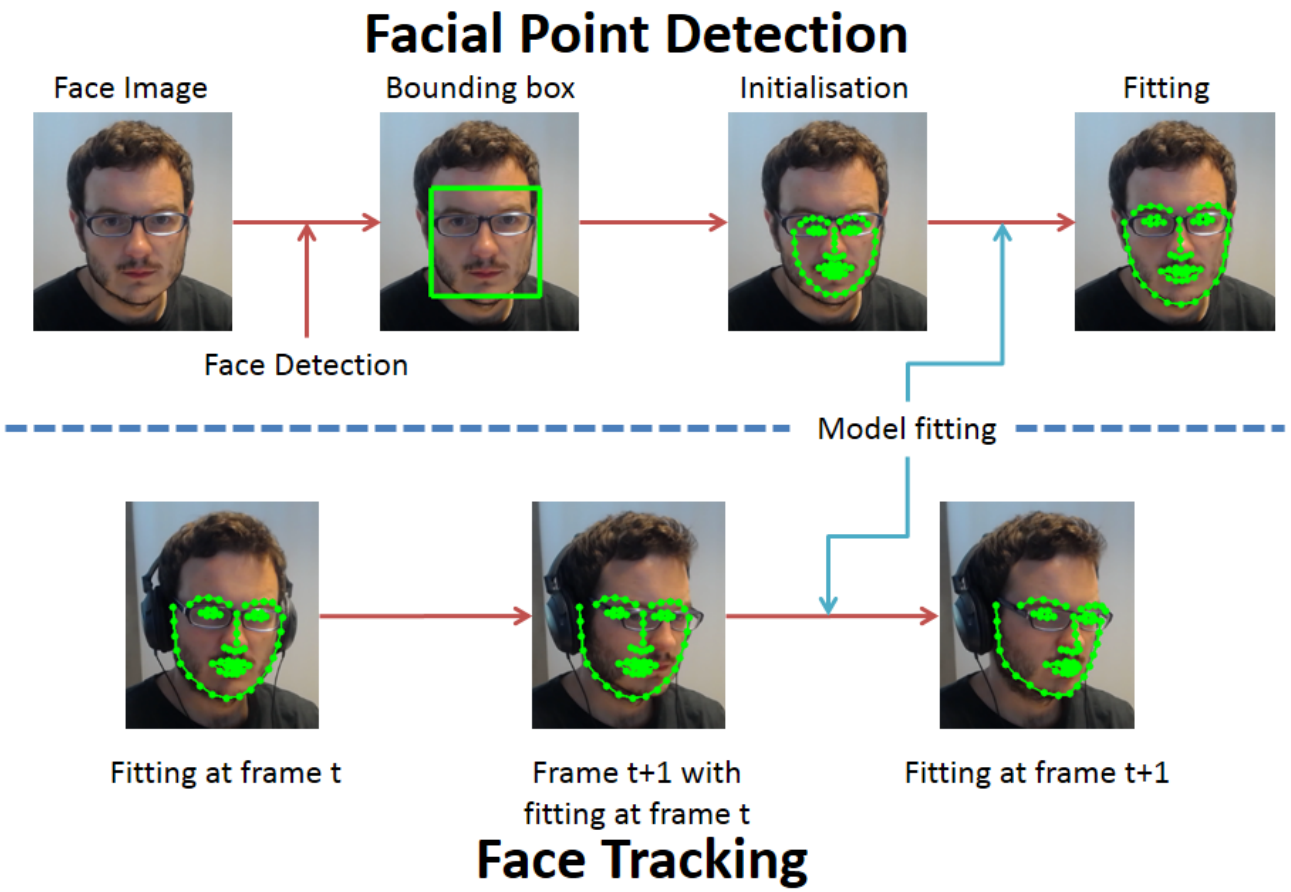

Figure 2.1: Fitting process for both detection (**Top**) and tracking (**Bottom**) tasks. The detection starts with roughly locating the face and initialising the points with the mean shape. Then, the fitting is done using a learnt model. Tracking differs from detection in the way the initialisation (and subsequently, the training) is done. Given the image at frame $t+1$, the fitting starts with the points located at frame t.

vectorisation of a matrix will be represented as $vec(\mathbf{B})$ and consists of concatenating all the column vectors of $\mathbf{B}$ into a single column vector. Finally, an upper-asterisk will represent the ground-truth, or annotated, data. For instance, $\mathbf{s}^*$ will represent an annotated shape, and $\mathbf{p}^*$ will be its corresponding ground-truth shape parameters.

2.3 Shape Model

Faces are structured deformable objects, in the sense that landmarks vary as a whole. It is thus a common approach for the task of Facial Landmark Localisation to rely on a *Shape Model*, which encodes and constrains variations of shapes in a lower dimensional space, known

as the space of **parameters**. The modelling of faces under a lower dimensional space of parameters allows for the shapes to be kept constrained to "plausible" faces (i.e., real faces), and to constrain the search to just a few dozens of dimensions, rather than the $2n$ dimensions in which shapes vary. A Shape Model (often called Point Distribution Model, PDM, as well[1]), is built from shapes that make up the training set. A standard approach to building a shape model is to first normalise the training shapes using Generalised Procrustes Analysis (Goodall 1991)[2], which removes all global *rigid information*, i.e., the rotation, translation, and scale. Normalised shapes are different to each other in what refers to non-rigid deformations only, which are modelled by performing Principal Component Analysis (PCA, Cootes et al. (1992)) on them. PCA generates a shape basis $\mathbf{B}_s \in \mathbb{R}^{2n \times k}$, which are the eigenvectors corresponding to the $k << n$ largest eigenvalues of the covariance matrix computed from the normalised shapes. k will then represent the amount of *non-rigid shape parameters*. We can represent any shape with its k low-dimensional parameters by $\tilde{\mathbf{s}} = \mathbf{s}_0 + \mathbf{B}_s\mathbf{c}$, where $\mathbf{c} \in \mathbb{R}^{k \times 1}$ represents the shape parameters, and $\mathbf{s}_0 = (x_1^0, \ldots, x_n^0, y_1^0, \ldots y_n^0)$ represents the mean shape, computed from the training shapes before applying PCA. The shape $\tilde{\mathbf{s}} = (\tilde{x}_1, \ldots, \tilde{y}_n)$ does not contain any rigid information. This rigid information needs however to be fitted as well, and thus we have to model it along with non-rigid information. Rigid transformations, i.e., scale s, rotation θ and translation $\{t_x, t_y\}$ are applied to $\tilde{\mathbf{s}}$ to obtain $\mathbf{s}$, as follows:

$$\begin{bmatrix} x_i \\ y_i \end{bmatrix} = \begin{bmatrix} s\cos\theta & -s\sin\theta \\ s\sin\theta & s\cos\theta \end{bmatrix} \begin{bmatrix} \tilde{x}_i \\ \tilde{y}_i \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}. \tag{2.1}$$

However, as we shall see in Chapter 3, many proposed methods for face alignment have typically looked for a linear update of either shapes or shape parameters. However, modelling rigid parameters in a linear way using Equation 2.1 is not feasible. In order to be able to do so, we can generate a set of linear basis (Matthews & Baker 2004), aiming to model rigid information

[1] Although it is often to use both terms to refer to the definition shown in this thesis, a Shape Model would also include other ways of modelling shapes

[2] It is worth noting that Procrustes Analysis is the process of aligning a shape to a reference shape, while Generalised Procrustes Analysis is the process of aligning a set of shapes, without a reference shape given

in a linear way. More specifically, we can define a set of linear basis as:

$$\mathbf{b}_1 = \mathbf{s}_0 = (x_1^0, \ldots, x_n^0, y_1^0, \ldots, y_n^0)^T / \|\mathbf{s}_0\|, \tag{2.2}$$

$$\mathbf{b}_2 = (-y_1^0, \ldots, -y_n^0, x_1^0, \ldots, x_n^0)^T / \|\mathbf{s}_0\|, \tag{2.3}$$

$$\mathbf{b}_3 = (1, 1, \ldots, 0, 0)^T / \sqrt{n}, \tag{2.4}$$

$$\mathbf{b}_4 = (0, 0, \ldots, 1, 1)^T / \sqrt{n}. \tag{2.5}$$

Then, the matrix $\mathbf{B}_q \in \mathbb{R}^{2n \times 4}$, in which each column vector is defined by $\mathbf{b}_i$, defines a linear span of rigid transformations (the reader might be convinced that the rigid bases are orthonormal). We can readily see that, for a given set of rigid parameters $\mathbf{q} = (q_1, \ldots, q_4)$, we would be able to apply a linear transformation as:

$$\begin{bmatrix} a & b \\ -b & a \end{bmatrix} \begin{bmatrix} x_i^0 \\ y_i^0 \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} = \begin{bmatrix} x_i^0 \\ y_i^0 \end{bmatrix} + \mathbf{B}_q \mathbf{q} \tag{2.6}$$

Where now $a = s\cos\theta$ and $b = s\sin\theta$. The relation between $\mathbf{q}$ and a, b, t_x, t_y is given as:

$$a = 1 + \frac{q_1}{\|\mathbf{s}_0\|}\,,\; b = \frac{q_2}{\|\mathbf{s}_0\|}\,,\; t_x = \frac{q_3}{\sqrt{n}}\,,\; t_y = \frac{q_4}{\sqrt{n}} \tag{2.7}$$

The process of computing the shape parameters $\mathbf{p} = [\mathbf{q}; \mathbf{c}]$ is known as *shape registration*, and can be done using Algorithm 1. The process starts from first computing the rigid parameters $\mathbf{q}$, from which a, b, t_x, t_y can be obtained. These parameters are however those that align the mean shape $\mathbf{s}_0$ to the input shape $\mathbf{s}$. However, we need to align $\mathbf{s}$ to $\mathbf{s}_0$, for the non-rigid parameters to be computed. That is to say, we need to compute the inverse "trail" $\hat{a}, \hat{b}, \hat{t_x}, \hat{t_y}$ so that, after computing $\left[\begin{smallmatrix} \hat{x}_i \\ \hat{y}_i \end{smallmatrix}\right] = \left[\begin{smallmatrix} \hat{a} & \hat{b} \\ -\hat{b} & \hat{a} \end{smallmatrix}\right] \left[\begin{smallmatrix} x_i \\ y_i \end{smallmatrix}\right] + \left[\begin{smallmatrix} \hat{t}_x \\ \hat{t}_y \end{smallmatrix}\right]$, we have removed the rigid information. The parameters $\hat{a}, \hat{b}$ can be computed from the inverse of the projection matrix $\mathbf{P} = \left[\begin{smallmatrix} a & b \\ -b & a \end{smallmatrix}\right]$, whereas computing $\hat{t}_x, \hat{t}_y$ is not straightforward. However, we can see that these are indeed not necessary, given that the projection $\mathbf{c} = \mathbf{B}_s^T(vec(\mathbf{s}\mathbf{P}^{-1} + \left[\begin{smallmatrix} \hat{t}_x \\ \hat{t}_y \end{smallmatrix}\right] - \mathbf{s}_0)) = \mathbf{B}_s^T(vec(\mathbf{s}\mathbf{P}^{-1} - \mathbf{s}_0))$. That is to say, $\mathbf{B}_s^T \left[\begin{smallmatrix} \hat{t}_x \\ \hat{t}_y \end{smallmatrix}\right] = 0$, given that the columns of $\mathbf{B}_s$ are orthonormal. Therefore, we can compute the shape parameters without computing $\hat{t}_x, \hat{t}_y$. Now, the process of computing a shape from the

Algorithm 1 Shape Registration

1: Input data: shape $\mathbf{s}$, Shape Model : $\{\mathbf{s}_0, \mathbf{B}_q, \mathbf{B}_s\}$
2: $\mathbf{q} = \mathbf{B}_q^T(\mathbf{s} - \mathbf{s}_0)$
3: Compute a, b, and projection matrix $\rightarrow \mathbf{P} = \begin{bmatrix} a & b \\ -b & a \end{bmatrix}$.
4: $\mathbf{c} = \mathbf{B}_s^T(vec(\mathbf{s}\mathbf{P}^{-1} - \mathbf{s}_0))$.
5: Return $\mathbf{p} = [\mathbf{q}; \mathbf{c}]$

shape parameters is known as *shape reconstruction*, and can be done using Algorithm 2. We can see that any shape $\mathbf{s}$ can be represented by its shape parameters $\mathbf{p} = [\mathbf{q}; \mathbf{c}]$, and that we can therefore represent any shape $\mathbf{s}$ as a function of its shape parameters as:

$$\mathbf{s}(\mathbf{q}, \mathbf{c}) = vec\left(\begin{bmatrix} 1 + q_1/\|\mathbf{s}_0\| & q_2/\|\mathbf{s}_0\| \\ -q_2/\|\mathbf{s}_0\| & 1 + q_1/\|\mathbf{s}_0\| \end{bmatrix} \begin{bmatrix} (\mathbf{B}_s^x\mathbf{c} + \mathbf{s}_0^x)^T \\ (\mathbf{B}_s^y\mathbf{c} + \mathbf{s}_0^y)^T \end{bmatrix} + \begin{bmatrix} \frac{q_3}{\sqrt{n}} \\ \frac{q_4}{\sqrt{n}} \end{bmatrix} \mathbf{1}_n^T\right) \tag{2.8}$$

where $\mathbf{B}_s^x$ represents the submatrix of $\mathbf{B}_s$ accounting for the first n rows, $\mathbf{B}_s^y$ is the submatrix of $\mathbf{B}_s$ for rows ranging from $n+1$ to $2n$, $\mathbf{s}_0^x$ represents the x coordinates of $\mathbf{s}_0$, and $\mathbf{s}_0^y$ represents the y coordinates of $\mathbf{s}_0$.

Algorithm 2 Shape Reconstruction

1: Input data: shape $\mathbf{p} = [\mathbf{q}; \mathbf{c}]$, Shape Model : $\{\mathbf{s}_0, \mathbf{B}_q, \mathbf{B}_s\}$
2: $\tilde{\mathbf{s}} = \mathbf{s}_0 + \mathbf{B}_s\mathbf{c}$
3: $a = 1 + \frac{q_1}{\|\mathbf{s}_0\|}$, $b = \frac{q_2}{\|\mathbf{s}_0\|}$, $\rightarrow \mathbf{P} = \begin{bmatrix} a & b \\ -b & a \end{bmatrix}$.
4: $\mathbf{t} = [q_3\,,\, q_4]/\sqrt{n}$
5: $\mathbf{s}_{\text{recons}} = \tilde{\mathbf{s}}\mathbf{P} + \mathbf{t}$
6: Return $\mathbf{s}_{\text{recons}}$

We can see then that with a Shape Model, faces are uniquely represented by the set of **shape parameters** $\mathbf{p} = [\mathbf{q}, \mathbf{c}] \in \mathbb{R}^{(k+4)\times 1}$. Figure 2.2 (**Right**) shows the variation associated with the first two principal components corresponding to non-rigid parameters. As can be seen, the first one corresponds to variation in pose. In fact, it is possible to remove, from the first parameter, any non-rigid deformation but pose changes by computing the shape model in an augmented training set including the mirrored (or symmetric) shapes. This is specially useful when faces need to be frontalised for face recognition (González-Jiménez & Alba-Castro 2007), as it has been shown that mirroring enforces the posed-related non-rigid parameter to be linear with

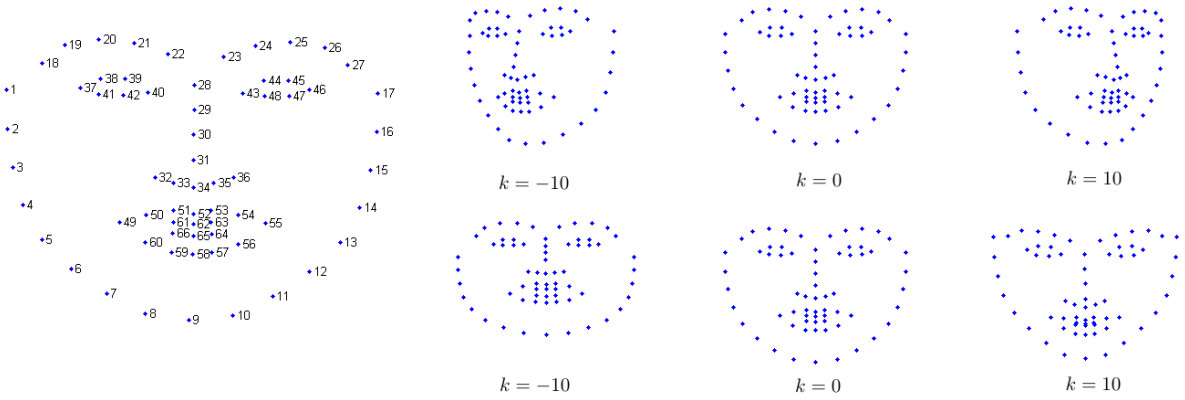

Figure 2.2: **Left** Shape example, labelled following a specific distribution. During training, all faces are labelled following this distribution, so that the problem is consistent. **Right** Rows correspond to the variation of the first two modes of the shape model, respectively. These faces were generated by setting the i-th element of $\mathbf{c}$ to k, and the remaining parameters to zero. That is, $\mathbf{s} = \mathbf{s}_0 + k\,\mathbf{B}_s^i$, where k is the constant used to show the variation, and $\mathbf{B}_s^i$ is the i-th column (1 in the upper row of the image, 2 in the lower row) of the shape basis.

respect to the pose angle. However, such symmetrical enforcement might be harmful for the case of facial point localisation, mainly due the fact that faces are actually asymmetric (Liu et al. 2003). This is still an interesting question to address, which lies out of the scope of this thesis. In this thesis, both types of models were tested, in which the model with mirrored shapes showed to have a lower reconstruction error (i.e., the distance between $\mathbf{s}$ and $\mathbf{s}_{\text{recons}}$) for the databases used in the experiments.

2.4 Appearance Model

Apart from shapes, faces are also represented by what they look like, i.e., their appearances. The appearance is, in short, what we see. We see and interpret an image and we are able to identify where the key points are, and we want machines to automatically do the same. What we see, as well as any higher level description of it, allows us to identify a face and its parts, and is referred to as the appearance. In a computerised image, the appearance can be represented by either the pixels, or any higher-level representation derived from them. We will refer to both as the **features**[3]. That is to say, the **features** are simply a way to represent and describe

[3]The reader might notice that different works refer to these two concepts in a different way. In this thesis, both concepts will be defined in a manner that will be consistent through the whole document

images, and, more specifically, faces. Early works in Computer Vision used the raw pixels as image descriptors, but these were soon found to be poor object descriptors, mainly due their high-variability (256 values per pixel in a typical *uint8* image), and their non-smooth transition between positions and consecutive images. For instance, the same image under a small change in illumination would produce a completely different feature vector. These problems imply that we need a huge variance to describe objects, and make the problem of "object identification" pretty hard.

Thus, descriptors invariant to some of these nuisance factors were proposed, such as Local Binary Patterns (LBP, Ojala et al. (1996)), or Gabor filters (Fogel & Sagi 1989). LBPs describe each pixel location by the difference with respect to its neighbors. The difference is further binarised to be 0 or 1 depending upon whether the difference is greater or lower than zero. This way, variances in, e.g., illumination, barely affect the descriptors. Gabor filters were proposed as a set of Gaussian-attenuated sinusoidal filters that are supposed to simulate the way that human vision processes information towards detecting key points. However, recent descriptors based on the gradients of the image pixels have proven to work better than LBP and Gabor. These are the Histograms of Oriented Gradients (HOG, Dalal & Triggs (2005)), and the closely related Scale-Invariant Feature Transform (SIFT, Lowe (2004)). The process of extracting the HOG features starts with the computation of the image gradients in both the x and y coordinates. Gradients of image pixels are computed by applying a 1D point-centred discrete derivative, or even more complex filters such as the Sobel mask. Gradients will have a magnitude and orientation. Then, a histogram of orientations is computed, in which each point votes to the bin containing the orientation of its gradient, weighted by its magnitude. The histogram is further organised and normalised in blocks and finally re-normalised as a whole.

The way we represent faces is key for the task of face alignment and tracking to work adequately. We have seen some typical descriptors with which we can efficiently represent faces, but we can also analyse how we perform the feature extraction step. In particular, we can distinguish some common patterns that different approaches share depending on how they extract the image features. More specifically, we can distinguish between different approaches for image feature extraction: those that are holistic or part-based. Holistic approaches try to model the

representation of the face as a whole, whereas part-based approaches model the image features in local neighborhoods surrounding each point. In holistic approaches, the image is deformed by applying a warping function, aligning it to the reference shape, and then the appearance belonging to the convex hull defined by the points is extracted. A warping function $\mathcal{W}(\mathbf{s}, \mathbf{p})$ is a one-to-one correspondence (i.e., is invertible), between all the pixel locations within the convex hull $\Omega(\mathbf{s})$ defined by any shape $\mathbf{s}$ and their corresponding locations in a shape defined by $\mathbf{p}$. We have to note that $\mathcal{W}$ returns the 2D coordinates of all the corresponding points of $\Omega(\mathbf{s})$ translated by $\mathcal{W}(\mathbf{s}, \mathbf{p})$. This defines the unique trail with which we can project the points from the given image back to the reference frame, as well as the other way around. Typical warping functions are the Piecewise Affine and the Thin Plate Splines (Cootes & Taylor (2004)). An example is depicted in Figure 2.3 (**Top**), in which the warped image is computed as $\mathbf{I}(\mathcal{W}(\mathbf{s}_0, \mathbf{p}))$, with $\mathbf{s}_0$ the mean shape. That is to say, we register the pixels of the image that lie within the estimated points, to the pixel locations in the reference frame. In contrast, part-based models define a *patch* around each point, and extract the features within them, this process is depicted in Figure 2.3 (**Bottom**). In part-based approaches there is still an image deformation, tasked with removing scale, rotation, and translation (i..e, the rigid information). In a convenient abuse of notation, we will indistinctly use the function $\mathcal{W}$ to denote such transformation. Nevertheless, in both approaches, when the image descriptors are extracted, these are typically collated to form the **feature vector**, which will ultimately represent the specific face under the points describing it.

To simplify notation, we will denote the feature vector as $\mathbf{x} = f(\mathbf{I}, \mathbf{s})$, where f is the feature extraction function of image $\mathbf{I}$ given the shape $\mathbf{s}$. Often, we will indistinctly refer to $\mathbf{x} = f(\mathbf{I}, \mathbf{p})$, in which $\mathbf{p} = [\mathbf{q}, \mathbf{c}]$ represents the shape parameters. Note that the image $\mathbf{I}$ will be either the warped image in holistic approaches, or the registered images in part-based approaches. Even though the variance and the dimensionality of newer descriptors is lower than that of pixels, these are still subject to a high-variability and to the curse of dimensionality. It is thus a common approach to apply a dimensionality reduction technique to the image features. Given an available training set, the image descriptors are extracted under the set of training shapes, and Principal Components Analysis is computed over them. This gives us a linear

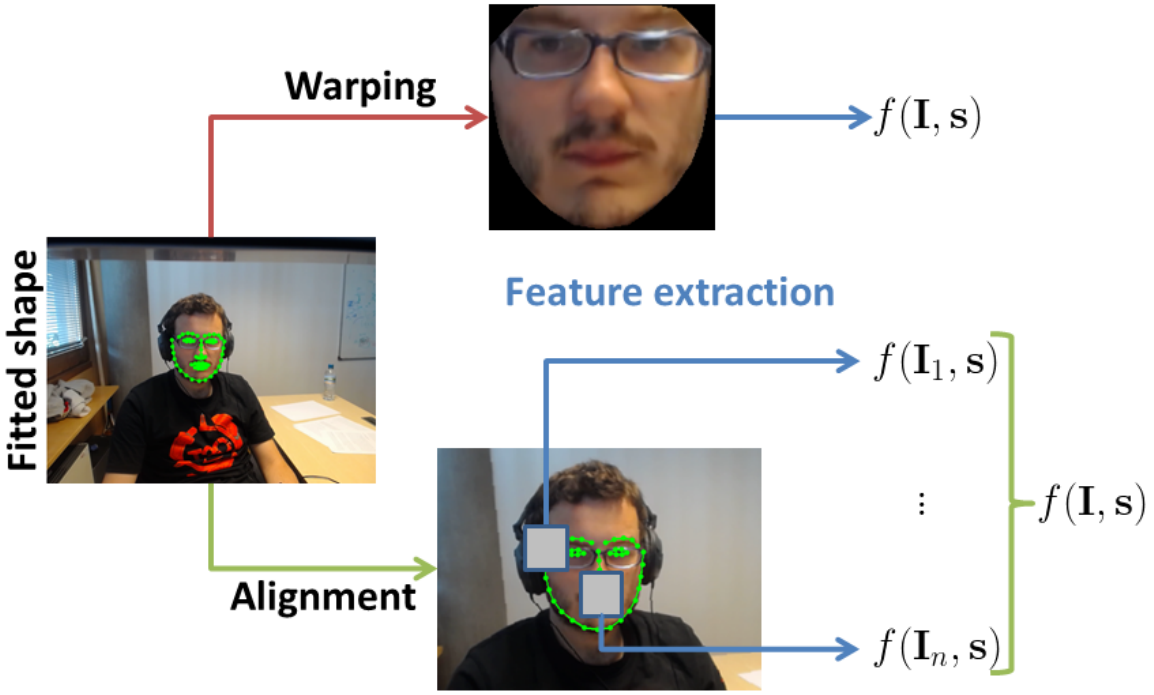

Figure 2.3: Different methods for extracting features from a facial image. **Top** image depicts the **holistic** representation of the face. The face is warped back from the original image to the mean shape, and then the whole convex hull defined by the external points defines the image within which features are extracted. **Bottom** image depicts the **part-based** feature extraction. First of all, the image is aligned so that scale, rotation and translation are removed, so that these artifacts do not affect the feature extraction process. Then, a patch around each point is defined, from which features are extracted, and further concatenated to form the final descriptor.

representation of faces $\mathbf{x} = \mathbf{x}_0 + \mathbf{B}_a\boldsymbol{\lambda}$, where $\mathbf{x} \in \mathbb{R}^{D\times 1}$ is the feature vector of dimension D, $\mathbf{x}_0$ is the mean feature vector, $\mathbf{B}_a \in \mathbb{R}^{D\times d}$ is the appearance basis, and $\boldsymbol{\lambda} \in \mathbb{R}^{d\times 1}$ is the vector of **appearance parameters**. The set $\{\mathbf{x}_0, \mathbf{B}_a\}$ is called the **Appearance Model**, and it is typically used in *generative methods* for fitting (see Section 2.5). In other approaches, the appearance model is just used to reduce the dimensionality of the input vector. In these cases, the set of apperance parameters is just the representation of the input feature vector in a lower dimensional space. As a convenient abuse of notation, in situations where we are interested in the feature vector itself as a descriptor, and not in its appearance parameters, we will keep the symbol $\mathbf{x} \in \mathbb{R}^{d\times 1}$ to refer to the feature vector in the d dimensional space.

2.5 Fitting approaches

There are many ways to classify existing approaches for Face Alignment. Perhaps the most extensive classification splits methods based on whether they are *generative*, *discriminative*, or *statistical*. Generally speaking, these are:

Generative Methods typically optimise both shape and appearance parameters $\mathbf{p}$ and $\boldsymbol{\lambda}$. In generative methods, the alignment is achieved by minimising a cost function w.r.t. the shape (and appearance) parameters. The cost function is typically the distance between a model instance (i.e., an image generated using $\mathbf{p}$ and $\boldsymbol{\lambda}$), and the feature vector extracted at the current estimation of the input shape. This distance is typically called the reconstruction error, and is computed by projecting back the appearance parameters. Often, generative methods use a *holistic* representation of the face, and PCA is computed on the ground-truth data.

Discriminative Methods directly learn a mapping from the image features $\mathbf{x}$, to the shape parameters $\mathbf{p}$, efectively marginalising out the need to minimise the reconstruction error. This mapping can be obtained by linear regression, complex regression models, or through a hard decision classifier (e.g., SVM), which outputs whether the model is well aligned or not. Thus, discriminative methods attempt to locate the points where the alignment is correct, by means of how well the feature vector discriminates between it being in a good or bad location. Contrary to generative methods, discriminative methods typically deploy a *part-based* representation of the face, and PCA is typically used to reduce the dimensionality of the input vector, and thus is done over the appearances of perturbed shapes, rather than over the ground-truth data.

Statistical Methods combine both generative and discriminative approaches. The appearance model is substituted by local models, called patch experts, which locally model the goodness of a fit (i.e., how well a current shape is fitted). Given the experts, a classifier function, which outputs whether the shape is correctly aligned or not, is maximised w.r.t. the shape parameters. Typically, this process entails two steps: an exhaustive local search per point, which does not take into account other landmark localisations, and an optimisation step, which attempts to find the optimal parameters that describe the closest shape to that found by the local experts.

Chapter 3

Literature Review

The goal of this chapter is to provide a comprehensive literature review on the relevant topics addressed in this thesis. The main topic and application of this thesis lies in the field of Face Alignment and Tracking, and thereby this is the major topic to be reviewed. However, the work presented in this thesis also lies in the field of Functional Regression, and thus it is also necessary to review related research in that field.

3.1 Literature Review in Face Alignment and Tracking

The problem of Face Alignment and Tracking is strongly related to the modelling of deformable objects. As such, faces are nothing but a class of objects with a huge variance. Faces can vary in identity, expression, pose, or illumination. However, despite all these changes, they remain, obviously, faces. It is thus important to find a natural way to model faces and the way they typically vary, so that we can guarantee that any new face can be modelled accurately, and hence the location of its key points is plausible. It is thus almost natural to see that the work that opened research in face alignment and tracking was mainly focused on proposing a novel and accurate way to model faces: Active Shape Models. Today, it is commonly accepted that the roots of face alignment and tracking start with the seminal work of Cootes et al. (Cootes et al. 1992, Cootes & Taylor 1992).

3.1.1 Active Shape Models

At the beginning of the 90's, the state-of-the-art in the topic of modelling deformable objects was yet to be unearthed, and existing works were mainly constrained to rigid variations. Briefly speaking, the field of face tracking was mainly constrained to the template matching proposed by Lucas & Kanade (1981), in which only a bounding box, manually located at the first frame, was tracked through a video sequence. At that time, there were no reliable works exploring ways to model deformable objects by means of non-rigid deformations, such as those that occur when the object changes its position with respect to the camera, or undergoes a deformation. In this sense, the modelling of faces as deformable objects was completely unexplored, and existing works were mainly focused on the tracking of faces as rigid objects (Lucas & Kanade 1981).

Thus, it was important to introduce a good method to model object deformations without compromising robustness. Even though a variety of works were proposed, the most successful one, which is today assumed as standard, is the Point Distribution Model, presented by Cootes et al. (1992). PDMs (see Chapter 2), model shapes by applying Principal Component Analysis to a set of annotated training shapes. Each of the eigenvectors corresponding to the covariance matrix of the zero-mean training shapes represent, in descendant order, the variance of training data. The bases of the PDM model how face shapes vary as a whole.

This simple yet effective way of modelling deformable objects was immediately introduced as the main component of Active Shape Models, or ASMs (Cootes & Taylor 1992, Cootes et al. 1995). Typically, ASMs are said to be the first method aiming to fit a PDM onto a new face, so that the points composing the PDM match the points used to model the target object. Basically, ASMs perform a local search, followed by a global optimisation. The local search finds, for each point, the best potential location within a neighbourhood of its current estimate, without regard to any other point. The local search is performed independently for each point, and is constrained to all the potential locations along a line segment, perpendicular to the boundary upon which the point is meant to be (e.g., the contour of the face, or the eye corners). Each possible location along the perpendicular is the centre of a line with fixed length, that is compared to

the template trained for the target point. The template is used to measure the likelihood of each potential location to be the correct one, and is defined as the mean and covariance of the lines corresponding to that point in the training set, taken at the centres defined by the ground-truth locations. The mathematical way of finding the most suitable location is through the Mahalanobis distance. The mean and covariance of the appearance are taken as the line *experts*.

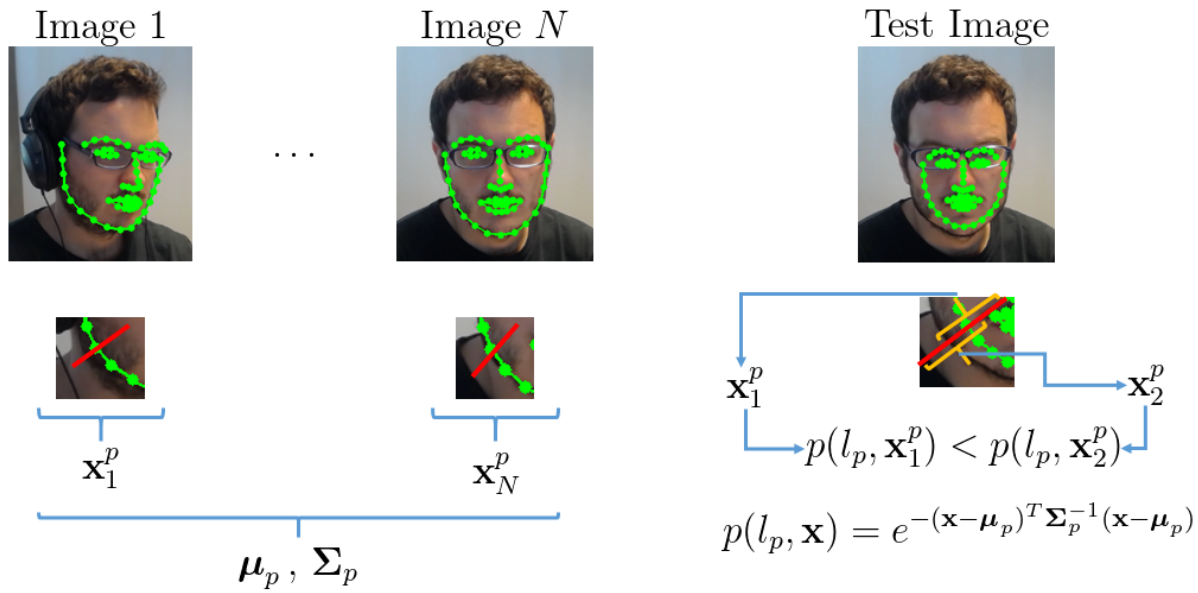

Figure 3.1: **Left:** The training of a patch expert for point p, done by collecting the pixels along the normal surface of target point (red line), and computing the average and covariance of the normals for the training set ($\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$). **Right:** The local search consists of sampling along different candidates on the normal surface to find which candidate yields the minimum Mahalanobis distance. As we can see, the second candidate yields a higher probability of being correctly aligned, and thus we select its centre.

Now, turning to a formal description, we can illustrate the ASMs local search as follows: given a current estimate or guess of where a point might be, we look along the normal of its boundary for the region that is the most similar to the template (also known as patch, or line, expert). That is to say, for a given point, we select a set of "candidates" along the normal line, and then compute the probability of each of them to be better or worse than our actual estimation. The probability is given by the Mahalanobis distance. The Mahalanobis distance assigns to a given line a "distance" to the centre of the Gaussian parameterised by the computed mean and covariance. The lower the Mahalanobis distance is, the more likely the point is to be correctly aligned. The centre corresponding to the maximum likelihood is selected to be the candidate

for each target point. Figure 3.1 illustrates the ASMs training and fitting.

Once the local search has been done, we need to "constrain" the points so that these represent an object belonging to the class of objects we are aiming to fit (faces in our case). This *global optimisation* prevents outliers, and corrects those points that are not well fitted after the local search is completed. This is where the Point Distribution Model plays its role. We want to find the shape parameters that best approximate the points found by the local search. Here, the reader might think that a simple projection of the PDM back (to obtain the shape parameters) and forth (to reconstruct the shape), as shown in Chapter 2, would be enough. However, this operation does not guarantee that the reconstructed points are the closest to the originally found points. Thus, an optimisation process is needed, aiming to locate the shape parameters for which the reconstructed shape is the closest to the found points. The "closeness" measure between the reconstructed shape and the located points in the ASMs is simply the L2 distance. This global optimisation will be further analysed in Section 3.1.4. After global optimisation, a new shape is given, and the process is repeated until convergence.

ASMs, and the use of a PDM, was a breakthrough in research on deformable objects in general, and faces in particular. Of course, this new approach was still far from being optimal, regarding accuracy and complexity. Apart from being slow, Active Shape Models were prone to local minima. We can immediately see that the local search would easily get stuck into any boundary, so convergence would be only guaranteed in the case that the initial points were close enough to their respective true boundaries. In this sense, we can see that initialisation plays a crucial role: the farther the initialisation with respect to the actual ground-truth the worse. If a point is closer to other boundaries, then the fit is prone to be poor.

3.1.2 Active Appearance Models

In order to overcome this flaw, Cootes et al. (2001) introduced a novel component to enhance the modelling of faces: the Appearance Model (see Chapter 2 for a description of what an Appearance Model is). Instead of modelling local patches along normal lines, the Appearance Models used a holistic representation of the faces appearance. Similar to how the PDM can

model shape deformations, the Appearance Model was introduced to reconstruct the texture of a given face using a model trained over an available training set. Instead of performing a local search, the Appearance Model can be used to see how far an instance of the model looks like a given texture. Furthermore, the Appearance Model allows modelling appearances with a few dozen parameters, rather than the thousands of pixels that lie within the normalised reference frame. Also, Cootes et al. (2001) proposed an upper level of modelling faces with a joint Shape and Appearance model. The joint model (coined Active Appearance Model) is built by extracting the ground-truth shape and appearance parameters on the training set, and then performing PCA again over the concatenation of both. The AAM then models both appearance and shape with a few parameters. The fitting of an AAM to a given image then consists of finding the AAM parameters, namely $\mathbf{a}$, that best describe the face. Once $\mathbf{a}$ is obtained, computing the shape parameters, and therefore the point locations, is straightforward. Contrary to the ASMs, the fitting process does not alternate between a local search and a global optimisation. Naturally, we want to find $\mathbf{a}$ so that the shape and appearance generated by the model resemble the target face. In order to do so, we have to vary $\mathbf{a}$ (i.e., the shape location, and the appearance reconstruction), until we find the parameters that best describe the target face. In order to do so, from a given instance $\mathbf{a}$, we aim to find $\delta\mathbf{a}$ so that the new object is "closer" to the target face. This is an optimisation process, and the cost function to be minimised is a *residual*, which measures how "far" a current estimation is from the parameters that describe both these points and the appearance within them. In other words, the residual measures how different an instance of the model generated with the current parameters $\mathbf{a}$ is with respect to the appearance lying under the current shape estimation.

Then, we need to find the $\delta\mathbf{a}$ that minimises the residual. Mathematically speaking, for a given initial shape and appearance estimate, the residual $\mathbf{r}$ is defined as the difference between the appearance generated by the model $\mathbf{x}_m$ and the texture captured within the generated shape $\mathbf{x}_s$, i.e., $\mathbf{r}(\mathbf{a}) = \mathbf{x}_s - \mathbf{x}_m$. The residual is computed in the reference frame, given that $\mathbf{x}_m$ is the model's appearance, and $\mathbf{x}_s$ is computed after warping the input image, and the points defined by the current estimation of the shape, to the reference frame (see Chapter 2). If we apply a first-order Taylor expansion of the residual we can express it linearly with respect to

the displacement $\delta\mathbf{a}$:

$$\mathbf{r}(\mathbf{a}+\delta\mathbf{a}) \approx \mathbf{r}(\mathbf{a}) + \frac{\partial \mathbf{r}}{\partial \mathbf{a}}\delta\mathbf{a}. \tag{3.1}$$

This way, we can "deform" both the appearance of the model and the points of the current shape, so that the model appearance $\mathbf{x}_m$ is closer to the input appearance $\mathbf{x}_s$, and the residual distance is minimised. The residual, as defined in Equation 3.1, is a vector, and therefore it is not a good choice for minimisation. To find a proper displacement, we can then apply a *Gradient Descent* method, in which $\delta\mathbf{a}$ is chosen so as to minimise a scalar measure of the residual, computed at $\mathbf{a}+\delta\mathbf{a}$. Such a scalar would be defined as $E(\mathbf{a}+\delta\mathbf{a}) = \mathbf{r}^T\mathbf{r} = \|\mathbf{r}(\mathbf{a}) + \frac{\partial \mathbf{r}}{\partial \mathbf{a}}\delta\mathbf{a}\|_2^2$. Thus, our goal is to find $\delta\mathbf{a}$ so that $E(\mathbf{a}+\delta\mathbf{a})$ is minimised. The solution for $\delta\mathbf{a}$ is given by

$$\delta\mathbf{a} = -\left(\frac{\partial \mathbf{r}}{\partial \mathbf{a}}^T \frac{\partial \mathbf{r}}{\partial \mathbf{a}}\right)^{-1} \frac{\partial \mathbf{r}}{\partial \mathbf{a}}^T \mathbf{r}(\mathbf{a}). \tag{3.2}$$

Once $\delta\mathbf{a}$ is obtained, we have to update the current parameters as $\mathbf{a} \leftarrow \mathbf{a}+\delta\mathbf{a}$, and repeat the process until the residual no longer decreases. Under this setting, Equation 3.2 requires computing the Jacobian of the residual with respect to the model parameters at each iteration of the optimisation process, which is quite expensive. In order to avoid this computationally expensive operation, Cootes et al. (2001) assumed that the Jacobian barely changes when computed within the reference frame, and hence proposed to obtain an average Jacobian matrix, which is calculated only once. This matrix is obtained by numerical differentiation. A set of random perturbations is applied to the ground truth data, and the average difference is computed over the training set. For a training set consisting of N images, the Jacobian of the residual with respect to parameter a_i is computed as:

$$\frac{\partial \mathbf{r}}{\partial a_i} = \sum_{j=1}^{N} \frac{\mathbf{r}(a^*_{i,j} + \delta a_{i,j}) - \mathbf{r}(a^*_{i,j})}{\delta a_{i,j}}. \tag{3.3}$$

The pseudoinverse of the Jacobian is then used to build a final regressor as $\mathbf{R} = \left(\frac{\partial \mathbf{r}}{\partial \mathbf{a}}^T \frac{\partial \mathbf{r}}{\partial \mathbf{a}}\right)^{-1} \frac{\partial \mathbf{r}}{\partial \mathbf{a}}^T$. This way, the minimisation process (the fitting), at iteration k is just defined as $\mathbf{a}_k = \mathbf{a}_{k-1} + \mathbf{R}\mathbf{r}(\mathbf{a}_{k-1})$, where $\mathbf{R}$ is fixed. An example illustrating the fitting of AAMs is shown in Figure 3.2. The AAMs became the state-of-the-art method for face alignment for several years, and are

still a basis for many recent works (Tzimiropoulos & Pantic (2014), Tzimiropoulos (2015)). This novel way of describing faces, and how the regressor bypasses the need to compute the Jacobian, made AAMs a seminal work in face alignment research. Built upon the AAMs work, many extensions appeared. In particular, Cootes & Taylor (2004) proposed several extensions to the original fitting of AAMs. We can see that the regressor $\mathbf{R}$ can be learnt, e.g., using linear regression. We will review some of them in Section 3.1.5.

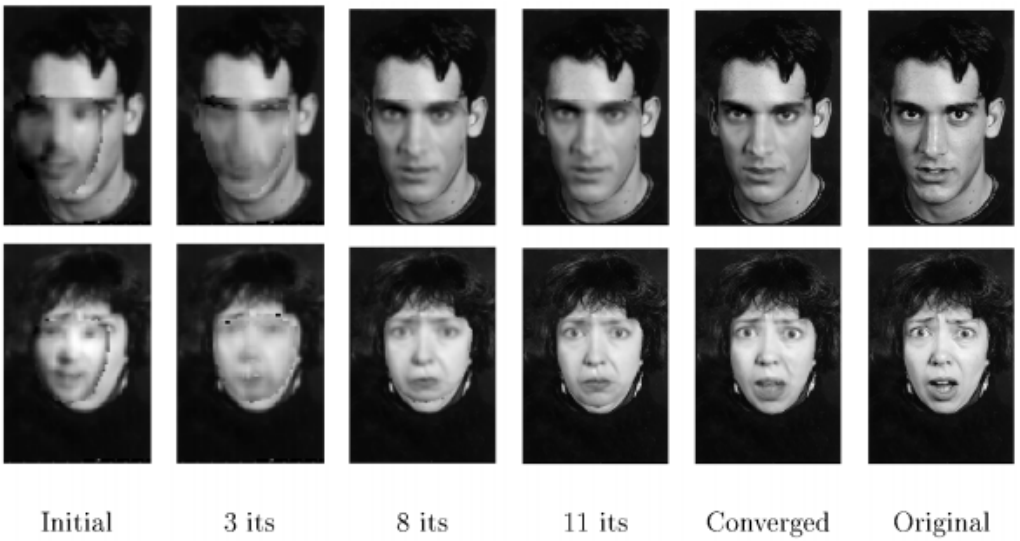

Figure 3.2: The process of fitting an AAM to an image. The shape is deformed so that an instance of the appearance model matches as much as possible to the input image. This image has been taken from Cootes et al. (2001)

3.1.3 The Inverse Compositional framework

One of the main assumptions that was key to the success of Active Appearance Models was the linearisation of the residual with respect to the model parameters in the reference frame, as well as the use of a fixed Jacobian, which is approximated through numerical differentiation, on the training set. This avoids the Jacobian being computed at each iteration, meaning a huge computational advantage. However, Matthews & Baker (2004) argued that this approximation yields suboptimal displacements. We have seen in Chapter 2 that the appearance of the input face in the reference frame is computed through a *warping* function (Figure 2.3, Chapter 2). A warping function defines a unique correspondence between the current image and the reference frame, once $\mathbf{s}$ and $\mathbf{p}$ are defined. Suppose now we are given a face image, and an initial shape

configuration (not the ground-truth). If we scale the image, and apply the same scaling to the shape, we would have the same warped image, although the real "distance" between the initial shape and the ground-truth is different in both scenarios. Since AAMs use a fixed Jacobian, and given that both warped images would be the same, we can see that the update would be exactly the same in both scenarios. However, in the second scenario we would expect a bigger, or lower, displacement, depending on whether the scaling factor is greater or lower than 1. It is then obvious that in many cases the displacements would not be optimal. That is to say, the fitting of AAMs is highly non-linear, and the assumption of a fixed Jacobian in the reference frame is highly suboptimal.

In order to overcome this limitation, Matthews & Baker (2004) proposed a novel framework based on Lucas-Kanade (Lucas & Kanade 1981) template matching, called *Inverse Compositional*, in which some novel components were introduced. First, Matthews & Baker (2004) pointed out that the use of a coupled AAM model has several disadvantages, and thus proposed to model appearance and shape using separate bases. Second, they showed that the appearance can be "projected-out" from the reconstruction error. Third, rather than linearising the image with respect to the shape parameters' displacement, they proposed to linearise the model instance (the template). Finally, they proposed the use of an *inverse compositional* update, in which rather than updating the shape parameters in a linear way, the entire warp is updated by composing the warp of the current image with the inverse of the warped image computed at the incremental update.

Mathematically speaking, Matthews & Baker (2004) proposed the following cost function, in which the optimisation cost is formulated with respect to both the shape and appearance parameters as:

$$\sum_{\Omega(\mathbf{s}_0)} [\underbrace{\mathbf{x}_0 + \mathbf{B}_a\boldsymbol{\lambda}}_{\mathbf{x}_m} - \mathbf{x}_s\left(\mathcal{W}(\mathbf{s}_0, \mathbf{p})\right)]^2 = \| \underbrace{\mathbf{x}_0 + \mathbf{B}_a\boldsymbol{\lambda}}_{\mathbf{x}_m} - \mathbf{x}_s\left(\mathcal{W}(\mathbf{s}_0, \mathbf{p})\right) \|_2^2, \tag{3.4}$$

where the latter expression is used as a convenient abuse of notation. We can recall from Chapter 2 that the appearance model instance $\mathbf{x}_m$ is the texture that lies within the convex hull of the reference shape $\mathbf{s}_0$. Equation 3.4 can be minimised directly with respect to both

$\boldsymbol{\lambda}$ and $\mathbf{p}$, which would lead to the *Simultaneous Inverse Compositional* algorithm (applying first the linearisation and then the project-out, see below). However, an algorithm tasked with updating both the appearance and shape parameters simultaneously is really slow. To bypass the need to update the appearance parameters at each iteration, Matthews & Baker (2004) noticed that the appearance can be *projected out* from the minimisation process. We can see that

$$\|\mathbf{x}_0 + \mathbf{B}_a\boldsymbol{\lambda} - \mathbf{x}_s\left(\mathcal{W}(\mathbf{s}_0, \mathbf{p})\right)\|_2^2 = \quad \|\mathbf{x}_0 + \mathbf{B}_a\boldsymbol{\lambda} - \mathbf{x}_s\left(\mathcal{W}(\mathbf{s}_0, \mathbf{p})\right)\|^2_{\mathbf{E}-\mathbf{B}_a\mathbf{B}_a^T+\mathbf{B}_a\mathbf{B}_a^T} = \tag{3.5}$$

$$\|\mathbf{x}_0 + \mathbf{B}_a\boldsymbol{\lambda} - \mathbf{x}_s\left(\mathcal{W}(\mathbf{s}_0, \mathbf{p})\right)\|^2_{\mathbf{B}_a\mathbf{B}_a^T} + \quad \|\mathbf{x}_0 + \mathbf{B}_a\boldsymbol{\lambda} - \mathbf{x}_s\left(\mathcal{W}(\mathbf{s}_0, \mathbf{p})\right)\|^2_{\mathbf{E}-\mathbf{B}_a\mathbf{B}_a^T} = \tag{3.6}$$

$$\|\mathbf{x}_0 + \mathbf{B}_a\boldsymbol{\lambda} - \mathbf{x}_s\left(\mathcal{W}(\mathbf{s}_0, \mathbf{p})\right)\|^2_{\mathbf{B}_a\mathbf{B}_a^T} + \quad \|\mathbf{x}_0 - \mathbf{x}_s\left(\mathcal{W}(\mathbf{s}_0, \mathbf{p})\right)\|^2_{\mathbf{E}-\mathbf{B}_a\mathbf{B}_a^T}. \tag{3.7}$$

The first term in Equation 3.6 corresponds to the reconstruction error in the subspace spanned by $\mathbf{B}_a$, whereas the second corresponds to the orthogonal complement of that subspace. Moreover, when working in the orthogonal complement, the appearance variation vanishes thanks to the fact that $\mathbf{B}_a$ is column-wise orthogonal. The minimum of the first term is always zero; i.e., once the optimal $\mathbf{p}$ is found, $\boldsymbol{\lambda}$ can be computed as $\boldsymbol{\lambda} = \mathbf{B}_a^T(\mathbf{x}_s\left(\mathcal{W}(\mathbf{s}_0, \mathbf{p}) - \mathbf{x}_0\right)$. This way, only the second term of Equation 3.7 needs to be iteratively minimised; this speeds up the process. Now, as shown before, Matthews & Baker (2004) proposed to overcome the problem of a fixed Jacobian by linearising the template, instead of the input image. The main idea behind this approach is to "deform" the template to be closer to the input image, and then compute the increment back in the image frame (hence "inverse"). When linearising the template, the cost function is formulated as:

$$\|\mathbf{x}_0\left(\mathcal{W}(\mathbf{s}_0, \delta\mathbf{p})\right) - \mathbf{x}_s\left(\mathcal{W}(\mathbf{s}_0, \mathbf{p})\right)\|^2_{\mathbf{E}-\mathbf{B}_a\mathbf{B}_a^T}, \tag{3.8}$$

w.r.t. $\delta\mathbf{p}$. In order to solve Equation 3.8, we can expand $\mathbf{x}_0\left(\mathcal{W}(\mathbf{s}_0, \delta\mathbf{p})\right)$ by its first-order Taylor expansion: $\mathbf{x}_0\left(\mathcal{W}(\mathbf{s}_0, \delta\mathbf{p})\right) \approx \mathbf{x}_0\left(\mathcal{W}(\mathbf{s}_0, \mathbf{0})\right) + \frac{\partial \mathbf{x}_0}{\partial \mathbf{p}}\delta\mathbf{p}$, where $\mathbf{x}_0\left(\mathcal{W}(\mathbf{s}_0, \mathbf{0})\right) = \mathbf{x}_0$ is the identity warp. Given the holistic nature of Active Appearance Models, the algorithm needs to compute the derivatives of the warping function and the *Steepest Descent* images (the projection of the

derivative of the mean appearance w.r.t. the derivative of the warping function), which will be used to compute the Jacobian and Hessian matrices, fixed for the Project-Out algorithm. For the sake of clarity, we omit the derivation here. The update can be computed as:

$$\delta \mathbf{p} = \mathbf{H}_{\mathbf{P}}^{-1} \mathbf{J}_{\mathbf{P}}^{T} \left(\mathbf{x}_s(\mathcal{W}(\mathbf{s}_0; \mathbf{p})) - \mathbf{x}_0 \right), \tag{3.9}$$

where $\mathbf{J}_{\mathbf{P}} = \left(\mathbf{E} - \mathbf{B}_a \mathbf{B}_a^T \right) \frac{\partial \mathbf{x}_0}{\partial \mathbf{s}} \frac{\partial \mathcal{W}}{\partial \mathbf{p}}$, and $\mathbf{H}_{\mathbf{P}} = \mathbf{J}_{\mathbf{P}}^T \mathbf{J}_{\mathbf{P}}$. Here, $\mathbf{P} = \mathbf{E} - \mathbf{B}_a \mathbf{B}_a^T$ is used to denote that the Jacobian and the Hessian are "projected out" from the reconstruction error. Finally, the nature of Equation 3.9 was used to propose the inverse compositional update. Now, instead of updating the model parameters as $\mathbf{p} \leftarrow \mathbf{p} + \delta \mathbf{p}$, the update is given as:

$$\mathcal{W}(\mathbf{s}, \mathbf{p}) \leftarrow \mathcal{W}(\mathbf{s}, \mathbf{p}) \circ \mathcal{W}(\mathbf{s}, \delta \mathbf{p})^{-1}. \tag{3.10}$$

The algorithm that minimises Equation 3.8, using the update rule shown in Equation 3.10 was coined *Project-Out Inverse Compositional*, and resulted in an efficient method to fitting AAMs. Summarising, the key contribution of Project-Out Inverse Compositional (PO-IC) were the linearisation of the template rather than the image, which allows the use of fixed *Descent Directions*, whilst avoiding the need to compute the Jacobian of the warp with respect to the shape parameters at each iteration. This is possible because the template remains unchanged, and no ambiguity can arise.

3.1.4 Constrained Local Models

One of the key aspects of both the original AAMs fitting approach and the Inverse Compositional framework is that they rely on an Appearance Model and a reconstruction error. That is to say, in both cases, the alignment error is based on how much the appearance of a given face matches the given instance of the appearance model. Thereby, the less the input face looks like the training faces, the less likely the alignment will be accurate. For instance, let us assume that the training faces do not contain glasses at all, and we want to fit the model to a face of someone wearing glasses. In such cases, the fitting is prone to failure. As we have previously

seen, we cast the AAM as a **generative** method, and one of the major drawbacks they suffer from is a lack of generalisation. We can refer to that problem in the past tense, mainly because the problem of generalisation was due to the lack of annotated data. By 2004, there were few annotated databases, and thus their generalisation power was poor. Today, we can find many databases that have been collected in the wild, and have been properly annotated, following the distribution shown herein, which has been recently assumed as standard. As we shall see, the recent availability of annotated databases has rekindled interest in generative methods.

With the problem of generalisation being caused by the lack of data, different alternatives emerged. We have previously seen that there are alternatives to generative methods, which are cast as either **discriminative** or **statistical** methods, although many works have categorised both as simply discriminative. Herein, discriminative methods are referred to as those that directly infer the shape parameters from the input features, with no associated cost function, whereas probabilistic methods are those that try to infer the locations for which a maximum likelihood function is maximised, i.e., those having an actual (probabilistic) cost function. The latter are also known as *Constrained Local Models* (CLMs), and will be addressed in this Section. The reader might however notice that many generative and discriminative methods have an alternative probabilistic formulation. For instance, the Least-Squares error associated with the reconstruction error shown above can be formulated as the log-likelihood of a Gaussian distribution, in which case the error captures how likely it is that the predicted shape corresponds to the correct location. In probabilistic terms, this is equivalent to finding the locations that maximise the probability of them to be the correct positions. Given this ambiguity, many authors cast the CLMs as simply discriminative methods. Herein, the term CLM will simply refer to a set of methods in which a local search is followed by a global optimisation, formulated by means of a probabilistic function. The way that either the local search or the global optimisation are performed differ for different works, although in all cases the methodology can be seen to follow this pattern. The reader might already be familiar with these methods. Effectively, the ASMs shown above can be seen as a particular case of CLMs. We will shortly see this connection. Mathematically speaking, the CLMs optimisation problem can be seen as:

$$\mathbf{p}^* = \arg\max_{\mathbf{p}} \left[p(\mathbf{s}(\mathbf{p})|\{l_i\}_{i=1}^n, \mathbf{I}) \propto p(\mathbf{s}(\mathbf{p})) \prod_{i=1}^{n} p(l_i|\mathbf{I}) \right], \tag{3.11}$$

where l_i represents the event of landmark i being correctly aligned. We can see that the term on the left is the probability of finding the landmarks in $\mathbf{s}(\mathbf{p})$, given the alignment correctness (that is to say, given that they have been correctly predicted, or aligned), and the image $\mathbf{I}$. The second term stems from applying the Bayes' rule to the left term, and from the fact that the likelihood of each landmark to be correctly aligned, given only its local appearance, is independent from the other landmarks. The optimisation procedure is repeated iteratively until convergence, and alternates between the local search and the global optimisation. An example is illustrated in Figure 3.3.

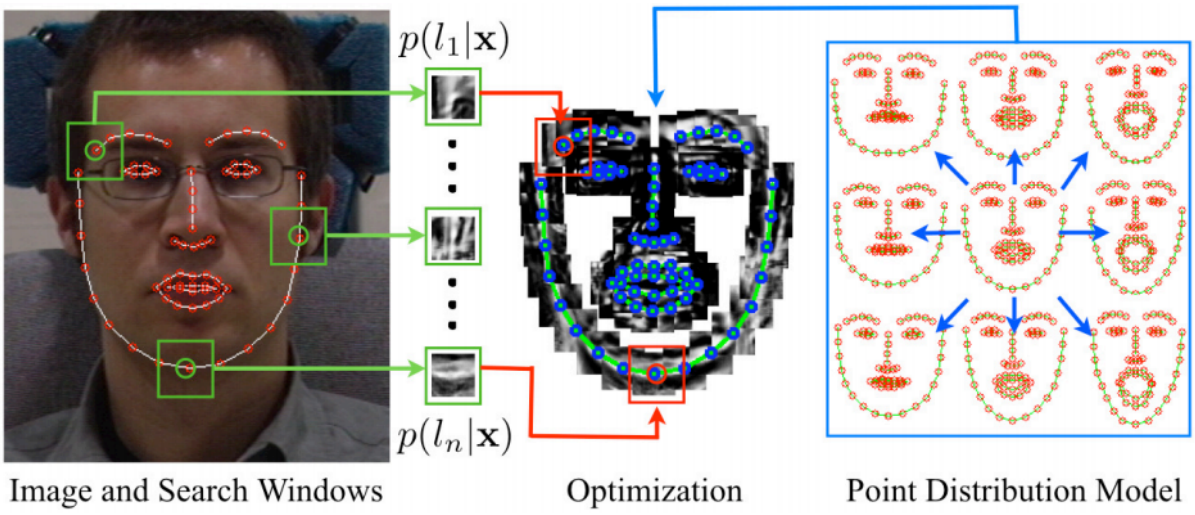

Figure 3.3: This image has been taken from Saragih et al. (2011), and illustrates the main CLMs steps. First of all, a local search returns the probability of each of the landmarks to be correctly aligned. Then, we have to find the PDM instance that best fits the located points.

Let us assume that the current shape is given by $\mathbf{s}_c(\mathbf{p})$. The local search generates a set of *response maps* surrounding each of the points in $\mathbf{s}_c = \{\mathbf{s}_c^i\}_{i=1}^n$. A response map is basically a visual representation of a likelihood map, indicating for each location the probability of being correctly located. The maps are simply generated using a template, which is filtered with the image features. In other words, if we are given a template picture of the appearance surrounding a point, with it being located in the centre, we "move", or slide, the template, pixel by pixel, within a neighbourhood of the current location of the point (bigger than the

template size), and then we measure how likely each of these locations is to be the correct one for our template. The response maps are the collection of these measures. The peaks of the response maps are, a priori, the locations that are most likely to be correct. Let $\boldsymbol{\mu}^i = (x_i, y_i)$ be the maximum, or the peak, of the response map for point i. Once all the maxima are taken, we need to perform a global optimisation, in which we aim to find the shape parameters that best describe the found locations. That is to say, we need to find the best displacement $\delta\mathbf{p}$ so that Equation 3.11 is maximised. As pointed out in Saragih et al. (2011), this operation can be carried out by expanding the current shape by its first order Taylor expansion and then minimising a weighted Least-Squares error against each $\boldsymbol{\mu}^i$ w.r.t. $\delta\mathbf{p}$. Mathematically speaking, we can see that $\mathbf{s}_c(\mathbf{p} + \delta\mathbf{p}) \approx \mathbf{s}_c + \frac{\partial \mathbf{s}}{\partial \mathbf{p}}\delta\mathbf{p}$. Then[1], we want to minimise the Least-Squares error against $\boldsymbol{\mu}^i$, given by

$$\sum_{i=1}^{n} \|\mathbf{s}_c^i - \boldsymbol{\mu}^i\|^2_{\mathbf{W}_i}. \tag{3.12}$$

Saragih et al. (2011) pointed out that this optimisation process can be seen from the point of view of a regularised cost function. If we take the negative of the log-likelihood of the right side of Equation 3.11, we can see that it is transformed as:

$$\mathbf{p}^* = \arg\min_{\mathbf{p}} \left[\log p(\mathbf{s}(\mathbf{p})) + \sum_{i=1}^{n} \log p(l_i|\mathbf{I}) \right] = \arg\min_{\mathbf{p}} \left[\mathcal{R}(\mathbf{p}) + \sum_{i=1}^{n} \mathcal{C}_i(\mathbf{x}_s^i, \mathbf{I}) \right] \tag{3.13}$$

This way, and as shown above, different cost functions can have a probabilistic interpretation. Saragih et al. (2011) unified the CLM framework using the formula above, showing the link that exists between the cost function and its corresponding probabilistic point of view. Now, we can see that the ASMs shown above are a particular case of this framework, in which the local search is done through the normal line instead of on a patch surrounding a point. Moreover, in ASMs, the way we "annotate" each location, i.e., the way we generate the response maps, is through the Mahalanobis distance. In general, a sigmoid function, or any other tool, can be used for that purpose. The global optimisation is equivalent to asking what is the probability that, once the peak corresponding to point i is given at $\boldsymbol{\mu}^i$, the *real* location is at that point, or at any of its neighbours. The answer in ASMs is that the probability of each of the points

[1]The computation of the Jacobian of the shape with respect to the PDM parameters will be addressed in Chapter 5

to be the real one is given by an isotropic Gaussian distribution. That is to say, Saragih et al. (2011) linked ASMs with the cost function in Equation 3.12, in which $\mathbf{W}_{\text{ASM}} = \mathbf{E}$. From the probabilistic point of view, this is equivalent to assuming that $p(l_i)$ is given by an isotropic Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}^i, \mathbf{E})$. Saragih et al. (2011) pointed out that other existing works could also be cast into the CLM framework, in which each would assume that the final response maps would be approximated by different functions. In all methods, the probabilistic point of view would assume $\mathbf{W}$ as the inverse of the covariance matrix.

Under this framework, different approaches can be followed to estimate the response maps, or obtain the posterior likelihood that each landmark is well aligned. For example, Cristinacce & Cootes (2008) replaced the use of the Mahalanobis distance to generate the response maps by the correlation between an instance of an appearance model and the local patches taken surrounding each point. Indeed, the term Constrained Local Models was first coined by Cristinacce & Cootes (2008). In Saragih et al. (2011), the response maps are computed by using a sigmoid function over the output of a classifier, which gives a score accounting for how well each point is fitted. Other works, such as Zhou et al. (2005), Wang et al. (2008), in which a Convex Quadratic Function (CQF) is fitted to the negative of the log of the response map, can be seen as approximating $\mathbf{W}$ with an anisotropic Gaussian, in which the covariance matrix is given by a Maximum Likelihood solution.

Of course, we can use more than one response map, or landmark detector, per point. For example, Gu & Kanade (2008) used a Gaussian Mixture Model (GMM) to estimate the global response maps, and Saragih et al. (2011) showed that this would be equivalent to replacing the cost function by $\sum_{i=1}^{n} \sum_{k=1}^{K_i} \|\mathbf{s}_c^i - \boldsymbol{\mu}_k^i\|^2_{\mathbf{W}_{ik}}$, where K_i is the number of maps for point i. In this line, Saragih & Göcke. (2009), Saragih et al. (2011) proposed what is probably the most successful CLM method to date, in which the response maps are substituted by a non-parametric probability distribution, which is estimated by a Kernel Density Estimation method. In their proposed CLM, they do not use a single peak estimation for the response maps, but rather select a grid of candidates, from which one is selected through a Mean-Shift algorithm. This way, more candidates are taken into account, without decreasing the speed of the search. Moreover, a non-parametric estimation of the response maps would handle occlusions better than using a

fixed set of estimators, such as those used in previous works. The KDE-CLM framework was shown to be fast, and the authors' implementation is among the most downloaded methods for face tracking, mainly thanks to its speed. Figure 3.4, taken from Saragih et al. (2011), illustrates the mentioned interpretations for the response maps.

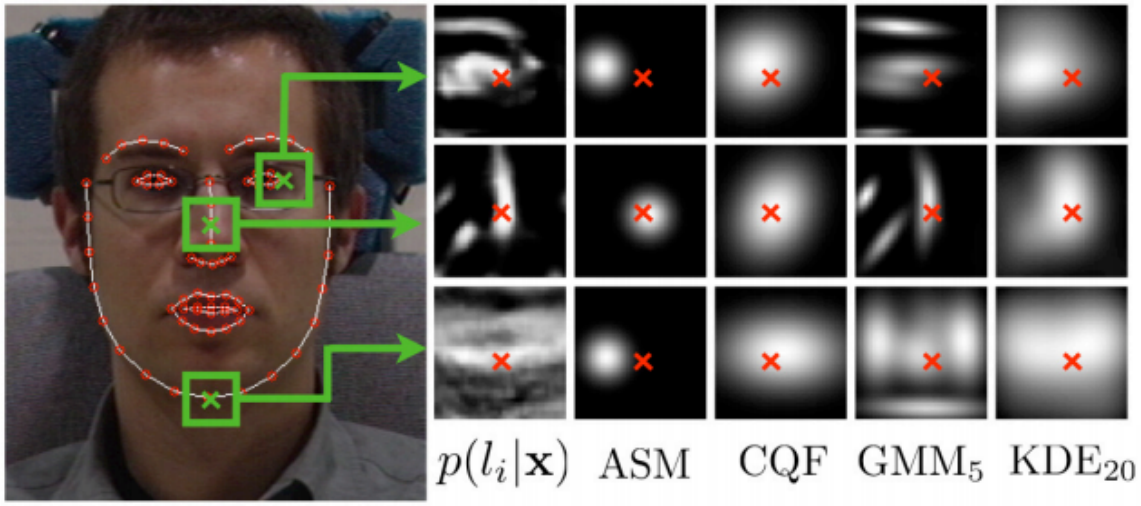

Figure 3.4: This image has been taken from Saragih et al. (2011), and illustrates the different ways of approximating the response maps.

Some extensions have also been proposed upon the CLM framework. Worth mentioning are those proposed by Martins et al. (2012, 2014, 2016), who proposed a novel Bayesian approach, and who replaced the local detectors by MOSSE filters (Bolme et al. 2010), which have proven to work fast and to generate accurate responses. Moreover, Belhumeur et al. (2011, 2013) presented a non-parametric approach that could be used instead of a PDM.

3.1.5 Regression Methods

The appearance of CLMs coincides in time with the increasing interest and research in **discriminative** methods. Sometimes, the term discriminative includes, among others, the work of Liu (2007, 2009), who proposed to replace the use of appearance models, as presented in Cootes et al. (2001), by Boosted Appearance Models (BAMs). The key feature of a BAM is the use of Haar features (Viola & Jones 2004), which were successfully used for the task of face detection. Haar features are just simply binary features that tell whether the sum of pixels

in a specific area is bigger or lower than the sum of pixels in, e.g., the opposite area. Haar features are computed over different sized patches, and might have different configurations. That is to say, there might be even millions of Haar features that can be extracted for a single image. The use of Adaboost allows to select the most discriminative ones for a given task. In the BAMs framework, the main idea of using Adaboost (Friedman 2001) is to select the set of Haar features that best discriminate correctly aligned shapes from those that are not. More specifically, the BAMs warp images to the mean shape, from both the ground truth points and perturbed shapes, and then use the Haar features and Adaboost to learn a model that, for a given image and a shape, returns a score accounting for how well the shape is aligned w.r.t. the image. However, we can see that the fitting process is very similar to the fitting of AAMs, in which now the displacement is selected to optimise the model's score. That is to say, BAMs are still a class of generative methods, and do not completely overcome the problems that are typically attached to these methods. Probably, the fact that Adaboost selects the most discriminative features for a given classification task has led many authors to wrongly cast the BAMs as discriminative methods.

We have to recall then the definition of discriminative methods shown before. Briefly speaking, discriminative methods try to predict the shape displacements from the input features. That is to say, no error or cost function is (a priori) minimised. One of the simplest discriminative methods found in the literature is based on the use of Linear Regression. Cootes & Taylor (2004) have shown that it is possible to bypass the use of an appearance model by predicting shape displacements from features, through a linear mapping. In their report, the shape parameters' displacement was formulated as $\delta\mathbf{p} = \mathbf{R}\mathbf{x}_s = \mathbf{R}f(\mathbf{I}, \mathbf{p})$. Apparently, this is the same update rule used for AAMs. However, the prediction now does not consider the residual, but rather the features, only. Moreover, the regressor is not learnt by numerical differentiation. Instead, the regressor is learnt using a Least-Squares formulation, by perturbing the ground-truth shapes and minimising the average loss:

$$\mathcal{L}_{lin}(\delta\mathbf{p}, \mathbf{R}) = \|\delta\mathbf{p} - \mathbf{R}f(\mathbf{I}, \mathbf{p}^* + \delta\mathbf{p})\|_2^2. \tag{3.14}$$

Now, the update rule is again linear: $\mathbf{p} \leftarrow \mathbf{p} + \delta\mathbf{p}$, and the feature vector is computed using a part-based representation of the face, as shown in Chapter 2 (Tresadern et al. 2010). The use of a linear regressor to directly infer the shape parameters' displacement results in a very fast algorithm, as only sampling and matrix multiplications are required (Tresadern et al. 2012).

However, a single linear regression, used in an iterative manner, has found to be a poor choice, as it would need to account for too much input variance. It is known that, the bigger the variance of the chosen $\delta\mathbf{p}$ in Equation 3.14, the lower the robustness of the regressor $\mathbf{R}$. However, when a single regressor is iteratively used we need it to consider the input shapes, as well as all possible shapes that might result after applying the regressor. That is to say, when a single regressor is used, we need it to cover all the possible trails that would lead to the original inputs to converge to a point near to the ground truth. In order for it to do so, we need to reinforce it. Oversimplifying, if we train the regressor using a set of perturbations generated from the ground truth, then we need to incorporate the possible outputs that result after applying the regressor to the training data, then train with all the new shapes, and so forth, until the training error no longer decreases. This sort of regressor reinforcement can be addressed with the use of *Gradient Boosting* (Friedman 2001, Duffy & Helmbold 2002). Gradient boosting for regression adapts the idea of AdaBoost (Friedman 2001), to the regression task (Duffy & Helmbold 2002). More specifically, let us assume we are given a training set $\mathcal{S} = \{(\mathbf{I}_1, \mathbf{s}_1^*), \ldots, (\mathbf{I}_N, \mathbf{s}_N^*)\}$[2]. We can generate a set of perturbations, e.g., running the face detector on each of the training images and extend the training set with those shapes that would initialise the fitting on them: $\mathcal{S}_{\text{ext}} = \{(\mathbf{I}_1, \mathbf{s}_1^*, \mathbf{s}_1^0), \ldots, (\mathbf{I}_N, \mathbf{s}_N^*, \mathbf{s}_N^0)\}$, where $\mathbf{s}_j^0$ represents the initial shape for image j. Furthermore, we can even *extend* the training set by generating a set of K random initialisations per image. This process is known as *data augmentation*. Then, our training set will become

$$\mathcal{S}_{\text{aug}} = \{(\mathbf{I}_1, \mathbf{s}_1^*, \{\mathbf{s}_1^{0,k}\}_{k=1\ldots K}), \ldots, (\mathbf{I}_N, \mathbf{s}_N^*, \{\mathbf{s}_N^{0,k}\}_{k=1\ldots K})\} \tag{3.15}$$

[2]Please, note that we can here indistinctly use $\mathbf{s}$ or $\mathbf{p}$ as there is a unique correspondence between both the shapes and the parameters

We can then build a regressor targeted with estimating the displacement $\delta\mathbf{s}_i^k = \mathbf{s}_i^* - \mathbf{s}_i^{0,k}$. Let $\mathcal{R}^0$ be a regression function (not necessarily a linear regressor), that receives the image features as an input, and returns the displacement. We can choose $\mathcal{R}^0$ to be:

$$\mathcal{R}^0 = \arg\min_{\mathcal{R}} \sum_{i=1}^{N}\sum_{k=1}^{K} \mathcal{L}(\delta\mathbf{s}_i^k, \mathcal{R}(\mathbf{I}_i, \mathbf{s}_i^{0,k})), \tag{3.16}$$

where $\mathcal{L}$ is a loss function of our choice, which measures the correctness of the prediction given by the regressor $\mathcal{R}$ with respect to the real measure $\delta\mathbf{s}_i^k$. Once $\mathcal{R}^0$ is chosen, we can generate a new training set, by applying to all $\mathbf{s}_i^{0,k}$ the learnt regressor. This will give us a new training set $\hat{\mathcal{S}} = \{(\mathbf{I}_1, \mathbf{s}_1^*, \{\mathbf{s}_1^{1,k}\}_{k=1\ldots K}), \ldots, (\mathbf{I}_N, \mathbf{s}_N^*, \{\mathbf{s}_N^{1,k}\}_{k=1\ldots K})\}$, where $\mathbf{s}_i^{1,k} = \mathbf{s}_i^{0,k} + \mathcal{R}(\mathbf{I}_i, \mathbf{s}_i^{0,k})$. In a boosting regression scenario, this new set is then incorporated to the original one (now the *master regressor*) as

$$\mathcal{R} = \mathcal{R}^0 + \alpha^1\mathcal{R}^1 \tag{3.17}$$

where α^1 and $\mathcal{R}^1$ are computed as:

$$(\alpha^1, \mathcal{R}^1) = \arg\min_{\alpha,\mathcal{R}} \sum_{i=1}^{N}\sum_{k=1}^{K} \mathcal{L}(\delta\mathbf{s}_i^k - \mathcal{R}^0(\mathbf{I}, \mathbf{s}_i^{0,k}), \alpha\mathcal{R}(\mathbf{I}_i, \mathbf{s}_i^{1,k})). \tag{3.18}$$

This way, we can add as many regressors as we might want, until the training error no longer decreases. Each of the separate regressors are known as *weak regressors*. Note that the fitting applies the master regressor $\mathcal{R}$ repeatedly until convergence. That is to say, given a test image $\mathbf{I}$ and initial shape $\mathbf{s}$, the fitting consists of updating $\mathbf{s}$ as

$$\mathbf{s} \leftarrow \mathbf{s} + \mathcal{R}(\mathbf{I}, \mathbf{s}). \tag{3.19}$$

This operation is repeated until convergence. The choice of the regressor $\mathcal{R}$ will differ from each work. The simplest choice is the aforementioned Linear Regression (Tresadern et al. 2010), shown in Equation 3.14.

Some other regressors have been proposed for that task as well, to name few: the *random forests* (Cootes et al. 2012), or *support vector regression* (Valstar et al. 2010, Martinez et al.

2013). Of course, the formulation above is a general framework, and each work has its own methodology. In Cootes et al. (2012), the random forests' decisions are based on a grid search in the manner of the CLMs described above. The predictions of the random forests are combined in an additive manner, as shown above. Moreover, Martinez et al. (2013) *aggregates* the single decisions of possible candidates using a mixture of Gaussians. Apart from that, both Valstar et al. (2010) and Martinez et al. (2013) modelled the spatial relations using a probabilistic framework. Instead of optimising the log-likelihood shown in Section 3.1.4, they used a Markov Random Field (Bishop 2006). Similarly, Baltrušaitis et al. (2014) presented a Continuous Conditional Neural Field to solve the regression problem, formulated in terms of a probabilistic graphical model.

3.1.6 Cascaded Regression

Arguably, all the methods shown above are optimisation problems, and therefore their performance relies a great deal on how these are initialised. Most of the methods shown above do not consider the initialisation process during the training stage. That is to say, neither the AAMs, nor the CLMs, include the way shapes would be initialised, at test time, during the training process. Also, early works using linear regression were trained using random perturbation from the ground-truth, and thus were not appropriate for the task of detecting landmarks from the face bounding box, especially for non-frontal faces. This is again because the way the shape would be initialised was not considered during training. Moreover, many of the methods shown above were utilising perturbations of the ground-truth data to assess the accuracy, or were tested in databases containing near-frontal faces only. Therefore, most of the methods shown above were not capable of dealing with big displacements such those that appear when the initialisation is an average shape and the target face is near profile. We have seen however a general framework in which we can *boost* a general regressor to account for a bigger variance space, using a set of weak regressors that jointly contribute to a master regressor. However, even though boosting regression has proven to increase generalisation with respect to previous works, it is still prone to fail when dealing with near to profile poses or occlusions. All these

methods are likely to fail when fitting non frontal faces. This clearly illustrates the limits that a single regressor can reach. Also, even when reinforcing a regressor, there is a limit to the variance that a regressor can handle without compromising accuracy. Look at Figure 3.5. The left image depicts an example in which the training variance is lower, whereas the right image depicts a training scenario in which a bigger variance is considered. We can see that the first scenario shows better accuracy for the test sample (in orange), than the right scenario. This example serves roughly to explain the problem of the variance when training a linear regressor.

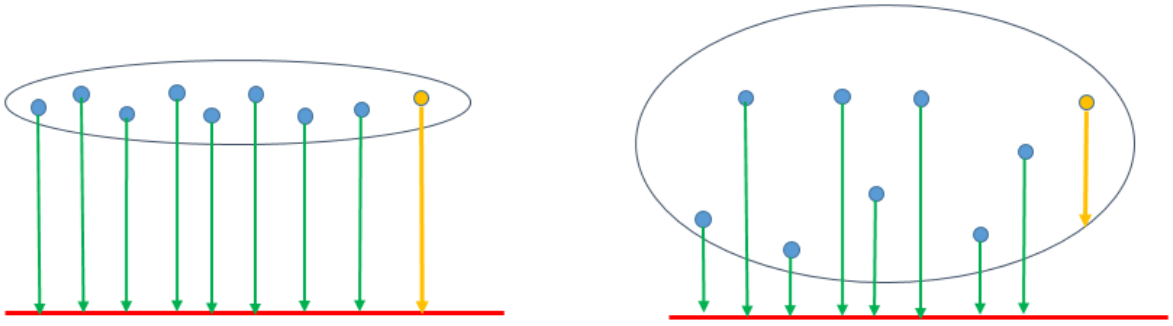

Figure 3.5: Example illustrating the problem of variance in Linear Regression. Blue dots are training examples, and green lines correspond to displacements towards the ground-truth, depicted by the red line. In a convenient abuse of terminology, we see a linear regressor in this one dimensional example as the average training displacement. The orange dot is a test sample, for which a prediction is made, resulting in the orange arrow. In both cases the test sample is the same. However, in the first scenario, with lower training variance, the regressor is capable of accurately predict the displacement, while in the second scenario, the regressor is far from being optimal for the given test sample.

One might consider either one of two directions, or both, to alleviate this problem. One is reducing the variance of the input features; the other is reducing the variance of the regressors to be trained. The first approach was proposed for classification problems by Fleuret & Geman (2008), in which a new type of features, called *weakly invariant*, were proposed. The use of weakly invariant features was exploited for regression tasks by Dollár et al. (2010). Basically, such features assume certain invariance given a specific pose parameter. For instance, assume that we are given a face image for which its ground truth points have been rotated with a specific angle θ_1. Now, let us assume that the whole image is rotated with an angle θ_2, and that the rotated points are rotated accordingly. In both cases, we want to recover the same rotation angle θ_1. However, we can see that, in general, we cannot guarantee that the features extracted in the same part of the face will be equal for both images, given that they are

"different". This means that the variance of the input is increased for a fixed deformation θ_1. Given this affine transformation of the same image, we expect the extracted features to be the same. Such features are then weakly invariant. Formally, let θ_1, θ_2 be two configurations of pose parameters (e.g., rigid parameters). Let θ_1 be the pose used to transform the input image $\mathbf{I}$ to $\mathbf{I}_1(\theta_1) = \mathbf{I}(\mathcal{W}(\theta_1))$, and θ_2 to be a relative pose parameter with respect to $\mathbf{I}_1$ (note the convenient abuse of notation). If we have a feature extraction function $f(\mathbf{I}_1(\theta_1), \theta_2)$, we say that it is weakly invariant if, for any $\delta\theta$, $f(\mathbf{I}_1(\theta_1 \circ \delta\theta), \theta_2 \circ \delta\theta) = f(\mathbf{I}_1(\theta_1), \theta_2)$, where now $\circ$ can be any operation such that $(\Theta, \circ)$, with Θ the space of pose parameters, forms a group. A simple example of weakly invariant features are the *pose-indexed* features, extracted at certain control points. In Dollár et al. (2010) each control point feature is computed as the difference of two image pixels at predefined image locations. Figure 3.6 illustrates the weak features.

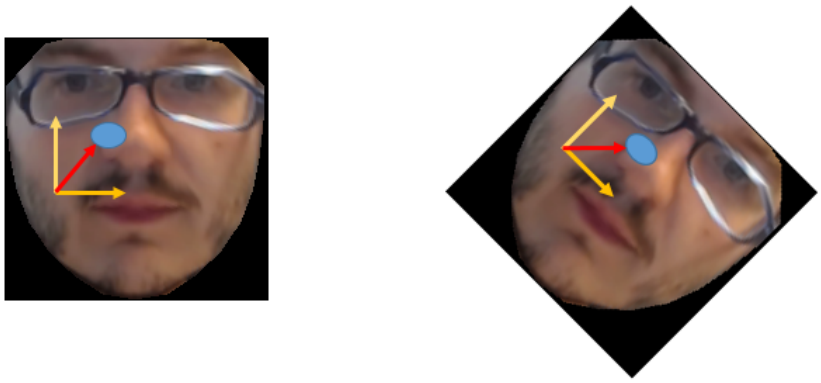

Figure 3.6: Weak invariant features. The yellow coordinates represent the rotation of the given image, and the red arrow represents the pose relative to the local coordinates (control points). The second image results after applying $\delta\theta$ to both the image and the control points, thus resulting in the same features. Thereby, we can say this kind of features to be weakly invariant.

Now, suppose we extract image features at some point $\mathbf{s}$. When using pose-indexed features, we have to "anchor" a control point, so that the features are simply intensity differences with respect to it. Our regressor gives us a shape estimation, and then we repeat the feature extraction process for the next iteration. We can keep the control point on its previous position, and then apply the same regressor again, or we can move the control point according to the shape displacement. If we apply the latter approach, we need to use a new regressor, since now the feature extraction process is defined with respect to a different reference point. That is to say, under this setting, the boosting regression approach moves into a novel framework, coined

cascaded regression (Cascaded Pose Regression in Dollár et al. (2010) to be precise [3]). The main idea is to use a set of weak learners in an additive manner, as shown above. However, in order to be able to use the pose-indexed features, one needs to extract the features each time that a new control point is to be defined. Otherwise, features would no longer meet the weakly invariant property.

This approach was used by Cao et al. (2012, 2014) to extend Cascaded Regression to the problem of face alignment. More specifically, given the original augmented training set $\mathcal{S}$ shown before, the first regressor can be learnt through Equation 3.16. Then, similar to boosting regression, we have to generate the new training set $\hat{\mathcal{S}}$, by applying our regressor to the training data. Now, the second regressor is again learnt as shown in Equation 3.18, with the only difference that now α is no longer needed. When the anchor remains unchanged we can add the resulting regressors as in Equation 3.17. However, when we move the anchor, the regressor can not be added to the previous learnt regressors. This means that now, when fitting, we no longer use the same master regressor iteratively, but rather a collection of them, which are placed in cascade. In other words, the main difference of Cascaded Regression with respect to previous methods comes now when carrying out the fitting process. Instead of *leveraging* the regressors into the master level, we have to apply them in a Cascaded fashion. The process of fitting is summarised in Algorithm 3.

Algorithm 3 Cascaded Regression

1: Input data: $\mathbf{I}$, $\mathbf{s}_0$, $\{\mathcal{R}^l\}_{l=1,\dots,L}$
2: **for** $l = 1$ to L **do**
3: Extract $f(\mathbf{I}, \mathbf{s}_{l-1})$
4: Update $\mathbf{s}_l = \mathbf{s}_{l-1} + \mathcal{R}^l(\mathbf{I}, f(\mathbf{I}, \mathbf{s}_{l-1}))$
5: **end for**
6: Return $\mathbf{s}_L$

For each Cascaded level j, Cao et al. (2012) proposed to learn an ensemble of weak classifiers, by using *random ferns* (Ozuysal et al. 2007). That is to say, Cao et al. (2012) combined a two-level cascaded approach, in which each level is generated by the boosting regression presented above, and for each level a new set of features is extracted (i.e., anchors are moved). Furthermore,

[3] It is also worth highlighting that a similar approach was used to fit AAMs Saragih & Göcke. (2007, 2009), however, as Dollár et al. (2010) mentioned, "these methods require manually defined energy functions that measure goodness of fit", and thus its complexity makes it hard to be properly implemented

Cao et al. (2012) proposed the regressor to be driven by the training set, by assigning to a specific fern the average displacement seen during the training process. This means that the output of the weak regressors are just simply a shape displacement that has been seen during training. This way, we no longer need to use a PDM, given that the shapes will always be a linear combination of training shapes. That is to say, the final output shape $\mathbf{s}_L$ will always be a linear combination of training shapes, and thus it will be implicitly constrained to lie within the subspace spanned by the training set. Therefore, Cao et al. (2012) proposed to directly infer the shape positions, without the need of using a shape model.

The fitting of Cascaded Regression shown in Algorithm 3 is a general form in which the choice of the regressor $\mathcal{R}_l$ is flexible. Moreover, even though that the use of weakly invariant features implies Boosting Regression to be transformed into the Cascaded Regression, the contrary is not necessarily required. That is to say, Cascaded Regression does not imply the use of weakly invariant features. This is because this method (the cascaded regression) intrinsically limits the variance of the input space for each regressor. Therefore, we are tackling the problem of variance without compromising robustness. This way, the Cascaded Regression framework became a breakthrough in the field of face alignment, and many methods were built upon it.

Within all methods, the most successful approach of Cascaded Regression was however derived from a set of completely different preliminaries. We are referring now to the **Supervised Descent Method** (SDM, Xiong & De la Torre (2013)). In practice, the SDM is a simple approach to Cascaded Regression, in which $\mathcal{R}_l$ is just a linear regressor, and the features used were the SIFT (Lowe 2004), which intrinsically maintain the weakly invariant property. This simple approach was shown to be a very fast method, yet highly accurate.

In the SDM, however, the goal was not to perform boosting regression, but rather to minimise a cost function. The cost function was formulated as the distance between the features extracted at the current shape positions, and the features extracted at the ground-truth. However, given that the ground-truth features are not available during testing, we need to perform an alternative approach. More specifically, the SDM was motivated by the minimisation of the

following cost function:

$$\mathcal{L}(\hat{\mathbf{s}}) = \|f(\mathbf{I}, \hat{\mathbf{s}}) - f(\mathbf{s}^*)\|_2^2 \tag{3.20}$$

where $\mathbf{s}^*$ represents the ground-truth shape, and $\hat{\mathbf{s}}$ is the estimated shape. The main idea of SDM is to solve Equation 3.20, w.r.t. $\hat{\mathbf{s}}$, applying a Newton descent method. That is to say, the Newton descent method minimises Equation 3.20 by updating $\hat{\mathbf{s}}$ as follows:

$$\hat{\mathbf{s}}_{l+1} = \hat{\mathbf{s}}_l - 2\mathbf{H}^{-1}\mathbf{J}^T(f(\hat{\mathbf{s}}_l) - f(\mathbf{s}^*)) \tag{3.21}$$

where $\mathbf{H}$ and $\mathbf{J}$ are the Hessian and Jacobian of the loss function, with respect to $\mathbf{s}$, evaluated at $\hat{\mathbf{s}}_l$. The reader may have already noticed the similiarity between Equation 3.21 and Cascaded Regression. Since we are given no access to neither the ground truth, nor the ground-truth features, we need to *learn* them. The SDM proposes then to learn the Descent Directions $-2\mathbf{H}^{-1}\mathbf{J}^T$. Basically, the SDM reformulates Equation 3.21 as

$$\hat{\mathbf{s}}_{l+1} = \hat{\mathbf{s}}_l - \mathbf{R}_l f(\hat{\mathbf{s}}_l) - \mathbf{b}_l. \tag{3.22}$$

This way, $\mathbf{R}_l = 2\mathbf{H}^{-1}\mathbf{J}^T$ is a *descent direction*. The main idea of SDM is to learn $\mathbf{R}_l$ and $\mathbf{b}_l$ from the training data (hence "*supervised*"). The process now is very similar to the Cascaded Regression shown above. We first learn $\mathbf{R}_0$ and $\mathbf{b}_0$ from the given set of initial training shapes $\mathbf{s}_i^{0,k}$, by minimising the average L2 loss function, defined as:

$$\{\mathbf{R}_0, \mathbf{b}_0\} = \arg\min_{\mathbf{R},\mathbf{b}} \sum_{i=1}^{N}\sum_{k=1}^{K} \|\mathbf{s}^* - \mathbf{s}_i^{0,k} + \mathbf{R}f(\mathbf{I}_i, \mathbf{s}_i^{0,k}) + \mathbf{b}\|_2^2. \tag{3.23}$$

Once $\{\mathbf{R}_0, \mathbf{b}_0\}$ are learnt, we can update the training shapes, and repeat the process, to generate the subsequent average descent directions. That is to say, given the training shapes $\mathbf{s}_i^{l,k} = \mathbf{s}_i^{l-1,k} - \mathbf{R}_{l-1}f(\mathbf{I}_i, \mathbf{s}_i^{l-1,k}) + \mathbf{b}_k$, we can generate the l-th descent direction as:

$$\{\mathbf{R}_l, \mathbf{b}_l\} = \arg\min_{\mathbf{R},\mathbf{b}} \sum_{i=1}^{N}\sum_{k=1}^{K} \|\mathbf{s}^* - \mathbf{s}_i^{l,k} + \mathbf{R}f(\mathbf{I}_i, \mathbf{s}_i^{l,k}) + \mathbf{b}\|_2^2. \tag{3.24}$$

The minimisation in Equation 3.24 is the well-known Least Squares problem. First of all, let us

denote $\hat{\mathbf{R}}_l = \{\mathbf{R}_l, \mathbf{b}_l\}$, in which now we consider the bias $\mathbf{b}_l$ to be appended as the last column of $\hat{\mathbf{R}}_l$. Then, if we denote $\hat{f}(\mathbf{I}_i, \mathbf{s}_i^{l,k}) = [f(\mathbf{I}_i, \mathbf{s}_i^{l,k}); 1]$, i.e., append the features $f(\mathbf{I}_i, \mathbf{s}_i^{l,k})$ with a 1, we can re-formulate Equation 3.24 in a more compact form as:

$$\hat{\mathbf{R}}_l = \arg\min_{\mathbf{R}} \sum_{i=1}^{N} \sum_{k=1}^{K} \|\mathbf{s}^* - \mathbf{s}_i^{l,k} + \mathbf{R}\hat{f}(\mathbf{I}_i, \mathbf{s}_i^{l,k})\|_2^2. \tag{3.25}$$

From now on, for the sake of clarity, the bias term will be omitted, $\hat{\mathbf{R}}_l$ will be simply referred to as $\mathbf{R}_l$, and it will be implicitly assumed that the bias term has been added to the feature vector $f(\mathbf{I}_i, \mathbf{s}_i^{l,k})$. If we collect the shape displacements $\delta\mathbf{s}_i^{l,k} = \mathbf{s}^* - \mathbf{s}_i^{l,k}$ as the column vector of a matrix $\mathbf{Y}_l$, and the corresponding features $f(\mathbf{I}_i, \mathbf{s}_i^{l,k})$ as column vectors of a matrix $\mathbf{X}_l$, the solution to Equation 3.25 is given as [4]:

$$\mathbf{R}_l = \mathbf{Y}_l \mathbf{X}_l^T \left(\mathbf{X}_l \mathbf{X}_l^T\right)^{-1}. \tag{3.26}$$

The fitting process is carried out using Algorithm 3, in which $\mathcal{R}^l(\mathbf{I}, f(\mathbf{I}, \mathbf{s}_{l-1})) = \mathbf{R}_l f(\mathbf{I}, \mathbf{s}_{l-1})$. That is to say, in practice, the Supervised Descent Method is just a class of Cascaded Regression methods. However, contrary to the derivation shown in Cao et al. (2012), Xiong & De la Torre (2013) argued that each regressor plays now the role of an average descent direction, steering the initial shape towards the local minima. Moreover, they showed that, opposed to existing discriminative methods, there is an error function linked to the Cascaded Regression (Equation 3.20), which, contrary to previous works, introduces the ground-truth data[5]. Furthermore, instead of using complex functions and raw pixels, Xiong & De la Torre (2013) showed that impressive results can be attained by simply using a set of linear regressors and SIFT features. Of course, even though the SDM was formulated in a non-parametric way, the use of a Shape Model would be straightforward. A graphical illustration is shown in Figure 3.7

The SDM immediately revolutionised the research field, and Cascaded Regression became the state-of-the-art technique for face alignment and tracking. Interestingly, despite that both

[4]Sometimes, it is also beneficial to include a regularisation term, to avoid overfitting. For the sake of clarity, it is not included in the derivations shown in this thesis. However, including a regularisation term would be straightforward

[5]We might argue that $f(\mathbf{s}^*)$ could be seen as a "template", which in the Inverse Compositional framework shown above would be defined as an instance of the appearance model

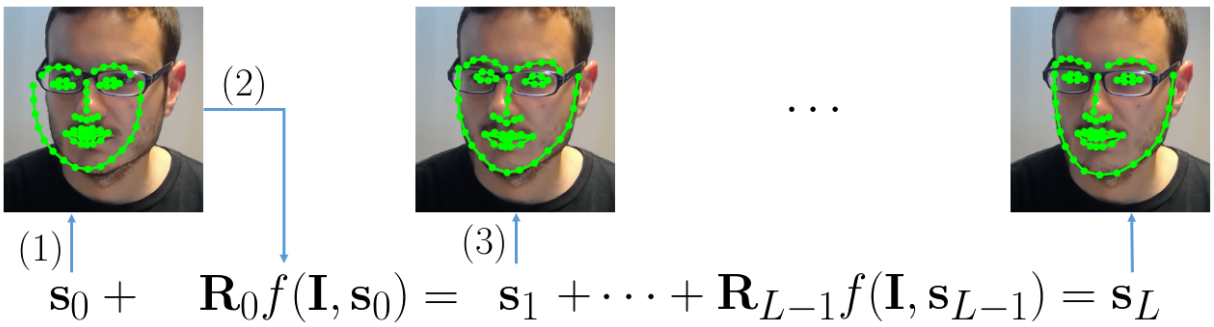

Figure 3.7: Cascaded regression fitting. First of all (1) we are given an initial shape $\mathbf{s}_0$ from the face detection bounding box. Then, the first step consists of estimating $\mathbf{s}_1$ from the features extracted in $\mathbf{s}_0$ and the regressor $\mathbf{R}_0$ learnt for the initial step. Then, this process is repeated for the set of L regressors learnt for the model.

the work of Cao et al. (2012) and Xiong & De la Torre (2013) are fairly similar, the latter attracted much more attention, perhaps thanks to the computational simplicity with which it was presented. In any case, both Cao et al. (2012) and Xiong & De la Torre (2013) have inspired the majority of recent works in the field. One of these works is that of Asthana et al. (2014), which is perhaps one of the works upon which this thesis relies the most.

We have seen that, naturally, Cascaded Regression algorithms are trained sequentially. That is to say, once we train the l-th level, we need to update the training set to generate the $l+1$-th level. Now, suppose we are given a model with L cascade levels, and that we want to add a new subset of images to it. A priori, we would need to re-train the models including all the images, given that all levels are affected by the previous levels. That is to say, do we have to go through the expensive process of training the models from scratch, just to introduce some images?

Before we answer this question, we need to further illustrate why solving this is crucial to improve Cascaded Regression. Recall that during face tracking, we are given a generic model with which we aim to track a specific person. In this setting, we know that the model would perform probably better if we could include some information of the target's face. Given that building person-specific models is not applicable in many cases, we can take an alternative approach: updating a generic model with correctly tracked faces. This problem, known as Incremental Learning, updates an existing model with new images (those corresponding to the target face). This way, the problem of "how to incorporate new images to a model without the

need to re-train the models" becomes crucial.

An alternative to the sequential SDM training was proposed by Asthana et al. (2014). Briefly speaking, Asthana et al. (2014) realised that the SDM can be trained in *parallel*, without compromising accuracy. That is to say, they proposed a *parallel-CLR* (Cascaded Linear Regression). The main idea of par-CLR is to first train a standard SDM, with a given training set, and then measure the statistics of the output of each level with respect to the ground truth.

More specifically, assume we have a trained SDM, for which we store all the training shapes $\{\mathbf{s}_i^{l,k}\}_{i=1\ldots N,k=1\ldots K}$ for each of the $l = 1 \ldots L$ cascade levels, as well as the distances of each of them with respect to their corresponding ground-truth: $\delta\mathbf{s}_i^{l,k} = \mathbf{s}_i^{l,k} - \mathbf{s}_i^*$. The main idea of par-CLR is that we can train each of the cascade levels *independently*, assuming the $\delta\mathbf{s}_i^{l,k}$ to be generated by a Gaussian distribution with mean $\boldsymbol{\mu}^l$ and covariance matrix $\boldsymbol{\Sigma}^l$, computed from the given $\delta\mathbf{s}_i^{l,k}$. That is to say, if we first train a sequential SDM, and then we measure the statistics for each of the $\delta\mathbf{s}_i^{l,k}$ remaining for each level, and train a new model in which each cascade level is trained by simply generating perturbations for $\delta\mathbf{s}_i^{l,k}$ (that is to say, without computing the outputs of the previous level), we would reach a model that is equivalent in performance to that trained sequentially. The reader might wonder why would we want to train a new model that is equivalent the original SDM, given that first we have to train the sequential-SDM to measure the statistics. The answer is quite simple: we can use these statistics to train new models, *with additional training shapes*. Given a sufficiently large original training set, we can assume that $\boldsymbol{\mu}^l$ and $\boldsymbol{\Sigma}^l$ generalise well to the observed $\delta\mathbf{s}_i^{k,l}$, and therefore we can generate new displacements for each level, assuming that the original statistics would not change if we were to train a model including all images.

The benefits of the parallel-CLR are clear: 1) it allows the training of new models with a different number of images taking advantage of parallel computing, given that we can train each level independently, and 2) it allows images to be added to trained models without the need of re-training them from scratch. These benefits open the possibility of incremental learning under the Cascaded Regression framework. Also, in order to perform incremental learning, we can apply the well-known recursive least-squares approach, which allows the models to be

updated without explicitly computing Equation 3.26.

Assume we are given a regressor $\mathbf{R}_l$, and an image $\mathbf{I}_n$, for which we know the ground-truth $\mathbf{s}_n^*$. First of all, for level l, we can generate the perturbations, given that we have seen that they follow a Gaussian distribution parameterised by $\boldsymbol{\mu}^l$ and $\boldsymbol{\Sigma}^l$. This way, we generate a set of random perturbations $\delta\mathbf{s}_n^{l,k}$, which are used to extract the features $f(\mathbf{I}_n, \mathbf{s}_n^* + \delta\mathbf{s}_n^{l,k})$. Let $\mathbf{X}_{(n)}$ be the matrix storing all $\mathbf{x}_n^{l,k}$, and $\mathbf{Y}_{(n)}$ be the matrix storing the corresponding $\delta\mathbf{s}_n^{l,k}$. Then, a regressor considering $\hat{\mathbf{X}}_l = [\mathbf{X}_l\, \mathbf{X}_{(n)}]$ and $\hat{\mathbf{Y}}_l = [\mathbf{Y}_l\, \mathbf{Y}_{(n)}]$, would be computed as $\hat{\mathbf{R}}_l = \hat{\mathbf{Y}}_l\hat{\mathbf{X}}_l^T\left(\hat{\mathbf{X}}_l\hat{\mathbf{X}}_l^T\right)^{-1}$. Now, we can apply the Woodbury identity (Brookes 2011) to the covariance matrix. Let us denote the covariance matrix of the training samples as $\mathbf{V} = \left(\mathbf{X}_l\mathbf{X}_l^T\right)^{-1}$, and the updated covariance matrix as $\hat{\mathbf{V}} = \left(\hat{\mathbf{X}}_l\hat{\mathbf{X}}_l^T\right)^{-1}$. We can see that

$$\mathbf{U} = (\mathbf{E} + \mathbf{X}_{(n)}^T\mathbf{V}\mathbf{X}_{(n)})^{-1} \tag{3.27}$$

$$\mathbf{Q} = \mathbf{X}_{(n)}\mathbf{U}\mathbf{X}_{(n)}^T\mathbf{V} \tag{3.28}$$

$$\hat{\mathbf{V}} = \mathbf{V} - \mathbf{V}\mathbf{Q} \tag{3.29}$$

$$\hat{\mathbf{R}}_l = \mathbf{R}_l - \mathbf{R}_l\mathbf{Q} + \mathbf{Y}_{(n)}\mathbf{X}_{(n)}^T\hat{\mathbf{V}}. \tag{3.30}$$

This way, it is possible to compute $\hat{\mathbf{R}}_l$ as a function of simply $\mathbf{R}_l$. Asthana et al. (2014) pointed out that the inversion in Equation 3.27 is extremely efficient, and therefore the incremental learning becomes feasible. In fact, Asthana et al. (2014) pointed out that, when updating a single image only, the dimensionality of $\mathbf{U}$ reduces to one, and therefore the inversion is just a division, which is extremely efficient. However, in Asthana et al. (2014) the whole tracking process is reported to be > 4 seconds, including the fitting and the incremental learning. Since the fitting runs in real-time, the main bottleneck resides in the incremental learning rules. What Asthana et al. (2014) did not realise is that the main bottleneck does not reside in computing the inverse, but rather in computing $\mathbf{VQ}$ in Equation 3.29. We can see that if we re-arrange the incremental learning computations, the complexity can be reduced by one order of magnitude, with respect to the dimensionality of the feature vector (the highest overload!). We will later analyse the computational complexity of this method, and we will see that re-arranging the computations there is a more efficient way to perform it.

In any case, the incremental parallel-CLR proposed by Asthana et al. (2014) allowed for the first time the possibility of incorporating new frames into the models whilst tracking is ongoing. In the tracking setting, the image $\mathbf{I}_n$ is the face image from a given frame, and $\mathbf{s}_n^*$ is the estimated shape for that frame, assuming it to be accurately located.

The Cascaded Regression approach is today considered the state-of-the-art method in face alignment and tracking. We have seen that it can be derived as the natural extension of Boosting Regression to weakly invariant features, as well as from the Newton Descent method. However, we could even think of Cascaded Regression as a coarse-to-fine search strategy (Zhang et al. (2014), Zhu et al. (2015)), in which an initial estimate is sequentially upgraded to be closer to the ground truth. Some recent works apply this concept, using different regressors or features. In this line, a cascaded regression with local binary features was shown to work at more than 1000 fps, making it the fastest method to date (Ren et al. 2014).

Also, it is worth mentioning that cascaded regression has also proven to be a good approach for generative methods as well. In fact, the PO-IC algorithm shown above can be formulated using Cascaded Regression. The Project Out Cascaded Regression (POCR) (Tzimiropoulos 2015), replaces the use of Steepest Descent images computed in the reference frame, by a Cascaded Regression, used to compute the Jacobian and Hessian in Equation 3.9. We can see that, if we expand Equation 3.8 with respect to $\delta\mathbf{p}$ we would have the following cost function:

$$\|\mathbf{x}_s(\mathcal{W}(\mathbf{s}_0, \mathbf{p})) - \mathbf{J}_{\mathcal{W}}\delta\mathbf{p} - \mathbf{x}_0\|^2_{\mathbf{P}}, \tag{3.31}$$

where $\mathbf{J}_{\mathcal{W}} = \frac{\partial \mathbf{x}_0}{\partial \mathbf{s}} \frac{\partial \mathcal{W}}{\partial \mathbf{p}}$. The POCR can then be formulated in a cascaded fashion, in which each descent direction is learnt as follows: first, an average $\hat{\mathbf{J}} = \mathbf{P}\mathbf{J}_{\mathcal{W}}$ is computed through minimising the Least Squares error, shown in Equation 3.31, for the training set and an initial set of displacements $\delta\mathbf{p}$ (i.e., following the cascaded approach shown above). Then, the Hessian is given by $\hat{\mathbf{H}} = \hat{\mathbf{J}}^T\hat{\mathbf{J}}$, and the descent direction is given by $\mathbf{R} = \hat{\mathbf{H}}^{-1}\hat{\mathbf{J}}$. This descent direction is utilised to update the training set, given that $\delta\mathbf{p} = \mathbf{R}(\mathbf{x}_s(\mathcal{W}(\mathbf{s}_0, \mathbf{p})) - \mathbf{x}_0)$. This solution using generative methods, along with newer databases collected in-the-wild, has shown promising results, and it is today classed among the state of the art methods. Importantly, the POCR

uses a generative method for a patch-based representation of the face, as it allows for a linear update of the models.

3.1.7 Towards the first benchmark in face tracking

The impressive results shown by Xiong & De la Torre (2013) and Asthana et al. (2014) have recently impulsed the field of face tracking, which was at that point barely exploited, mainly because of the difficulty of making existing works work in real-time and the lack of publicly annotated data. By 2013, most of the methods presented for face tracking were simply adaptations of face alignment methods, such as those shown above. As we have previously seen, the only difference appeared to be the initialisation. In the case of generative methods, this does not affect the training process. However, in the case of regression methods, in which the initialisations are included into the training process, the methods can not be directly extended to the face tracking problem. The recent interest in the field of tracking was displayed in the 300 Videos in the Wild Challenge (300VW Shen et al. (2015)), which was the first (and only one to date) contest on face tracking. The challenge collected 50 videos for training and 64 for testing, which were split into three categories, each of increasing difficulty. The videos are approximately 1 minute long, and were annotated using a semi-automated methodology (Shen et al. 2015, Tzimiropoulos 2015, Tzimiropoulos & Pantic 2014), and released afterwards for research purposes. All the video frames were annotated using a 68 points mark-up (see below). Some of the videos were however containing some frames in which faces were beyond profile, and therefore were excluded from the challenge evaluation. The tracking system developed as a result of the research conducted in this thesis is tested in the 300VW database. Different methods were submitted to the 300VW challenge, in which two methods (Xiao et al. 2015, Yang et al. 2015) clearly outperformed other contestants. Interestingly, both methods were based on Cascased Regression algorithms. Xiao et al. (2015) utilised a multi-stage regression, in which certain key points, such as eyes and mouth corners, help the initialisation of upcoming frames. Given that initialisation plays an important role in face tracking, Xiao et al. (2015) focused on reinforcing the initialisation of upcoming frames based on certain strongly semantical po-

sitions, which appear to be easier to detect. In Yang et al. (2015) a spatio-temporal cascade shape regression is used, using a multi-view regression approach to reduce the variance of input regressors. As we shall later see, the implementation developed for this thesis outperforms both methods, evaluated under the same conditions.

3.1.8 Databases

Apart from the only benchmark that exists to date for Face Tracking, many databases containing static images have appeared since the appearance of Active Appearance Models. This section briefly summarises the databases that are used in this thesis.

Helen (Le et al. 2012) : The Helen database was constructed using images from Flickr, which were manually annotated using the Amazon Mechanical Turk. The annotations are made for a set of 192 facial points, which follow the distribution shown in the example shown in Figure 3.8 (upper row, left corner). The dataset consists of a training set of 2000 images, and a test set composed of 330 images.

LFPW (Belhumeur et al. 2011) : The LFPW (Labelled Faces Parts in the Wild) database was collected from Flickr, Google, and Yahoo. Originally, the database was released using only the URLs to the images, and a set of files containing a set of 19 points per image. Some of these URLs were no longer available. However, the remaining images were finally stored, and can now be found within the 300W dataset (see below). The dataset is divided into a training set of 811 images, and a test set of 224 testing images. An example can be seen in Figure 3.8 (upper row, centre)

Multi-PIE (Gross et al. 2010) : The Multi-PIE (Pose, Identity and Expression) dataset is a lab-environmentally collected database, aiming to model different expressions, illuminations, poses and identities. The dataset consists of more than 750,000 images of 337 people recorded in up to four sessions over the span of five months. For each session, subjects were recorded under

15 view points and 19 illumination conditions while displaying a range of facial expressions. In addition, high resolution frontal images were acquired as well. The annotation protocol follows the 66 points shown in Figure 2.2, along with two more points, located in the inner corners of the mouth. An example is depicted in Figure 3.8 (lower row, left corner).

AFW (Zhu & Ramanan 2012) : The AFW (Annotated Faces in the Wild) is a database consisting of 205 images, containing 468 faces, downloaded from Flickr. In the AFW, faces contain a wide range of face poses. All faces are annotated with the head pose angle, along with 6 key points: the outer eye's corners, the nose tip, the mouth corners, and the mouth centre. An example is shown in Figure 3.8 (upper row, right corner).

iBug (Sagonas et al. 2013*b*) : The iBug dataset is a collection of 135 images, following the 68 points mark-up of Multi-PIE, that aims to introduce a set of training images under uncontrolled conditions. The iBug dataset, despite being the smallest, has been reported to be the most challenging one. An example is shown in Figure 3.8 (lower row, centre).

300W (Sagonas et al. 2013*a*) : The 300W (300 labelled faces in the wild) is a dataset made of 600 images, annotated following the 68 points mark-up of Multi-PIE. The dataset is divided into two parts, corresponding to either indoor or outdoor scenes. The dataset was originally used as a benchmark for the first (Sagonas et al. 2013*a*) and second (Sagonas et al. 2016) editions of the 300W challenge, aiming for participants to test their face alignment methods under the same conditions. The database was released afterwards. Figure 3.8 shows an example (lower row, right corner).

300W re-annotations (Sagonas et al. 2013*a*) : All the databases shown above follow their own point distribution, thus making hard to check different methods in all of them. With the aim of unifying all databases towards a single protocol, and under the context of the 300W challenge, Sagonas et al. (2013*a*) released all the annotations of Helen, LFPW and AFW databases, following the 68 points mark-up of Multi-PIE. That is to say, the 300W annotations

unify Helen, LFPW and AFW with the 68 points configuration of Multi-PIE, iBug, and 300W datasets.

Figure 3.8: Images with annotations from different databases. From left to right, top to bottom: Helen, LFPW, AFW, Multi-PIE, Ibug, 300W.

3.2 Review of Functional Regression

This thesis builds upon linear regression, which is the standard learning tool for the Supervised Descent Method. Through this Chapter, linear regression has been referred to as the learning method for cascaded regression approaches. Basically, when using linear regression, we are referring to a specific method, in which images are *sampled* in a perturbed space, in order to, concisely speaking, "correct" the perturbation that has been applied to the image. However, linear regression is also a widely used method in statistics, in which the model is used to *explain* observed variables, given a set of responses. Given a set of responses and observations, linear regression aims to model the relation that exists between them, using the assumption the relationship is linear. Linear regression is typically used in economic research to model the relation that exists between product prices and the amount of sales.

However, in such scenarios, linear regression can model observed data, but can not make any predictions, or model incomplete data. Linear regression can fail to model the latent structure of data. The limitations of modelling observed data are widely studied within the field of Functional Data Analysis (Ramsay & Silverman 1997). Basically, Functional Data Analysis (FDA) is a branch of statistics that aims to model realisations of stochastic processes as continuous functions (Ramsay & Silverman 1997). Similar to the relation that exists between product prices and the amount of sales, we can see that when we record an Electroencephalography (EEG), measures are taken within a given frequency, and thus observations are discrete, while the EEG signal is instead continuous . Thus, FDA, which attempts to model observations assuming they belong to a continuum, is of wide interest for data modelling. The data to be modelled is called *functional data*.

Concisely, FDA assumes that observations, and/or responses, are outcomes of continuous processes (Morris 2015). Typically, functional data lies on time series (*longitudinal data*), although it has been recently extended to spatial and imaging domains, among others. The key idea of FDA is the parameterisation of functions by means of structured objects, or basis functions, rather than a set of samples. These bases are the building blocks of FDA, and FDA methods try to fit these functions to the observed data. Several basis functions have been proposed for different domains. Worth mentioning are the Radial Basis Functions (RBF) (Quak et al. 1993), B-splines (Marx & Eilers 1999), or Fourier Series (Ratliffe et al. 2002). Briefly speaking, Radial Basis Function (RBF) is a function in which the output varies according to the distance from the evaluation point to a given centre, i.e., any function of the form $f(\|\mathbf{x} - \mathbf{x}_i\|)$, where $\mathbf{x}_i$ is a given centre, is considered an RBF function. B-splines are just piecewise polynomial functions, and Fourier Series approximate any function by the sum of sinusoids, which are modulated in frequency by the harmonics of the function to be approximated.

A typical use of Functional Data Analysis is in the field of Regression, namely *Functional Regression*, where either the responses or the observed data, or both, are modelled as continuous functions. In other words, we may say that Functional Regression can be seen as an extension of Multivariate Polynomial Regression to the continuous domain, in which complex structures are used to fit the observed data.

As an example, Quak et al. (1993) used the Least Squares approximation to estimate which function underlies observed data. Using RBF, the CLS problem is iteratively solved by applying a partition of the space in an iterative manner, so that different regions are covered by their centres. Typically, bases are used to approximate the observed data, although, in some cases, it is also possible to model the responses on basis spaces too, and some elegant solutions have been proposed in closed-form (Morris 2015). However, in practice, it is not computationally feasible to approximate the observed outcomes of an image (i.e., the image features), by means of RBF or Fourier Series.

3.2.1 Functional Regression in Computer Vision

It is worth noting that, despite being a widely studied field in statistics, FDA has been barely used in Computer Vision, mainly because the intractability of the data by means of Fourier series or B-splines.

The work that can perhaps be linked most closely to the research conducted in this thesis (from the Functional Regression perspective, of course), is the extension of Principal Component Analysis (PCA) to the convex set of linear combinations of training images (Levin & Shashua 2002). More specifically, Levin & Shashua (2002) introduced the continuous domain into the PCA framework, making it more robust to illumination changes.

Mathematically speaking, we can see that PCA requires computing the covariance of the training data: the bases $\mathbf{B}_a$ are the eigenvectors of the covariance matrix, sorted so that the corresponding eigenvalues go in descendant order. Let $\{\hat{\mathbf{x}}_i\}_{i=1...N}$ be the zero-mean training samples, and $\mathbf{X} \in \mathbb{R}^{D \times N}$ the matrix storing them. The sampled covariance is defined as:

$$Cov = \sum_{i=1}^{N} \hat{\mathbf{x}}_i \hat{\mathbf{x}}_i^T = \mathbf{X}\mathbf{X}^T. \tag{3.32}$$

The main contribution of (Levin & Shashua 2002) is to extend the covariance matrix to the whole set of convex combinations of the training samples $\hat{\mathbf{x}}_i$, so that any linear combination of the sampled data (including the sampled data itself) is included in the covariance matrix as

well. The sampled covariance matrix then transforms into a functional covariance matrix. Let W be the polytope defined by the convex combinations of $\hat{\mathbf{x}}_i$, then, the functional covariance matrix can be written as:

$$Cov(W) = \frac{1}{V}\int_{\hat{\mathbf{x}}\in W} \hat{\mathbf{x}}\hat{\mathbf{x}}^T d\hat{\mathbf{x}}, \tag{3.33}$$

where V is the volume of the polytope defined by W. The inverse of the volume outside the integral indicates that a uniform density function when sampling the points $\hat{\mathbf{x}} \in W$ is assumed. Let $\mathcal{D}_N = \{\alpha = (\alpha_1, \ldots, \alpha_N) | \sum_i \alpha_i = 1, \alpha_i \geq 0\}$ be the manifold of combinations. We can see that $\mathbf{X}\alpha \in W, \forall \alpha \in \mathcal{D}_N$. This way, Equation 3.33 can be written as:

$$Cov(W) = \frac{1}{V(\mathcal{D}_N)}\mathbf{X}\left[\int_{\alpha\in\mathcal{D}_N} \alpha\alpha^T d\alpha\right]\mathbf{X}^T. \tag{3.34}$$

The matrix $\mathbf{A} = \int_{\alpha\in\mathcal{D}_N} \alpha\alpha^T d\alpha$ does not depend on the input data, and has a closed-form solution. Therefore, the functional covariance matrix can be represented by means of a $N \times N$ matrix as (derivation can be found in Levin & Shashua (2002)):

$$Cov(W) = \frac{1}{N(N+1)}\mathbf{X}(\mathbf{E} + \mathbf{1}_N\mathbf{1}_N^T)\mathbf{X}^T, \tag{3.35}$$

where $\mathbf{1}_N$ is a vector whose N elements are all equal to 1. Thus, a closed-form solution is given for a potential infinite set of convex combinations of a given data consisting of N samples. Moreover, the solution can be further processed to transform the eigendecomposition problem into a $N \ll D$ dimensional problem, which implies a fixed computational complexity for a fixed number of basis. Results proved this extension to be effective, yet computationally simple.

We will see in Chapter 5 that the contributions of this thesis can be linked to the work of Levin & Shashua (2002). More specifically, we will see that the Least-Squares problem can be formulated by means of covariance matrices, and that the proposed approach can be used to compute the functional covariance matrix. The main difference of the functional covariance of Levin & Shashua (2002), and the one presented in this thesis, is that the approach above extends the training data to the possible set of combinations, whereas in this thesis we need to compute the covariance of the training data in the perturbed image space. In other words,

the functional covariance matrix of Levin & Shashua (2002) does not account for perturbed images, but rather for combinations of training examples, whereas in this thesis we need to generate perturbations of the training data, and the Continuous Regression presented in this thesis bypasses the problem of generating samplings by applying a first-order Taylor expansion of the input features. That is to say, despite that both the work of Levin & Shashua (2002) and that presented in this thesis present a solution for the functional covariance matrix, the work of Levin & Shashua (2002) solves it in the given training images (i.e., the ground-truth), and the work of this thesis solves the functional covariance matrix in the space of perturbations. Moreover, we will see how the volume in the integral relates to the probability density function. The use of a uniform distribution in Levin & Shashua (2002) is based on the fact that this way the limits are bounded, and the integral can be solved. For a different sampling distribution, the solution is not straightforward. In this thesis, the Continuous Regression is solved for general sampling distributions in Chapter 5.

Apart from the work of (Levin & Shashua 2002), it is worth highlighting the Continuous Generalised Procrustes Analysis framework introduced in Igual & De la Torre (2010), Igual et al. (2014). We have previously seen (Chapter 2) that one of the process that is key to construct a Shape Model is first normalise the training shapes, removing rigid information from them[6]. The process of normalisation is done using Generalised Procrustes Analysis (GPA, Goodall (1991)), in which shapes are iteratively registered, whilst an average shape is also generated. Mathematically speaking, the GPA optimises the following cost function:

$$\min_{\mathbf{s}_0,\{\mathbf{T}_i\}_{i=1,\dots,N}} \sum_{i=1}^{N} \|\mathbf{s}_i - \mathbf{T}_i\mathbf{s}_0\|^2, \tag{3.36}$$

where $\mathbf{s}_i$ is each of the N training shapes, $\mathbf{s}_0$ is the mean shape, and $\mathbf{T}_i = \left[\begin{smallmatrix} s_i\cos\theta_i & s_i\sin\theta_i & t_{x_i} \\ -s_i\sin\theta_i & s_i\cos\theta_i & t_{y_i} \end{smallmatrix}\right]$ is the transformation matrix, for each of the training shapes. That is to say, Equation 3.36 jointly minimises $\mathbf{s}_0$ and the transformations $\mathbf{T}_i$.

The Continuous Generalised Procrustes Analysis then extends the Least-Squares problem as-

[6]Strictly speaking, we can apply PCA on the shapes without normalising them beforehand. However, this would result in the first modes to account for rigid variation, although somehow mixed with certain non-rigid information, which results in highly impractical models.

suming that the rigid transformations $\{s, \theta, t_x, t_y\}$ are part of a continuum. Moreover, Igual & De la Torre (2010), and Igual et al. (2014) proposed to model 2D shapes from projecting an available set of 3D shapes into the 2D space, using the same angles that apply to the rigid deformation. The main idea of this approach is to project the 3D shapes into the 2D space assuming all possible 3D rotations of them, and then compute the affine transformation of the mean shape (to be computed as well) that would lead to the alignment of the 2D projected objects. In Igual & De la Torre (2010) and Igual et al. (2014), the transformations $\mathbf{T}_i$ are modelled using Euler angles, and then CGPA optimises the following cost function:

$$\min_{\mathbf{s}_0, \{\mathbf{T}_i\}_{i=1,\dots,N}} \sum_{i=1}^{n} \int_{\Omega} \|\mathbf{P}(\omega)\mathbf{s}_i - \mathbf{T}_i(\omega)\mathbf{s}_0\|^2 d\omega, \tag{3.37}$$

where ω accounts for the Euler angles that would rotate the 3D object. The solution to Equation 3.37 has a closed-form for both $\mathbf{s}_0$ and $\mathbf{T}_i$, which can therefore be optimised in an iterative manner. However, strictly speaking, despite the generalisation of the CGPA with respect to the classical GPA, the contributions of Igual & De la Torre (2010) do not extend Functional Data Analysis to the domain of images. In this thesis, the least-squares problem associated to linear regression is extended to the continuous domain in the same way as that of Igual & De la Torre (2010), Igual et al. (2014). However, the least-squares problem of Igual & De la Torre (2010) and Igual et al. (2014) is formulated to solve a different problem, rather than a linear regression. Contrary to the CGPA, the work of this thesis is based on linearising the images with respect to the shape displacements, thus extending the image features to the infinite set of training perturbations. Also, in practice, the CPGA requires an available set of 3D shapes. In the case of faces, the availability of databases containing annotated 3D shapes, is very limited.

3.3 Challenges and Opportunities

Research in Face Alignment and Tracking has recently grown fast. Many databases and methods have appeared, each proposing different solutions and alternatives. At the beginning of the

research conducted in this document (2011), there were some problems and challenges that have rapidly changed with the unearthing of newer solutions and datasets. Resources have grown fast, and today we can deal with thousands of images in a much faster way than few years ago. It is thus important to briefly remark how the motivation and challenges have evolved whilst the research on Continuous Regression was ongoing. We can see that, for instance, early methods were evaluated using only perturbations of ground-truth data, leaving aside the problem of initialising the facial points under a real scenario. Nevertheless, early annotated databases were environmentally controlled, and comprised near-frontal faces only. This way, there was no significant difference between evaluating the methods from a face detector than from perturbing the ground-truth data. Today, face alignment and tracking methods can be evaluated in a fully automated manner, thus making the research more focused on real scenarios.

This way, the motivation of this thesis was first driven by the theoretical and practical limitations that linear regression as a learning method has (Sánchez-Lozano et al. 2012). After the appearance of SDM, the research was focused on introducing continuous regression into the Cascaded Regression framework, and thus overcoming the limitations that SDM has: the high cost of its learning, and its inability of it to deal with real-time incremental face tracking (Sánchez-Lozano, Martinez, Tzimiropoulos & Valstar 2016). At the same time, a new theoretical framework tackling the limitations of classical functional regression methods was proposed (Sánchez-Lozano, Tzimiropoulos, Martinez, De la Torre & Valstar 2016). More specifically, this thesis approaches the limitations of SDM, which have been barely explored.

However, it would be unfair not to recognise that Face Alignment and Tracking research has many other alternatives to undergo rather than Continuous Regression, and whilst the tracker developed to fulfill this thesis is the first to date incorporating real-time incremental learning, we can not leave aside some recent solutions that have proven to attain impressive results as well. That is to say, it is fair, as well as necessary, to mention these recent works that yield similar results, and that work in real-time as well. For example, we find that the SDM is the driving method of Intraface (De la Torre et al. 2015). Other open source solutions were recently released, to name few, the dlib `dlib.net`, using the local binary features of Ren et al. (2014) and the OpenFace (Baltrušaitis et al. 2016), which uses the CLNF (Baltrušaitis et al. 2014).

These methods have proven to show real-time capabilities, but can not be updated in real-time. The solution presented in this thesis will include a novel updating strategy, capable of working in real-time, which is crucial to attain state-of-the-art results in the 300VW benchmark.

The contributions of this thesis are twofold. In one hand, the Continuous Regression is a theoretical framework that considers a set of *infinite* samples, and presents a computational advantage with respect to sampling based methods. On the other hand, Continuous Regression allows real-time incremental face tracking. The solution presented in this thesis has been successfully implemented in C++, and runs at more than 30fps. It has been further incorporated into a facial expression recognition system, which makes use of the points given by the tracker to subsequently perform emotion recognition. That is to say, the solution presented in this thesis is not only theoretical, but also has led to have a real-time face tracking system that is already running as a basic tool for facial expression recognition.

Chapter 4

Continuous Regression

This Chapter introduces Continuous Regression, a Functional Regression solution to the Least-Squares problem. In this Chapter, Continuous Regression is formulated by means of classical functional regression approaches, introducing a novel element with respect to previous works. More specifically, instead of approximating the input space with complex bases, such as B-splines or Fourier Series, Continuous Regression approximates it using a simple first-order Taylor expansion of the image features with respect to shape displacements. Throughout this Chapter, Continuous Regression is presented as a solution to the Least-Squares problem in the continuous domain of shape displacements, and no consideration is given to its inclusion within the Cascaded Regression framework. Chapters 5 and 6 will address this feature.

First, the problem is presented and solved for the 1-dimensional space of shape displacements (the x coordinate of a single point). Then, Continuous Regression is extended to the $2n$ dimensional shape. Finally, a set of experiments are presented to assess the correctness of the assumptions made throughout this Chapter. It is important to remark that the experiments conducted to validate the building blocks of Continuous Regression are not meant to outperform state of the art methods, and thus are kept simple on purpose. Chapter 5 will present Continuous Regression for correlated variables, and a real-time state of the art system is built in Chapter 6 to conduct a comparison with respect to state of the art methods. The contributions of this Chapter have been published in Sánchez-Lozano et al. (2012).

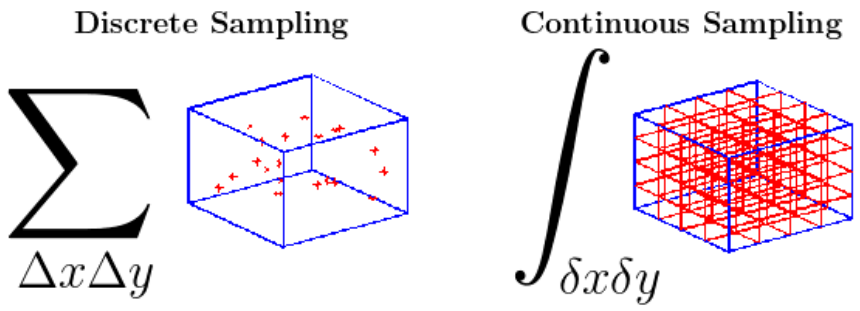

Figure 4.1: Difference between sampling-based regression and continuous regression. In Continuous Regression, we consider all the samples within a predefined neighbourhood, whereas in sampling-based approaches we need to generate the data from a given distribution.

4.1 Introduction

Briefly speaking, Continuous Regression extends sampling-based linear regression to the infinite set of perturbations that would be defined for a set of predefined limits (Figure 4.1). The basic idea consists of extending the number of perturbations to the infinite, where the sum would be replaced by a definite integral. Of course, it is computationally intractable to take infinite samples. However, we can approximate the samples by a Taylor expansion, so that samples are replaced by an estimate given by the ground-truth features and Jacobians. In order to better understand this, we can think of Continuous Regression from a different point of view. Let us consider the approach shown in Figure 4.2. We have an image and a given ground-truth position (the green dot in the nose tip). In the sampling-based approach (a), we generate random perturbations of the ground-truth locations. These perturbations are represented by the black arrows and yellow crosses. Then, we extract the image features in a patch surrounding the generated locations. We also store the displacement causing the perturbations, and then train a regressor $\mathbf{R}$ aimed to predict the displacement toward the ground-truth from the image features extracted at a perturbed location. The first step in Continuous Regression consists of replacing the features extracted at the ground-truth position by a first-order Taylor expansion. We can approximate the features at the perturbed locations as a linear expansion of the image features, extracted at the ground-truth positions, and their Jacobian. In this new scenario (b), we simply extract the image features and the Jacobian, at the ground-truth point (the

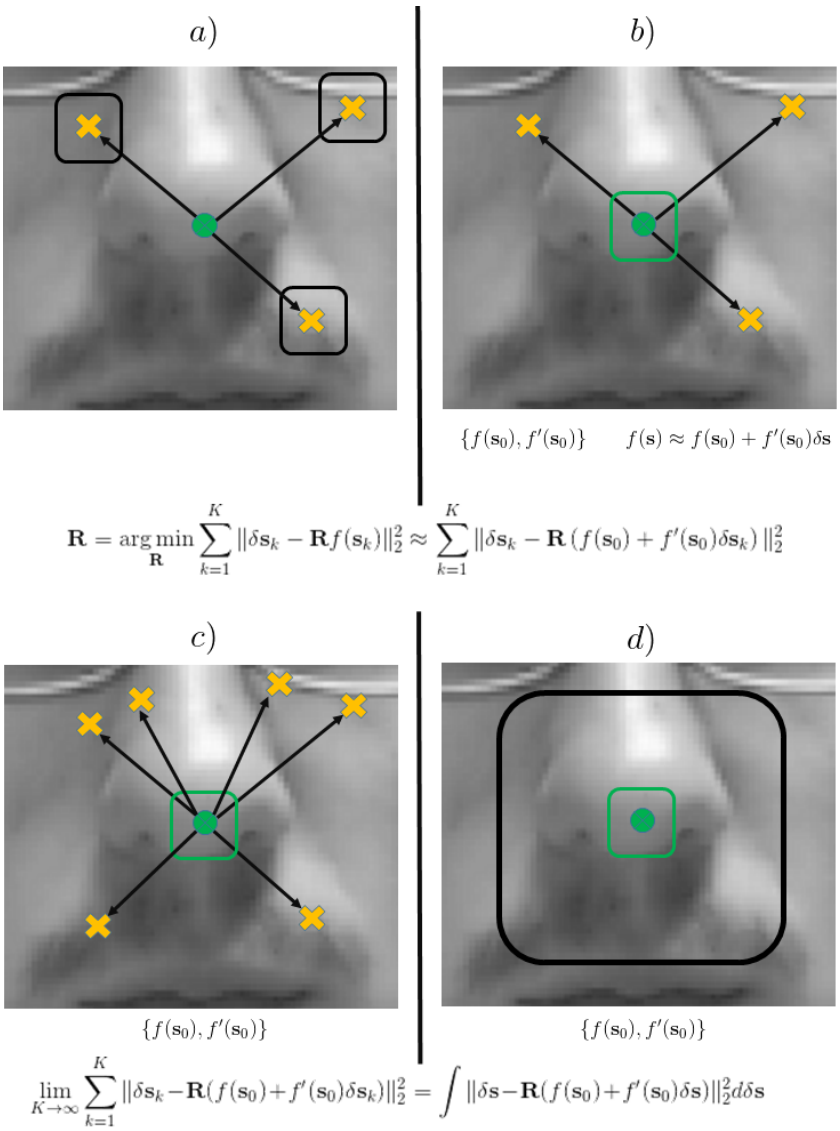

Figure 4.2: We can think of Continuous Regression in a different way. The green dot represents the ground-truth position, black arrows represent the shape displacements, yellow crosses represent the perturbations. a) In sampling-based regression we generate a set of random perturbations and then we extract the features around the perturbed samples (black square around the yellow crosses). b) We can then replace the sampled features by a Taylor approximation; to do so we only need the ground-truth features (the green box surrounding the ground-truth position). The features at the yellow crosses are given by a Taylor approximation. c) The Taylor expansion linearises the features with respect to the shape displacements, and hence we can add as many samples as we want, being these approximated by the Taylor expansion given by the image features and Jacobians, extracted at the ground-truth positions. d) We can extend the number of samples to the infinite, covering all possible displacements within a neighbourhood of the ground-truth positions (black box). This implies that the sum over perturbations is replaced by an integral. In this Chapter we will see that there is a closed form solution to this problem.

green dot), and then approximate the features at the perturbed locations by just computing the Taylor approximation (see below). Replacing the features in the perturbed space by an approximation that is given by the ground-truth features entails a very simple sampling process: we only need to extract and store the ground-truth features. If we want to generate a set of different features, or if we want to simply add more, we just use the Taylor estimation. This way, we can consider as many samples as we might want (c): nothing impedes the extension of the number of perturbations to the infinite, covering the continuous space within a defined region. This is nothing but replacing the summation by a definite integral (d). It is important to remark however that even though that Continuous Regression can be conceived this way (first replacing samples by a Taylor expansion then extending the process to the infinite), the most natural way to think of Continuous Regression is the former: we first want to solve the Least-Squares problem for an infinite set of samples; to solve this we approximate the input space by a first-order Taylor expansion. The problem to be solved is how to extend sampling-based linear regression to use a set of infinite samples, and the way we solve this is by using a Taylor expansion. We could use other Functional Regression techniques to solve it, although the intractability of the data using other basis would make the problem hard to be solved.

In this Chapter, we will see that there exists a closed-form solution for $\mathbf{R}$ in Continuous Regression. The solution depends only on the image features, and the derivatives of these features, extracted on the ground-truth positions. Contrary to previous solutions, Continuous Regression samples over images, but not over perturbations. Aside from the theoretical contributions of Continuous Regression, it is important to note its computational advantages with respect to sampling-based approaches: while in sampling-based regression images need to be sampled (i.e., features need to be taken) in perturbed shapes, Continuous Regression only needs the ground-truth data. This way, if we want to train a new model, we don't need to repeat the sampling process. We will later see how this implies a huge computational advantage.

We have previously seen that, despite its success, Cascaded Regression still suffers from three major limitations:

- 1) The Least-Squares formulation needs an exponential number of samples with respect

to the dimensionality of the input vector in order for the models not to be biased.

- 2) Training an SDM model is computationally expensive: it needs the data to be sampled once per cascade level. This means that, both the memory and the time needed for training increase dramatically with the number of perturbations. Therefore, training models under different configurations (e.g., for a different face detector) is in most cases very costly.

- 3) Proposed approaches for the recursive Least-Squares formulation, namely Incremental Learning (Xiong & la Torre 2014, Asthana et al. 2014) are reported to be very slow. This inibits the SDM and its extensions to from producing a "learning on the go" system capable of working in real-time, thus limiting its use.

The first limitation, as well as partially the second, is an intrinsic problem of sampling-based linear regression, which is the standard tool of Cascaded Regression methods, and is addressed within this Chapter. Chapter 5, and Chapter 6, will further address the limitations of Cascaded Regression methods, as well as the problem of incremental learning.

We note that the first limitation is mainly theoretical, and refers to which sampling distribution should be used when creating the shape displacements in the Least-Squares problem, as well as how many. When Cascaded Regression is used for face detection, the process of data augmentation consists of generating random bounding boxes around the one given by the detector. However, for the task of face tracking, the initial shapes are generated simulating the way faces vary from frame to frame. The question of distribution underlies such change is a discussion yet to be addressed. In many cases, when creating samples for Cascaded Regression training, the main assumption is to consider a Gaussian distribution. However, both the SDM (Xiong & De la Torre 2013) and Chehra (Asthana et al. 2014), consider a very low number of samples: 10. In this case, the samples generated tend to be *biased*. That is to say, the samples rarely meet the Gaussian distribution. Moreover, the bigger the variance of the Gaussian distribution, the bigger the number of samples needed in order to avoid bias in the samples. Even though it is hard to empirically tell the influence of this concept given the impressive results of Cascaded Regression for the task of facial landmark localisation, this is a question that still remains

unanswered. The nature of Continuous Regression bypasses this question, provided that an *infinite* set of samples are considered close to the sampled points.

This Chapter extends the Least-Squares formulation, which is the standard learning method for regression in Computer Vision, to the continuous domain, in which the target variable (the shape displacements) is treated as a continuum. Unlike existing approaches for Functional Regression, which approximate the input space by means of B-splines (Marx & Eilers 1999), Fourier Series (Ratliffe et al. 2002), or Radial Basis Functions (Quak et al. 1993), Continuous Regression proposes to use instead a first-order Taylor expansion of the feature space. This way the input space becomes linear with respect to shape displacements, yielding a closed-form solution.

4.1.1 Contributions

The main contributions of the research conducted in this Chapter are as follows:

- A complete formulation for Continuous Regression is presented, in which a first-order Taylor expansion of the input space is used as the basis functions, rather than other complex bases.
- A closed-form solution for Continuous Regression is derived, based on the first derivatives of the input space only.
- The computational benefits and the scope of Continuous Regression are studied, enabling the use of Continuous Regression as the building block of Cascaded Regression.

4.2 Linear Regression for Face Alignment

Before introducing the formulation of Continuous Regression, it is worth first reviewing the notation that will be used, as well as the main problem upon which Continuous Regression is built. A face image is represented by $\mathbf{I}$. A face shape $\mathbf{s} \in \mathbb{R}^{2n}$ is a vector describing the

location of the n landmarks. Vector shapes are represented as $\mathbf{s} = (x_1, \ldots, x_n, y_1, \ldots y_n)^T$, i.e., the convention that is followed here consists of first storing the x coordinates, then the y coordinates. Furthermore $\mathbf{x} = f(\mathbf{I}, \mathbf{s}) \in \mathbb{R}^d$ will be the feature vector extracted on image $\mathbf{I}$ at the points described by the shape $\mathbf{s}$. An asterisk represents the ground truth, e.g., $\mathbf{s}_i^*$ is the ground truth shape for image i.

4.2.1 Training

The problem of Linear Regression (in the context of face alignment), can be described as the problem of learning a mapping matrix $\mathbf{R}$ between image features, extracted in a perturbed version of the corresponding ground-truth shape, and the displacement, $\delta\mathbf{s}$, causing that perturbation. Mathematically speaking, the mapping is represented as $\delta\mathbf{s} = \mathbf{R}f(\mathbf{I}, \mathbf{s}^* + \delta\mathbf{s})$. Assuming a large set of N training images, and a sufficiently large set of K perturbations per image, the mapping matrix $\mathbf{R}$ is learnt through the typical least squares problem:

$$\sum_{i=1}^{N}\sum_{k=1}^{K} \|\delta\mathbf{s}_i^k - \mathbf{R}f(\mathbf{I}_i, \mathbf{s}_i^* + \delta\mathbf{s}_i^k)\|_2^2. \tag{4.1}$$

The minimisation of Equation 4.1 can be computed in a closed-form. Let $\mathbf{X} \in \mathbb{R}^{d\times KN}$ and $\mathbf{Y} \in \mathbb{R}^{2n\times KN}$ be the matrices whose columns contain the input feature vectors $f(\mathbf{I}_i, \mathbf{s}_i^* + \delta\mathbf{s}_i^k)$, and the target outputs $\delta\mathbf{s}_i^k$, respectively, the optimal regressor $\mathbf{R}$ can be computed as:

$$\mathbf{R} = \mathbf{Y}\mathbf{X}^T\left(\mathbf{X}\mathbf{X}^T\right)^{-1}. \tag{4.2}$$

In the context of Cascaded Regression, for each cascade level l (starting with $l = 0$), we are given a set of initial shapes $\{\mathbf{s}_i^{l,k}\}_{i=1\ldots N, k=1\ldots K}$ which can be expressed as $\mathbf{s}_i^* + \delta\mathbf{s}_i^{l,k}$. Then, we use Equation 4.2 to train each regressor. In parallel Cascaded Regression, we can train each of the levels by randomly generating $\delta\mathbf{s}_i^{l,k}$, once the mean $\boldsymbol{\mu}^l$ and covariance matrix $\boldsymbol{\Sigma}^l$, for each level, are given. That is to say, after the SDM training, we can compute the mean $\boldsymbol{\mu}^l$ and covariance $\boldsymbol{\Sigma}^l$ of the differences obtained after updating the training shapes: $\delta\mathbf{s}_i^{l,k} = \mathbf{s}_i^{l,k} - \mathbf{s}_i^*$. Then, for

each cascade level, a set of random perturbations $\delta\mathbf{s}_i^{l,k}$ is drawn from a Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}^l, \boldsymbol{\Sigma}^l)$, and then each $\mathbf{R}^l$ is computed using Equation 4.2. This way, computing each regressor is independent from the other levels, thus meaning that the training can be performed in parallel. We will see in Chapter 5 that Continuous Regression can be easily introduced within the Cascaded Regression framework, with the advantage that Continuous Regression only needs the data to be sampled once, whereas both the SDM and parallel SDM (onwards Chehra, as coined by the authors) require a sampling step for each Cascade Level.

During testing, an image $\mathbf{I}$ is given, along with a shape guess $\mathbf{s}^l$ (again, starting with $l = 0$). The shape guess $\mathbf{s}^0$ is obtained either from a face detector, or from the estimation of a previous frame. Then, a new shape estimate can be computed using the regressor $\mathbf{R}$ to the features extracted at $\mathbf{s}^l$, as follows:

$$\mathbf{s}^{l+1} = \mathbf{s}^l - \mathbf{R}f(\mathbf{I}, \mathbf{s}^l). \tag{4.3}$$

In a tracking scenario, the shape $\mathbf{s}^L$ will serve as the initial guess $\mathbf{s}^0$ for the upcoming frame.

4.3 Continuous Regression

This section formulates the Continuous Regression problem, and presents an approximation of the input space based on the first-order Taylor expansion of the image features. This leads to finding a closed-form solution, which only depends on the image features and derivatives, extracted in the ground-truth positions. We will see its computational advantages with respect to the sampling-based approach. For the sake of clarity, the formulation and experiments carried out in this Chapter are focused on a single-level model.

Let us consider the Linear Regression problem shown in Equation 4.1. We want to consider the variable $\delta\mathbf{s}$ as part of a continuum, so that we can integrate over all possible displacements. For the sake of clarity, the formulation will start with a one-dimensional example, that will later be extended to the $2n$ dimensions that are considered in the space of shape displacements.

4.3.1 1-D Continuous Regression

Let us start with a one-dimensional variable. In this configuration, the goal is to find a regressor $\mathbf{R} \in \mathbb{R}^{1\times d}$ tasked with predicting the displacement for the x coordinate of a single landmark. Let us assume then that perturbations are taken by displacing the x coordinate along the ground truth. That is to say, perturbed points take the form of $x + \delta x$, where δx lies within some predefined limits. Without loss of generality, we can assume that displacements are unbiased: the limits for δx will lie within the range $[-a\,,\,a]$, where a is a constant accounting for how large the displacement can be. For the sake of clarity, the level index is omitted, given that the problem is the same for all cascade levels.

Thus, our problem is now finding a regressor $\mathbf{R}$ to predict the displacement δx from the features extracted at the perturbed point $x^* + \delta x$. Under this setting, sampling-based linear regression training is defined, for a set of K perturbations per image, as

$$\sum_{i=1}^{N}\sum_{k=1}^{K} \|\delta x_i^k - \mathbf{R} f(\mathbf{I}_i, x_i^* + \delta x_i^k)\|_2^2. \tag{4.4}$$

The perturbations are randomly taken within the limits $[-a\,,\,a]$. Let us assume then that we want to consider *all* the possible perturbations that would lie within these limits. In this case, we would consider δx to be part of a continuum, and the sum would then be seen as the Riemannian sum, or a definite integral, as:

$$\sum_{i=1}^{N} \int_{-a}^{a} \|\delta x - \mathbf{R} f(\mathbf{I}_i, x_i^* + \delta x)\|_2^2 d\delta x. \tag{4.5}$$

Equation 4.5 assumes that the target variable δx is continuous. Unfortunately, despite the image features being a function of δx, there is no analytical form for it. That is to say, there is no analytical form with which we can express the image features as a function of δx. In order to circumvent this problem, we can apply an approximation that would help us solve Equation 4.5. Functional Regression approaches would approximate image features by B-splines, Fourier Series, or even more complex bases. Such bases would make it difficult for Equation 4.5 to have a closed-form solution, and some empirical approximations would be need

to be found instead. The extra complexity introduced would dramatically reduce the interest in such solutions.

Therefore, unlike previous works on Functional Regression, this thesis proposes the use of a first-order Taylor expansion to approximate the input feature space. The first-order Taylor expansion is defined as:

$$f(\mathbf{I}_i, x_i^* + \delta x) \approx f(\mathbf{I}_i, x_i^*) + \frac{\partial f(\mathbf{I}_i, x_i^*)}{\partial x}\delta x. \tag{4.6}$$

Through this section, and for the sake of clarity, we will denote $f_{*,i} = f(\mathbf{I}_i, x_i^*) \in \mathbb{R}^d$ and $f'_{*,i} = \frac{\partial f(\mathbf{I}_i, x_i^*)}{\partial x} \in \mathbb{R}^d$. The reader might have noticed that we have apparently moved from one problem to another: if we were unable to solve $f(\delta x)$ analytically, we would not be able to find its derivatives either. Effectively, images do not have analytical derivatives. However, despite the fact that we cannot evaluate the analytical derivative of image pixels with respect to spatial coordinates, we can apply an empirical approximation. For example, we have already seen that the computation of HOG features requires first computing the image derivatives. Typically, these are computed by filtering the image with a Sobel filter, which is a discrete differentiation operator. The Sobel filter is defined, for the x coordinate, as

$$S_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}. \tag{4.7}$$

The Sobel filter, for the y-th coordinate, is simply defined as $S_y = S_x^T$. The derivative of the image pixels with respect to the x coordinates is approximated as $\nabla_x \mathbf{I} \approx \mathbf{I} \star S_x$, where here $\star$ denotes the convolution operation. However, such a filter needs to perform 6 operations per pixel in order to compute the derivatives, only for the x coordinate. This might be quite expensive for a real-time application. In order to have an efficient and fast method for computing derivatives, in this thesis, the differentiation operator will be simply defined as the central differences:

$$S_x = \frac{1}{2}\begin{bmatrix} -1 & 0 & 1 \end{bmatrix}, \tag{4.8}$$

with, again, $S_y = S_x^T$. This way, only two operations are needed to compute the derivatives, for each coordinate. Also, instead of applying the convolution to the whole image, we might want to apply it to only the positions of interest. For the x derivative, this can be done by just extracting the features at the points defined by both the *left* and *right* positions of the points the derivative is being obtained from (*upper* and *lower* in the case of the y derivative), and then computing the weighted difference. Mathematically, we can express this concept as:

$$\begin{aligned} f'_x &= \frac{\partial f(\mathbf{I}, x)}{\partial x} \approx \frac{f(\mathbf{I}, x + \Delta x) - f(\mathbf{I}, x - \Delta x)}{2\Delta x} \\ f'_y &= \frac{\partial f(\mathbf{I}, y)}{\partial y} \approx \frac{f(\mathbf{I}, y + \Delta y) - f(\mathbf{I}, y - \Delta y)}{2\Delta y}. \end{aligned} \tag{4.9}$$

Interestingly, this idea can be applied to other complex features, such as SIFT or HOG. This implies that, f can be defined to be any kind of feature representation, upon which we can compute the derivatives using the central differences. That is to say, f in Equation 4.9 is defined as a generic feature extraction system. In practice, Δx is set to the minimum displacement, i.e., $\Delta x = 1$. The effect of assuming a higher value for Δx is studied in Section 4.6.1. Given the approximation of Equation 4.6, the original problem of Equation 4.5 can be expressed as:

$$\begin{aligned} &\sum_{i=1}^{N} \int_{-a}^{a} \|\delta x - \mathbf{R} f(\mathbf{I}_i, x_i^* + \delta x)\|_2^2 d\delta x \approx \\ &\sum_{i=1}^{N} \int_{-a}^{a} \Big(\delta x^2 - 2\delta x \mathbf{R}(f_{*,i} + f'_{*,i}\delta x) + \\ &(f_{*,i} + f'_{*,i}\delta x)^T \mathbf{R}^T \mathbf{R}(f_{*,i} + f'_{*,i}\delta x) \Big) d\delta x. \end{aligned} \tag{4.10}$$

In order to find the minimum of Equation 4.5 w.r.t. $\mathbf{R}$, we need to find its vanishing points. To do so, we can first solve the above integral with respect to δx, and then derive with respect

to $\mathbf{R}$. Grouping independent, linear, and quadratic terms, with respect to δx, we have[1]:

$$\begin{aligned}\int_{-a}^{a} \|\delta x - \mathbf{R} f(\mathbf{I}_i, x_i^* + \delta x)\|_2^2 d\delta x \approx \\ \int_{-a}^{a} (1 - \mathbf{R} f'_{*,i})^2 \delta x^2 d\delta x \\ -2 \int_{-a}^{a} (f'^T_{*,i} \mathbf{R}^T \mathbf{R} f_{*,i} - \mathbf{R} f_{*,i}) \delta x d\delta x + \\ + \int_{-a}^{a} f^T_{*,i} \mathbf{R}^T \mathbf{R} f_{*,i} d\delta x. \end{aligned} \tag{4.11}$$

The solution for the integrals in Equation 4.11 is straightforward, and leads to:

$$\begin{aligned}\int_{-a}^{a} \|\delta x - \mathbf{R} f(\mathbf{I}_i, x_i^* + \delta x)\|_2^2 d\delta x \approx \\ \frac{2}{3} a^3 (1 - \mathbf{R} f'_{*,i})^2 + 2\, a f^T_{*,i} \mathbf{R}^T \mathbf{R} f_{*,i} \end{aligned} \tag{4.12}$$

Each of the derivatives in Equation 4.12 with respect to $\mathbf{R}$ are given as:

$$\begin{aligned}\frac{\partial}{\partial \mathbf{R}} \frac{2}{3} a^3 (1 - \mathbf{R} f'_{*,i})^2 = \frac{4}{3} a^3 (f'^T_{*,i} - \mathbf{R} f'_{*,i} f'^T_{*,i}) \\ \frac{\partial}{\partial \mathbf{R}} 2\, a f^T_{*,i} \mathbf{R}^T \mathbf{R} f_{*,i} = 4\, a \mathbf{R} f_{*,i} f^T_{*,i}. \end{aligned} \tag{4.13}$$

Therefore, we get the solution for the one-dimensional continuous regression:

$$\mathbf{R} = \frac{1}{3} a^2 \sum_{i=1}^{N} f'^T_{*,i} \left(\sum_{i=1}^{N} f_{*,i} f^T_{*,i} + \frac{1}{3} a^2 f'_{*,i} f'^T_{*,i} \right)^{-1}. \tag{4.14}$$

Let us have a closer look to Equation 4.14. We can readily see that the solution depends on the image features, extracted at the ground-truth positions, and the derivatives, extracted at the ground-truth points as well. That is to say, the solution to Continuous Regression needs only to sample over images, but not over perturbations. This implies that, if we were to define a different limit for the displacement on the x coordinate, we would not need to generate perturbations again, but rather change the variable a in Equation 4.14. This results in computational saving for Continuous Regression with respect to the sampling-based approach.

[1]For the sake of clarity, and when not strictly necessary, the summation term is ommited, and the subscript i refers to the i-th training image

4.4 2n-dimensional Continuous Regression

We can readily move to the $2n$ dimensional case, in which we consider the $2n$ dimensions that form shapes. The natural extension of the Functional Regression form presented in Section 4.3.1 to the multidimensional case assumes that variables are independent from each other. In the sampling-based scenario, we would generate samples for each dimension independently from each other, by defining the corresponding limits separately. We will later see why. The Continuous Regression problem is formulated as:

$$\sum_{i=1}^{N} \int_{-a_1}^{a_1} \dots \int_{-a_{2n}}^{a_{2n}} \|\delta\mathbf{s} - \mathbf{R}f(\mathbf{I}_i, \mathbf{s}_i^* + \delta\mathbf{s})\|_2^2 d\delta\mathbf{s}, \tag{4.15}$$

where the line element is defined as $\delta\mathbf{s} = \delta x_1 \dots \delta x_n \delta y_1 \dots \delta y_n$, and the limits are bounded, and defined independently for each dimension. Without loss of generality, we will assume again that limits are symmetric, and the displacement assumed for each of the points is therefore unbiased. In the case of face tracking, with no prior information given, there is no reason to assume that a specific point is more likely to move in one direction than in the opposite.
Now, we will expand the image features with respect to the shape coordinates as:

$$f(\mathbf{I}_i, \mathbf{s}_i^* + \delta\mathbf{s}) \approx f(\mathbf{I}_i, \mathbf{s}_i^*) + \mathbf{J}_i^* \delta\mathbf{s}, \tag{4.16}$$

where $\mathbf{J}_i^* = \frac{\partial f(\mathbf{I}_i, \mathbf{s})}{\partial \mathbf{s}}|_{(\mathbf{s}=\mathbf{s}_i^*)} \in \mathbb{R}^{d \times 2n}$, evaluated at $\mathbf{s} = \mathbf{s}_i^*$, is the Jacobian of the feature representation of image $\mathbf{I}_i$, with respect to shape coordinates $\mathbf{s}$, at $\mathbf{s}_i^*$. Then, the k-th column of the Jacobian $\mathbf{J}^*$, for $k = 1, \dots, n$, is given by f'_{x_k}, whereas the $(k+n)$-th column of the Jacobian is given by f'_{y_k}. That is to say, $\mathbf{J}^{*k} = \frac{\partial f(\mathbf{I}_i, x_k)}{\partial x_k}$. Given the approximation of Equation 4.16, the original problem can be expressed as:

$$\begin{aligned}
&\int_{a_1 \dots a_{2n}} \|\delta\mathbf{s} - \mathbf{R}f(\mathbf{I}_i, \mathbf{s}_i^* + \delta\mathbf{s})\|_2^2 d\delta\mathbf{s} \approx \\
&\int \|\delta\mathbf{s} - \mathbf{R}\left(f(\mathbf{I}_i, \mathbf{s}_i^*) + \mathbf{J}_i^*\delta\mathbf{s}\right)\|_2^2 d\delta\mathbf{s} = \\
&= \int \Big[\delta\mathbf{s}^T\delta\mathbf{s} - 2\delta\mathbf{s}^T\mathbf{R}(\mathbf{x}_i^* + \mathbf{J}_i^*\delta\mathbf{s}) + \\
&\quad (\mathbf{x}_i^* + \mathbf{J}_i^*\delta\mathbf{s})^T\mathbf{R}^T\mathbf{R}(\mathbf{x}_i^* + \mathbf{J}_i^*\delta\mathbf{s})\Big] d\delta\mathbf{s}
\end{aligned} \tag{4.17}$$

We can group independent, linear, and quadratic terms with respect to $\delta\mathbf{s}$:

$$\begin{aligned}&\int \|\delta\mathbf{s} - \mathbf{R}\left(f(\mathbf{I}_i, \mathbf{s}_i^*) + \mathbf{J}_i^*\delta\mathbf{s}\right)\|_2^2 d\delta\mathbf{s} = \\ &\approx \int \left[\delta\mathbf{s}^T\mathbf{A}_i\delta\mathbf{s} + 2\delta\mathbf{s}^T\mathbf{b}_i + \mathbf{x}_i^{*T}\mathbf{R}^T\mathbf{R}\mathbf{x}_i^*\right] d\delta\mathbf{s},\end{aligned} \tag{4.18}$$

where $\mathbf{A}_i = (\mathbf{E} - \mathbf{R}\mathbf{J}_i^*)^T(\mathbf{E} - \mathbf{R}\mathbf{J}_i^*)$ and $\mathbf{b}_i = \mathbf{J}_i^{*T}\mathbf{R}^T\mathbf{R}\mathbf{x}_i^* - \mathbf{R}\mathbf{x}_i^*$. Given the independence between each dimension, the linear term equals to zero for symmetric limits. For the quadratic term, the solution stems from the fact that

$$\int \delta\mathbf{s}^T\mathbf{A}_i\delta\mathbf{s}d\delta\mathbf{s} = \sum_{u,v}\int A_i^{uv}\delta s_u\delta s_v d\delta\mathbf{s}, \tag{4.19}$$

where A_i^{uv} is the $\{u, v\}$-th entry of $\mathbf{A}_i$, and s_u, s_v refer indistinctly to any pair of elements of $\mathbf{s}$. We can see that

$$\int A_i^{uv}\delta s_u\delta s_v d\delta\mathbf{s} = \begin{cases} A_i^{uu}\frac{a_u^2}{3}\prod_k 2a_k & \text{if } u = v \\ 0 & \text{otherwise} \end{cases} \tag{4.20}$$

Let $V = \prod_k 2a_k$, and let $\mathbf{\Sigma} \in \mathbb{R}^{2n}$ be a diagonal matrix, the entries of which are defined by $\frac{a_u^2}{3}$. We can see that $\int \delta\mathbf{s}^T\mathbf{A}_i\delta\mathbf{s}d\delta\mathbf{s} = V\sum_u A_i^{uu}\frac{a_u^2}{3}$. We can further observe that $\sum_u A_i^{uu}\frac{a_u^2}{3} = Tr(\mathbf{A}_i\mathbf{\Sigma})$. Similarly, we can see that $\int \mathbf{x}_i^{*T}\mathbf{R}^T\mathbf{R}\mathbf{x}_i^* d\delta\mathbf{s} = \mathbf{x}_i^{*T}\mathbf{R}^T\mathbf{R}\mathbf{x}_i^* V$. Then, the solution to Equation 4.18 is given by:

$$Tr(\mathbf{A}_i\mathbf{\Sigma})V + \mathbf{x}_i^{*T}\mathbf{R}^T\mathbf{R}\mathbf{x}_i^* V \tag{4.21}$$

In order to find $\mathbf{R}$, we need to find the vanishing points of Equation 4.21. We can remove the constant $V = \prod_k 2a_k$, since it appears in both terms and does not depend on $\mathbf{R}$. The derivatives of both terms are calculated as follows:

$$\begin{aligned}\frac{\partial}{\partial\mathbf{R}}Tr(\mathbf{A}_i\mathbf{\Sigma}) &= 2\mathbf{R}\mathbf{J}_i^*\mathbf{\Sigma}\mathbf{J}_i^{*T} - 2\mathbf{\Sigma}\mathbf{J}_i^{*T} \\ \frac{\partial}{\partial\mathbf{R}}\mathbf{x}_i^{*T}\mathbf{R}^T\mathbf{R}\mathbf{x}_i^* &= 2\mathbf{R}\mathbf{x}_i^*\mathbf{x}_i^{*T}.\end{aligned} \tag{4.22}$$

This way, the solution for $\mathbf{R}$ is as follows:

$$\mathbf{R} = \mathbf{\Sigma} \left(\sum_{i=1}^{N} \mathbf{J}_i^{*T} \right) \left(\sum_{i=1}^{M} \mathbf{x}_i^* \mathbf{x}_i^{*T} + \mathbf{J}_i^* \mathbf{\Sigma} \mathbf{J}_i^{*T} \right)^{-1} . \tag{4.23}$$

Again, it is immediate to see that, with Equation 4.23, we sample over images, but not over perturbations. This means that the solution only needs the ground-truth features and their corresponding Jacobians. Once the data has been sampled, training a new regressor for different chosen limits is straightforward, as it does not require to generate perturbations again. This implies that, for the Cascaded Regression framework, we only need to sample the images once. We can see that Equation 4.21 depends on $\mathbf{\Sigma}$, which is indeed a function of $\{a_1, \dots a_{2n}\}$. Therefore, we can change the chosen limits and introduce a new $\mathbf{\Sigma}$ into the solution, without the need of sampling again. This is a huge time saving with respect to the sampling-based SDM. The Continuous Regression will be introduced into the Cascaded Regression framework in Chapter 5, and the benefits of training with Continuous Regression will be studied.

4.5 Complexity

Aside from the theoretical contributions, Continuous Regression offers a computational advantage with respect to sampling-based linear regression. This Section analyses the theoretical complexity of the proposed Continuous Regression, compared to sampling-based linear regression. We can leave aside the analysis of complexity at test time, given that once both regressors have been trained, their use is exactly the same. We will only notice the difference when it comes to incremental learning, which will be addressed in Chapter 6. We have seen that Continuous Regression only needs to sample over images, but not over perturbations. This means that Continuous Regression needs to store the ground-truth data for the training set. For each image, Continuous Regression stores the ground-truth features and their Jacobians, thus meaning that a matrix of $D \times (2n + 1)$ is needed per image, where D is the dimensionality of the feature vector, and n represents the number of points of interest. Typically, PCA is applied to the input space, keeping $d \ll D$ dimensions. However, PCA has to be applied in the

perturbed image space. We can not perform PCA directly on the ground-truth features and Jacobians. In the sampling-based approach, we can perform PCA directly over the extracted samples, given these to have been directly taken from the perturbed image space. Nevertheless, the Continuous Regression framework offers an alternative solution to the PCA problem in the perturbed space, which will be studied in the next Chapter. Therefore, in the experiments conducted in this Chapter, no dimensionality reduction is used. The reader might however be convinced at this point that D can be reduced to d in Continuous Regression, without any loss of generality.

In Continuous Regression, the feature extraction process entails 5 operations per image, as we need to extract the image features at the ground-truth positions, as well as its adjacent points (see Equation 4.9). The feature extraction process is thus fixed and does not depend on the number of perturbations. The computational complexity of a single feature extraction operation will be denoted as $\mathcal{O}(q)$. This will be independent of the chosen method (i.e., Continuous Regression or sampling-based regression). Furthermore, for Continuous Regression, it is necessary to obtain the inverse of the covariance matrix (we shall see in Chapter 5 that the invertible part of Equation 4.23 is the actual functional covariance matrix of the data). This operation is given by the $2n + 1$ outer products of the training features. We can see the invertible part of Equation 4.23 as the weighted sum of the products $\mathbf{X}_f\mathbf{X}_f^T$, where $\mathbf{X}_f \in \mathbb{R}^{D\times N}$ is a matrix containing either the ground-truth features or an indexed Jacobian. If we define a vector $\mathbf{f} = (1, a_1, \dots a_{2n})$, we can see that

$$\left(\sum_{i=1}^{M} \mathbf{x}_i^* \mathbf{x}_i^{*T} + \mathbf{J}_i^* \mathbf{\Sigma} \mathbf{J}_i^{*T}\right) = \sum_{f=1}^{2n+1} \mathbf{f}_f \mathbf{X}_f \mathbf{X}_f^T, \tag{4.24}$$

where $\mathbf{X}_f$ results from concatenating the features $\mathbf{x}_i^*$ in a single matrix, if $f = 1$, and the features $\mathbf{J}_{f-1}^*$ if $f > 1$. The outer product of each $\mathbf{X}_f$ has a cost of $\mathcal{O}(ND^2)$. Finally, the cost of inverting the final covariance is $\mathcal{O}(D^3)$, although this cost is exactly the same as inverting the covariance matrix in the sampling-based approach. Thus, for a given set of N training images, we can see that the timing complexity $\mathcal{C}_{cont}^t$, and memory complexity $\mathcal{C}_{cont}^m$, of training

a regressor using Continuous Regression, is given as

$$\mathcal{C}^t_{cont} = \mathcal{O}(D^3) + \mathcal{O}((2n+1)ND^2) + \mathcal{O}(5qN)$$
$$\mathcal{C}^m_{cont} = \mathcal{O}(D(2n+1)) \tag{4.25}$$

On the other hand, sampling-based linear regression needs the data to be extracted for each of the given perturbations. Once the data has been extracted, we need to store the matrices $\mathbf{Y} \in \mathbb{R}^{2n \times NK}$ and $\mathbf{X} \in \mathbb{R}^{D \times NK}$. Then, we have to compute the outer product of the feature vectors, and then the inverse of the covariance matrix. Therefore, we can see that the timing complexity $\mathcal{C}^t_{samp}$, and memory complexity $\mathcal{C}^m_{samp}$, of training a regressor using perturbations is given as:

$$\mathcal{C}^t_{samp} = \mathcal{O}(D^3) + \mathcal{O}(NKD^2) + \mathcal{O}(qNK)$$
$$\mathcal{C}^m_{samp} = \mathcal{O}(DNK) + \mathcal{O}(2nNK) \tag{4.26}$$

However, Continuous Regression is particularly better than sampling-based linear regression, in terms of its complexity, when we want to train a new regressor using the same training images, but under a different set of chosen limits. The timing complexity of Continuous Regression would only be the cost of computing the weighted sum (for which the outer products have already been stored), and the inverse of the covariance matrix. However, training a new regressor under the sampling-based approach needs the data to be extracted again. This means that the training complexity of Continuous Regression would be only $\mathcal{C}^t_{cont} = \mathcal{O}(D^3)$, whereas the computational complexity of re-training a sampling-based linear regression would not change: $\mathcal{C}^t_{samp} = \mathcal{O}(D^3) + \mathcal{O}(NKD^2) + \mathcal{O}(qNK)$. The computational saving is obvious. This is also highlighted under the context of Cascaded Regression, which will be analysed in Chapter 5.

4.6 Experiments

Before moving forward to the extension of Continuous Regression to correlated variables, as well as to the Cascaded Regression framework, it is necessary to evaluate the correctness of the preliminary assumptions presented in this Chapter. First of all, the conclusions derived from the 1-dimensional case are evaluated. That is to say, it is necessary first to evaluate the scope of the solution, i.e., the influence of the chosen limits in the performance of Continuous Regression. We can analyse the limits whilst evaluating other parameters: the use of HOG or Pixels, and the value of Δx when computing the central differences.

Then, the $2n$-dimensional Continuous Regression is evaluated. To do so, the experiments shown will rely on the use of a single regressor to predict displacements applied to the ground-truth, given a different set of limits. It is important to remark that these experiments are meant to evaluate the correctness of Continuous Regression, element by element. Chapter 6 will present a complete set-up for both training and validating the Continuous Regression for the task of accurately tracking faces in existing benchmarks. Therefore, the experiments in this Chapter are not focused on the generalisation of Continuous Regression to test scenarios, but rather on the evaluation of the aforementioned hypotheses. The experiments conducted towards evaluating the correctness of Continuous Regression are done using the training partition of the LFPW database (Belhumeur et al. 2011), which has been described in Chapter 3.

4.6.1 1d-Continuous Regression

For the 1-dimensional scenario, the goal is to evaluate the scope of Continuous Regression given that samples are substituted by the Jacobian approximations. It is well known that approximating a function using a first-order Taylor expansion is valid only within some predefined limits. These limits can differ depending on the feature extraction system, as well as on the Δx in the derivatives. We can evaluate any of the shape points that will conform our shapes. However, for the sake of clarity, the nose tip, indexed as 31 in the 66 point configuration shown above, was chosen as the target point for the 1-dimensional experiments. This is because we

aim at evaluating the correctness of Continuous Regression, not the scope of each of the possible points, given that finally these will be predicted as a whole. The nose tip is typically surrounded by pixels that belong to skin as well, not to the background, as is the case of some contour points. This context makes the variation of pixels "smooth", validating the Taylor expansion within larger limits than those that could be applied for "less smooth" points. For a variable set of training images and perturbations, a linear regression using a sampling-based approach was trained, using different levels of a. Then, a set of models were trained, using Continuous Regression, for each of the chosen limits, using the same number of training images.

Scope of CR

First, the scope of Continuous Regression is measured. For a varying set of training images, a different set of perturbations were considered for the sampling-based linear regression approach. To test both Continuous Regression and sampling-based linear regression, a set of perturbations were generated for the training set, using the same limits that were chosen for the training of both models. That is to say, the evaluation is made over the same set of images, but with a different set of given perturbations. Under this setting, the test set consists of a new set of perturbations. The reason not to use a different set of images for the evaluation is that the number of images is fairly low, and therefore the generalisation capabilities of both Continuous Regression and sampling-based linear regression is too poor. In this curve, Continuous Regression performance is displayed in green, whereas sampling-based linear regression is shown in red. The initial error (i.e., the error given by the random perturbation), is shown in blue. The error is measured as the mean squared root of the distance between the point, and the ground-truth annotation. Each plot shows the results for a varying limit a. We can see that a represents the variance of the perturbations, in pixels, that are applied to the ground-truth shapes to generate both the testing perturbations, and the training perturbations for the sampling-based linear regression case. Interestingly, we can see that, even for $a = 10$, Continuous Regression is still capable of reducing the error. Also, the better results shown for big values of a clearly show the problem of variance explained in Chapter 3. We can see that, in the case of $a = 50$ (i.e., a variance of $a^2/3 \approx 800$ pixels, which corresponds to an error $\approx \sqrt{800} = 28.3$), the regressor

that is capable of reducing the error fairly well is the one that has been trained with only one perturbation per image. This is because the higher the number of perturbations considered, the most probable the generated samples to be within intermediate levels. The Continuous Regression considers the extreme case, in which the infinite set of samples are considered.

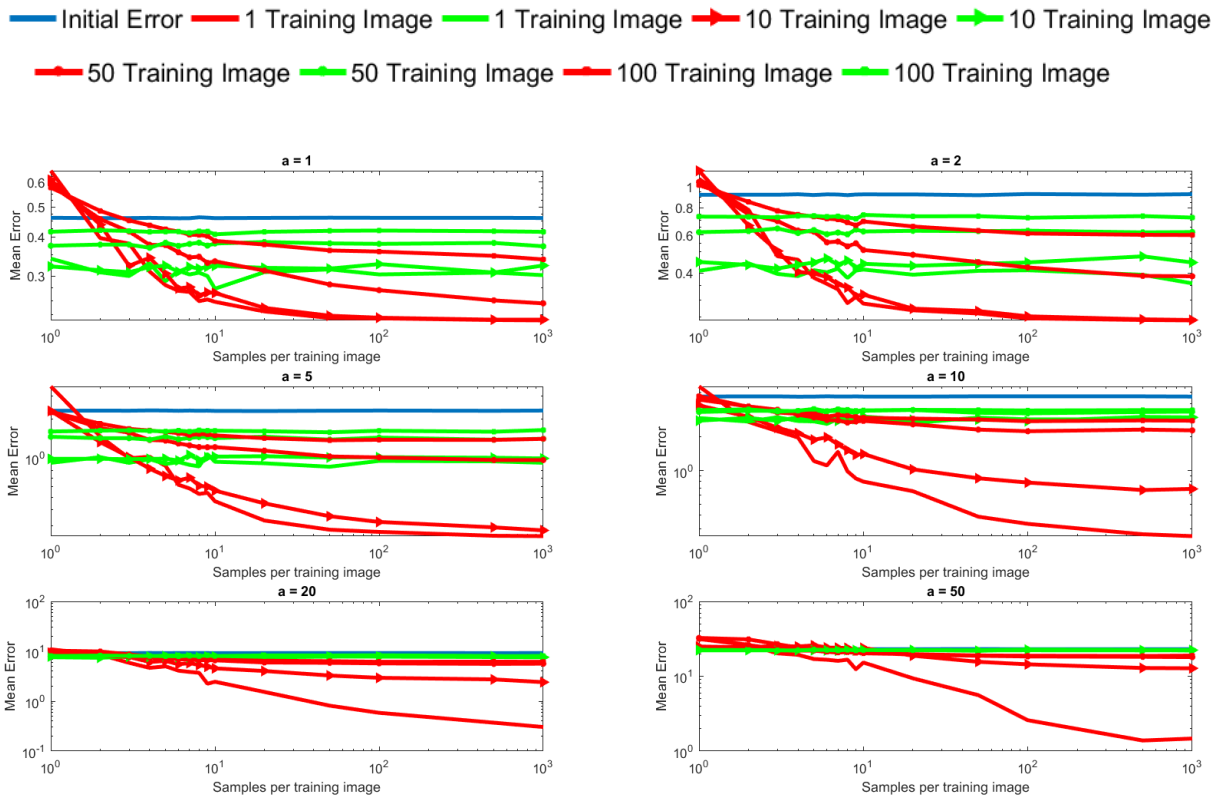

Figure 4.3: Mean errors for the predictions given by discrete regressors (in **red**) and continuous regressors (in **green**). Each plot shows a configuration in which the chosen limit is set to the value shown in the corresponding title (i.e., $a = 1, 2, 5, 10, 20, 50$). Errors are measured in the same subset of images composing the training set, the number of which is also varied in this experiment. It can be seen that the Continuous Regression gives a constant error no matter the number of training samples, given that it does not account for perturbations, but rather for image features. Also, it can be seen that, the bigger the input variance, the lower the capacity of Continuous Regression. In this setting, the discrete regressor needs 1000 perturbations to perform reasonable well.

Influence of Δx

Second, it is worth evaluating the influence of Δx in the performance of Continuous Regression. Recall that Δx is used in the empirical derivative of the image features (Equation 4.9) as the distance of adjacent points w.r.t. the ground truth. Again, this is done for a varying set of

training images, as well as for a varying set of chosen limits. Basically, $\Delta x > 1$ would imply that the sampling would be performed farther than in the closest point to the ground-truth solution. Intuitively, the derivatives should be more descriptive the closer they are taken to the ground-truth positions. However, given the non-linear nature of the features, it is possible that a bigger step would somehow "smooth" the derivatives. The results of this experiment are shown in Figure 4.4. We can see that the best results are mostly given when Δx is set to the minimum distance.

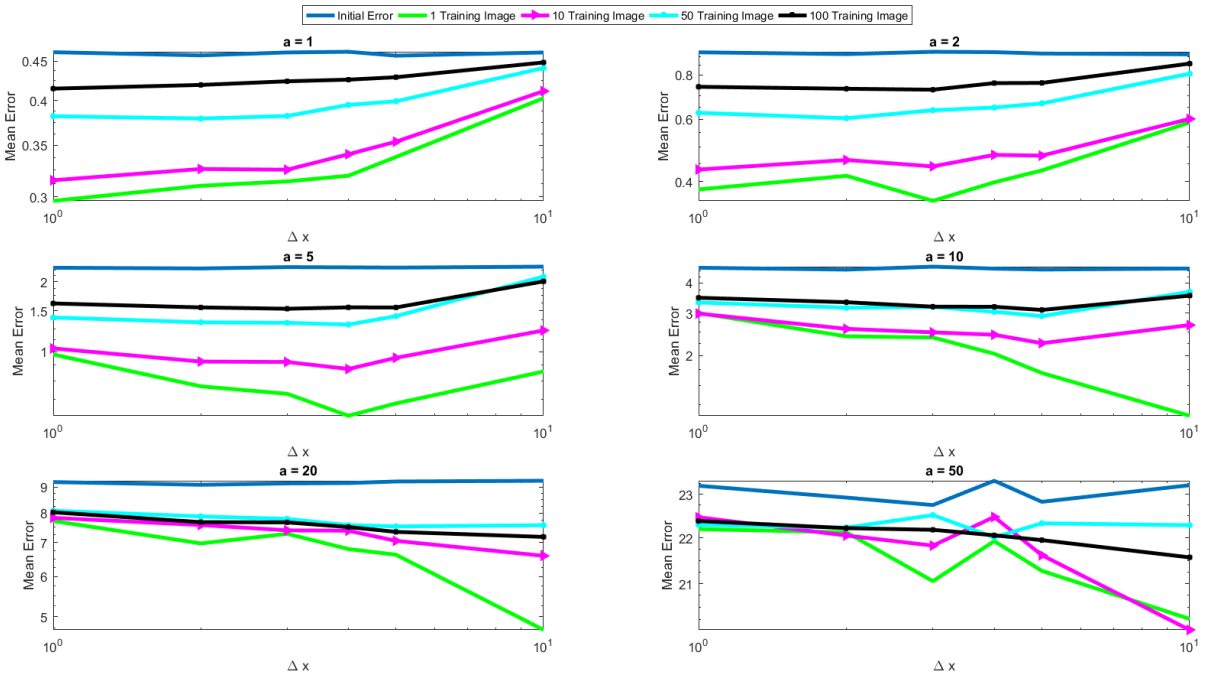

Figure 4.4: Mean errors for the predictions given by Continuous Regression under different configurations. We can see that the best step is given by $\Delta x = 1$, although for bigger limits this assumption stops being correct. However, the behaviour of Continuous Regression for such limits can not be associated to the chosen step, given that all configurations generally fail to recover from a huge initial error.

HOG vs Pixels

Finally, an experiment evaluating the difference between HOG and pixels as feature descriptors for Continuous Regression was designed. For pixels, a patch of size 24×24 is considered, resulting in a 576 dimensional vector for the feature descriptor. Results comparing both kinds of features are shown in Figure 4.5. It can be seen that, for small perturbations, pixels can perform

better. However, the Taylor approximation becomes a big constraint for bigger displacements, thus meaning that pixels are highly non-linear.

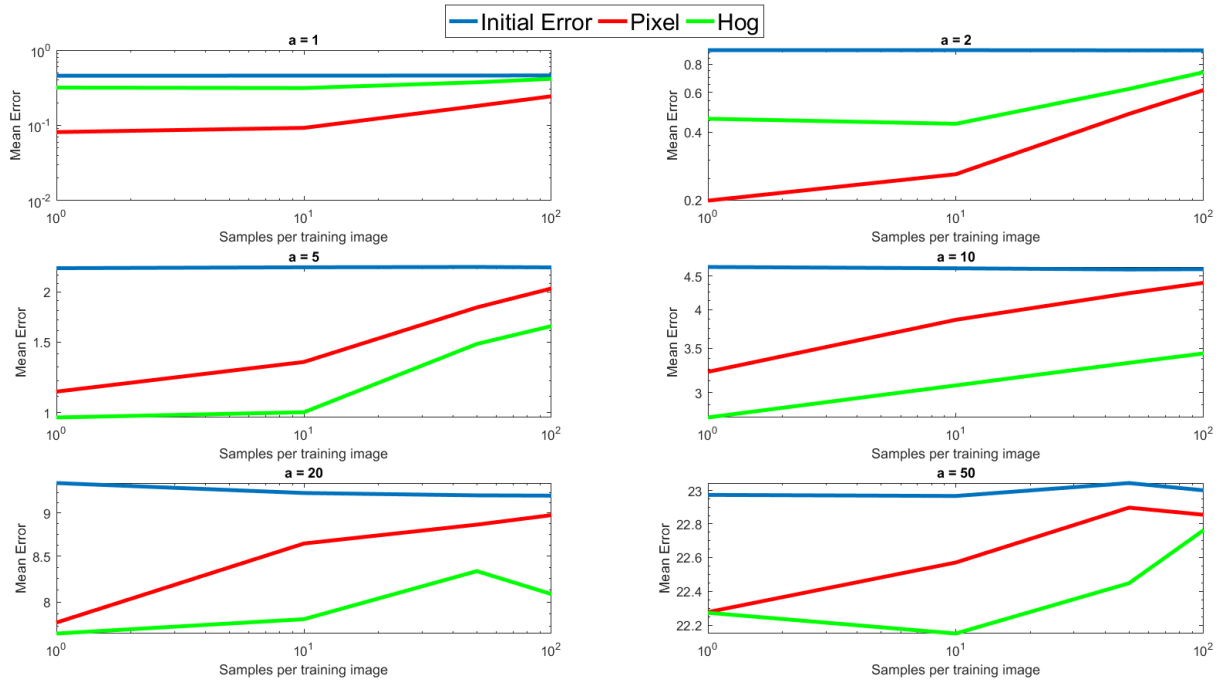

Figure 4.5: Comparison between performance given by a regressor trained with HOG, and a regressor trained with Pixels. We can see that the pixels perform better than HOG only for very small perturbations given that the Taylor approximation is still valid. However, when considering further distances, we have to note that a "smoother" feature descriptor, such as HOG, attains better results than pixels.

4.6.2 2nd-Continuous Regression

The aim of this experiment is now to check that the partial results shown above generalise to the whole set of shape displacements. To do so, the parameters evaluated above were fixed to those shown to have the best results, i.e., in this experiment, all regressors were trained using Continuous Regression utilising HOG features, with $\Delta x = 1$. In this experiment, the regressors are meant to predict the whole set of 66 points forming a shape. The error measure is then the average point-to-point Euclidean distance, normalised by the interocular distance, interpreted as the Euclidean distance between the eyes' outer corners. This error measure was introduced in the field of Face Alignment to avoid scaling factors influencing the total error. This normalisation ensures that errors represent a typical deviation from the ground-truth

points, regardless of the size of the target image. In this experiment, the error is represented through Cumulative Error Distribution curves (CED), which is the standard representation form for displaying error results. The CED curves represent the percentage of images (y-axis) with error lower than a given threshold (x-axis). It is an increasing function, meaning that, the closer the curve to the top of the plot, the better the results.

In this experiment, all the points were randomly displaced using a shared perturbation parameter a. In order to avoid the generation of non-plausible shapes, the perturbations were generated through a PDM. The use of a PDM will be introduced in Section 5.6. Once the parameter a is set for a given evaluation, all shape parameters are perturbed according to the same value, scaled by the energy each is retaining. The parameter a is used to generate both the training and testing perturbations. This experiment evaluates the influence of using a variable training set. The regressors are tested in a random subset of 500 images from the Helen dataset(Le et al. 2012). Figures 4.6 and 4.7 show the results for a variable training set size, a variable amount of perturbations per image, and a variable parameter a. It can be seen that this experiment validates the contributions shown through this Chapter. As a becomes bigger, the capacity of regressors to fit correctly in a single iteration becomes smaller. Also, the capacity of continuous regression to fit as good as a single regressor reduces as a becomes bigger as well.

4.7 Conclusions

This Chapter presented the basic foundations of Continuous Regression as the extension of sampling-based linear regression to the continuous domain. Built upon a Taylor expansion as the feature basis, it is possible to find a closed-form solution for the problem of Continuous Regression. Interestingly, the solution only needs to account for the ground-truth data, and therefore no further sampling is needed. The fact that samples are replaced by a Taylor expansion entails a huge computational sampling, although the cost of this approximation needs to be considered for each use case. The experimental validation shows that it is beneficial

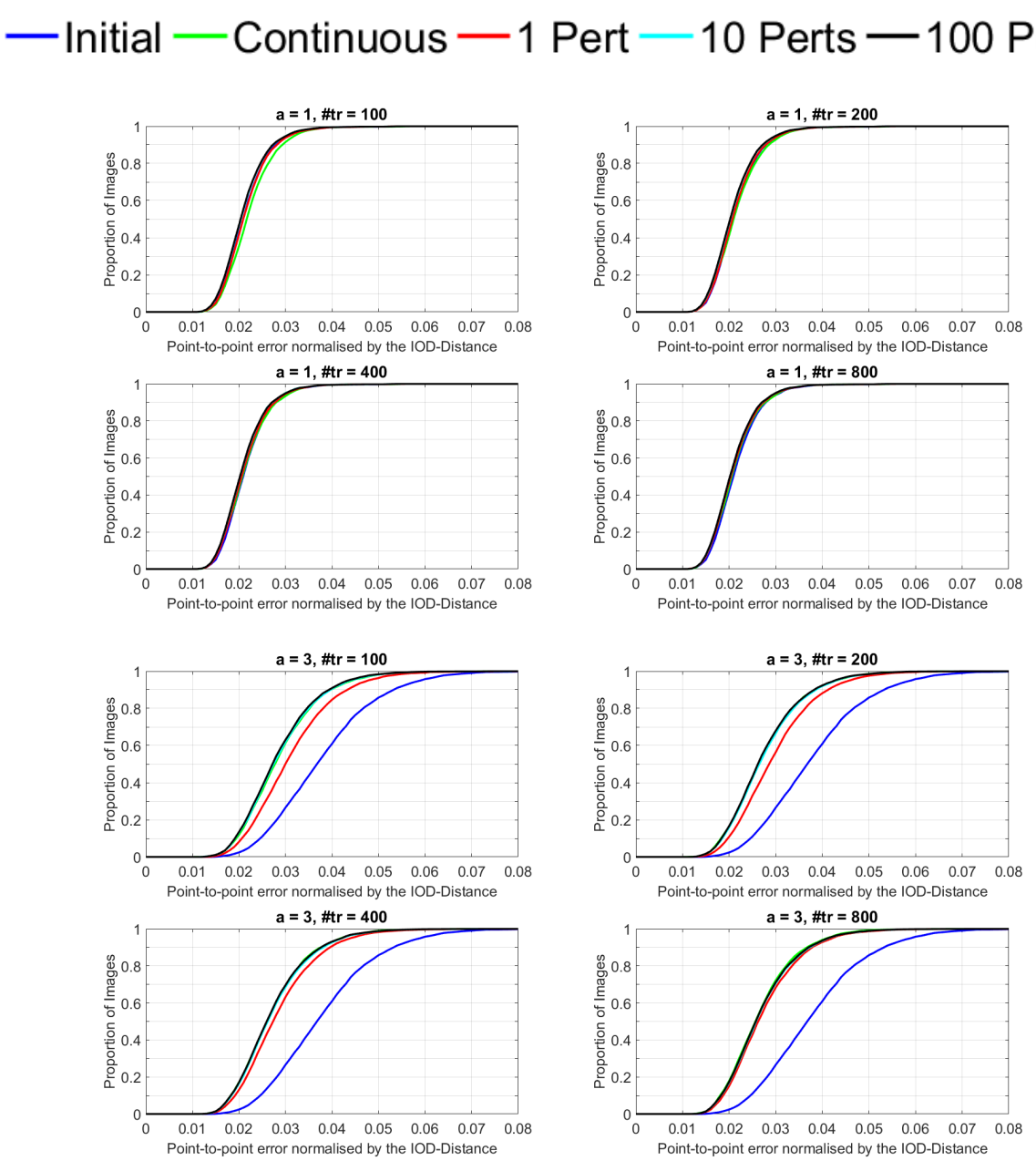

Figure 4.6: Comparison of Continuous Regression and Sampling-based Linear Regression, in which both training and testing perturbations depend on the value of a. $\#tr$ represents the number of training images.

under certain conditions, but can be harmful when the Taylor expansion breaks down. In this Chapter, several aspects of Continuous Regression were studied, showing that there exists a limit for the Taylor expansion to work fairly well. We can see that certain image features might appear to be "smoother" under some conditions than others. The experiments shown in this Chapter validate the foundations of Continuous Regression, although it is important to remark that for specific scenarios Continuous Regression can not replace sampling-based regression,

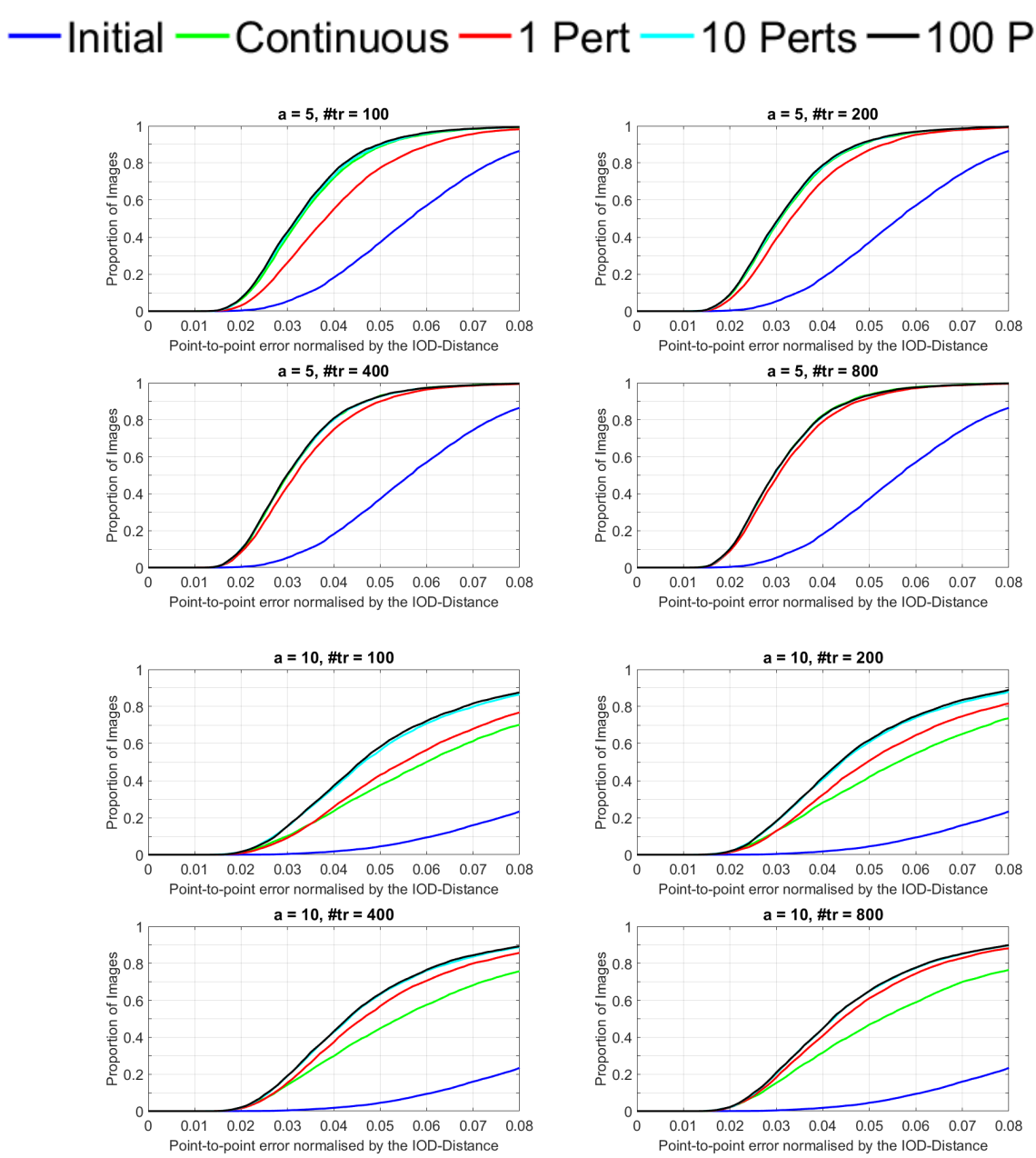

Figure 4.7: Comparison of Continuous Regression and Sampling-based Linear Regression, in which both training and testing perturbations depend on the value of a. $\#tr$ represents the number of training images.

such as when dealing with very large displacements. Future work will explore the use of different linearisation points, that would allow the extension of Continuous Regression without compromising the accuracy of the approximations. Nevertheless, we can state that the results shown in this Chapter validate the theoretical derivations, and hence can be used as a basis for Cascaded Regression, which is studied in the next Chapter.

Chapter 5

Continuous Regression on Correlated Variables

This Chapter presents an alternative formulation to Continuous Regression, allowing the modelling of variables that might be correlated, which can not be considered using the classical Functional Regression form used in Chapter 4. One important aspect of Continuous Regression in the form presented in Chapter 4 is that the formulation intrinsically assumes that an isotropic uniform distribution is being used. This constrains the scope of Continuous Regression to variables that are uncorrelated, and zero-mean. We will see that faces vary from frame to frame in a correlated manner, thus meaning that we need to assume such type of samples when training. Therefore, we need to enable Continuous Regression to model correlated variables. To do so, this Chapter introduces a novel measure to the problem formulation, encoding a "data term", tasked with correlating the different dimensions over which the problem is solved. The main assumption that underlies the new formulation resides in the fact that correlation can be encoded through a full-covariance matrix, which ultimately defines sufficient statistics for probability density functions. This way, further to Continuous Regression being a solution that integrates over an infinite set of samples, it can be shown that there is a link between the probability density function (i.e., the way samples are taken), and the space of perturbations that is being considered in the infinite sampling.

In this Chapter, the solution to Continuous Regression is introduced to the Cascaded Regression framework, and the benefits of training using Continuous Regression with respect to the standard SDM are shown. We will see that, beyond Continuous Regression being of reduced computational complexity, it attains the same performance as SDM. We will see that Continuous Regression depends on the sampling statistics: the mean and covariance. This way, its inclusion within the Cascaded Regression framework is straightforward, meaning a huge computational saving with respect to sampling-based methods. As we have previously seen, contrary to previous works, Continuous Regression can be trained under different configurations in a very fast and efficient manner, without the need for re-sampling.

The contributions of this Chapter have been partially presented at ECCV 2016 (Sánchez-Lozano, Martinez, Tzimiropoulos & Valstar 2016), and further submitted for review in IEEE Transactions on Pattern Analysis and Machine Intelligence (preprint available at `https://arxiv.org/pdf/1612.02203v1.pdf`). These can be summarised as follows:

- An alternative optimisation problem for Continuous Regression is presented, in which the target variable (i.e., shape dimensions), can be internally correlated, something not possible under the classical Functional Regression form.

- The solution is further incorporated into the Cascaded Regression framework. It is shown that under the same training settings, Cascaded Regression with Continuous Regression performs comparably to the SDM. The method is coined Cascaded Continuous Regression (CCR).

- The statistical meaning of the new formulation is linked with the actual space of perturbations within which samples are taken.

- A link between Continuous Regression and previous works is provided, and then the possibility of computing PCA in the perturbed image space is given.

5.1 Introduction

The solution to Continuous Regression presented in Chapter 4 implies a computational saving with respect to sampling-based methods. However, it suffers from a major flaw that is crucial to its performance when working with faces. We have previously seen that, in sampling-based linear regression, the samples $\delta\mathbf{s}$ are randomly generated from an adequate distribution. The chosen distribution is typically Gaussian, and therefore samples are taken, for level l in Cascaded Regression, following $\mathcal{N}(\boldsymbol{\mu}^l, \boldsymbol{\Sigma}^l)$. The statistics $\boldsymbol{\mu}^l \in \mathbb{R}^{2n}$ and $\boldsymbol{\Sigma}^l \in \mathbb{R}^{2n\times 2n}$ are given by the training shapes, and do not necessarily need to be isotropic. For the task of tracking, the statistics for the first level are given by measuring how shapes vary from frame to frame. That is to say, the statistics are computed by measuring the mean and covariance of the differences of shapes in each two consecutive frames, on annotated data. These statistics will serve to generate the initial shapes in the Cascaded Regression training. The initial shapes are given by applying random perturbations to static images, following the learnt statistics. Therefore, we can readily see that $\boldsymbol{\mu}$ is not necessarily a vector of zeros, and $\boldsymbol{\Sigma}$ can be a full covariance matrix. This means that the generation of samples does not assume independence between different dimensions, and the way we sample the δx_k coordinate can affect how we sample the rest of the shape coordinates. Mathematically, we can express this concept as:

$$p(\delta x_k|\{\delta\mathbf{s}_c\}_{c=1\ldots 2n, c\neq k}) = p(\delta x_k) \iff \boldsymbol{\Sigma}(k,j) = 0\,,\ \forall j \neq k, \tag{5.1}$$

where $\boldsymbol{\Sigma}(k,j)$ is the k,j entry of $\boldsymbol{\Sigma}$. A closer look to the formulation shown in Equation 4.15 reveals that, when grouping terms, we simply apply the *separability* rule of variables, given that there is no constraint or function impeding its application. When applying this rule, we see that the integral splits into each of the different dimensions composing the line element. Statistically speaking, this means that the way we sample each dimension does not affect how we sample other dimensions. This means that we can first take the infinite samples lying within the range $[-a_1\,,\,a_1]$, while others are kept constant, then we move to the second dimension, and so forth, i.e., we *fix* all dimensions but the one over which we are integrating. This means that the equality shown in Equation 5.1 holds, and therefore we are assuming an isotropic

distribution. Also, the integrals in Equation 4.15 assume an *isotropic uniform distribution*: we are taking all samples within some bounded limits, while a Gaussian distribution would consider some samples to go over these limits, although the occurrence of these would be very low. This means that Continuous Regression, as presented in Chapter 4, intrinsically assumes an isotropic uniform distribution, whereas in sampling-based linear regression we would consider generating samples from a full-covariance Gaussian distribution. After measuring the statistics in the training set of videos, the covariance matrix is given as shown in Figure 5.1.

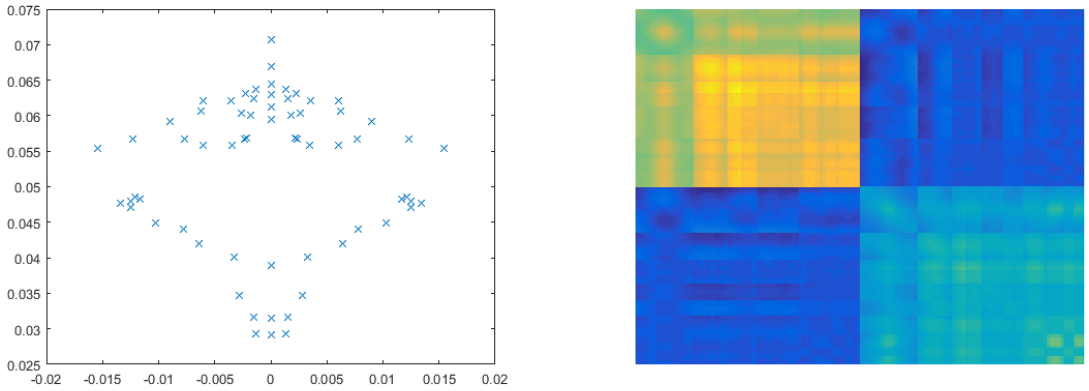

Figure 5.1: **Left:** Average shape $\boldsymbol{\mu}$ resulting after computing the shape displacements across the training set of videos. It can be seen that it is almost zero, thus we can assume that displacements are unbiased. **Right:** Covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{132\times132}$ of the shape displacements measured across the set of training videos. As can be seen, the covariance matrix is far from being diagonal. Therefore, we cannot approximate the covariance matrix by its diagonal elements.

At this point, the reader might wonder why the sampling assumption is important in Continuous Regression, given that its solution does not account for samples, but rather for the image features and Jacobians. If we were to take *all* samples, why should we consider the way we take them? We will later see how the sampling distribution is indeed related with the region within which samples are taken, and that the isotropic uniform distribution does not allow for correlated variables, hence assuming independence between dimensions. We will see how the probability distribution actually defines the space within which samples are taken, thus being crucial for the Continuous Regression to perform effectively.

This Chapter extends the Continuous Regression formulation to the space of correlated variables. It will be shown that, assuming a different measure, it is possible to find a closed-form

solution for a general case. The new solution is then introduced into the Cascaded Regression framework, and is shown to provide a huge training time saving with respect to the standard SDM training, without compromising the accuracy of the models. Then, a link to previous work is given, and a solution to PCA in the perturbed space for Continuous Regression is presented. After that, a geometrical interpretation is given, and an extension of Continuous Regression to the space of shape parameters is derived. Finally, a set of experiments are conducted validating the contributions of this Chapter.

5.2 A new measure for Functional Regression

The problem of Continuous Regression presented in Chapter 4 assumes an isotropic uniform distribution, thus limiting the extent of its solution. We can see that the main problem of existing Functional Regression approaches is that the integral is defined with respect to the Lebesgue measure, which implicitly assumes that an (unnormalised) uniform distribution, in which samples are uncorrelated, is used. Thus, it is not possible to solve it for correlated dimensions. To overcome such a limitation, we need to solve the integral with respect to a different measure μ. The measure that fits to the Least-Squares problem is the *probability measure*, i.e., $\mu = \Pr$. We will shortly see why. Let us first define the problem with respect to μ:

$$\sum_{i=1}^{N} \int_{\delta \mathbf{s}} \|\delta \mathbf{s} - \mathbf{R} f(\mathbf{I}_i, \mathbf{s}_i^* + \delta \mathbf{s})\|_2^2 d\Pr(\delta \mathbf{s}). \tag{5.2}$$

Applying the Riemannian form, we can write Equation 5.2 as

$$\sum_{i=1}^{N} \int_{\delta \mathbf{s}} \|\delta \mathbf{s} - \mathbf{R} f(\mathbf{I}_i, \mathbf{s}_i^* + \delta \mathbf{s})\|_2^2 p(\delta \mathbf{s}) d\delta \mathbf{s}, \tag{5.3}$$

where now $p(\delta \mathbf{s})$ accounts for the pdf of the sampling distribution. We can see that, when formulating the integral with respect to a probability measure, we are actually formulating the Continuous Regression by means of the average *expected loss function*, rather than from the classical Functional Regression perspective, which minimises the *empirical loss function*. The expected loss is actually the function from which the empirical loss is derived, for example,

in Xiong & De la Torre (2013), the expected loss function is reduced to the empirical loss by applying a Monte Carlo sampling approximation. This clearly illustrates how the probability measure helps correlating dimensions, and why this measure is appropriate for the problem being considered. Figure 5.2 illustrates these differences. The top (a) scheme depicts the approach studied in Chapter 4, where the Functional Regression approach was directly applied to the Least-Squares problem, resulting in the empirical loss function. The bottom figure (b) illustrates a different view from which we can think of Continuous Regression. While the problem is still solved using a Functional Regression approach, it *can be seen* as the inverse of a Monte-Carlo sampling estimation: we now are solving the expected loss function.

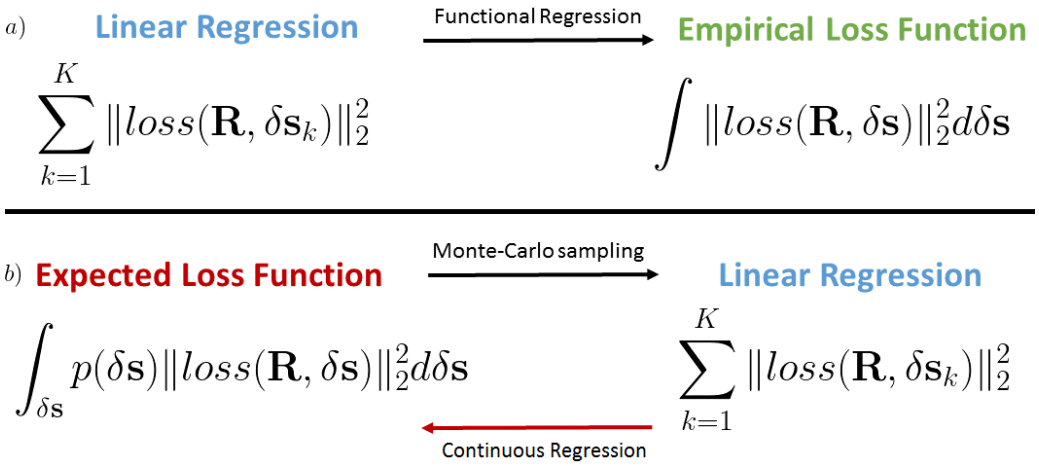

Figure 5.2: a) Functional Regression as explained in Chapter 4. b) The new approach to Continuous Regression, studied in this Chapter, *can be seen* as the inverse of a Monte Carlo sampling estimation. The probability density function defined by $p(\delta\mathbf{s})$ defines the geometric space within which samples are taken.

Again, to solve Equation 5.3, the feature space is approximated by its first-order Taylor expansion. The integrals are now, a priori, unbounded (they become bounded for e.g. a uniform distribution). Following the steps carried out in Section 4.4, we expand the features, and group linear, quadratic, and independent terms, with respect to $\delta\mathbf{s}$:

$$\begin{aligned} &\int_{\delta\mathbf{s}} p(\delta\mathbf{s}) \|\delta\mathbf{s} - \mathbf{R} f(\mathbf{I}_i, \mathbf{s}_i^* + \delta\mathbf{s})\|_2^2 d\delta\mathbf{s} \approx \\ \approx &\int_{\delta\mathbf{s}} p(\delta\mathbf{s}) \left[\delta\mathbf{s}^T \mathbf{A}_i \delta\mathbf{s} + 2\delta\mathbf{s}^T \mathbf{b}_i + \mathbf{x}_i^{*T} \mathbf{R}^T \mathbf{R} \mathbf{x}_i^*\right] d\delta\mathbf{s}, \end{aligned} \tag{5.4}$$

where, using a similar approach to that of Chapter 4, we define $\mathbf{A}_i = (\mathbf{E} - \mathbf{R}\mathbf{J}_i^*)^T (\mathbf{E} - \mathbf{R}\mathbf{J}_i^*)$

and $\mathbf{b}_i = \mathbf{J}_i^{*T}\mathbf{R}^T\mathbf{R}\mathbf{x}_i^* - \mathbf{R}\mathbf{x}_i^*$. Let us have a closer look to Equation 5.4, and let us assume that the pdf $p(\delta\mathbf{s})$ can be parameterised by its mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$. The pdf of the sampling distribution is only required to be parameterised by a mean and covariance. A Gaussian, or a uniform distribution, would be completely defined by these first and second order moments. Then, for any symmetric matrix $\mathbf{A}$, the following properties hold (Brookes 2011):

$$\begin{aligned} \int_{\delta\mathbf{s}} p(\delta\mathbf{s})d\delta\mathbf{s} = 1, \quad \int_{\delta\mathbf{s}} \delta\mathbf{s}p(\delta\mathbf{s})d\delta\mathbf{s} = \boldsymbol{\mu}, \\ \int_{\delta\mathbf{s}} p(\delta\mathbf{s})\delta\mathbf{s}^T\mathbf{A}\delta\mathbf{s}d\delta\mathbf{s} = Tr(\mathbf{A}\boldsymbol{\Sigma}) + \boldsymbol{\mu}^T\mathbf{A}\boldsymbol{\mu}. \end{aligned} \tag{5.5}$$

Importantly, these properties hold *no matter how the pdf is defined*. This means that, henceforth, we are not required to define the actual pdf, but only its first and second order moments. For an isotropic uniform distribution, defined for the limits $[a_1, \dots a_{2n}]$, we would have $\boldsymbol{\mu} = 0$, and $\boldsymbol{\Sigma}(j,j) = a_j^2/3$. The expected error for the i-th training example is given as:

$$\begin{aligned} \int_{\delta\mathbf{s}} p(\delta\mathbf{s})\|\delta\mathbf{s} - \mathbf{R}f(\mathbf{I}_i, \mathbf{s}_i^* + \delta\mathbf{s})\|_2^2 d\delta\mathbf{s} \approx \\ Tr(\mathbf{A}_i\boldsymbol{\Sigma}) + \boldsymbol{\mu}^T\mathbf{A}_j\boldsymbol{\mu} + 2\boldsymbol{\mu}^T\mathbf{b}_i + \mathbf{x}_i^{*T}\mathbf{R}^T\mathbf{R}\mathbf{x}_i^*. \end{aligned} \tag{5.6}$$

Again, $\mathbf{R}$ is obtained after minimising Equation 5.6 w.r.t. $\mathbf{R}$, in which the derivatives are obtained as follows:

$$\begin{aligned} \frac{\partial}{\partial\mathbf{R}}Tr(\mathbf{A}_i\boldsymbol{\Sigma}) &= 2\mathbf{R}\mathbf{J}_i^*\boldsymbol{\Sigma}\mathbf{J}_i^{*T} - 2\boldsymbol{\Sigma}\mathbf{J}_i^{*T} \\ \frac{\partial}{\partial\mathbf{R}}\boldsymbol{\mu}^T\mathbf{A}_i\boldsymbol{\mu} &= 2\mathbf{R}\mathbf{J}_i^*\boldsymbol{\mu}\boldsymbol{\mu}^T\mathbf{J}_i^{*T} - 2\boldsymbol{\mu}\boldsymbol{\mu}^T\mathbf{J}_i^{*T} \\ \frac{\partial}{\partial\mathbf{R}}2\boldsymbol{\mu}^T\mathbf{b}_i &= 4\mathbf{R}\mathbf{x}_i^*\boldsymbol{\mu}^T\mathbf{J}_i^{*T} - 2\boldsymbol{\mu}\mathbf{x}_i^{*T} \\ \frac{\partial}{\partial\mathbf{R}}\mathbf{x}_i^{*T}\mathbf{R}^T\mathbf{R}\mathbf{x}_i^* &= 2\mathbf{R}\mathbf{x}_i^*\mathbf{x}_i^{*T}. \end{aligned} \tag{5.7}$$

This leads to the closed-form solution:

$$\boxed{\mathbf{R} = \left(\sum_{i=1}^{N}\boldsymbol{\mu}\mathbf{x}_i^{*T} + (\boldsymbol{\Sigma} + \boldsymbol{\mu}\boldsymbol{\mu}^T)\mathbf{J}_i^{*T}\right) \cdot \left(\sum_{i=1}^{N}\mathbf{x}_i^*\mathbf{x}_i^{*T} + 2\mathbf{x}_i^*\boldsymbol{\mu}^T\mathbf{J}_i^{*T} + \mathbf{J}_i^*(\boldsymbol{\Sigma} + \boldsymbol{\mu}\boldsymbol{\mu}^T)\mathbf{J}_i^{*T}\right)^{-1}}. \tag{5.8}$$

The similarities between the closed form error in Equation 5.6 and that of Equation 4.21 are clear. More specifically, if $p(\delta\mathbf{s})$ is defined as a zero-mean uniform distribution with diagonal covariance matrix with entries defined as $\frac{a^2}{3}$ (i.e., defined for the limits $a_1, \ldots, a_{2n}$), the error in Equation 4.21 would be the same as Equation 5.6, but scaled by a volume factor of $V = \prod_k 2a_k$. However, in both cases, Equation 5.8 and Equation 4.23 would be the same. We can see that this volume factor is actually similar to the one that appears in Levin & Shashua (2002), in which a uniform distribution was considered.

Remarkably, Continuous Regression bypasses the question of which distribution should be used when sampling the data: the solution does not depend on the actual sampling pdf, but rather on its first and second order moments. Moreover, aside from the theoretical differences between Continuous Regression and sampling-based methods like SDM, the proposed formulation has important computational advantages: once the training images have been sampled, training a new regressor under a different configuration requires very little computation as opposed to the sampling-based approach. We have seen the computational complexity of both Continuous Regression and sampling-based linear regression in Chapter 4. While in Chapter 4 the benefits of training a different regressor for different chosen limits were shown, now we can extend such a claim to a different set of chosen statistics. That is to say, Continuous Regression does not require the sampling process to be repeated, but rather just changing the sampling statistics. This becomes noteworthy under the context of Cascaded Regression: given the statistics for each level of the cascade with respect to the ground truth, there is no need to sample the images more than once, which yields a huge time saving. Furthermore, we can see that there is a matrix form that can help computing Equation 5.8. Let us introduce the following shorthand notation: $\mathbf{M} = [\boldsymbol{\mu}, \boldsymbol{\Sigma} + \boldsymbol{\mu}\boldsymbol{\mu}^T]$, $\mathbf{B} = \left(\begin{smallmatrix} 1 & \boldsymbol{\mu}^T \\ \boldsymbol{\mu} & \boldsymbol{\Sigma}+\boldsymbol{\mu}\boldsymbol{\mu}^T \end{smallmatrix}\right)$, $\mathbf{D}_i^* = [\mathbf{x}_i^*, \mathbf{J}_i^*]$ and $\bar{\mathbf{D}}^* = [\mathbf{D}_1^*, \ldots, \mathbf{D}_N^*]$. Then:

$$\mathbf{R} = \mathbf{M}\left(\sum_{i=1}^{N} \mathbf{D}_i^*\right)^T \left(\bar{\mathbf{D}}^* \hat{\mathbf{B}} \bar{\mathbf{D}}^{*^T}\right)^{-1}, \tag{5.9}$$

where $\hat{\mathbf{B}} = \mathbf{B} \otimes \mathbf{E}_N$, and $\otimes$ denotes the Kronecker product. Since $\hat{\mathbf{B}}$ is sparse, computing Equation 5.9, once the data is given, is a matter of seconds. This way, we can readily introduce the Continuous Regression into the Cascaded Regression framework.

5.3 Cascaded Continuous Regression

The next step is to naturally extend Continuous Regression to the Cascaded Regression framework. To do so, we can see that the statistics defined in Equation 5.8 correspond to those used in the parallel SDM training proposed in Asthana et al. (2014), described in Chapter 3. In this context, we can train each cascade level using the corresponding $\boldsymbol{\mu}^l$ and $\boldsymbol{\Sigma}^l$. The proposed method is coined Cascaded Continuous Regression (CCR, Sánchez-Lozano, Martinez, Tzimiropoulos & Valstar (2016)). The main advantage of CCR with respect to SDM is that CCR only needs to sample the training images once, given that Equation 5.8 only needs to account for the ground-truth features and their corresponding Jacobians. That is to say, the **sampling process in the CCR training is done only once**.

Sometimes however it is not possible to train a SDM beforehand, from which statistics can be taken. In such cases, we can still train a CCR model by just generating the statistics per-level, by applying the training in a sequential manner. Of course, in such a case we would partially lose the benefits of parallel training. Algorithm 4 summarises the training of CCR when no trained SDM is given. Given the training images and ground-truth points, CCR needs the data to be pre-computed, by computing all $\mathbf{D}_i^*$. Then, the process of CCR training basically consists of updating the initial given statistics $\boldsymbol{\mu}^0$ and $\boldsymbol{\Sigma}^0$ (computed for a training set of annotated videos), to generate a new regressor for each cascade level. The statistics for each level are generated by computing the difference of the outputs of the previous level with respect to the corresponding ground-truth points. For the first level, a set of initial shapes $\mathbf{s}_j^0$ are generated by randomly perturbing the ground-truth with $\boldsymbol{\mu}^0$ and $\boldsymbol{\Sigma}^0$.

5.3.1 Complexity

Now, we can extend the complexity analysis presented in Section 4.5 to the Cascaded Regression framework. We can see that, denoting the cost of sampling an image as $\mathcal{O}(q)$, for a set of N images, K perturbations, and L cascade levels, the total sampling cost of SDM is $\mathcal{O}(LKNq)$. In CCR however, the sampling is done only once, which is $\mathcal{O}(5qN)$, given that extracting

Algorithm 4 CCR Training

Input data: $\{\mathbf{I}_i, \mathbf{s}_i^*, \mathbf{s}_i^0\}_{i=1:N}$, $\boldsymbol{\mu}^0$ and $\boldsymbol{\Sigma}^0$; L levels
Pre-compute: Extract $\bar{\mathbf{D}} = \{\mathbf{D}_i^*\}_{i=1:N}$
Pre-compute: $\hat{\mathbf{D}} = \left(\sum_{i=1}^{N} \mathbf{D}_i^*\right)^T$
for $l = 1$ to L **do**
    Compute $\mathbf{M}_l$ and $\mathbf{B}_l$
    $\mathbf{R}_l = \mathbf{M}_l \hat{\mathbf{D}} \left(\bar{\mathbf{D}} \hat{\mathbf{B}}_l \bar{\mathbf{D}}^T\right)^{-1}$
    Apply $\mathbf{R}_l$ to $\mathbf{s}_i^{(l)}$ to generate $\mathbf{s}_i^{(l+1)}$
    Compute distances $\delta\mathbf{s}_i = \mathbf{s}_i^{(l+1)} - \mathbf{s}_i^*$
    Update $\boldsymbol{\mu}^{l+1} = mean(\{\delta\mathbf{s}\})$ and $\boldsymbol{\Sigma}^{l+1} = cov(\{\delta\mathbf{s}\})$
end for
Return $\{\mathbf{R}_l\}_{l=1...L}$

the ground-truth features and Jacobians is $\mathcal{O}(5q)$ (see Section 4.5). Therefore, CCR training presents a computational advantage, with respect to SDM, of $LK/5$. The configuration used hereinafter is set up to, $L = 4$ and $K = 10$, meaning that the Continuous Regression sampling is 8 times faster (in FLOPS). However, we have to note that, if we were to train a different model **for a different set of statistics**, we would not need to sample any data again. This means that, under a different configuration, we only need Equation 5.9 to be computed, which can be done in seconds. In such a case, we can see that the total sampling cost of SDM training would remain $\mathcal{O}(LKMq)$, while retraining a CCR does not entail any sampling, thus having a null cost. The training set used to generate the final model is of $N \approx 8000$ images, meaning that **the computational saving of retraining a CCR, with respect to SDM, is** $\mathcal{O}(10^6)$.

5.4 Functional Covariance for PCA in CCR

So far, the Continuous Regression is the solution to the Least-Squares problem, where the target variable is considered to belong to a continuum. However, there is an alternative formulation, that would be somehow linked to the work of Levin & Shashua (2002). More specifically, we can see that one can first compute the functional covariance in a similar fashion to that of Levin & Shashua (2002), and then link it to the Least-Squares solution. To do so, we can observe that the normal equations shown in Equation 4.2 can actually be written by means of a covariance

matrix as (Wang et al. 2015)[1]:

$$Cov(X, Y) = Cov(X, X)\mathbf{R}, \tag{5.10}$$

where the term Cov refers to the *data* covariance, as opposed to $\mathbf{\Sigma}$ referring to the covariance of the sampling pdf. While the work of Levin & Shashua (2002) applies a polytope approximation to estimate the covariance matrix, Continuous Regression relies on the first-order Taylor expansion of the feature space. We have to note that Levin & Shashua (2002) approximate the covariance matrix by considering combinations of training samples, while in this case we want to generate perturbations of the training data, and therefore the polytope approximation would not be adequate for solving the Least-Squares problem in the perturbed space of images. We can compute the functional covariance matrix as follows:

$$Cov(X, X) = \sum_i \int_{\delta\mathbf{s}} p(\delta\mathbf{s}) f(\mathbf{I}, \mathbf{s}_i^* + \delta\mathbf{s}) f(\mathbf{I}, \mathbf{s}_i^* + \delta\mathbf{s})^T d\delta\mathbf{s}. \tag{5.11}$$

If we approximate the input features by its first-order Taylor expansion, we can see that:

$$\begin{aligned} f(\mathbf{I}, \mathbf{s}_i^* + \delta\mathbf{s}) f(\mathbf{I}, \mathbf{s}_i^* + \delta\mathbf{s})^T \approx \\ \mathbf{x}_i^* \mathbf{x}_i^{*^T} + \mathbf{x}_i^* \delta\mathbf{s}^T \mathbf{J}_i^{*^T} + \mathbf{J}_i^* \delta\mathbf{s} \mathbf{x}_i^{*^T} + \mathbf{J}_i^* \delta\mathbf{s} \delta\mathbf{s}^T \mathbf{J}_i^{*^T}. \end{aligned} \tag{5.12}$$

We can use Equation 5.5, and the fact that $\int p(\delta\mathbf{s}) \delta\mathbf{s} \delta\mathbf{s}^T = \mathbf{\Sigma} + \boldsymbol{\mu}\boldsymbol{\mu}^T$, to further expand the covariance as:

$$Cov(X, X) = \sum_i \mathbf{x}_i^* \mathbf{x}_i^{*^T} + 2\mathbf{x}_i^* \boldsymbol{\mu}^T \mathbf{J}_i^{*^T} + \mathbf{J}_i^* (\mathbf{\Sigma} + \boldsymbol{\mu}\boldsymbol{\mu}^T) \mathbf{J}_i^{*^T}, \tag{5.13}$$

which is, exactly, the invertible part of Equation 5.8. We can readily see that the covariance can be expressed as:

$$Cov(X, X) = \bar{\mathbf{D}}^* \hat{\mathbf{B}} (\bar{\mathbf{D}}^*)^T. \tag{5.14}$$

[1]Without loss of generality, we can assume that the input data and the shape displacements are zero-mean

Similarly, we can see that:

$$Cov(X,Y) = \sum_i \int_{\delta\mathbf{s}} \delta\mathbf{s} f(\mathbf{I}, \mathbf{s}_i^* + \delta\mathbf{s})^T d\delta\mathbf{s} \approx$$
$$\approx \sum_i \boldsymbol{\mu}\mathbf{x}_i^{*T} + (\boldsymbol{\Sigma} + \boldsymbol{\mu}\boldsymbol{\mu}^T)\mathbf{J}_i^{*T}. \tag{5.15}$$

Obtaining Equation 5.8 from Equation 5.10 is straightforward given Equation 5.15 and Equation 5.14.

Aside the theoretical connection of Continuous Regression with previous work, it is worth noting that we can take advantage of this approach to generate the PCA models for each of the cascade levels (recall the problem of PCA in Continuous Regression from Section 4.5). These models are used to reduce the dimensionality of the feature space, and therefore need to be computed in the perturbed image space. This way, we can easily generate each of the models just by applying an eigen decomposition of the functional covariance of Equation 5.14. The invertible part of Equation 5.8 corresponds to the functional covariance matrix, supporting the theoretical assumption that Continuous Regression is the natural extension of sampling-based regression to the infinite set of perturbations, where the samples are approximated using the ground-truth data.

5.5 Geometric interpretation

We have seen that, for a zero-mean uniform distribution with diagonal covariance, the empirical error would be the same as the expected error, but scaled by a volume factor. This volume factor comes from the fact that the classical Functional Regression, formulated with respect to the Lebesgue measure, does not consider that the measure of the whole manifold of shape displacements should sum up to one, as is the case of the probability measure. We can see that using the Lebesgue measure, the error would diverge for unbounded limits.

Furthermore, we can analyse the actual meaning of the "data term" in the Continuous Regression framework, i.e., the role of a pdf given that an infinite set of samples is taken. To do

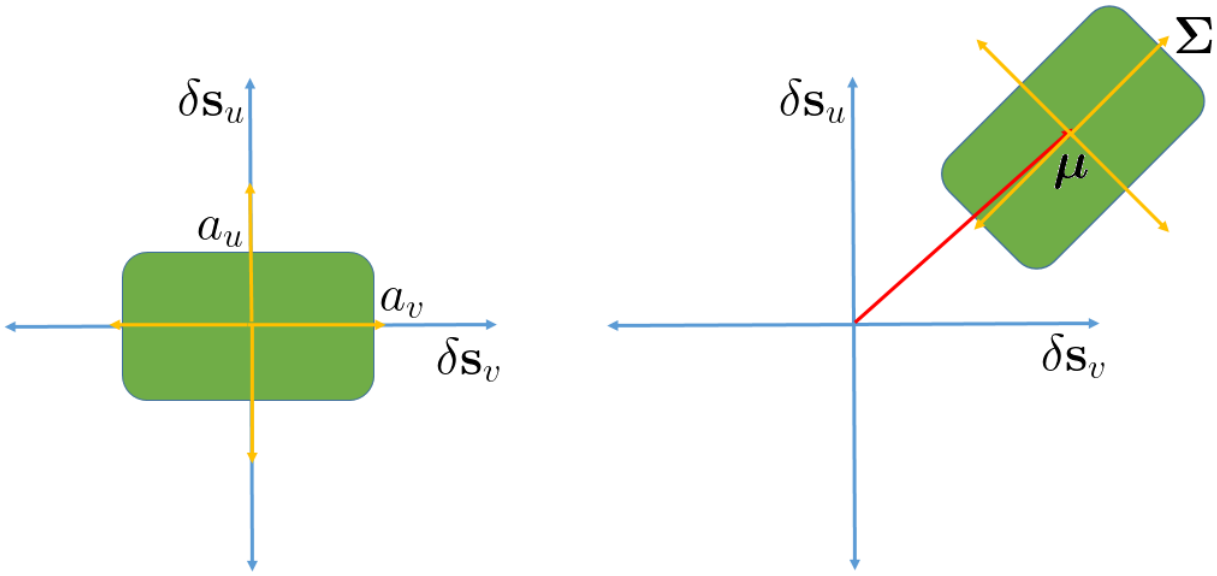

Figure 5.3: Differences between the classical Functional Regression formulation (**Left**) and the CR in correlated variables (**Right**). The shadowed green area represents the volume within which samples are taken. Left image corresponds to a diagonal covariance matrix, with entries defined as $\frac{a^2}{3}$. Right image corresponds to a full covariance matrix and a non-zero mean vector. The sampling space is moved to the centre of coordinates defined by $\boldsymbol{\mu}$, and rotated according to the eigenvectors of $\boldsymbol{\Sigma}$.

so, the solution presented through this Chapter can be seen from the Information Geometry perspective (Amari 1985, Pennec 2006), in which a geometrical interpretation can be applied to the probability measure. In the uncorrelated case, i.e. when using a diagonal covariance matrix, the manifold of shape displacements is a $2n$-dimensional parallelepiped, spanned by the Cartesian basis, defined for each of the shape dimensions. When we extend the Continuous Regression to full covariance matrices, we are actually rotating the manifold of shape displacements according to the eigenvectors of $\boldsymbol{\Sigma}$. Furthermore, the origin of the coordinates axes is displaced to $\boldsymbol{\mu}$. Figure 5.3 illustrates the geometrical interpretation in a graphical manner.

Summarising, we can think of Continuous Regression as follows: instead of generating samples, we "ask" the Jacobians how the input space would be in a point defined by $\delta\mathbf{s}$. Given the reliability of such predictions, we only need to store the image features and Jacobians. Then, the mathematical solution presented in this thesis allows to take the infinite $\delta\mathbf{s}$ within a defined space, and then sum over all of them. That is to say, strictly speaking, the Continuous Regression sums over the infinite approximations given by the Taylor expansion of the image features

in the ground-truth positions, within the manifold of shape displacements $\delta\mathbf{s}$, defined as a $2n$-d parallelepiped, with volume defined by the limits chosen for each of the points, centred at the point described by the mean vector $\boldsymbol{\mu}$, and rotated according to the eigenvectors of $\boldsymbol{\Sigma}$. If we want to train a new model using a different set of statistics, we just "ask" the stored data (ground-truth features and Jacobians), how "they think" the perturbed space would be in that specific region. With Continuous Regression, we can simply use the statistics to "ask", with a single shot, how the set of infinite samples within a specific volume would be approximated by the data.

5.6 Use of a PDM

We can also extend the Continuous Regression formulation to parameterised shapes, applying the parameterisation shown in Chapter 2. We have seen that the SDM was first formulated aiming to predict the location of the n landmarks composing a shape. However, the SDM can be easily extended to the use of a Shape Model. For instance, Asthana et al. (2014) presented the parallel-SDM (known as Chehra) to predict the shape parameters. The idea is exactly the same, but now, instead of working with the direct estimation of the landmarks, we aim to predict the shape parameters.

Let us first recall the main components of a PDM. In a PDM, a shape $\mathbf{s}$ is parameterised in terms of $\mathbf{p} = [\mathbf{q}, \mathbf{c}] \in \mathbb{R}^{4+k}$, where $\mathbf{q} \in \mathbb{R}^4$ represents the rigid parameters and $\mathbf{c}$ represents the flexible shape parameters, so that $\mathbf{s} = t_{\mathbf{q}}(\mathbf{s}_0 + \mathbf{B}_s\mathbf{c})$, where t represents the rigid information (scale, rotation, and translation), parameterised by $\mathbf{q}$. We have previously seen that $\mathbf{B}_s \in \mathbb{R}^{2n\times k}$ and $\mathbf{s}_0 \in \mathbb{R}^{2n}$ are learnt during training and represent the linear subspace of flexible shape variations. Under the use of a PDM, the Continuous Regression formulation would be simply transformed into:

$$\sum_{i=1}^{N} \int_{\delta\mathbf{p}} p(\delta\mathbf{p}) \|\delta\mathbf{p} - \mathbf{R} f(\mathbf{I}_i, \mathbf{p}_i^* + \delta\mathbf{p})\|_2^2 d\delta\mathbf{p}, \tag{5.16}$$

where now the Jacobians need to be computed with respect to the PDM. To do so, it suffices

to apply the chain rule:

$$\frac{\partial f(\mathbf{I}, \mathbf{p} + \delta\mathbf{p})}{\partial \mathbf{p}} = \frac{\partial f(\mathbf{I}, \mathbf{p} + \delta\mathbf{p})}{\partial \mathbf{s}} \frac{\partial \mathbf{s}}{\partial \mathbf{p}}, \tag{5.17}$$

where $\frac{\partial \mathbf{s}}{\partial \mathbf{p}}$ can be analytically computed from the shape model, and $\frac{\partial f(\mathbf{I}, \mathbf{p}+\delta\mathbf{p})}{\partial \mathbf{s}}$ is computed as in Equation 4.9. This way, the Continuous Regression solution is exactly the same as shown in Equation 5.8, with the Jacobians now being computed with respect to the PDM.

That is to say, when using a PDM, the only thing that changes in our setting is that now the derivatives need to be computed with respect to the shape parameters. The derivative of the shape with respect to the PDM can be computed from the definition shown in Chapter 2. We can recall that the shape $\mathbf{s}$ is a function of its shape parameters:

$$\mathbf{s}(\mathbf{q}, \mathbf{c}) = vec\left(\begin{bmatrix} 1 + q_1/\|\mathbf{s}_0\| & q_2/\|\mathbf{s}_0\| \\ -q_2/\|\mathbf{s}_0\| & 1 + q_1/\|\mathbf{s}_0\| \end{bmatrix} \begin{bmatrix} (\mathbf{B}_s^x \mathbf{c} + \mathbf{s}_0^x)^T \\ (\mathbf{B}_s^y \mathbf{c} + \mathbf{s}_0^y)^T \end{bmatrix} + \begin{bmatrix} \frac{q_3}{\sqrt{n}} \\ \frac{q_4}{\sqrt{n}} \end{bmatrix} \mathbf{1}_n^T \right). \tag{5.18}$$

Then, computing the derivatives of the PDM is straightforward. We can see that $\frac{\partial \mathbf{s}}{\partial \mathbf{p}} = [\frac{\partial \mathbf{s}}{\partial \mathbf{q}}, \frac{\partial \mathbf{s}}{\partial \mathbf{c}}]$. Let $\tilde{\mathbf{s}} = vec(\mathbf{s}_0 + \mathbf{B}_s\mathbf{c}) = (\tilde{x}_1, \ldots, \tilde{x}_n, \tilde{y}_1, \ldots, \tilde{y}_n)^T$ be the non-rigid reconstruction of the shape, and let $\hat{\mathbf{s}} = (-\tilde{y}_1, \ldots, -\tilde{y}_n, \tilde{x}_1, \ldots, \tilde{x}_n)^T$, then:

$$\frac{\partial \mathbf{s}}{\partial \mathbf{q}} = \left[\frac{1}{\|\mathbf{s}_0\|}\tilde{\mathbf{s}}, \frac{1}{\|\mathbf{s}_0\|}\hat{\mathbf{s}}, \mathbf{b}_3, \mathbf{b}_4 \right]. \tag{5.19}$$

Now, for the derivative of the PDM w.r.t. the non-rigid shape parameters, we can see that, if $\mathbf{P} = \begin{bmatrix} 1+q_1/\|\mathbf{s}_0\| & q_2/\|\mathbf{s}_0\| \\ -q_2/\|\mathbf{s}_0\| & 1+q_1/\|\mathbf{s}_0\| \end{bmatrix}$, then, $\frac{\partial \mathbf{s}}{\partial \mathbf{c}_i} = vec(\mathbf{P}\,[\mathbf{B}_s^{x_i}, \mathbf{B}_s^{y_i}])$. We can express $\frac{\partial \mathbf{s}}{\partial \mathbf{c}}$ in a compact form as:

$$\frac{\partial \mathbf{s}}{\partial \mathbf{c}} = (\mathbf{P} \otimes \mathbf{1}_n)\, \mathbf{B}_s \tag{5.20}$$

The reader might be wondering whether the use of a PDM would be beneficial or not. Let us recall from Section 4.5, that the computational cost of Continuous Regression was a function of the number of dimensions being considered. When working with points, we have:

$$\mathcal{C}_{cont}^t = \mathcal{O}(D^3) + \mathcal{O}((2n+1)ND^2) + \mathcal{O}(5qN)\,, \mathcal{C}_{cont}^m = \mathcal{O}(D(2n+1)) \tag{5.21}$$

When working with shape parameters, we have that now shapes are represented by $m = k + 4$ parameters. Therefore, both the timing and memory complexity benefit from working with a PDM. The complexity is reduced to

$$\mathcal{C}^t_{cont} = \mathcal{O}(D^3) + \mathcal{O}((m+1)ND^2) + \mathcal{O}(5qN)\,,\, \mathcal{C}^m_{cont} = \mathcal{O}(D(m+1)). \tag{5.22}$$

Given that the number of points being considered in this thesis is $2n = 132$, and the total number of parameters is $m = 24$, the computational saving is obvious. Besides, this becomes crucial for the recursive least-squares problem (i.e., incremental learning), which will be presented in Chapter 6.

5.7 Experiments

This Section empirically analyses the contributions of Continuous Regression presented in this Chapter. As in Chapter 4, we are interested in evaluating the different aspects of each of the elements introduced in this Chapter, without targeting state of the art results. However, the experiments are carried out on the benchmark for video tracking, that is ultimately the database upon which the tracker resulting from this thesis attains state of the art results. The experiments are evaluated by means of CED curves (Cumulative Error Distribution). We can recall from Chapter 4 that these represent the percentage of images (y-axis) with an error lower than a given threshold (x-axis). It is an increasing function, meaning that, the closer the curve to the top of the plot, the better the results.

5.7.1 Equivalence with SDM

Once Cascaded Continuous Regression has been presented, we need to ensure its performance is capable of competing with the sampling-based counterpart, the SDM. To do so, both methods were used to train each model, under the same conditions, and were evaluated on the testing videos of the 300VW dataset (Shen et al. 2015) corresponding to Category 3 (a complete

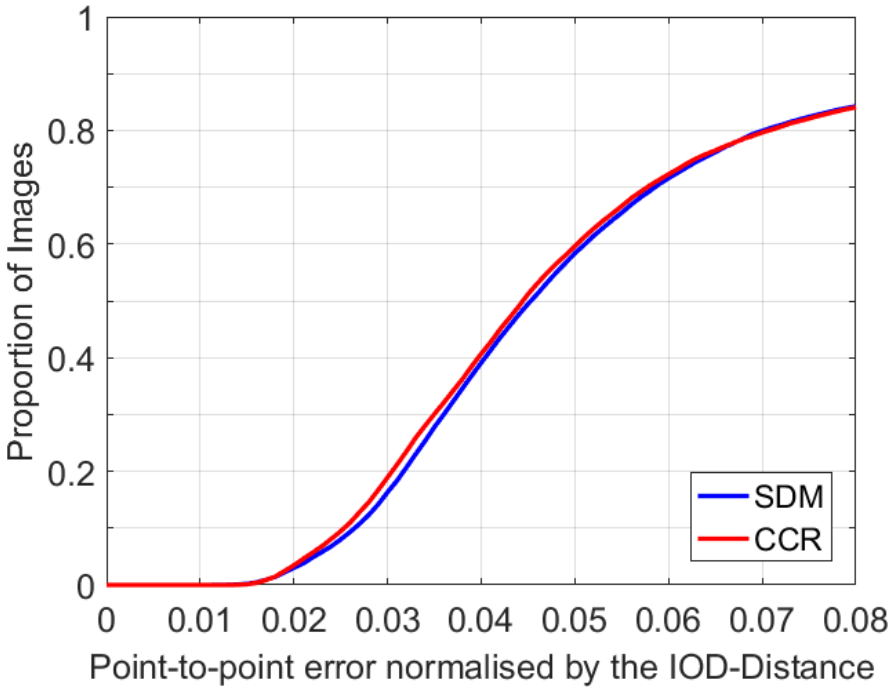

Figure 5.4: Cumulative Error Distribution (CED) curve for both the CCR and SDM methods in the test partition of the 300VW. The Area Under the Curve for both methods is the same. This illustrates that CCR and SDM are equivalent.

description of the experimental set-up can be found in Section 6.4.1). More specifically, both methods were trained using the training partition of Helen dataset (Le et al. 2012). Results are shown in Figure 5.4. It can be seen that the CCR and the SDM provide similar performance, thus implying that a first-order Taylor expansion is enough to achieve the same performance as SDM.

5.7.2 The impact of the data term

Furthermore, an experiment was designed to empirically demonstrate the theoretical influence of the data term, which allows for correlated variables. This experiment compares the correlated Continuous Regression approach (cor-CR), with the original (uncorrelated) one (uncor-CR), in which the limits are chosen for each point and coordinate separately. To do so, the covariance matrix for the landmark displacements from the training set of the 300-VW dataset (Shen et al. 2015) is computed (see Section 6.4). In cor-CR, a full covariance matrix is used, whereas

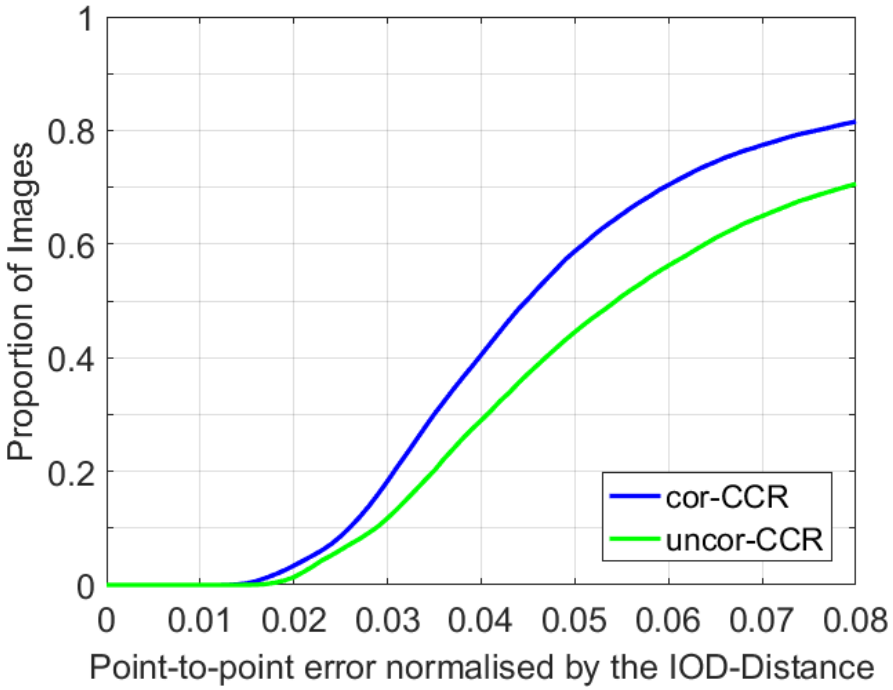

Figure 5.5: Cumulative Error Distribution (CED) curve for both the uncor-CR (green) and cor-CR (blue). Results are shown for the 49 points configuration. The contribution of a full covariance matrix is clear

in uncor-CR, the covariance matrix is forced to be diagonal. Other components, such as the number of cascade levels, or the PCA components, remain the same for both methods. Both models were evaluated in the most challenging subset, using the 300-VW error measure. Results in Figure 5.5 show the Cumulative Error Distribution (CED). It is immediate to see that the contribution of the cor-CR is significant, and that shape dimensions are clearly correlated. The result of this experiment clearly demonstrates the importance of solving the Continuous Regression in spaces of correlated variables.

5.7.3 Use of a PDM

Finally, it is worth evaluating the performance of CCR when using a PDM. To do so, two models were identically trained using the Helen database, and were tested in Category 3 of the 300VW dataset. Results are shown in Figure 5.6. We can immediately see that both models attain the same results, thus meaning that the computational advantages of using a PDM do not affect

performance. However, it is worth noting that the use of a PDM "decorrelates" the dimensions, thus meaning that the covariance becomes almost diagonal (in a general scenario, which does not necessarily need to always be the case). In such a case, we can see that the influence of using a data term is not as big as previously shown. However, the fact that the covariance matrix does not become strictly linear (see Figure 5.7) still implies that not considering the new measure introduced in this Chapter would result in a drop in performance. This is illustrated in Figure 5.8. Thus, the theoretical contributions of this Chapter still apply to the use of PDMs, although the impact of the data term is less obvious when evaluating the results using it.

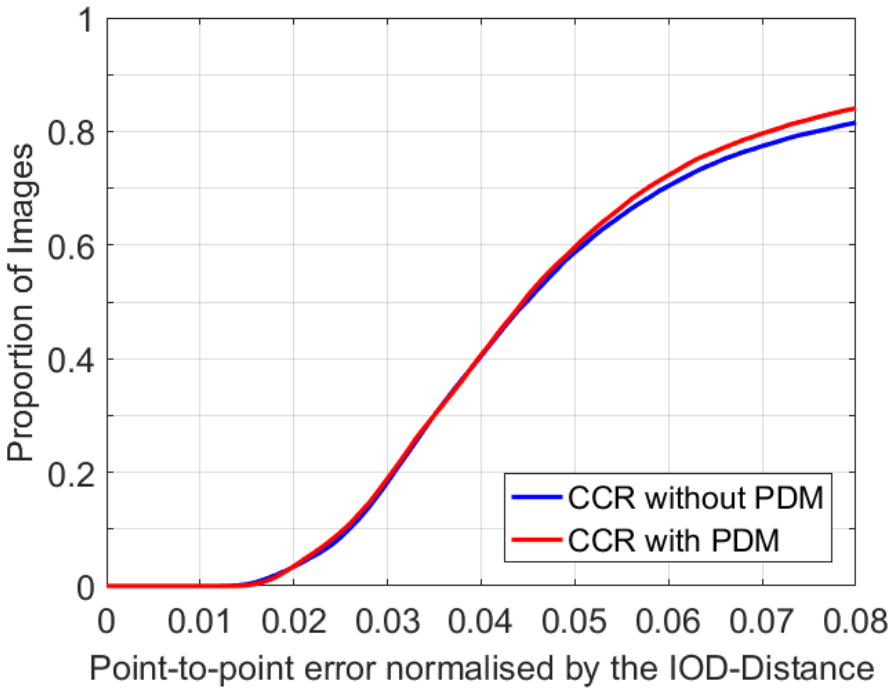

Figure 5.6: Performance achieved by a model using a PDM, and a model not using it. The results show that both methods perform equally, and therefore the use of a PDM is beneficial, given its computational advantages.

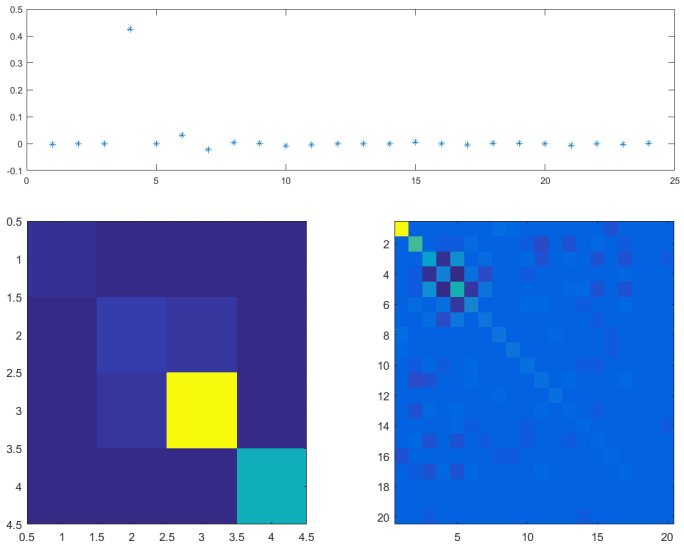

Figure 5.7: **Top**: Mean shape parameters displacement across the training set of videos ($\boldsymbol{\mu}$). **Bottom**: Covariance matrix ($\boldsymbol{\Sigma}$) of shape parameter displacements. For the sake of visualisation, left image shows the covariance of rigid parameters, and right image shows the covariance of non-rigid parameters.

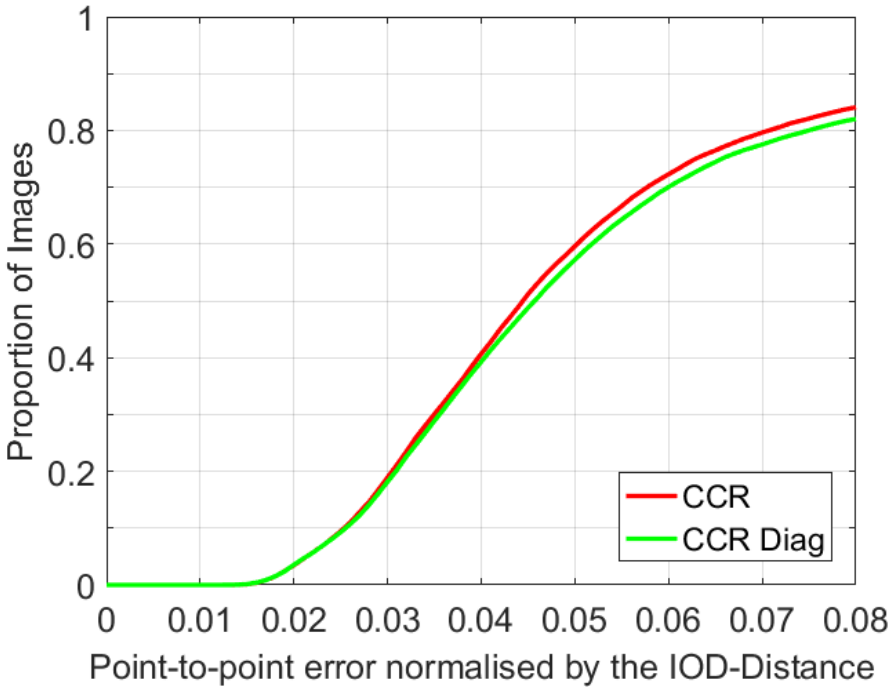

Figure 5.8: Results comparing a model trained using full covariance matrices with a model trained using only diagonal covariance matrices. Both models use the same PDM. We can see that, given that the covariance matrix is not completely diagonal, the use of a full covariance matrix, and hence the approach presented in this Chapter, is still beneficial.

Chapter 6

Incremental Cascaded Continuous Regression

Hitherto, Cascaded Continuous Regression can be seen as the cheapest method, computationally speaking, among the family of Cascaded Regression methods. We have seen that it implies a huge computational saving with respect to sampling-based linear regression, whilst achieving the same performance. Moreover, Continuous Regression does not need to generate perturbations, but rather accounts for the infinite approximations yielded by the Taylor expansion of the image features, in a neighbourhood surrounding the ground-truth points.

There is still a problem of Cascaded Regression methods that is yet to be solved: the development of an effective and computationally reliable method for incremental learning. Basically, SDM, CCR, and in general Cascaded Regression methods, are trained using an extensive training set, so that the generalisation of their models is good enough to track under unconstrained conditions. That is to say, under different configurations in illumination, pose, and identity. However, a generic model will rarely perform as well as a person-specific one. If we know in advance the specific conditions the tracker is meant to work with, then we can reduce the variance of each of the regressors to work with the proper conditions that are specific to a sequence. For example, let us assume we have a model trained on the LFPW training set partition, and a model trained using the first 100 frames of a specific video. The LFPW model was trained

using $>$ 800 images, whereas the person-specific model is built using only 100 images. The person-specific models were trained using the statistics per level generated after training the LFPW model. Figure 6.1 shows two examples extracted from Category 3 of 300VW dataset. From the curves shown in Figure 6.1, it is clear that person-specific models clearly outperform generic models when the conditions are known.

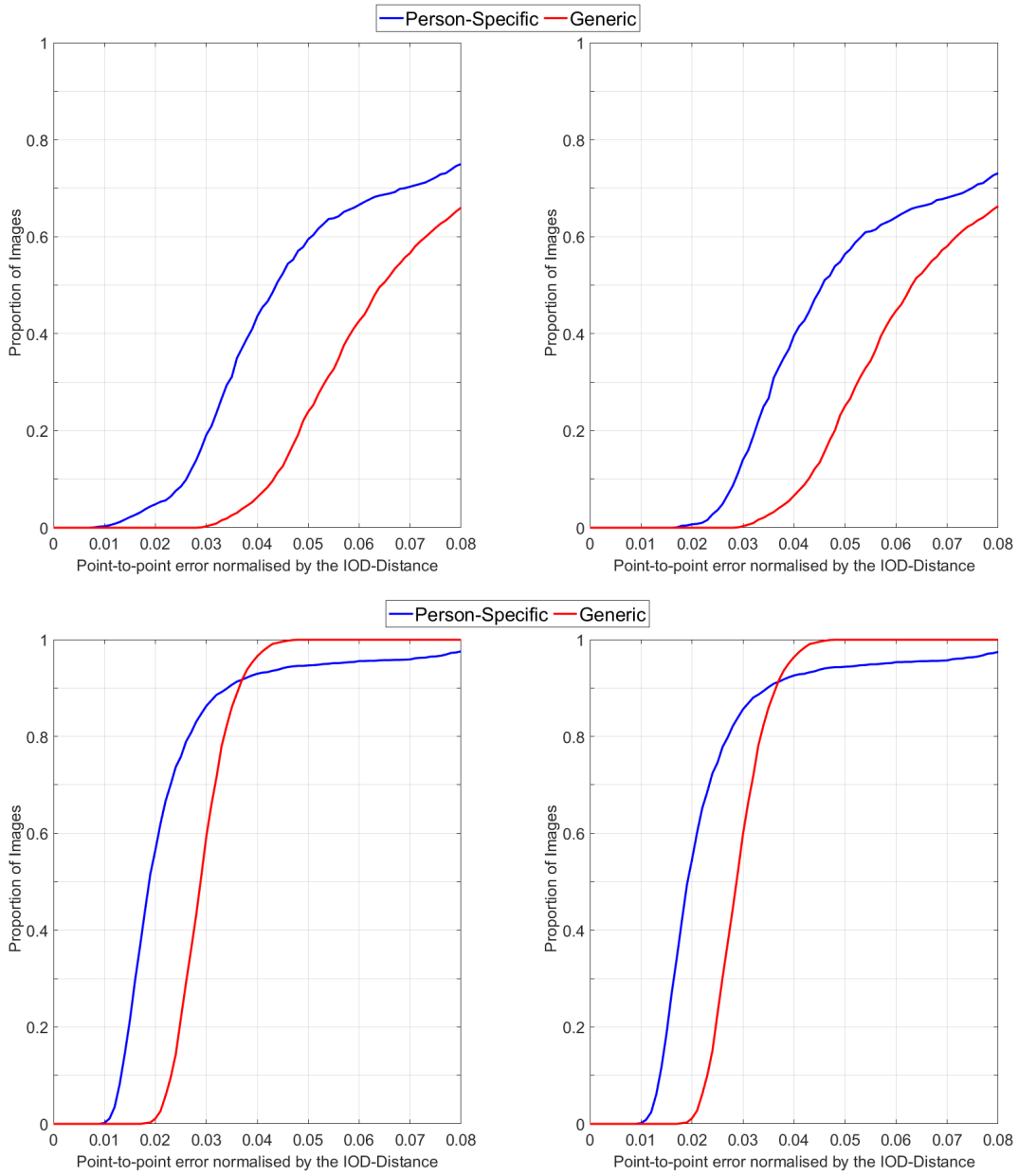

Figure 6.1: Cumulative curves for a generic model (red) trained using the LFPW training partition set, and using the first 100 frames of the specific video (blue). Left image shows the cumulative error for the whole video, and right image shows the cumulative error for frames 101 to end.

However, building a person-specific model is in practice not tractable in most cases. Therefore, in order to improve the generalisation capabilities of generic models, we want to incorporate new images to the model, corresponding to the subject being tracked. Of course, we want to introduce these images to the models under certain conditions. For instance, we will not incorporate images we might know to have been wrongly tracked, because such information would cause the tracker to drift. Also, we need this process to be as fast as possible. If the update is too slow, then the scope of the tracker would be limited to off-line videos. In such cases, when real-time requirements are no longer an issue, it will be better track the whole video and add all correctly tracked frames to the model, and then track the video from the beginning.

As of the end of 2016, there were no existing tracking methods using Cascaded Regression capable of incorporating person-specific information in real-time. To date, only the SDM (Xiong & la Torre 2014) and Chehra (Asthana et al. 2014) have mentioned the possibility of performing incremental learning within the context of Cascaded Regression, but none of them are capable of working in real-time. Actually, the SDM does not incorporate incremental learning, and only such a possibility is mentioned in Xiong & la Torre (2014). However, the sequential training of an SDM clearly impedes any reliable form of incremental learning. The parallel-SDM (Chehra, Asthana et al. (2014)) is reported to be as slow as 4 seconds per frame. We will shortly see why.

This Chapter presents Incremental Cascaded Continuous Regression (iCCR), a Cascaded Regression framework using Continuous Regression incorporated with real-time incremental learning capabilities. As such, the Matlab prototype, of which a protected version has been publicly released, is capable of working at 5 frames per second, using a sequential update. This clearly brings to iCCR the possibility of real-time incremental learning, something not possible under existing approaches.

In this chapter, the SDM and CCR update rules are presented first followed by an analysis of their complexity. Finally, a fully experimental set-up is presented, and both the iCCR and CCR methods are evaluated under the same conditions and compared against those methods

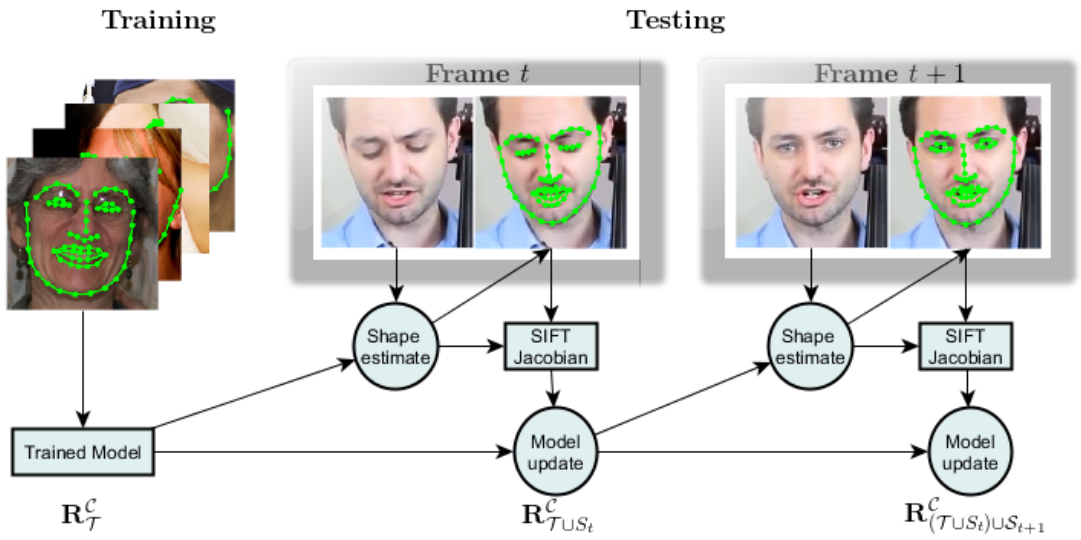

Figure 6.2: Overview of our incremental cascaded continuous regression algorithm (iCCR). The originally model $\mathbf{R}_{\mathcal{T}}$ learned offline is updated with each new frame, thus sequentially adapting to the target face.

that were ranked winner and runner-up on the 300VW dataset. We will see that, thanks to the incremental learning capabilities, the iCCR tracker is capable of competing with, and even surpassing, state of the art methods.

6.1 Introduction

Once a regressor has been trained, we might want to incorporate new images to the model, without computing Equation 5.8 again. This is very important for the task of face tracking, since it has been reported that generic models perform worse than person-specific models. Since training a person-specific model is something not possible in most cases, adding samples to a generic model as the face is being tracked might help the model to later track a specific person better. An overview of the incremental learning procedure is depicted in Figure 6.2. However, we can not retrain the models again incorporating new images, as it would be far too slow, and therefore is not computationally tractable. Some previous works (Xiong & la Torre 2014, Asthana et al. 2014) have attempted to incorporate online learning to their current models, by just applying a recursive least squares to a trained model. However, both methods are very slow, and thus are impractical for real-time tracking.

The incremental learning rules for Linear Regression are simply derived from the well-known recursive least-squares problem. The recursive least-squares problem has been widely used, e.g., in the field of Signal Processing, to update the recovery of noisy signals. Therefore, while Xiong & la Torre (2014) only pointed out that the nature of SDM would allow the use of incremental learning, it was the work of Asthana et al. (2014) that realised that the sequential-SDM would make incremental learning far too slow, and then proposed the parallel-SDM, upon which the recursive least-squares would be easier to use.

We can recall from Chapter 3 the solution presented by Asthana et al. (2014). For a given cascade level l, a regressor $\mathbf{R}_{\mathcal{T}}$ has been trained using the training set $\mathcal{T}$ (the level is omitted in the derivation below for the sake of clarity). Then, we want to update the trained model with a new set of images $\mathbf{I}_{\mathcal{S}}$, for which the ground-truth *shape parameters* $\mathbf{p}^*_{\mathcal{S}}$ are either known (in the case of annotated images), or estimated (in the case of tracking). That is to say, the recursive least-squares does not depend on the nature of the update data. In sampling-based linear regression, we generate perturbations of the ground-truth parameters, given the statistics $\boldsymbol{\mu}^l$ and $\boldsymbol{\Sigma}^l$ for each level l. This way, we generate a set of random perturbations $\delta\mathbf{p}^{l,k}_{\mathcal{S}}$, which are used to extract the features $f(\mathbf{I}_n, \mathbf{p}^*_n + \delta\mathbf{p}^{l,k}_n)_{n\in\mathcal{S}}$. Let $\mathbf{X}_{\mathcal{S}}$ be the matrix storing all $\mathbf{x}^{l,k}_n$, and $\mathbf{Y}_{\mathcal{S}}$ be the matrix storing the corresponding $\delta\mathbf{p}^{l,k}_n$. Then, a regressor considering $\mathbf{X}_{\mathcal{T}\cup\mathcal{S}} = [\mathbf{X}_{\mathcal{T}}\,,\,\mathbf{X}_{\mathcal{S}}]$ and $\mathbf{Y}_{\mathcal{T}\cup\mathcal{S}} = [\mathbf{Y}_{\mathcal{T}}\,,\,\mathbf{Y}_{\mathcal{S}}]$, would be computed as

$$\mathbf{R}_{\mathcal{T}\cup\mathcal{S}} = \mathbf{Y}_{\mathcal{T}\cup\mathcal{S}}\mathbf{X}^T_{\mathcal{T}\cup\mathcal{S}}\left(\mathbf{X}_{\mathcal{T}\cup\mathcal{S}}\mathbf{X}^T_{\mathcal{T}\cup\mathcal{S}}\right)^{-1}. \tag{6.1}$$

We have analysed in Chapter 4 the computational complexity of Equation 6.1, which would be dominated by the inverse of the covariance matrix, which is $\mathcal{O}(d^3)$ (assuming the feature space to have undergone a dimensionality reduction). However, we can apply the Woodbury identity to the inverse of the covariance matrix (Brookes 2011)[1]. Denoting the covariance matrix as $\mathbf{V}_{\mathcal{T}\cup\mathcal{S}} = \mathbf{X}_{\mathcal{T}\cup\mathcal{S}}\mathbf{X}^T_{\mathcal{T}\cup\mathcal{S}}$, and assuming $K = \#\mathcal{S}$ the number of samples to be updated, we can see that

$$\mathbf{V}^{-1}_{\mathcal{T}\cup\mathcal{S}} = \left(\mathbf{V}_{\mathcal{T}} + \mathbf{X}_{\mathcal{S}}\mathbf{X}^T_{\mathcal{S}}\right)^{-1} = \mathbf{V}^{-1}_{\mathcal{T}} - \mathbf{V}^{-1}_{\mathcal{T}}\mathbf{X}_{\mathcal{S}}\left(\mathbf{E}_K + \mathbf{X}^T_{\mathcal{S}}\mathbf{V}^{-1}_{\mathcal{T}}\mathbf{X}_{\mathcal{S}}\right)^{-1}\mathbf{X}^T_{\mathcal{S}}\mathbf{V}^{-1}_{\mathcal{T}}. \tag{6.2}$$

[1]The Woodbury identity establishes the equality $(A+UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1}+VA^{-1}U)^{-1}VA^{-1}$

Therefore, we only need to store the inverse of the covariance matrix, not the actual one. Furthermore, we can see that the matrix to be inverted becomes K-dimensional, where K is the number of samples to be updated. Thus, computing the updated covariance matrix becomes feasible. Using this property, Asthana et al. (2014) rearranged the update rules as:

$$\mathbf{U} = (\mathbf{E}_K + \mathbf{X}_{\mathcal{S}}^T \mathbf{V}_{\mathcal{T}}^{-1} \mathbf{X}_{\mathcal{S}})^{-1} \tag{6.3}$$

$$\mathbf{Q} = \mathbf{X}_{\mathcal{S}} \mathbf{U} \mathbf{X}_{\mathcal{S}}^T \mathbf{V}_{\mathcal{T}}^{-1} \tag{6.4}$$

$$\mathbf{V}_{\mathcal{T}\cup\mathcal{S}}^{-1} = \mathbf{V}_{\mathcal{T}}^{-1} - \mathbf{V}_{\mathcal{T}}^{-1} \mathbf{Q} \tag{6.5}$$

$$\mathbf{R}_{\mathcal{T}\cup\mathcal{S}} = \mathbf{R}_{\mathcal{T}} - \mathbf{R}_{\mathcal{T}} \mathbf{Q} + \mathbf{Y}_{\mathcal{S}} \mathbf{X}_{\mathcal{S}}^T \mathbf{V}_{\mathcal{T}\cup\mathcal{S}}^{-1}. \tag{6.6}$$

Furthermore, Asthana et al. (2014) pointed out that in updating one sample at a time, the inverse becomes one-dimensional, and therefore the update is extremely efficient. However, this approach has a bottleneck that can not be overcome by reducing the number of samples to be updated. Actually, using this approach, updating one sample each time is harmful for the updating process, in terms of computational complexity. The main bottleneck of this approach resides in the cost of computing $\mathbf{V}_{\mathcal{T}}^{-1}\mathbf{Q}$. We can see that $\mathbf{V}_{\mathcal{T}}^{-1} \in \mathbb{R}^{d\times d}$, and $\mathbf{Q} \in \mathbb{R}^{d\times d}$ thus meaning that $\mathbf{V}_{\mathcal{T}}^{-1}\mathbf{Q}$ is $\mathcal{O}(d^3)$. However, if we rearrange the update rules as follows:

$$\mathbf{Z} = \mathbf{X}_{\mathcal{S}}^T \mathbf{V}_{\mathcal{T}}^{-1} \tag{6.7}$$

$$\mathbf{U} = (\mathbf{E}_K + \mathbf{Z}\mathbf{X}_{\mathcal{S}})^{-1} \tag{6.8}$$

$$\mathbf{Q} = \mathbf{X}_{\mathcal{S}} \mathbf{U} \mathbf{Z} \tag{6.9}$$

$$\mathbf{V}_{\mathcal{T}\cup\mathcal{S}}^{-1} = \mathbf{V}_{\mathcal{T}}^{-1} - \mathbf{Z}^T \mathbf{U} \mathbf{Z} \tag{6.10}$$

$$\mathbf{R}_{\mathcal{T}\cup\mathcal{S}} = \mathbf{R}_{\mathcal{T}} - \mathbf{R}_{\mathcal{T}} \mathbf{Q} + \mathbf{Y}_{\mathcal{S}} \mathbf{X}_{\mathcal{S}}^T \mathbf{V}_{\mathcal{T}\cup\mathcal{S}}^{-1}, \tag{6.11}$$

we can see that the most expensive operation becomes $\propto \mathcal{O}(d^2 K)$, given that $\mathbf{V}_{\mathcal{T}}^{-1}$, which is the only $d \times d$ matrix, is only multiplied by $\mathbf{X}_{\mathcal{S}}^T \in \mathbb{R}^{K\times d}$. This way, the computational complexity is reduced by one order of magnitude with respect to d, which is the highest valued variable.

6.2 Incremental Cascaded Continuous Regression

Now, we can devise the incremental learning update rule for Continuous Regression. To do so, we again assume that we have a regressor $\mathbf{R}_{\mathcal{T}}$, trained using Equation 5.9, on a training set $\mathcal{T}$. Also, the (functional) covariance matrix for the training data is denoted as $\mathbf{V}_{\mathcal{T}} := Cov(X, X) = \bar{\mathbf{D}}^*_{\mathcal{T}}\hat{\mathbf{B}}(\bar{\mathbf{D}}^*_{\mathcal{T}})^T$. Incremental learning aims to update $\mathbf{R}_{\mathcal{T}}$ with a new image $\mathcal{S}$, for which the ground-truth image features and Jacobians are extracted. Let $\mathbf{D}^*_{\mathcal{S}}$ be the data corresponding to the updating image[2]. We would compute the new regressor as:

$$\mathbf{R}_{\mathcal{T}\cup\mathcal{S}} = \mathbf{M}\left(\sum_{i=1}^{N}\mathbf{D}^*_j + \mathbf{D}^*_{\mathcal{S}}\right)^T (\mathbf{V}_{\mathcal{T}\cup\mathcal{S}})^{-1}, \tag{6.12}$$

where

$$(\mathbf{V}_{\mathcal{T}\cup\mathcal{S}})^{-1} = \left(\mathbf{V}_{\mathcal{T}} + \mathbf{D}^*_{\mathcal{S}}\mathbf{B}\mathbf{D}^{*T}_{\mathcal{S}}\right)^{-1}, \tag{6.13}$$

we recall $\mathbf{B} = \begin{pmatrix} 1 & \boldsymbol{\mu}^T \\ \boldsymbol{\mu} & \boldsymbol{\Sigma}+\boldsymbol{\mu}\boldsymbol{\mu}^T \end{pmatrix}$. Again, we can apply the Woodbury identity to Equation 6.13:

$$\mathbf{V}_{\mathcal{T}\cup\mathcal{S}}{}^{-1} = \mathbf{V}_{\mathcal{T}}{}^{-1} - \mathbf{V}_{\mathcal{T}}{}^{-1}\mathbf{D}^*_{\mathcal{S}}\left(\mathbf{B}^{-1} + \mathbf{D}^{*T}_{\mathcal{S}}\mathbf{V}_{\mathcal{T}}{}^{-1}\mathbf{D}^*_{\mathcal{S}}\right)^{-1}\mathbf{D}^{*T}_{\mathcal{S}}\mathbf{V}_{\mathcal{T}}{}^{-1}. \tag{6.14}$$

This way, obtaining the inverse of the covariance matrix is computationally feasible. Recall that during tracking, the set $\mathcal{S}$ consists of a tracked image with its estimated landmarks, assuming these to have been correctly fitted. In iCCR, the update rules are therefore as follows (let $\mathbf{D}^{\Sigma}_{\mathcal{T}} = \mathbf{M}(\sum_{i=1}^{N}\mathbf{D}^*_i)^T$ be the stored weighted sum of training data):

$$\mathbf{Q} = \mathbf{V}^{-1}_{\mathcal{T}}\mathbf{D}^*_{\mathcal{S}} \tag{6.15}$$

$$\mathbf{U} = (\mathbf{B}^{-1} + \mathbf{D}^{*T}_{\mathcal{S}}\mathbf{Q})^{-1} \tag{6.16}$$

$$\mathbf{V}^{-1}_{\mathcal{T}\cup\mathcal{S}} = \mathbf{V}^{-1}_{\mathcal{T}} - \mathbf{Q}\mathbf{U}\mathbf{Q}^T \tag{6.17}$$

$$\mathbf{D}^{\Sigma}_{\mathcal{T}\cup\mathcal{S}} = \mathbf{D}^{\Sigma}_{\mathcal{T}} + \mathbf{M}\left(\mathbf{D}^*_{\mathcal{S}}\right)^T \tag{6.18}$$

$$\mathbf{R}_{\mathcal{T}\cup\mathcal{S}} = \mathbf{D}^{\Sigma}_{\mathcal{T}\cup\mathcal{S}}\mathbf{V}^{-1}_{\mathcal{T}\cup\mathcal{S}}, \tag{6.19}$$

[2]Herein, the incremental learning is devised for a single image, corresponding to a given frame. The reader might notice that its extension to a set of more than one image is straightforward, replacing in the derivations the matrix $\mathbf{B}$ by $\mathbf{B}\otimes\mathbf{E}_{N_{\mathcal{S}}}$, with $N_{\mathcal{S}}$ the number of images considered

Also, sometimes it is beneficial to include a learning factor λ, also known as "forgetting" factor, in which the samples to be updated have more or less weight. It is out of the scope of this thesis to explore the influence of such a factor, although it is included for the sake of completeness. In the experimental set-up, the learning rate was fixed for the whole set of videos, attaining impressive results. Future work shall explore a variable learning rate upon the "quality" of the frame, and its suitability for the update.

When a learning rate is included, we are basically considering that, when adding a new sample, this is going to have a weight λ:

$$\mathbf{R}_{\mathcal{T}\cup\mathcal{S}} = \mathbf{M}\left(\sum_{i=1}^{N}\mathbf{D}_j^* + \mathbf{W}_\lambda^{-1}\mathbf{D}_\mathcal{S}^*\right)^T\left(\mathbf{V}_\mathcal{T} + \mathbf{D}_\mathcal{S}^*\mathbf{W}_\lambda^{-1}\mathbf{B}\mathbf{D}_\mathcal{S}^{*^T}\right)^{-1}, \tag{6.20}$$

where $\mathbf{W}_\lambda = \lambda\mathbf{E}_{m+1}$. The inverse of $\mathbf{W}$ in Equation 6.20 implies that $\lambda < 1$ corresponds to a higher contribution of the new samples, whereas $\lambda > 1$ corresponds to a lower contribution. It is assumed that the image features and Jacobians share the same learning factor. The update rules are simply transformed as follows

$$\mathbf{Q} = \mathbf{V}_\mathcal{T}^{-1}\mathbf{D}_\mathcal{S}^* \tag{6.21}$$

$$\mathbf{U} = (\mathbf{W}_\lambda\mathbf{B}^{-1} + \mathbf{D}_\mathcal{S}^{*T}\mathbf{Q})^{-1} \tag{6.22}$$

$$\mathbf{V}_{\mathcal{T}\cup\mathcal{S}}^{-1} = \mathbf{V}_\mathcal{T}^{-1} - \mathbf{Q}\mathbf{U}\mathbf{Q}^T \tag{6.23}$$

$$\mathbf{D}_{\mathcal{T}\cup\mathcal{S}}^\Sigma = \mathbf{D}_\mathcal{T}^\Sigma + \mathbf{M}\mathbf{W}_\lambda^{-1}\left(\mathbf{D}_\mathcal{S}^*\right)^T \tag{6.24}$$

$$\mathbf{R}_{\mathcal{T}\cup\mathcal{S}} = \mathbf{D}_{\mathcal{T}\cup\mathcal{S}}^\Sigma\mathbf{V}_{\mathcal{T}\cup\mathcal{S}}^{-1}, \tag{6.25}$$

In this thesis, the learning rate was explored to maximise the behaviour of incremental learning in a subset of videos, and was kept fixed for the experimental set-up. Given that the learning rate does not necessarily need to be the same for all cascade levels, the learning rate was set to lower values in the first level of the cascade, and was increased in the subsequent levels. The reason for doing so is that the quality of the fit might have a negative effect when performing incremental learning in the lower levels of the cascade, given that the variance of the regressors is smaller. In the upper levels of the cascade, a possible error when selecting a frame for the

model to be updated would have less effect, and therefore the contribution of a given frame in such levels of the cascade can be higher. The learning rate in this thesis was empirically explored and set to $[0.01\,0.025\,0.05\,0.1]$ for each cascade level, respectively. Future work shall include a further analysis of the learning rate, as well as a study of how a better goodness of fit measure would allow for variable learning rates. For a fair comparison with respect to state of the art methods, in this thesis the learning rate was not changed.

6.3 Complexity

One of the advantages of parallel-SDM (Asthana et al. 2014) with respect to the standard SDM (Xiong & De la Torre 2013), is that the sampling process in the former can be done in parallel, given the statistics for each level to be available. Therefore, it is immediate to see that the parallel-SDM can exploit this property during the updating process. Of course, this property can be extended to the CCR presented in this thesis. However, in the parallel-SDM, the sampling process has to be done for each of the cascade levels. We can see that this is not required when using Continuous Regression. That is to say, iCCR only needs the data to be sampled once, and does not need to be repeated for each cascade level. Denoting the sampling cost as $\mathcal{O}(q)$, we can see that the total sampling cost of iCCR update is fixed to $\mathcal{O}(5q)$, and does not depend on the number of cascade levels. In contrast, the total sampling cost in the incremental parallel-SDM has to be carried out for each cascade level, thus having a total cost of $\mathcal{O}(LKq)$. Given a model of $L = 4$ cascade levels, and $K = 10$ perturbations per image, the cost of updating a parallel-SDM would be 8 times slower than the cost of updating the iCCR.

Regarding the cost of performing the updates shown in Equation 6.6(Chehra) and Equation 6.25, we can see that Equation 6.25 needs to compute the inverse in matrix $\mathbf{U}$, which is m-dimensional, thus having a cost of $\mathcal{O}(m^3)$. On the other hand, $\mathbf{U}$ in Equation 6.6 is K dimensional, and thus its computation is $\mathcal{O}(K^3)$. Typically, $K = 10$, and m in this thesis is set to 24, meaning that the cost of computing $\mathbf{U}$ in Chehra is computationally less expensive than in iCCR, although the difference is very small. Furthermore, the main bottleneck in both

methods resides in computing $\mathbf{Q}$, which is approximately $\mathcal{O}(d^2)$ in both iCCR and Chehra.

Thus, we can conclude that iCCR is less expensive computationally speaking than Chehra thanks to the fact that the former only needs to sample the data once, whereas the latter needs the data to be sampled for each cascade level. In this thesis, for a configuration of $L = 4$ cascade levels, and $K = 10$ samples per image in the sampling-based models, the iCCR update takes ~ 0.2 seconds in total (in a sequential implementation), whereas the sampling-based update takes ~ 0.31 seconds.

In any case, it is worth highlighting that, using the updates shown in this thesis, the computational complexity of Chehra is much lower than that reported by the authors, and that iCCR benefits from these updates to bring the possibility of real-time incremental learning in the context of Cascaded Regression. To the best of my knowledge, the tracker developed as a result of this thesis is the first one using Cascaded Regression that incorporates real-time incremental learning.

6.4 Experimental results

Now, we are ready to evaluate the performance of both the iCCR and CCR under the same conditions, with both methods aiming to compete against state of the art methods. This Section describes the experimental set-up and results. Both the CCR and iCCR methods are evaluated in the most extensive benchmark that exists to date: the 300VW (Shen et al. 2015). Given that participants were asked to submit their tracking systems in a fully automated manner (i.e., incorporating, if necessary, a face detection system, or a failure detection mechanism), a fully automated system was developed, which has been made publicly available, including an automated initialisation, as well as a lost detection tool. This way, both methods are compared in exactly the same experimental conditions as the challenge participants. It will be shown that the tracking system developed towards fulfilling this thesis achieves state of the art results. Furthermore, the impact of incremental learning in attaining these results is shown, thus highlighting the importance of having a method capable of performing in real-time. The

tracking system, along with the publications that resulted out of the research conducted in this thesis, can be found in the project's webpage `https://continuousregression.wordpress.com`.

6.4.1 Experimental set-up

Test Data

All methods are tested in the 300VW(Shen et al. 2015) dataset, which is the most extended and recent benchmark in Face Tracking (see Chapter 3). The dataset consists of 114 videos, each ~ 1 minute long, divided into different categories. 50 videos are used for training, whereas the 64 remaining videos are subdivided into three categories, intended to represent increasingly unconstrained scenarios. Category 1 contains 31 videos recorded in controlled conditions, whereas Category 2 includes 19 videos recorded under severe changes in illumination. Category 3 contains 14 videos captured in totally unconstrained scenarios. All the videos are single-person, and have been annotated in a semi-supervised manner (Tzimiropoulos 2015, Chrysos et al. 2015). All the frames in which the face appears beyond profile-view have been removed from the challenge evaluation, and therefore are not considered in these experiments either. These frames were kindly provided by the challenge organisers.

Error measure

The error measure is the same that was defined for the challenge, which is also the same measure that has long been used to measure the quality of face alignment algorithms. The error is computed for each frame by dividing the average point-to-point Euclidean error by the inter-ocular distance, interpreted as the distance between the two outer eye corners. More specifically, if $\hat{\mathbf{s}}$ is the estimated shape, and $\mathbf{s}^*$ is the ground-truth shape, the RMSE is given as:

$$RMSE = \frac{\sum_{i=1}^{n} \sqrt{(\hat{x}_i - x_i^*)^2 + (\hat{y}_i - y_i^*)^2}}{d_{outer} n}, \tag{6.26}$$

where d_{outer} is the Euclidean distance between the points defined for the outer corner of the eyes, measured on the ground-truth. The results are summarised in the form of Cumulative Error Distribution curves (CED), along with the Area Under the Curve (AUC). The CED curve illustrate the proportion of images (y axis) under a certain threshold (x axis). It is a monotonically increasing function, meaning that the closer the curve is to the top of the plot, the better the results. The AUC basically consists of the area that lies under the CED curve, and its value ranges from 0 to 1, being 0 the worst scenario, and 1 the best.

Data

The training data was taken from different datasets of static images. Specifically, the training data consists of Helen (Le et al. 2012), LFPW (Belhumeur et al. 2011), AFW (Zhu & Ramanan 2012), IBUG (Sagonas et al. 2013*b*), a subset of MultiPIE (Gross et al. 2010), and a random subset of images from the training partition of 300VW (see Chapter 3 for a description of these datasets). The training set comprises ∼8000 images. The facial landmark annotations follow the configuration of the 300 faces in the wild challenge (Shen et al. 2015, Sagonas et al. 2013*a*) (see Chapter 3). The models are trained for a 66-point configuration[3]. In order to ensure the consistency of the annotations with respect to the test set, the Shape Model is constructed using the training partition of the 300VW, and comprises of 20 non-rigid parameters and 4 rigid parameters. See Chapter 2 for a further description of the Shape Model construction.

Training

As shown in Algorithm 4, the training of CCR starts from a set of given statistics ($\boldsymbol{\mu}^0$ and $\boldsymbol{\Sigma}^0$). In this experiment, the statistics are computed across the training sequences, by computing the differences between consecutive frames. That is to say, $\boldsymbol{\mu}^0$ and $\boldsymbol{\Sigma}^0$ are meant to model how the shapes vary from frame to frame. The main idea of this approach is to replicate a real scenario when "perturbing" the ground-truth. Given the ease of the training set with respect to the test

[3]The reason why a 66-point configuration was chosen resides in the fact that the tracking was built as a continuation of a work that was started with this configuration

set, the differences between each two and three frames were also considered. This way, higher displacements are also captured.

Cascade levels

In this experimental set-up, the number of cascade levels is fixed to $L = 4$. It is out of the scope of this thesis to explore the key features of Cascaded Regression methods, such as the number of cascade levels, or the best features that might optimise performance. All of these were however explored in previous research.

Features

The chosen feature selection is the same as that of previous Chapters, and consists of a self-implemented version of HOG(Dalal & Triggs 2005) (see Chapter 2). The developed HOG operates with a block-size of 24 pixels, subdivided in 4 blocks, and 9 bins, resulting in a 9504-dimensional vector. The dimensionality of the feature vector was reduced to 2000 components through PCA, taking advantage of the functional covariance presented in Chapter 5

6.4.2 Tracking System

Initialisation

As mentioned in Chapter 2, the first frame needs to be initialised from the bounding box given by a face detector. This initialisation step also needs to be repeated each time the tracker is detected to have lost a fitting (see details below). The initialisation utilises the open-source dlib face detection to locate the face bounding box (`dlib.net`), and then predicts the shape with a Context-based SDM (Sánchez-Lozano, Martinez & Valstar 2016). Then, a single CCR iteration is carried out.

Incremental learning update

The incremental learning needs to filter frames to decide whether a fitting is suitable for the models to be updated, or harmful. That is, in practice, it is beneficial to filter out badly-tracked frames by avoiding performing incremental updates in these cases. It is out of the scope of this thesis to explore the best way to estimate the "quality of fit", and therefore a simple heuristic was used. More specifically, a linear SVM was trained to decide whether a particular fitting is "correct" or not, otherwise referred to as being under a threshold error. Furthermore, given that videos are single-person, the tracker does not need to perform any face recognition to guarantee that updated models keep tracking the same person with which models have been updated. This task will be included as future work. Finally, for the iCCR update, it is also beneficial to include as much person-specific data as possible. A sliding window of 50 frames is used to collect person-specific statistics. That is to say, the same procedure used to generate the statistics for the training of the models is used to generate the statistics used to update the models. Finally, it is worth mentioning again that the same learning factor was used in all the tested videos.

Failure detection

Sometimes, it is not possible for the models to track a face. This occurs especially when there are important occlusions or faces beyond profile. In these cases, we need to detect that the tracker has lost a face, in order to reinitialise the next frame from the face detector. To keep the tracker simple, the same SVM learnt for detecting the goodness of fit was used to also detect whether the tracker is lost or not, by empirically selecting a suitable threshold for the score returned by the model.

6.4.3 Results

The tracking system was evaluated under exactly the same conditions that participants had during the challenge evaluation. In order to provide an analysis of performance, the tracker is

compared against the top two participants (Yang et al. 2015, Xiao et al. 2015). The results, shown in Figures 6.3, 6.4, 6.5, 6.6, 6.7, and 6.8, report the CED curves for both the 49 point and 66 point configurations. However, the CED curves in the 66 point configuration are compared against the 68 point configuration in which participants were evaluated. It is important to remark that the 66 point configuration used in this thesis does not consider two points that are part of the mouth, which are typically better fitted than those belonging to the contour. Therefore, results of methods performing a 66 point detection against methods detecting 68 points are typically balanced towards the latter configuration.

In order to compare the tracker against Yang et al. (2015) and Xiao et al. (2015), I asked participants to send their results, which were kindly provided. For the fairest comparison with respect to the Challenge evaluation, those frames that were not considered for the evaluation (i.e., those including faces beyond profile), were removed. In all the curves, the results of both the CCR and iCCR systems is included, showing the benefits of incremental learning. CED results are shown in Figures 6.3, 6.4, 6.5, 6.6, 6.7, and 6.8, and the AUC for each method is shown in Table 6.1 and 6.2. It is interesting to note that both Yang et al. (2015) and Xiao et al. (2015) emphasise either on the initialisation of the tracker in each frame, or the lost detection. In this thesis, the tracking system is kept simple on purpose, in order to assess the contributions of Continuous Regression. Therefore, even with a simple heuristics, iCCR is capable of attaining comparable or even superior performance than Yang et al. (2015) and Xiao et al. (2015). Also, the initialisation for each frame in iCCR is just the output of the previous frame. There are no multi-view models or progressive initialisation, as in Yang et al. (2015) or Xiao et al. (2015). Instead, the statistics used to generate the models are ensured to generalise well to unseen scenarios. Furthemore, it is worth highlighting that the CCR model had to be reinitialised in only $\sim 0.51\%$ of the total frame count (> 121000), whilst the iCCR only needed to be reinitialised in $\sim 0.34\%$ of the frames. This illustrates the stability of this simple (yet effective), approach. Finally, it is worth highlighting the importance of Incremental Learning in challenging sequences, such as those shown in Category 3, to achieve state of the art results. Two qualitative examples are shown in Figure 6.9 and Figure 6.10. That is to say, it seems that incremental learning is crucial to attain good results in challenging scenarios.

Besides, as shown through this Chapter, the complexity of the incremental learning update allows for real-time implementation, something that could not be achieved by previous works on Cascaded Regression.

Method	Category 1	Category 2	Category 3
Yang et al. (2015)	**0.5981**	0.6025	0.4996
Xiao et al. (2015)	0.5814	**0.6093**	0.4865
iCCR	0.5978	0.5918	**0.5141**
CCR	0.5657	0.5539	0.4410

Table 6.1: AUC for 49 points configuration for the different categories.

Method	Category 1	Category 2	Category 3
Yang et al. (2015)	**0.5291**	0.5167	0.4011
Xiao et al. (2015)	0.5235	**0.5527**	0.4030
iCCR	0.5171	0.5232	**0.4044**
CCR	0.4807	0.4680	0.3198

Table 6.2: AUC for 66 points configuration for the different categories (68 for Yang et al. (2015), Xiao et al. (2015)).

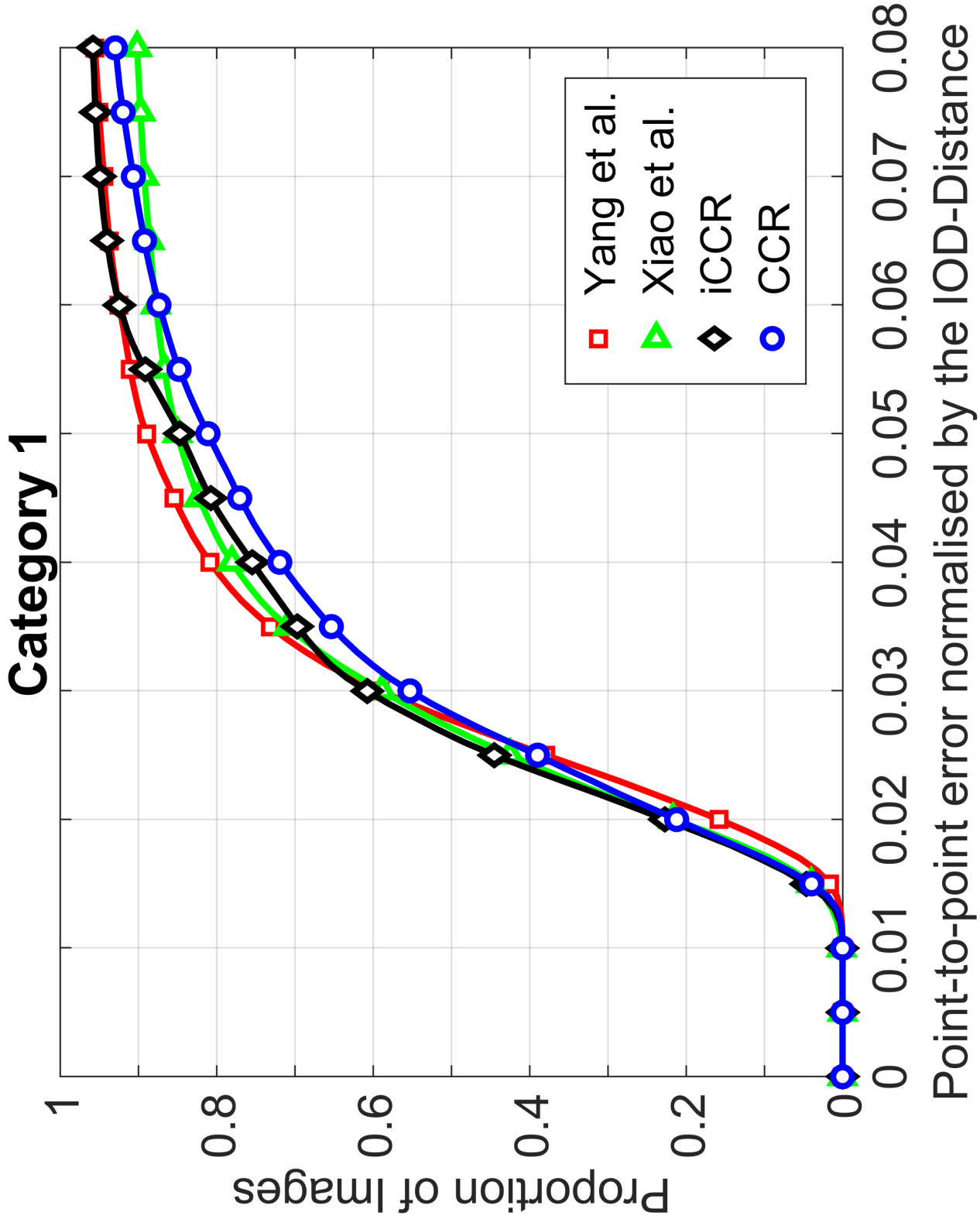

Figure 6.3: CED's for the 49-points configuration (category A).

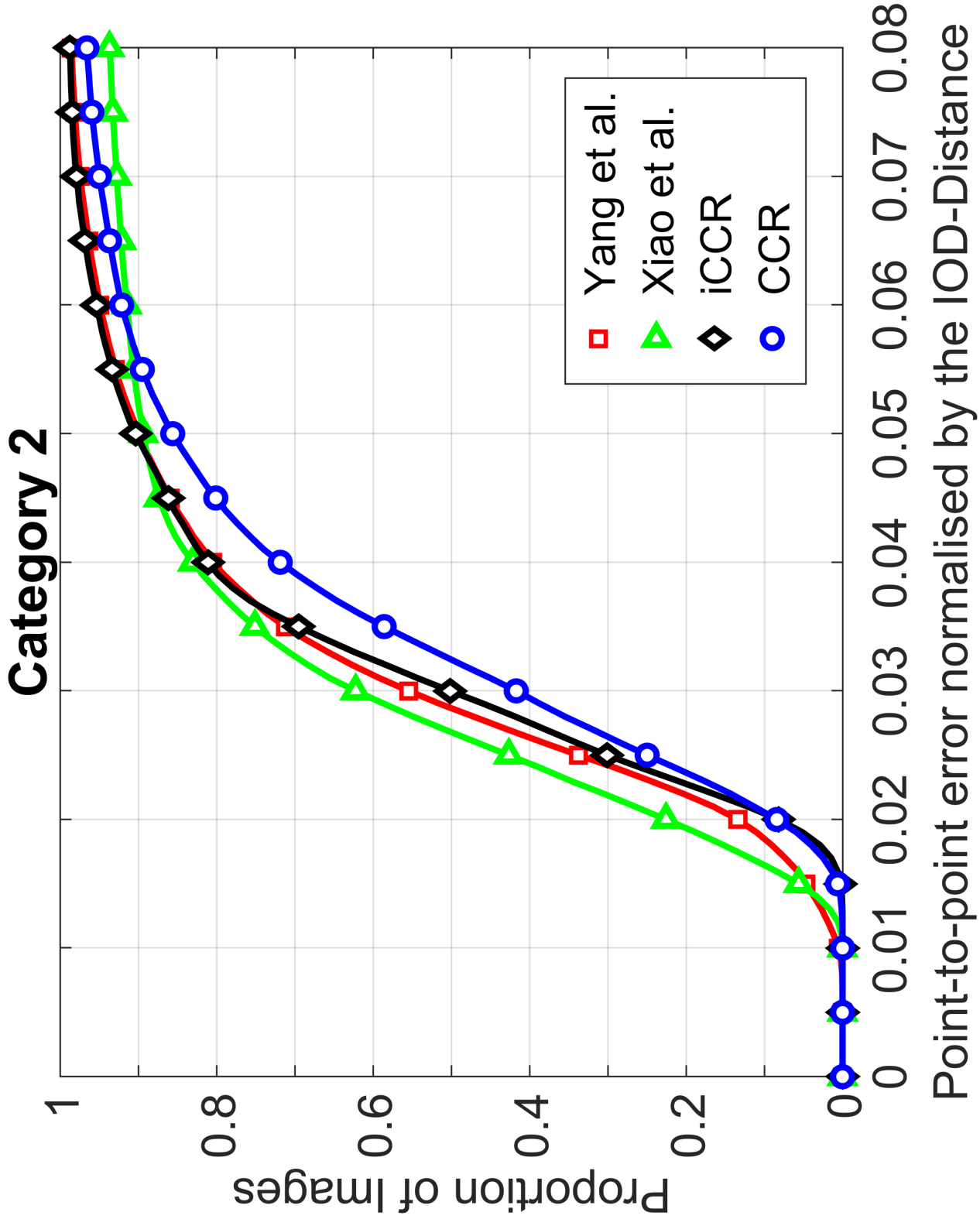

Figure 6.4: CED's for the 49-points configuration (category B).

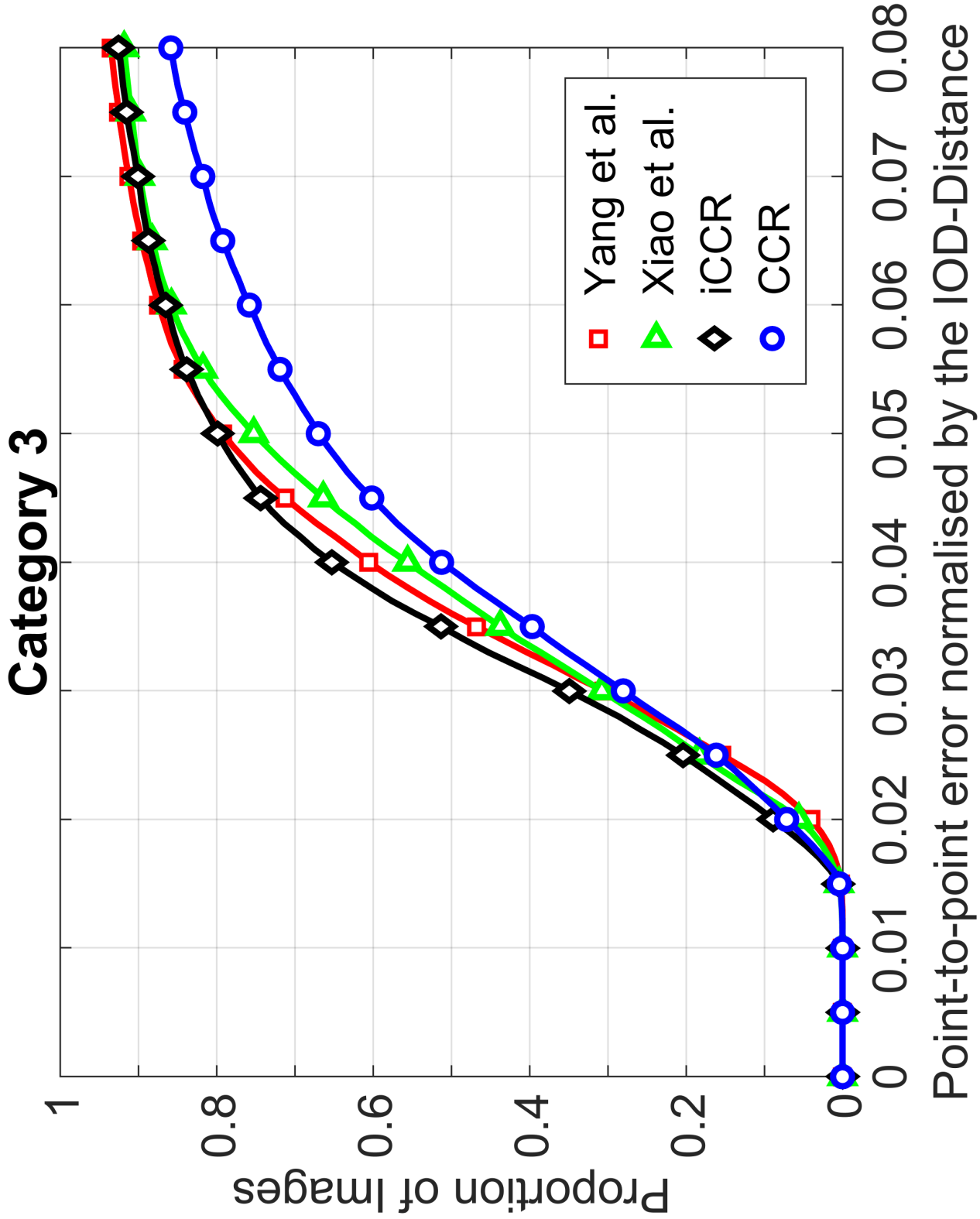

Figure 6.5: CED's for the 49-points configuration (category C).

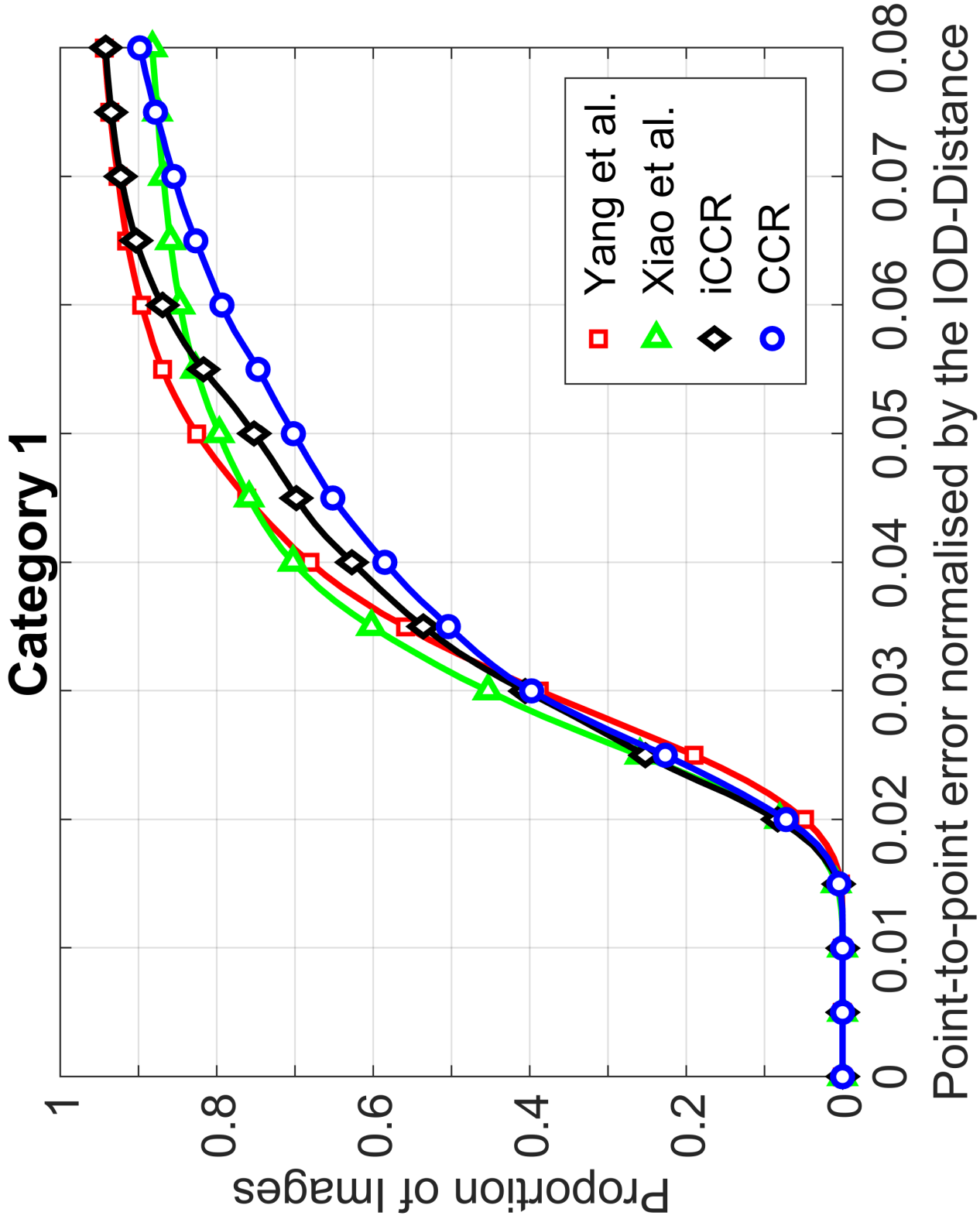

Figure 6.6: CED's for the 66-points configuration (category A).

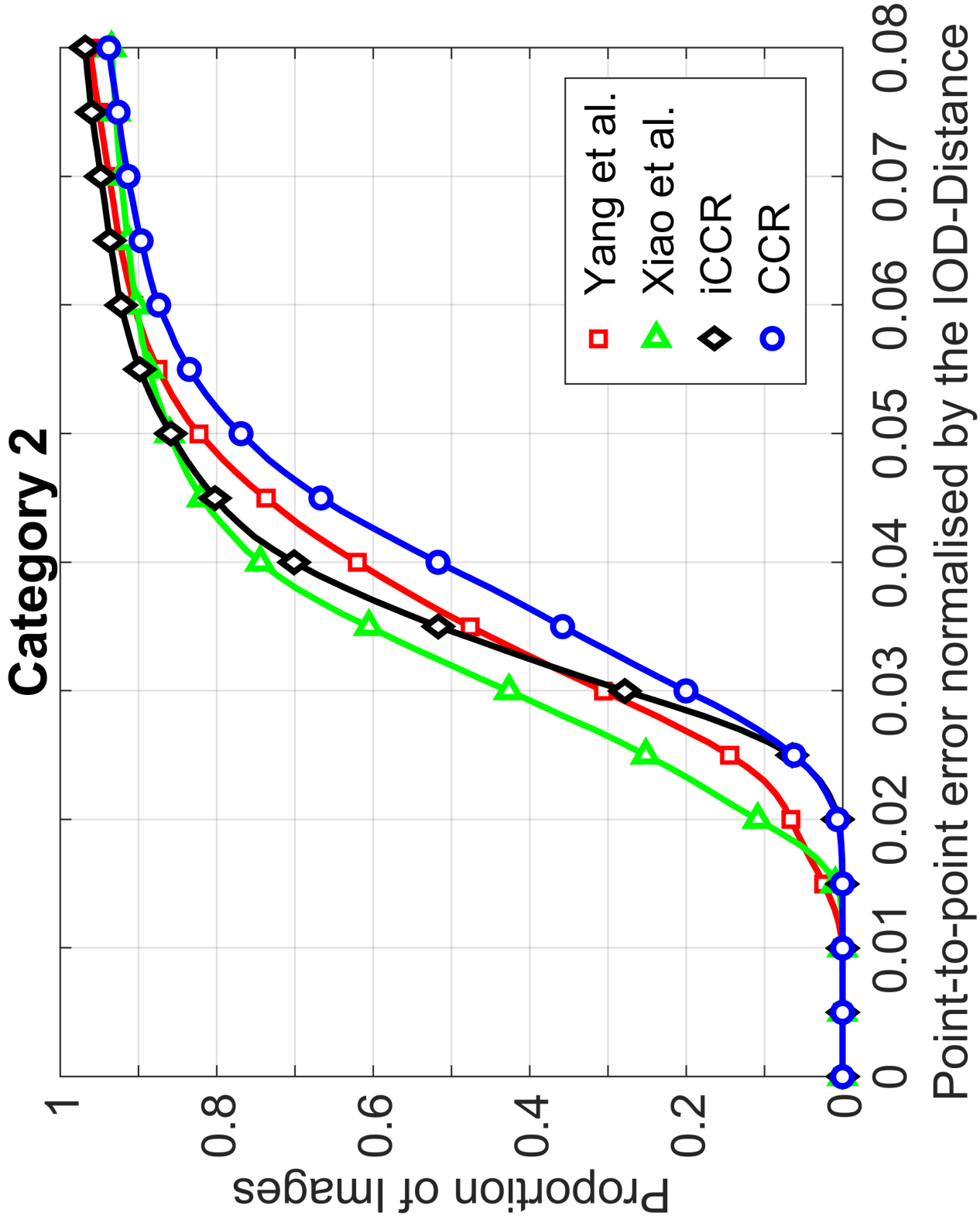

Figure 6.7: CED's for the 66-points configuration (category B).

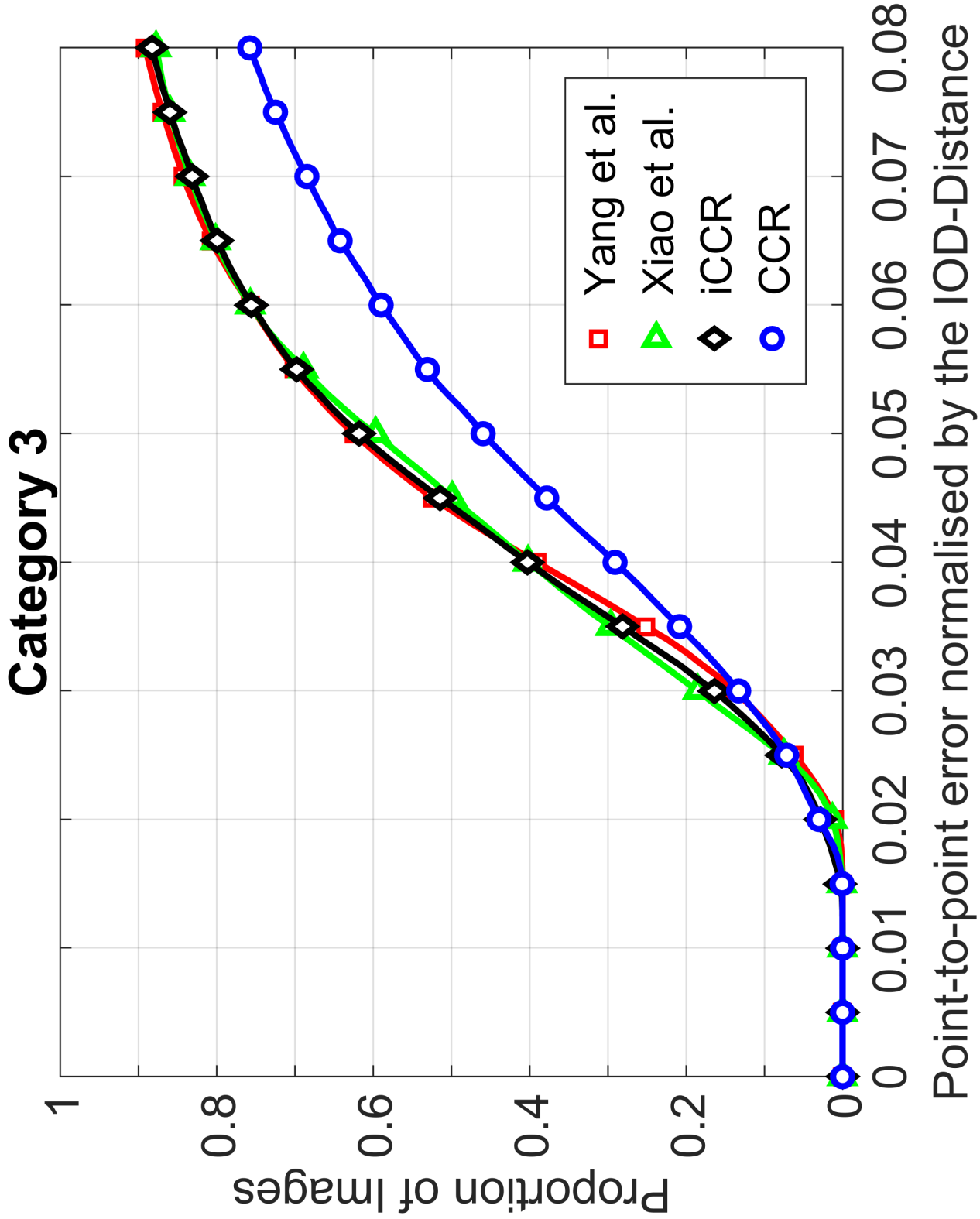

Figure 6.8: CED's for the 66-points configuration (category C).

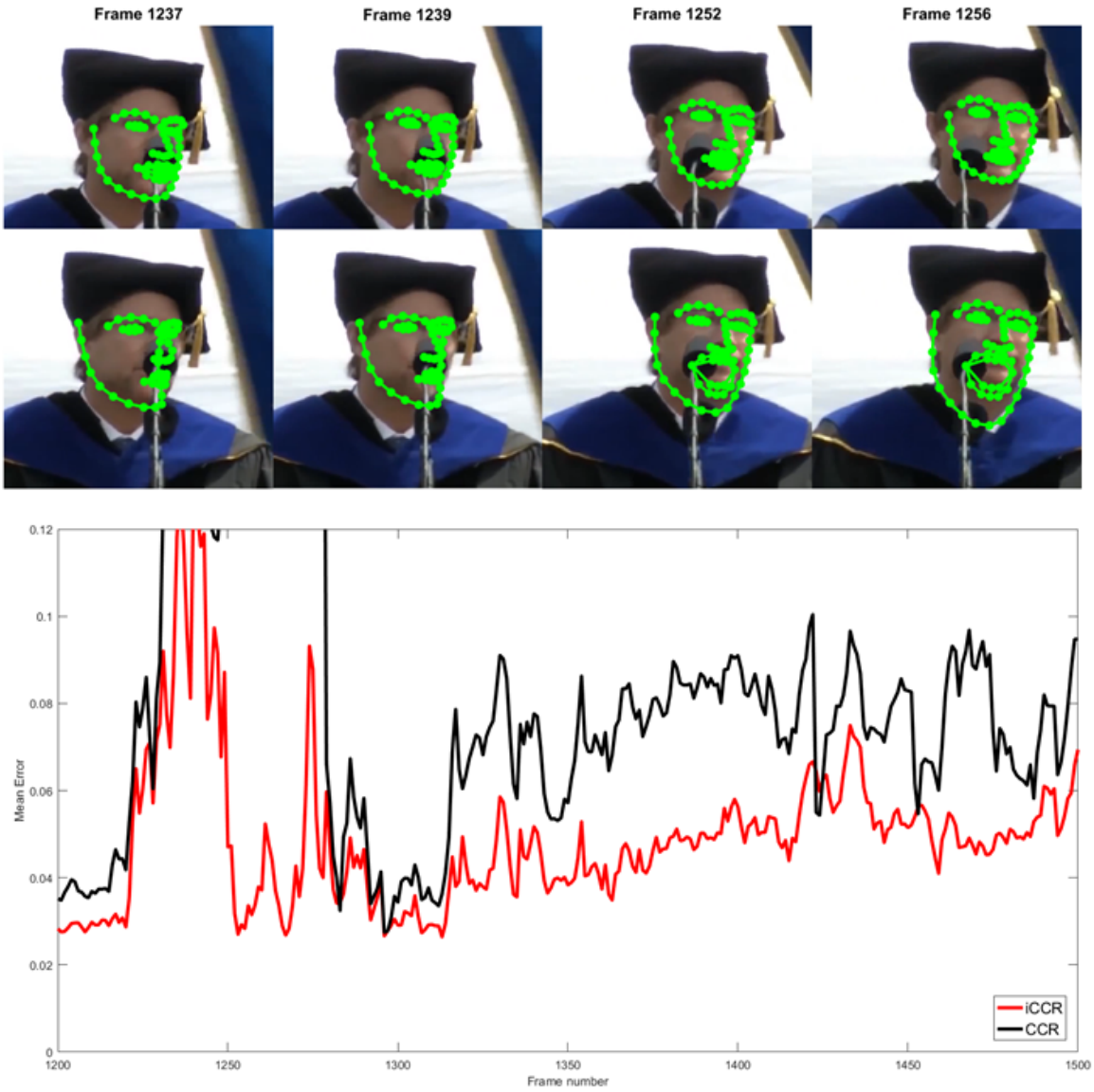

Figure 6.9: Qualitative results for a sample video of Category 3. Top row shows the tracked points using iCCR, whereas bottom row shows the results given by the CCR without incremental learning. The importance of incremental learning is therefore clear.

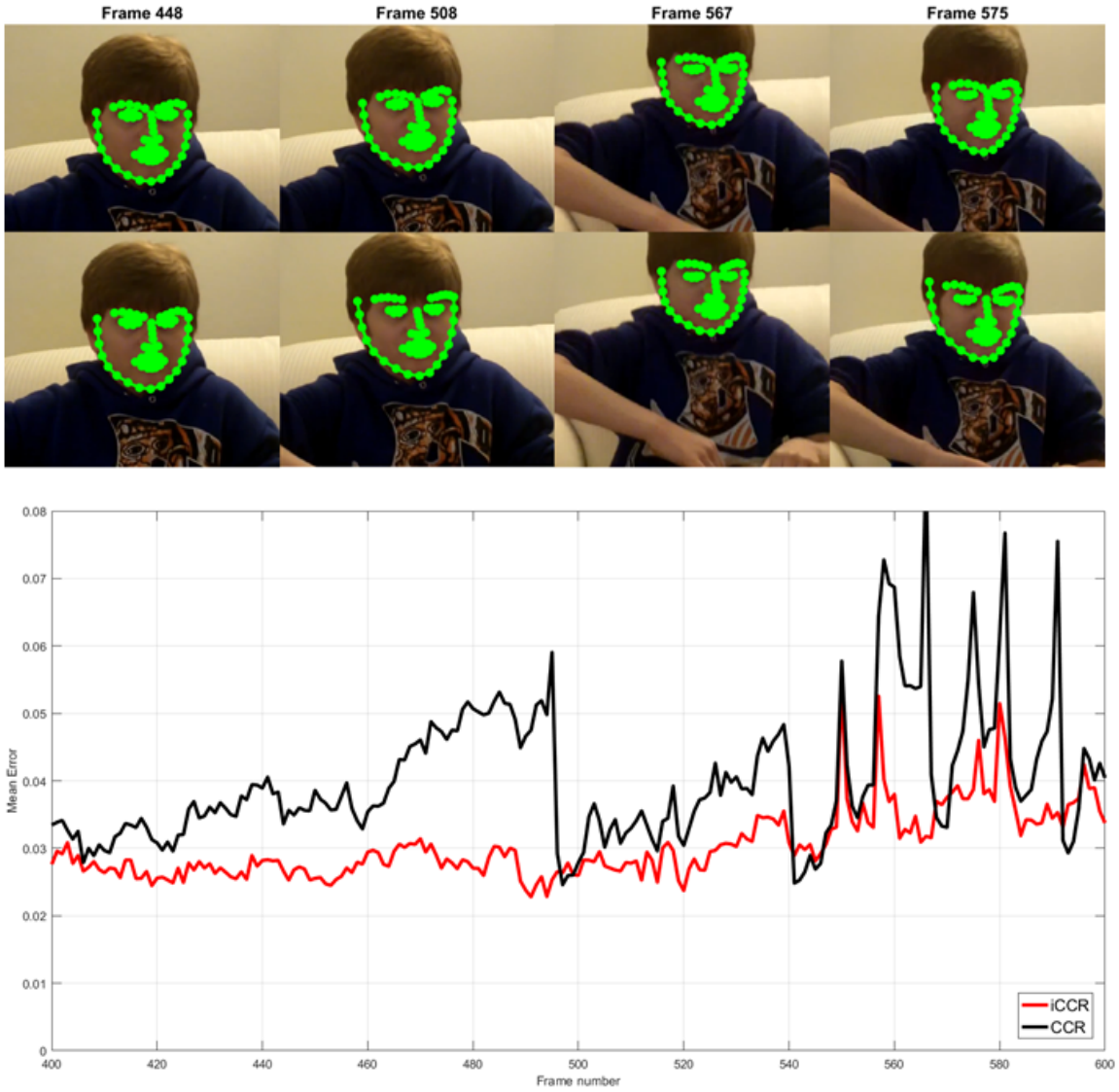

Figure 6.10: Qualitative results for a sample video of Category 3. Top row shows the tracked points using iCCR, whereas bottom row shows the results given by the CCR without incremental learning. The importance of incremental learning is therefore clear.

Chapter 7

Conclusion

This thesis proposed a real-time incremental face tracking system, built upon a novel Functional Regression solution for the Least-Squares problem, coined Continuous Regression. First, a novel basis for Functional Regression in the field of images was proposed: the use of a first-order Taylor expansion. The classical Functional Regression solution was found to be limited when dealing with correlated variables, like faces, specially during tracking. To overcome such limitation, a second novelty was proposed: the use of a probability measure to model correlated variables. This novel solution utilises the link between probability and geometry to provide a geometrical interpretation of the sampling distribution that underlies the new measure introduced in Continuous Regression.

However, besides the theoretical contributions, this thesis has important practical applications. First, the novel solution was introduced into the state-of-the-art Cascaded Regression framework, and the benefits of using Continuous Regression with respect to sampling-based linear regression were shown. Its complexity was studied deeply. Importantly, Cascaded Regression, using Continuous Regression (Cascaded Continuous Regression, CCR), has been shown to be computationally efficient, as opposed to sampling-based approaches, in which the whole process has to be repeated each time a new model has to be trained.

Finally, and most importantly, a novel incremental learning update rule was devised for both SDM and CCR, showing that the complexity of the latter allows real-time implementation. To

the best of my knowledge, the tracker resulting from this thesis is the first real-time incremental face tracker using Cascaded Regression. The results shown in this thesis support the importance of having a real-time incremental learning procedure, and the tracker that results from the research conducted in this thesis achieves state-of-the-art results in the only benchmark that exists to date, being capable of competing against the methods that ended winner and runner up in the 300VW competition.

7.1 Applications

At the beginning of this thesis, it was shown why a robust Facial Point Detection and Tracker system is key to the success of many Face Analysis systems. We can recall that a robust facial landmark localisation system is important for applications such as Face Recognition, Facial Expression Recognition, or Age and Gender Recognition, among others. Therefore, the results attained by the tracker developed in this thesis can be directly applied to all of these fields. In fact, the CCR system has been used to extract the features that are used for the FERA 2017 (Facial Expression Recognition and Analysis Challenge (Valstar et al. 2017)) baseline system, which are based on the tracked facial point locations. The results yielded by the geometric features under different poses show the importance of having an accurate face tracking system.

In addition, the CCR tracker has been already implemented in C++, and its incremental version is now underway. The C++ tracker has been introduced into the eMax software, which is the key part of the visual module of the ARIA-VALUSPA project `aria-agent.eu`. The ARIA-VALUSPA project is an European funded project that aims at creating a framework allowing the easy creation of Artificial Retrieval of Information Assistants (ARIAs) that are capable of holding multi-modal social interactions in challenging and unexpected situations. Therefore, the CCR tracker serves as a robust facial localisation system that allows the extraction of visual information for emotion recognition.

Apart from the publications that resulted from the research conducted in this thesis, part of the MATLAB code has been released for research purposes, allowing the experiments conducted

in Chapter 6 to be reproduced by any potentially interested researcher. A project webpage `continuousregression.wordpress.com` is available, and allows the use of both CCR and iCCR trackers. The tracker's skeleton is open, so that all the aspects discussed as future work (see below) can be investigated.

7.2 Future Work

We have seen in Chapter 6 that the tracker built towards validating the research conducted through this thesis yields state-of-the-art results. However, despite the impressive results yielded by the tracker, there is still a large margin to improve. Through this thesis, several things were relegated to future work, mainly because these appeared as new challenges that are now opened as a consequence of the achievements attained by the iCCR tracker. This Section summarises the main challenges and opportunities that remain to be solved.

Frame selection

As mentioned in Chapter 6, not all the tracked frames are used to update the generic models. The reason for not doing so resides in the fact that a bad fit might add noisy samples to the model that will ultimately cause the tracker to drift. If the tracker returns a bad fit, we need to skip that frame, and try to track the next one. That is to say, we will select only those frames for which we think the tracker has returned a reliable estimate, so that its inclusion in the model will be beneficial, not harmful. In this thesis, a simple heuristic was used. Basically, a linear SVM was trained, in which a set of images from the training set were taken, and their ground-truth were displaced according to certain perturbations. Then, the error was measured, and those perturbations above a certain threshold were selected as negative, whereas those below the error threshold were selected as positive, along with the ground-truth. After the SVM was trained, a threshold for the score returned by the SVM was used to select whether the fit was good enough or not. The threshold was chosen empirically using the training set. Therefore, a better frame selection would probably help the incremental learning procedure to

work even better. The use of better classifiers, or better image features, would probably allow for an improvement regarding the frame selection.

Failure detection

Similar to the frame selection issue, it is important to emphasise that having a better failure detection system would dramatically improve the performance of the tracker, specially in the most challenging category of the 300VW dataset. In this thesis, the same SVM trained for the frame selection procedure is used to detect whether the tracker has lost a frame or not! This highlights the benefits of the tracker, given that despite this simple approach, the results are impressive. We have seen that the failure rate for the whole 300VW dataset is less than 0.5%. After a visual inspection, it can be found that there are a few number of frames in which the tracker should have detected a failure. This would slightly increase the number of frames where the tracker would need to reinitialise, but would also increase its performance. This way, future work shall include a study of a better failure detection system.

Learning rate

We have also commented on the possibility of having a higher or lower rate for the frames that are used to update the models. This increases or reduces the contribution of each given frame. In this thesis, an empirical learning rate was found and fixed for the whole set of videos. The reason for this ws to avoid manually changing the learning rate which could lead to improvement of the iCCR tracker for each specific video and thus result in an unfair comparison with previous work. During intermediate experiments, I found several videos in which this fixed learning rate was indeed suboptimal, and even harmful for a few videos. However, on average, the fixed learning rate appeared to maximise the performance in a subset of videos. In order to maintain a fair comparison with respect to those methods that participated in the 300VW Challenge, I decided to keep the learning rate fixed. Participants were asked to submit their systems, not their results per video. Therefore, this was the fairest way to compare against these methods. However, given that now the iCCR tracker enables the possibility of real-time

incremental learning, the question that arises is how the learning rate affects the contribution of incremental learning. Furthermore, we can study a variable learning rate, depending on the quality of the fit for a current frame. If we know a current frame has been correctly fitted, but we decide its contribution might not be as good as other frames, we can set a lower learning rate. Also, if we know that the tracked subject is barely moving, we can disregard some of the frames belonging to that sequence, to avoid redundancy.

Track multiple faces

The tracker developed in this thesis is meant to track a single person. When the tracker detects a failure, it attempts to detect a new face. If there are more faces appearing in a video (which might be video streaming), then it is possible that the tracker can "jump" to a different person. In such case, it is not clear whether the updated models would need to restart from the generic models or not. Future work needs to study this issue. A simple heuristic would be to perform a face identification system, so that each person can keep a corresponding model. Also, the use of parallel programming would help tracking more than one face simultaneously without incurring an increase in complexity, and hence drop in performance. It is worth noting that all the videos belonging to the 300VW data are single person, and therefore this problem was not an issue during the evaluation of the developed tracker.

Face alignment in static images

Finally, it is important to remark why the CCR has not been used in this thesis for the task of Face Alignment from the bounding box given by a face detector. The answer is that CCR depends on the Taylor expansion, and therefore its scope is limited by how the Taylor expansion can be used to predict the input space in a given location. The input variance of a face detector is far too high for a CCR to perform well. During the research conducted towards this thesis, a model using CCR was trained from the output of a bounding box, although results were not as good as those given by SDM. However, as we have seen in this thesis, face alignment and face tracking differ in few aspects, and while the literature has profoundly addressed the problem

of face alignment, there are very few works addressing the problem of face tracking. We can therefore conclude that face alignment is a well-studied and almost solved problem, whereas face tracking still needs work to compete with face alignment in terms of performance. Thus, there is no reason why CCR should be extended to the problem of face alignment.

However, mathematically speaking, there is an alternative formulation to Continuous Regression that might enable its use within the context of Face Alignment. During this thesis, many intermediate experiments were proposed to evaluate the scope of Continuous Regression. Finally, the results given by the tracker support the use of a first-order Taylor expansion for Continuous Regression in the context of Face Tracking.

We can see that when the Taylor expansion appears to be a limitation, one might take either two possible alternatives: apply a higher-order expansion, or apply an intermediate sampling and then apply the first-order Taylor expansion again. The former results in a too expensive method (we would need the third-order statistics and the Hessians of the images!), but the latter has a closed form solution that results in an intermediate solution between sampling-based and continuous regression. The main idea is to consider the following approximation:

$$f(\mathbf{I}_i, \mathbf{s}_i^* + \delta\mathbf{s}_j + \delta\mathbf{s}) \approx f(\mathbf{I}_i, \mathbf{s}_i^* + \delta\mathbf{s}_j) + \mathbf{J}_i^{*,j}\delta\mathbf{s}, \tag{7.1}$$

where now the Jacobian would be computed in $\mathbf{s}_i^* + \delta\mathbf{s}_j$. Therefore, we sample on $\delta\mathbf{s}_j$, and then integrate over $\delta\mathbf{s}$. Clearly, this approximation contains both Continuous Regression and sampling-based linear regression. If $\delta\mathbf{s}$ is drawn from a zero-mean distribution with $\mathbf{\Sigma} = 0$, then the space of integration is null, whereas if all $\delta\mathbf{s}_j$ are set to zero, we are resorting to Continuous Regression again. This extension could be used to allow the use of CCR for Face Alignment.

Bibliography

Amari, S. (1985), 'Differential-geometrical methods in statistics', *Spring Science and Business Media* .

Ammar, M. B., Neji, M., Alimi, A. M. & Gouardres, G. (2010), 'The affective tutoring system', *Expert Systems with Applications* **37**(4), 3013 – 3023.

Asthana, A., Zafeiriou, S., Cheng, S. & Pantic, M. (2014), Incremental face alignment in the wild, *in* 'IEEE Conf. on Computer Vision and Pattern Recognition'.

Aung, M., Kaltwang, S., Romera-Paredes, B., Martinez, B., Singh, A., Cella, M., Valstar, M., Meng, H., Kemp, A., Elkins, A. et al. (2015), 'The automatic detection of chronic pain-related expression: requirements, challenges and a multimodal dataset', *Trans. on Affective Computing* .

Baltrušaitis, T., Robinson, P. & Morency, L.-P. (2014), Continuous conditional neural fields for structured regression, *in* 'European Conf. on Computer Vision'.

Baltrušaitis, T., Robinson, P. & Morency, L.-P. (2016), Openface: an open source facial behavior analysis toolkit, *in* 'Winter Conf. on Applications of Computer Vision'.

Belhumeur, P., Jacobs, D., Kriegman, D. & Kumar, N. (2013), 'Localizing parts of faces using a consensus of exemplars', *Trans. on Pattern Analysis and Machine Intelligence* **35**(12), 2930–2940.

Belhumeur, P. N., Jacobs, D. W., Kriegman, D. J. & Kumar, N. (2011), Localizing parts of faces using a consensus of exemplars, *in* 'IEEE Conf. on Computer Vision and Pattern Recognition', pp. 545–552.

Bishop, C. M. (2006), *Pattern Recognition and Machine Learning*, Springer.

Bolme, D., Beveridge, R., Draper, B. & Lui, Y. M. (2010), Visual object tracking using adaptive correlation filters, *in* 'IEEE Conf. on Computer Vision and Pattern Recognition'.

Brookes, M. (2011), 'The matrix reference manual'.
URL: *http://www.ee.imperial.ac.uk/hp/staff/dmb/matrix/intro.html*

Cao, X., Wei, Y., Wen, F. & Sun, J. (2012), Face alignment by explicit shape regression, *in* 'IEEE Conf. on Computer Vision and Pattern Recognition', pp. 2887–2894.

Cao, X., Wei, Y., Wen, F. & Sun, J. (2014), 'Face alignment by explicit shape regression', *Int'l Journal of Computer Vision* **107**(2), 177–190.

Chrysos, G. S., Antonakos, E., Zafeiriou, S. & Snape, P. (2015), Offline deformable face tracking in arbitrary videos, *in* 'Int'l Conf. Computer Vision - Workshop'.

Cootes, T. F., Edwards, G. J. & Taylor, C. J. (1998), Active appearance models, *in* 'European Conf. on Computer Vision', pp. 484–498.

Cootes, T. F., Edwards, G. J. & Taylor, C. J. (2001), 'Active appearance models', *Trans. on Pattern Analysis and Machine Intelligence* **23**(6), 681–685.

Cootes, T. F., Ionita, M. C., Lindner, C. & Sauer, P. (2012), Robust and accurate shape model fitting using random forest regression voting, *in* 'European Conf. on Computer Vision', pp. 278–291.

Cootes, T. F. & Taylor, C. J. (2004), 'Statistical models of appearance for computer vision'.

Cootes, T. F., Taylor, C. J., Cooper, D. H. & Graham, J. (1992), Training models of shape from sets of examples, *in* 'British Machine Vision Conf.'.

Cootes, T. F., Taylor, C. J., Cooper, D. H. & Graham, J. (1995), 'Active shape models-their training and application', *Comp. Vision and Image Understanding* **61**(1), 38–59.

Cootes, T. & Taylor, C. (1992), Active shape models - 'smart snakes', *in* 'British Machine Vision Conf.'.

Cristinacce, D. & Cootes, T. (2008), 'Automatic feature localisation with constrained local models', *Pattern Recognition* **41**, 3054–3067.

Dalal, N. & Triggs, B. (2005), Histograms of oriented gradients for human detection, *in* 'IEEE Conf. on Computer Vision and Pattern Recognition', pp. 886–893.

De la Torre, F., Chu, W.-S., Xiong, X., Vicente, F., Ding, X. & Cohn, J. (2015), Intraface, *in* 'Int'l Conf. on Face and Gesture'.

Dollár, P., Welinder, P. & Perona, P. (2010), Cascaded pose regression, *in* 'IEEE Conf. on Computer Vision and Pattern Recognition', pp. 1078–1085.

Duffy, N. & Helmbold, D. (2002), 'Boosting methods for regression', *Machine Learning* **47**, 153–200.

Egede, J., Valstar, M., Martinez, B. & Qiu, G. (2017), Fusing deep learned and hand-crafted features of appearance, shape, and dynamics for automatic pain estimation, *in* 'Proceedings of the IEEE 2017 Conference on Face and Gesture Recognition'.

Ekman, P. & Oster, H. (1979), 'Facial expressions of emotion', *Annual Rev. of Psychollogy* **30**, 527–554.

Fleuret, F. & Geman, D. (2008), 'Stationary features and cat detection', *Journal of Machine Learning Research* **41**, 85–107.

Fogel, I. & Sagi, D. (1989), 'Gabor filters as texture discriminator', *Biological Cybernetics* **61**(2), 103–113.

Friedman, J. (2001), 'Greedy function approximation: a gradient boosting machine', *Annals of Statistics* **29**, 1189–1232.

González-Jiménez, D. & Alba-Castro, J. (2007), 'Towards pose-invariant 2-d face recognition through point distribution models and facial symmetry', *Trans. on Information Forensics and Security* **2**(3), 413–429.

Goodall, C. (1991), 'Procrustes methods in the statistical analysis of shape', *J. R. Statist. Soc. B* **53**(2), 285–339.

Gordon, I., Pierce, M. D., Bartlett, M. S. & Tanaka, J. W. (2014), 'Training facial expression production in children on the autism spectrum', *Journal of Autism Dev Disord* **44**(10), 2486–2498.

Gross, R., Matthews, I. & Baker, S. (2005), 'Generic vs. person specific active appearance models', *Image and Vision Computing* **23**(11), 1080–1093.

Gross, R., Matthews, I., Cohn, J., Kanade, T. & Baker, S. (2010), 'Multi-pie', *Image and Vision Computing* **28**(5), 807–813.

Gu, L. & Kanade, T. (2008), A generative shape regularization model for robust face alignment, *in* 'IEEE Conf. on Computer Vision and Pattern Recognition'.

Igual, L. & De la Torre, F. (2010), Continuous procrustes analysis to learn 2d shape models from 3d objects, *in* '3rd CVPR Workshop on Non-Rigid Shape and Deformable Image Alignment', pp. 17–22.

Igual, L., Perez-Sala, X., Escalera, S., Angulo, C. & De la Torre, F. (2014), 'Continuous generalized procrustes analysis', *Pattern Recognition* **47**, 659–671.

Jaiswal, S., Valstar, M., Gillot, A. & Daley, D. (2017), Automatic detection of adhd and asd from expressive behaviour in rgbd data, *in* 'Proceedings of the IEEE 2017 Conference on Face and Gesture Recognition'.

Le, V., Brandt, J., Lin, Z., Bourdev, L. D. & Huang, T. S. (2012), Interactive facial feature localization, *in* 'European Conf. on Computer Vision', pp. 679–692.

Levin, A. & Shashua, A. (2002), Principal component analysis over continuous subspaces and intersection of half-spaces, *in* 'European Conf. on Computer Vision'.

Liu, X. (2007), Generic face alignment using boosted appearance model, *in* 'IEEE Conf. on Computer Vision and Pattern Recognition'.

Liu, X. (2009), 'Discriminative face alignment', *Trans. on Pattern Analysis and Machine Intelligence* **31**(11), 1941–1954.

Liu, X. (2011), Optimal gradient pursuit for face alignment, *in* 'Int'l Conf. on Face and Gesture'.

Liu, Y., Schmidt, K. L., Cohn, J. F. & Mitra, S. (2003), 'Facial asymmetry quantification for expression invariant human identification', *Computer Vision and Image Understanding* **91**(12), 138 – 159. Special Issue on Face Recognition.

Lowe, D. G. (2004), 'Distinctive image features from scale-invariant keypoints', *International Journal of Computer Vision* **60**(2), 91–110.

Lucas, B. & Kanade, T. (1981), An iterative image registration technique with an application to stereo vision, *in* 'International Joint Conference on Artificial Intelligence', pp. 674–679.

Martinez, B., Valstar, M. F., Binefa, X. & Pantic, M. (2013), 'Local evidence aggregation for regression based facial point detection', *Trans. on Pattern Analysis and Machine Intelligence* **35**(5), 1149–1163.

Martins, P., Caseiro, R. & Batista, J. (2014), Non-parametric bayesian constrained local models, *in* 'IEEE Conf. on Computer Vision and Pattern Recognition'.

Martins, P., Caseiro, R., Henriques, J. F. & Batista, J. (2012), Discriminative bayesian active shape models, *in* 'European Conf. on Computer Vision', pp. 57–70.

Martins, P., Henriques, J., Caseiro, R. & Batista, J. (2016), 'Bayesian constrained local models revisited', *Trans. on Pattern Analysis and Machine Intelligence* **38**(4), 704–716.

Marx, B. & Eilers, P. (1999), 'Generalized linear regression on sampled signals and curves: A p-spline approach', *Techno.* **41**, 1–13.

Matthews, I. & Baker, S. (2004), 'Active appearance models revisited', *Int'l Journal of Computer Vision* **60**(2), 135–164.

Morris, J. (2015), 'Functional regression', *Annual Review of Stats. and its App.* **2**, 321–359.

Ojala, T., Pietikinen, M. & Harwood, D. (1996), 'A comparative study of texture measures with classification based on featured distributions', *Pattern Recognition* **29**(1), 51 – 59.

Orozco, J., Martinez, B. & Pantic, M. (2015), 'Empirical analysis of cascade deformable models for multi-view face detection', *Image and Vision Computing* .

Ozuysal, M., Fua, P. & Lepetit, V. (2007), Fast keypoint recognition in ten lines of code, *in* 'IEEE Conf. on Computer Vision and Pattern Recognition'.

Pennec, X. (2006), 'Intrinsic statistics on riemannian manifolds: basic tools for geometrical measurements', *J. Math Imaging Vis* **25**, 127–154.

Pentland, A. (2007), 'Social signal processing', *IEEE Signal Processing Magazine* **24**, 108–111.

Ponce-Lopez, V., Chen, B., Oliu, M., Cornearu, C., Clapes, A., Guyon, I., Baro, S., Escalante, H. & Escalera, S. (2016), Chalearn lap 2016: First round challenge on first impressions - datasets and results, *in* 'European Conf. on Comp. Vision Workshop'.

Quak, E., Sivakumar, N. & Ward, J. (1993), 'Least squares approximation by radial functions', *SIAM J. Math. Anal.* **24**, 1043–1066.

Ramsay, J. & Silverman, B. (1997), *Functional Data Analysis*, Springer.

Ratliffe, S., Heller, G. & Leader, L. (2002), 'Functional data analysis with application to periodically stimulated foetal heart rate data. i: Functional regression', *Stat. in Med.* **21**, 1115–1127.

Ren, S., Cao, X., Wei, Y. & Sun, J. (2014), Face alignment at 3000 FPS via regressing local binary features, *in* 'IEEE Conf. on Computer Vision and Pattern Recognition', pp. 1685–1692.

Sagonas, C., Antonakos, E., Tzimiropoulos, G., Zafeiriou, S. & Pantic, M. (2016), '300 faces in-the-wild challenge: Database and results', *Image and Vision Computing* **47**, 3–18.

Sagonas, C., Tzimiropoulos, G., Zafeiriou, S. & Pantic, M. (2013*a*), 300 faces in-the-wild challenge: the first facial landmark localization challenge, *in* 'Int'l Conf. Computer Vision - Workshop'.

Sagonas, C., Tzimiropoulos, G., Zafeiriou, S. & Pantic, M. (2013*b*), A semi-automatic methodology for facial landmark annotation, *in* 'IEEE Conf. on Computer Vision and Pattern Recognition - Workshops'.

Sánchez-Lozano, E., Argones-Rua, E. & Alba-Castro, J. (2013), Blockwise linear regression for face alignment, *in* 'British Machine Vision Conf.'.

Sánchez-Lozano, E., De la Torre, F. & Gonzalez-Jimenez, D. (2012), Continuous regression for non-rigid image alignment, *in* 'European Conf. on Computer Vision', pp. 250–263.

Sánchez-Lozano, E., Martinez, B., Tzimiropoulos, G. & Valstar, M. (2016), Cascaded continuous regression for real-time incremental face tracking, *in* 'European Conf. on Computer Vision'.

Sánchez-Lozano, E., Martinez, B. & Valstar, M. (2016), 'Cascaded regression with sparsified feature covariance matrix for facial landmark detection', *Pattern Recognition Letters* **73**(1), 19–26.

Sánchez-Lozano, E., Tzimiropoulos, G., Martinez, B., De la Torre, F. & Valstar, M. (2016), 'A functional regression approach to facial landmark tracking', *Submitted to IEEE Trans. on Pattern Analysis and Machine Intelligence* .

Saragih, J. & Göcke., R. (2007), A nonlinear discriminative approach to aam fitting, *in* 'Int'l Conf. Computer Vision'.

Saragih, J. & Göcke., R. (2009), 'Learning aam fitting through simulation', *Pattern Recognition* **42**(11), 2628–2636.

Saragih, J. M., Lucey, S. & Cohn, J. F. (2011), 'Deformable model fitting by regularized landmark mean-shift', *Int'l Journal of Computer Vision* **91**(2), 200–215.

Shen, J., Zafeiriou, S., Chrysos, G. S., Kossaifi, J., Tzimiropoulos, G. & Pantic, M. (2015), The first facial landmark tracking in-the-wild challenge: Benchmark and results, *in* 'Int'l Conf. Computer Vision - Workshop'.

Solomon, C., Valstar, M. F., Morriss, R. K. & Crowe, J. (2015), 'Objective methods for reliable detection of concealed depression', *Frontiers in ICT* **2**, 5.

Torres Torres, M., Valstar, Michel, H. C., Ward, C. & Sharkey, D. (2017), Small sample deep learning for newborn gestational age estimation, *in* 'Proceedings of the IEEE 2017 Conference on Face and Gesture Recognition'.

Tresadern, P. A., Ionita, M. C. & Cootes, T. F. (2012), 'Real-time facial feature tracking on a mobile device', *Int'l Journal of Computer Vision* **96**, 280–289.

Tresadern, P., Sauer, P. & Cootes, T. F. (2010), Aditive update predictors in active appearance models, *in* 'British Machine Vision Conf.'.

Tzimiropoulos, G. (2015), Project-out cascaded regression with an application to face alignment, *in* 'IEEE Conf. on Computer Vision and Pattern Recognition', pp. 3659–3667.

Tzimiropoulos, G. & Pantic, M. (2014), Gauss-newton deformable part models for face alignment in-the-wild, *in* 'IEEE Conf. on Computer Vision and Pattern Recognition', pp. 1851–1858.

Valstar, M. (2014), Automatic behaviour understanding in medicine, *in* 'Proceedings of the 2014 Workshop on Roadmapping the Future of Multimodal Interaction Research including Business Opportunities and Challenges', ACM, pp. 57–60.

Valstar, M. F., Martinez, B., Binefa, X. & Pantic, M. (2010), Facial point detection using boosted regression and graph models, *in* 'IEEE Conf. on Computer Vision and Pattern Recognition', pp. 2729–2736.

Valstar, M. F., Sánchez-Lozano, E., Cohn, J. F., Jeni, L. A., Girard, J. M., Yin, L., Zhang, Z. & Pantic, M. (2017), Fera 2017 - addressing head pose in the third facial expression recognition and analysis challenge, *in* '12th IEEE Int'l Conf. and Workshops on Automatic Face and Gesture Recognition (FG)'.

Viola, P. & Jones, M. (2004), 'Robust real-time face detection', *Int'l Journal of Computer Vision* **57**(2), 137–154.

Wang, J., Chiou, J. & Muller, H. (2015), 'Review of functional data analysis', *arXiv* .

Wang, Y., Lucey, S. & Cohn, J. F. (2008), Enforcing convexity for improved alignment with constrained local models, *in* 'CVPR', pp. 1–8.

Xiao, S., Yan, S. & Kassim, A. A. (2015), Facial landmark detection via progressive initialization, *in* 'Int'l Conf. Computer Vision - Workshop'.

Xiong, X. & De la Torre, F. (2013), Supervised descent method and its applications to face alignment, *in* 'IEEE Conf. on Computer Vision and Pattern Recognition'.

Xiong, X. & la Torre, F. D. (2014), 'Supervised descent method for solving nonlinear least squares problems in computer vision', *arXiv* **abs/1405.0601**.

Yang, J., Deng, J., Zhang, K. & Liu, Q. (2015), Facial shape tracking via spatio-temporal cascade shape regression, *in* 'Int'l Conf. Computer Vision - Workshop'.

Zhang, J., Shan, S., Kan, M. & Chen, X. (2014), Coarse-to-fine auto-encoder networks (cfan) for real-time face alignment, *in* 'European Conf. on Computer Vision'.

Zhou, X. S., Gupta, A. & Comaniciu, D. (2005), 'An information fusion framework for robust shape tracking', *Trans. on Pattern Analysis and Machine Intelligence* **27**(1), 115–129.

Zhu, S., Li, C., Loy, C. & Tang, X. (2015), Face alignment by coarse-to-fine shape searching, *in* 'IEEE Conf. on Computer Vision and Pattern Recognition'.

Zhu, X. & Ramanan, D. (2012), Face detection, pose estimation, and landmark localization in the wild, *in* 'IEEE Conf. on Computer Vision and Pattern Recognition', pp. 2879–2886.