

Imperial College London
Department of Computing

Advances in Compositional Fitting of Active Appearance Models

Joan Alabort-i-Medina

November, 2016

Submitted in part fulfilment of the requirements for the degree of
Doctor of Philosophy in Computing and the Diploma of Imperial College.

Abstract

This thesis presents a detailed and complete study of *compositional gradient descent* (CGD) algorithms for fitting *active appearance models* (AAM) and advances the state-of-the-art in generative AAM fitting by incorporating: (i) novel *robust* texture representations; (ii) novel *cost functions* and *compositional types*; and (iii) *combined* fitting approaches with complementary deformable models; into the original CGD framework. In particular, a robust texture representation based on image gradient orientations is used to define a new type of generative deformable model that generalizes well to variations in identity, pose, expression, illumination and occlusions and that can be fitted to images using standard CGD algorithms. Moreover, a novel *Bayesian formulation* of the AAM fitting problem, which can be interpreted as a probabilistic generalization of the well-known *project-out inverse compositional* (PIC) algorithm, is proposed along with two new types of composition, *asymmetric* and *bidirectional*, that lead to better convergent and more robust CGD fitting algorithms. At the same time, interesting insights into existent strategies used to derive fast and exact simultaneous CGD algorithms are provided by reinterpreting them as direct applications of the *Schur complement* and the *Wiberg method*. Finally, CGD algorithms are combined with similar generative fitting techniques for *constrained local models* (CLM) to create a unified probabilistic fitting framework that combines the strengths of both models (AAM and CLM) and produces state-of-the-art results on the problem of non-rigid face alignment in the wild.

Acknowledgements

I would like to thank my supervisor Dr. Stefanos Zafeiriou for his invaluable guidance and support throughout the time I spent doing my PhD. I am also thankful to Prof. Maja Pantic for her advice and for recruiting me to the iBUG group. I would also like to thank Dr. Ioannis Marras and, in particular Dr. Georgios Tzimiropoulos, for their support and advice in the early stages of my PhD.

My most heartfelt gratitude goes to the members of the Menpo team and, in particular, to Nontas Antonakos, James Booth and Patrick Snape for being the most amazing colleagues one could hope to work with. My gratitude and appreciation also goes to all my colleagues at Imperial College London, and, especially, to all members of the iBUG group.

Finally, my deepest thanks and gratitude go to my parents, my brother and my soul-mate Adriana for their unconditional love and support. This thesis is dedicated to them and to my wonderful grandmother *Fina*.

Dedication

Pels meus pares, el meu germà i l'Adri; i per descomptat, la meva quita.

‘A mask tells us more than a face.’

Oscar Wilde

Contents

Abstract	i
Acknowledgements	iii
1 Introduction	1
1.1 Motivation and objectives	1
1.2 Contributions	4
1.3 Statement of originality	6
1.4 Publications	6
1.5 Copyright	7
2 Literature review	8
2.1 Introduction	8
2.2 Holistic deformable models	9
2.3 Part-based deformable models	10
2.4 Direct regression techniques	11

3	Active appearance models	13
3.1	What are AAM?	14
3.1.1	Shape model	14
3.1.2	Texture model	16
3.1.3	Deformation model	17
3.1.4	Formal definition	18
3.2	Fitting AAM	19
3.2.1	Sum of squared differences	20
3.2.2	Fitting algorithm	21
3.2.2.1	Compositional gradient descent	22
4	Active orientation models	28
4.1	Motivation	29
4.2	Weighted non-linear least squares	30
4.3	Image gradient orientations	31
4.4	IGO-based texture model	34
4.5	Fitting AOM	36
4.5.1	Project-out inverse compositional	36
4.5.2	Alternate inverse compositional algorithm	37
4.6	Results	39
4.6.1	XM2VTS	40

4.6.2	Multi-PIE	41
4.7	Conclusions	43
5	Unifying compositional AAM fitting	45
5.1	Motivation and outline	46
5.2	Cost Function	47
5.2.1	Sum of squared differences	48
5.2.1.1	Probabilistic formulation	49
5.2.2	Project-out	51
5.2.2.1	Probabilistic formulation	53
5.3	Type of Composition	55
5.3.1	Forward	57
5.3.2	Inverse	58
5.3.3	Asymmetric	59
5.3.4	Bidirectional	61
5.4	Optimization Method	62
5.4.1	Gauss-Newton	63
5.4.1.1	Simultaneous	65
5.4.1.2	Alternated	69
5.4.2	Newton	72
5.4.2.1	Simultaneous	74

5.4.2.2	Alternated	78
5.4.3	Wiberg	80
5.5	Relation to prior work	84
5.5.1	Project-out algorithms	85
5.5.2	SSD algorithms	86
5.6	Results	89
5.6.1	Comparison on LFPW	92
5.6.1.1	SSD Gauss-Newton algorithms	93
5.6.1.2	SSD Newton algorithms	94
5.6.1.3	SSD Wiberg algorithms	95
5.6.1.4	PO Gauss-Newton algorithms	95
5.6.1.5	PO Newton algorithms	96
5.6.1.6	PO Wiberg algorithms	96
5.6.2	Weighted Bayesian project-out	96
5.6.3	Optimal asymmetric composition	98
5.6.4	Sampling and number of iterations	99
5.6.5	Comparison on Helen and AFW	100
5.6.6	Analysis of the results	102
5.7	Conclusions	103

6	Combining AAM and CLM	115
6.1	Motivation	116
6.2	Constrained local models	118
6.2.1	Fitting CLM	119
6.2.1.1	Probabilistic interpretation	120
6.3	A unified approach for AAM and CLM fitting	122
6.4	Results	125
6.4.1	Comparison with AAM and CLM	126
6.4.2	Comparison with state-of-the-art	127
6.5	Conclusions	128
7	Conclusion	130
7.1	Future Work	132
	Bibliography	133
	Appendices	
A	Deformation models	147
A.1	Warp	147
A.1.1	Piece-wise affine	147
A.1.2	Thin plate splines	149
A.2	Warp Jacobian	151

A.2.1	Piece-wise affine	151
A.2.2	Thin plate splines	152
A.3	Warp inversion	154
A.4	Warp composition	154
A.4.1	Piece-wise affine	155
A.4.2	Thin plate splines	155
B	Efficient second-order minimization	158
C	Newton Hessian terms	162
C.1	Asymmetric	162
C.2	Bidirectional	163
D	Iterative solutions	165

List of Tables

D.1 Iterative solutions for all SSD algorithms studied in chapter 5. 166

D.2 Iterative solutions of all PO algorithms studied in chapter 5. . 167

List of Figures

3.1	Exemplar images from the <i>labeled faces parts in-the-wild</i> (LFPW) dataset [1] for which a consistent set of sparse landmarks representing the shape of the object being model (human face) has been manually defined [2, 3].	14
3.2	Visual representation of the shape model. A novel shape instance, $\mathbf{s}$, generated by the model is shown in the left most figure. The remaining figures show the mean shape, $\bar{\mathbf{s}}$, and the two first shape bases, $\mathbf{s}_1$ and $\mathbf{s}_2$, respectively. The shape bases are depicted as overlaid vectors on top of the mean shape.	15
3.3	Visual representation of the texture model. A novel texture instance, $\mathbf{a}$, generated by the model is shown in the left most figure. The remaining figures show the mean texture, $\bar{\mathbf{a}}$, and the two first texture bases, $\mathbf{a}_1$ and $\mathbf{a}_2$, respectively.	17

3.4	Visual representation of the deformation model. An image I is warped onto the reference frame Ω by the deformation model $\mathcal{W}(\mathbf{x}; \mathbf{p})$. The deformation model is completely defined by the relative position of the sparse set of landmarks defining the shape model (and controlled by the shape parameters $\mathbf{p}$) in both images, for which direct correspondences are always known.	18
3.5	Schematic representation of the forward and inverse compositions. Forward composition solves the image alignment problem iteratively by introducing the incremental warp $\mathcal{W}(\mathbf{x}; \Delta\mathbf{p})$ on the image side, solving for the increments $\Delta\mathbf{p}$ and composing the resulting incremental warp with the current estimate of the warp $\mathcal{W}(\mathbf{x}; \mathbf{p})$. Inverse composition solves the same problem by introducing the incremental warp $\mathcal{W}(\mathbf{x}; \Delta\mathbf{q})$ on the model side, solving for the increments $\Delta\mathbf{q}$, inverting the incremental warp and composing it with the current estimate of the warp $\mathcal{W}(\mathbf{x}; \mathbf{p})$	25
4.1	Results showing the fitting accuracy of AOM, Indep. [4], CLM [5] and AAM on the XM2VTS and Multi-PIE datasets.	44
5.1	The Bayesian project-out formulation fits AAM by minimizing two different distances: <i>i</i>) the Mahalanobis distance <i>within</i> the linear texture subspace; and <i>ii</i>) the Euclidean distance <i>to</i> the linear texture subspace (i.e. the Euclidean distance within its orthogonal complement) weighted by the inverse of the estimated image noise.	54

5.2	Exemplar initializations obtained by varying the percentage of uniform noise added to the similarity parameters. Note that, increasing the percentage of noise produces more challenging initialization.	89
5.3	Subset of pixels on the reference frame used to optimize the SSD and PO cost functions for different sampling rates. . . .	99
5.4	Results showing the fitting accuracy and fitting and cost convergence properties of the SSD Gauss-Newton algorithms on the LFPW test dataset initialized with 5% uniform noise. . .	105
5.5	Results showing the fitting accuracy and fitting and cost convergence properties of the SSD Newton algorithms on the LFPW test dataset initialized with 5% uniform noise.	106
5.6	Results showing the fitting accuracy and fitting and cost convergence properties of the SSD Wiberg algorithms on the LFPW test dataset initialized with 5% uniform noise.	107
5.7	Results showing the fitting accuracy and fitting and cost convergence properties of the PO Gauss-Newton algorithms on the LFPW test dataset initialized with 5% uniform noise. . .	108
5.8	Results showing the fitting accuracy and fitting and cost convergence properties of the PO Newton algorithms on the LFPW test dataset initialized with 5% uniform noise.	109
5.9	Results showing the fitting accuracy and fitting and cost convergence properties of the PO Wiberg algorithms on the LFPW test dataset initialized with 5% uniform noise	110

5.10	Results quantifying the effect of varying the value of the parameters $\rho = 1 - \gamma$ in PO Gauss-Newton algorithms on the LFPW test dataset initialized with 5% noise.	111
5.11	Results assessing the effectiveness of sampling for the best performing PO and SSD algorithms on the LFPW database. Colors encode overall fitting accuracy, from highest to lowest: red, orange, yellow, green, blue and purple.	112
5.12	Results quantifying the effect of varying the value of the parameters $\alpha = 1 - \beta$ for asymmetric algorithms on the LFPW dataset initialized with 5% noise.	113
5.13	Results showing the fitting accuracy for the SSD and PO asymmetric Gauss-Newton algorithms on the Helen test dataset and the entire AFW dataset both initialized with 5% noise. .	114
6.1	Results showing the fitting accuracy and convergence properties together with average run-time for the unified, AAM and CLM algorithms on the LFPW and Helen test datasets; and the fitting accuracy for the previous algorithms as well as for SDM and GNDPM on the entire AWF dataset.	129

Chapter 1

Introduction

1.1 Motivation and objectives

Due to their numerous applications in human computer interaction, face analysis and recognition, and medical imaging, the problem of *learning* and *fitting deformable models* (DM) [6, 7, 8, 9, 10, 11, 12, 5, 13, 1, 14, 4, 15, 16, 15, 17, 18, 19, 20, 21, 21, 22, 23, 24, 25, 26, 27, 28, 29] has been the focus of cutting edge research in *computer vision* and *machine learning* over the last two decades. Put in simple terms, this problem consist of:

1. Learning a deformable model by:
 - Annotating a set of landmarks defining the shape of the object being modelled (e.g. the human face) over a large collection of training images.
 - Learning a shape representation (known as shape model or point distribution model) that effectively represents the structure and variations of the object's shape.

- Learning an effective texture representation (known as texture model) of the object using the image region defined by the annotated landmarks.
2. Fitting the learned deformable model by using its shape and texture models to infer the location of the set of landmarks defining object's shape in a new unseen image. This, is a non-linear optimization problem that can be solved using regression and classification techniques as well as direct analytical optimization.

Active appearance models (AAM) [8, 10] are one of the most popular and well-established techniques for solving the previous problem. AAM use linear statistical models to describe the shape and texture of objects, and can be fitted to images by solving a non-linear optimization problem that requires the minimization of a error measure between the images and the texture model of the AAM. Several approaches [8, 30, 10, 31, 32, 33, 34, 35, 36, 37, 11, 38, 39, 40, 41] have been proposed to define and solve this optimization problem. Broadly speaking, they can be divided into two different groups:

- *Regression* based [8, 30, 31, 33, 36, 11]. Regression based techniques attempt to solve the problem by learning a direct function mapping between the error measure and the optimal values of the AAM's parameters. Most notable approaches include variations on the original [8] fixed linear regression approach of [30, 33], the adaptive linear regression approach of [31], and the works of [36] and [11] which considerably improved upon previous techniques by using boosted regression.
- *Optimization* based [10, 32, 34, 38, 40]. Optimization based methods

for fitting AAM were proposed by Matthews and Baker in [10]. These techniques are known as *compositional gradient decent* (CGD) algorithms and are based on direct analytical optimization of the error measure. Popular CGD algorithms include the very efficient *project-out inverse compositional* (PIC) algorithm [10], the accurate but costly *simultaneous inverse compositional* (SIC) algorithm [32], and the more efficient versions of SIC presented in [34] and [40].

Regardless of the approach, in the sub-field of *non-rigid face alignment*, AAM have often been criticized [42, 32] for their inability to generalize well beyond the conditions seen in the training set and have been regarded as inadequate for fitting *in the wild* images i.e. images obtained in uncontrolled recording setting typically containing large variations in terms of identity, pose, expression and illumination and containing occlusions. Consequently, AAM fitting has been most successfully applied, in terms of non-rigid face alignment, to controlled recording scenarios and person-specific applications.

The main objectives of this thesis are:

1. To offer a detailed study of AAM and, in particular, of CGD algorithms for AAM fitting.
2. To understand the limitations of these models and their fitting algorithms.
3. To develop novel and efficient CGD algorithms for AAM fitting that overcome the limitations of existing approaches and work well in the challenging problem of non-rigid face alignment in the wild.

The remainder of the thesis is structured as follows. Section 2 contains a con-

cise literature review of prior work in the field of non-rigid object alignment. Chapter 3 serves as a detailed introduction to AAM. Chapter 4 introduces a robust image representation based on image gradient orientations to increase the robustness and generalization abilities of CGD algorithms. Chapter 5 presents a unified framework for CGD algorithms and proposes several enhancements with respect to three of their main characteristics: (i) cost function (section 5.2); (ii) composition types (section 5.3); and (iii) optimization methods (section 5.4). Chapter 6 proposes a novel deformable fitting approach by combining AAM with another widely used and popular deformable model, *constrained local models* (CLM). Finally, conclusions and future work are presented in chapter 7.

1.2 Contributions

The contributions of this thesis are described below in further detailed:

1. **Active orientation models.** Chapter 4 introduces *active orientation models* (AOM); a new type of generative parametric model of shape and texture that generalizes better than *active appearance models* (AAM) to unseen variations in identity, expression and illumination, as well as occlusions. AOM are built upon AAM and like AAM, they are composed of: shape, texture, and deformation models. Their deformation model and shape models are the same ones used by AAM, but their texture model is defined using a robust subspace learning technique [43] that naturally leads to a robust formulation of existing *compositional gradient descent* (CGD) algorithms.
2. **Unified framework for CGD algorithms.** Chapter 5 presents

a unified and complete overview of the most relevant and recently published CGD algorithms for fitting AAM [10, 32, 34, 37, 38, 40] and proposes a unified framework in which these algorithms can be classified with respect to three of their main characteristics:

- (a) The cost function defining the fitting problem (section 5.1).
- (b) The type of composition used (section 5.3).
- (c) The method used to solve the non-linear optimization problem (section 5.4).

3. **Bayesian project-out cost function.** Section 5.2 reviews the probabilistic formulation of AAM and proposes a novel *Bayesian formulation* of the fitting problem. This novel formulation is obtained by assuming a probabilistic model for texture generation with both Gaussian noise and a Gaussian prior over a latent texture space. The resulting Bayesian cost function, which is obtained by marginalizing out the latent texture space, can be interpreted as a probabilistic generalization of the well-known *project-out* cost function [10].

4. **Asymmetric and bidirectional compositions.** Section 5.3 proposes the use of two novel types of composition that had never been used before in AAM: *i) asymmetric*; and *ii) bidirectional*. These types of composition have been widely used in the related field of parametric image alignment [44, 45, 46, 47] and make use of the gradients of both the image and texture model to derive faster convergent and more robust CGD algorithms.

5. **Schur and Wiberg methods.** Section 5.4 provides valuable insights into existent strategies used to derive fast and exact simultaneous CGD

algorithms [34, 37, 40] by reinterpreting them as direct applications of the *Schur complement* [48] and the *Wiberg method* [49, 50].

6. **Combining AAM with CLM.** Chapter 6 proposes to combine two of the most popular techniques in deformable model fitting: AAM and *constrained local models* (CLM) [6, 5]. To this end, a novel probabilistic formulation of the fitting problem that unifies the fitting procedures of both models is proposed. This approach explicitly and optimally combines the generative and discriminative texture models of AAM and CLM using a *maximum a posteriori* (MAP) estimation framework. The result is a unified cost function that can be iteratively solved using the Gauss-Newton method and in which the solution at each iteration is given by an optimal, in a MAP sense, weighted combination of the original AAM and CLM iterative solutions.

1.3 Statement of originality

This thesis is entirely my own work, and, except where otherwise indicated, describes my own research.

1.4 Publications

The work presented in this thesis resulted in the following publications:

- G. Tzimiropoulos, J. Alabort-i-Medina, S. Zafeiriou, and M. Pantic, “Generic active appearance models revisited,” in *IEEE Asian conference on computer vision (ACCV)*, 2012

- G. Tzimiropoulos, J. Alabort-i-Medina, S. Zafeiriou, and M. Pantic, “Active orientation models for face alignment in-the-wild,” *IEEE Transactions on information forensics and security (TIFS)*, 2014
- J. Alabort-i-Medina and S. Zafeiriou, “Bayesian active appearance models,” in *IEEE Conference on computer vision and pattern recognition (CVPR)*, 2014
- J. Alabort-i-Medina, E. Antonakos, J. Booth, P. Snape, and S. Zafeiriou, “Menpo: A comprehensive platform for parametric image alignment and visual deformable models,” in *ACM international conference on multimedia (ACMM)*, 2014
- J. Alabort-i-Medina and S. Zafeiriou, “Unifying holistic and parts-based deformable model fitting,” in *IEEE Conference on computer vision and pattern recognition (CVPR)*, 2015
- J. Alabort-i-Medina and S. Zafeiriou, “A unified framework for compositional fitting of active appearance models,” *International journal of computer vision (IJCV)*, 2016

1.5 Copyright

The copyright of this thesis rests with the author and is made available under a Creative Commons Attribution Non-Commercial No Derivatives licence. Researchers are free to copy, distribute or transmit the thesis on the condition that they attribute it, that they do not use it for commercial purposes and that they do not alter, transform or build upon it. For any reuse or redistribution, researchers must make clear to others the licence terms of this work.

Chapter 2

Literature review

This section reviews prior work on the prolific field of non-rigid object alignment. A strong emphasis is put on reviewing techniques used in the sub-field of *non-rigid face alignment* because this is the main area of application of the work presented in this thesis.

2.1 Introduction

Non-rigid object alignment has been an active area of research in computer vision and machine learning for over 20 years. Solving this problem consists of registering a shape model to an image such that the set of landmarks defining the model accurately describe the shape of the object being modelled on the image. Throughout the years, a large variety of techniques have been proposed to solve this problem [6, 7, 8, 9, 10, 11, 12, 5, 13, 1, 14, 4, 15, 16, 15, 17, 18, 19, 20, 21, 21, 22, 23, 24, 25, 26, 27, 28, 29].

In general, research in this area can be divided into three different groups:

- *Holistic deformable models* (HDM).
- *Parts-based deformable models* (PBDM).
- *Direct regression techniques* (DRT).

2.2 Holistic deformable models

HDM are global statistical representations of the shape and texture of a particular object. These models were popularized by the seminal works of Cootes et al. [8] and Blanz and Vetter [7]. They have a long tradition in computer vision and have been widely used to solve the problems of rigid and non-rigid object alignment and tracking.

Cootes et al. [8] proposed to fit *active apperance models* (AAM) by learning a fixed linear relation from texture differences (between the image and the global texture model) to shape parameters. The authors of [12] and [11] extended this approach by using boosted regression considerably improving the accuracy and convergence of the results. Cootes and Taylor [9] showed that the use of non-linear image features for texture representation led to better fitting performance in AAM fitting. On the other hand, Blanz and Vetter [7] use stochastic gradient descent to fit *3D morphable models* (3DMMs) to images. In [10], Matthews and Baker proposed a general *compositional gradient descent* (CGD) framework for fitting AAM and proposed the very efficient *project-out inverse compositional* (PIC) algorithm. Amberg et al. proposed several variations of the original PIC algorithm in [37]. Gross et al. [32] increased the accuracy of this algorithm, at expenses of its efficiency, by proposing the *simultaneous inverse compositional* (SIC) algorithm. Equivalent efficient versions of the SIC algorithm were introduced in [34] and [40].

The authors of [21] extended the CGD algorithms to the Fourier domain to efficiently enable convolution with *Gabor* filters, increasing their robustness. Antonakos et al. showed in [57] that highly-descriptive feature descriptors such as the *scale-invariant feature transform* (SIFT) [58] and *histogram of oriented gradients* (HOG) [59] could be used to define the texture representation of AAM leading to dramatic improvements in fitting accuracy. Very recently, the authors of [28] proposed to replace the linear shape and texture models typically used in AAM for *deep* shape and texture models based on *restricted Boltzmann machines* (RBM).

HDM, and in particular AAM, have been criticized due to the limited representational power of their global linear texture representation. However, the work presented in this thesis suggest that this limitation might have been over-stressed in the literature and that global linear atexture models can produce highly accurate results when the appropriate: (i) training data (chapters 5 and 6); (ii) texture representations (chapters 4, 5 and 6); and (iii) fitting strategies (chapters 4, 5 and 6) are employed.

2.3 Part-based deformable models

PBDM where first introduced by Cootes et al. in [6] and later re-popularized by the work of Saragih et al. [5]. PBDM define object texture as the combination of several independent parts typically constrained by a global shape representation. Just like HDM, PBDM have enjoyed a long-standing popularity in computer vision.

Cootes et al. [6] proposed an iterative search procedure for fitting *active shape models* (ASM) that approximated local texture responses with

isotropic Gaussian estimators. Cristinacce and Cootes [60] built upon ASM to propose *constrained local models* (CLM) which used a combined statistical model of all parts to generate local response maps. Saragih et al. [5] derived a probabilistic interpretation of CLM and extended their original fitting approach by assuming a non-parametric representation of the local response maps. This approach was later extended by introducing a sample specific shape prior in [13] and a non-parametric shape prior in [1]. The authors of [14] proposed to use several independent *principal component analysis* (PCA) priors to model the object’s shape. On the other hand, Asthana et al. [22] proposed a robust cascade-regression approach for fitting CLM. Recently, Martins et al. [23] proposed a CLM fitting strategy that defines a non-Gaussian posterior distribution and performs inference via efficient *regularized particle filters* (RPF). The same authors proposed to cluster the local texture around each landmarks and learn a separate classifier per cluster in order to increase the discriminative power of the response maps [61]. The authors of [24] recently proposed the use of *continuous conditional neural fields* (CCNF) to learn local patch experts. Finally, Zhu and Ramanan [4] used a tree-based shape model to derive a parts-based fitting strategy that could be simultaneously used for detection, pose estimation and landmark localization.

2.4 Direct regression techniques

DRT attempt to solve the problem of non-rigid object alignment directly, without the use of explicit shape and texture deformable models. To this end, they aim at learning a direct function mapping between the appearance of the object and the position of its landmarks on the image.

Cao et. al [15] proposed a two-level cascade boosted regression approach for learning the mapping between shape-indexed image features (computed across the whole image for each landmark) and landmark positions. Similarly, Xiong and De la Torre [20] proposed to learn a cascade of linear regressors from local image descriptors (*scale invariant feature transform* (SIFT) extracted around each landmark) to landmark positions. The authors of [19] explicitly incorporated information regarding the visibility of the landmarks into a similar cascade regression framework. Recently Asthana et al. [25], Kazemi et al. [26], Lee et al. [29] and Ren et al. [27] have augmented the original cascade regression framework of [15] by proposing an incremental algorithm for cascade regression learning, substituting linear regressors by ensembles of linear and *Gaussian processes* (GP) regression trees and by learning extremely descriptive binary features using regression forests, respectively. On the other hand, in [17] a method to effectively combine multiple landmark hypothesis using *structured support vector machines* (SSVM) was used to boost the results obtained with the previous cascade-regression frameworks. Finally, the authors of [16] and [18] used *kernel ridge regression* (KRR) and *deep convolutional neural network* (DCNN), respectively, to learn a direct mapping from the entire object image region (defined by the rectangular box obtained from object detection) to the position of the landmarks.

Chapter 3

Active appearance models

This chapter provides a detailed introduction to *active appearance models* (AAM) and serves as a necessary preamble to the advance concepts developed in forthcoming chapters.

The content of this chapter is based on the following publications:

- J. Alabort-i-Medina, “Advances in generic facial model fitting,” tech. rep., Department of computing, Imperial College London, 2012.
- J. Alabort-i-Medina and S. Zafeiriou, “Bayesian active appearance models,” in *IEEE Conference on computer vision and pattern recognition (CVPR)*, 2014.
- J. Alabort-i-Medina and S. Zafeiriou, “A unified framework for compositional fitting of active appearance models,” *International journal of computer vision (IJCV)*, 2016.



Figure 3.1: Exemplar images from the *labeled faces parts in-the-wild* (LFPW) dataset [1] for which a consistent set of sparse landmarks representing the shape of the object being model (human face) has been manually defined [2, 3].

3.1 What are AAM?

AAM [8, 10] are generative parametric models that explain visual variations, in terms of shape and texture, within a particular object class. AAM are built from a collection of images for which the spatial position of a sparse set of v landmark points $\mathbf{x}_i = (x_i, y_i)^T \in \mathbb{R}^2$ representing the shape $\mathbf{s} = (x_1, y_1, \dots, x_v, y_v)^T \in \mathbb{R}^{2v \times 1}$ of the object being modeled have been manually defined a priori.

AAM are themselves composed of three different models: (i) *shape model*; (ii) *texture model*; and (iii) *deformation model*.

3.1.1 Shape model

The shape model [63, 64], also referred to as *point distribution model* (PDM), is obtained by typically applying *principal component analysis* (PCA) to the set of object's shapes. The resulting shape model is mathematically expressed as:

$$\begin{aligned} \mathbf{s} &= \bar{\mathbf{s}} + \sum_{i=1}^n p_i \mathbf{s}_i \\ &= \bar{\mathbf{s}} + \mathbf{S} \mathbf{p} \end{aligned} \tag{3.1}$$

where $\bar{\mathbf{s}} \in \mathbb{R}^{2v \times 1}$ is the mean shape, and $\mathbf{S} \in \mathbb{R}^{2v \times n}$ and $\mathbf{p} \in \mathbb{R}^{n \times 1}$ denote the

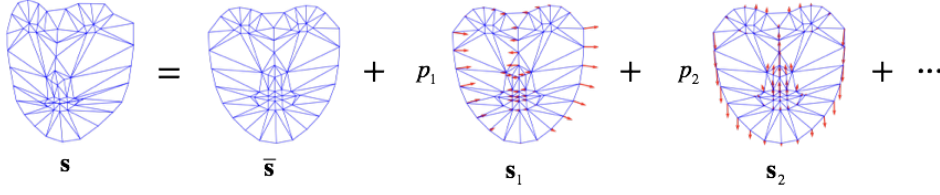


Figure 3.2: Visual representation of the shape model. A novel shape instance, $\mathbf{s}$, generated by the model is shown in the left most figure. The remaining figures show the mean shape, $\bar{\mathbf{s}}$, and the two first shape bases, $\mathbf{s}_1$ and $\mathbf{s}_2$, respectively. The shape bases are depicted as overlaid vectors on top of the mean shape.

shape bases and shape parameters, respectively. Figure 3.2 provides a visual descriptions of this model. In order to allow a particular shape instance $\mathbf{s}$ to be arbitrarily positioned in space, the shape model can be augmented with a global similarity transform. Note that this normally requires the initial shapes to be normalized with respect to a global similarity transform (typically using *Procrustes analysis* [63]) before PCA is applied. This results in the following expression for each landmark point of the shape model:

$$\mathbf{x}_i = s\mathbf{R}(\bar{\mathbf{x}}_i + \mathbf{X}_i\mathbf{p}) + \mathbf{t} \quad (3.2)$$

where s , $\mathbf{R} \in \mathbb{R}^{2 \times 2}$ and $\mathbf{t} \in \mathbb{R}^2$ denote the scale, rotation and translation applied by the global similarity transform, respectively. Using the orthonormalization procedure described in [10]¹, which consists in choosing an aux-

¹In this formulation shapes are able to move under a global similarity transform, as well as the original linear shape variation. This is not the same as moving under the linear shape variation followed by a global similarity transform as described by equation 3.2. However, the differences between both formulation are minimal and they only affect the relative scale of the shape parameters $\mathbf{p}$.

iliary set of shape bases $\mathbf{s}^*$ so that:

$$\begin{aligned}\mathbf{s}_1^* &= \bar{\mathbf{s}} = (\bar{\mathbf{x}}_1, \bar{\mathbf{y}}_1, \dots, \bar{\mathbf{x}}_v, \bar{\mathbf{y}}_v) \\ \mathbf{s}_2^* &= (-\bar{\mathbf{y}}_1, \bar{\mathbf{x}}_1, \dots, -\bar{\mathbf{y}}_v, \bar{\mathbf{x}}_v) \\ \mathbf{s}_3^* &= (1, 0, \dots, 1, 0) \\ \mathbf{s}_4^* &= (0, 1, \dots, 0, 1)\end{aligned}\tag{3.3}$$

, the final expression for the shape model can be compactly written as the linear combination of a set of shape bases:

$$\begin{aligned}\mathbf{s} &= \bar{\mathbf{s}} + \sum_{i=1}^4 p_i^* \mathbf{s}_i^* + \sum_{i=1}^n p_i \mathbf{s}_i \\ &= \bar{\mathbf{s}} + \mathbf{S} \mathbf{p}\end{aligned}\tag{3.4}$$

where $\mathbf{S} = (\mathbf{s}_1^*, \dots, \mathbf{s}_4^*, \mathbf{s}_1, \dots, \mathbf{s}_n) \in \mathbb{R}^{2v \times (4+n)}$ and $\mathbf{p} = (p_1^*, \dots, p_4^*, p_1, \dots, p_n)^T \in \mathbb{R}^{(4+n) \times 1}$ are redefined as the concatenation of the similarity bases $\mathbf{s}_i^*$ and similarity parameters p_i^* with the original $\mathbf{S}$ and $\mathbf{p}$ defined in equation 3.1, respectively.

3.1.2 Texture model

The texture model [65, 66] is obtained by first warping the original images onto a common reference frame and applying PCA to the obtained warped images. The warping procedure is explained in section 3.1.3. These warped images are referred to as *shape-free images* in the literature [67]. Mathematically, the texture model is defined by the following expression:

$$A(\mathbf{x}) = \bar{A}(\mathbf{x}) + \sum_{i=1}^m c_i A_i(\mathbf{x})\tag{3.5}$$

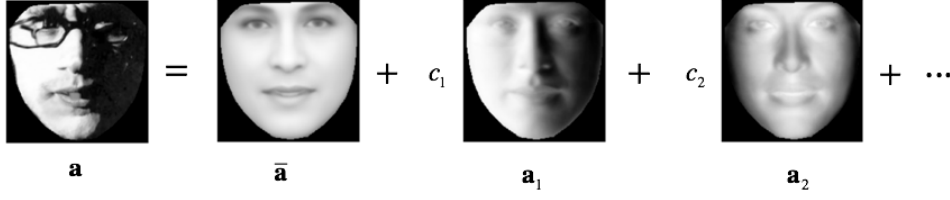


Figure 3.3: Visual representation of the texture model. A novel texture instance, $\mathbf{a}$, generated by the model is shown in the left most figure. The remaining figures show the mean texture, $\bar{\mathbf{a}}$, and the two first texture bases, $\mathbf{a}_1$ and $\mathbf{a}_2$, respectively.

where $\mathbf{x} \in \Omega$ denote a pixel position on the reference frame, and $\bar{A}(\mathbf{x})$, $A_i(\mathbf{x})$ and c_i denote the mean texture, the texture bases and texture parameters, respectively.

Let us define the operator vec as the operator that takes a shape-free image $A \in \mathbb{R}^{R \times C}$ and vectorizes it by concatenating its rows into a single column vector of length $F = R \times C$. Then, writing $\mathbf{a} = \text{vec}(A) \in \mathbb{R}^{F \times 1}$ as the vectorized version of the texture instance, equation 3.5 can be concisely written in vector form as:

$$\mathbf{a} = \bar{\mathbf{a}} + \mathbf{A}\mathbf{c} \quad (3.6)$$

where $\bar{\mathbf{a}} \in \mathbb{R}^{F \times 1}$ is the mean texture, and $\mathbf{A} \in \mathbb{R}^{F \times m}$ and $\mathbf{c} \in \mathbb{R}^{m \times 1}$ denote the texture bases and texture parameters, respectively. Figure 3.3 provides a visual representation of this model.

3.1.3 Deformation model

The role of the deformation model, denoted by $\mathcal{W}(\mathbf{x}; \mathbf{p})$, is to extrapolate the position of all pixel positions $\mathbf{x} \in \Omega$ in a reference frame to a particular image I and vice versa. The reference frame Ω is a template image defined using the mean shape $\bar{\mathbf{s}}$ of the shape model. The deformation model is

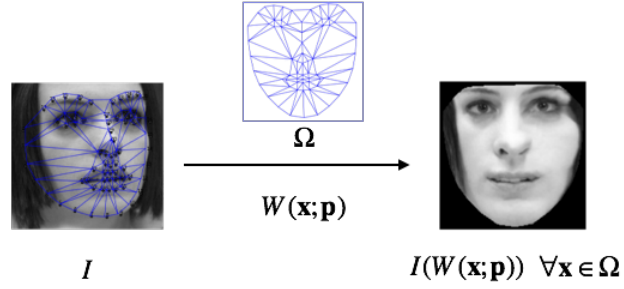


Figure 3.4: Visual representation of the deformation model. An image I is warped onto the reference frame Ω by the deformation model $\mathcal{W}(\mathbf{x}; \mathbf{p})$. The deformation model is completely defined by the relative position of the sparse set of landmarks defining the shape model (and controlled by the shape parameters $\mathbf{p}$) in both images, for which direct correspondences are always known.

completely defined by the relative position of the sparse set of landmarks defining the shape model (and controlled by the shape parameters $\mathbf{p}$) in both images, for which direct correspondences are always known. Figure 3.4 provides a visual representation of this model.

Classic deformation models for AAM are *piece-wise affine* (PWA) [67, 10] and *thin plate splines* (TPS) [67, 34] warps. A detailed description of PWA and TPS can be found in appendix A.

3.1.4 Formal definition

Given an image I containing the object of interest, its manually annotated ground truth shape $\mathbf{s}$, and a particular deformation model $\mathcal{W}(\mathbf{x}; \mathbf{p})$; the two main assumptions behind AAM are:

1. The ground truth shape of the object can be well approximated by the

shape model

$$\mathbf{s} \approx \bar{\mathbf{s}} + \mathbf{S}\mathbf{p} \quad (3.7)$$

2. The object's texture can be well approximated by the texture model after the image is warped, using the deformation model and the previous shape approximation, onto the reference frame:

$$\mathbf{i}[\mathbf{p}] \approx \bar{\mathbf{a}} + \mathbf{A}\mathbf{c} \quad (3.8)$$

where $\mathbf{i}[\mathbf{p}] \in \mathbb{R}^{F \times 1} = \text{vec}(I(\mathcal{W}(\mathbf{x}; \mathbf{p}))) \forall \mathbf{x} \in \Omega$ denotes the vectorized version of the warped image. Note that, the warp $\mathcal{W}(\mathbf{x}; \mathbf{p})$ which explicitly depends on the shape parameters $\mathbf{p}$, relates the shape and texture models and is a central part of the AAM formulation.

Because of the explicit use of the deformation model, these two assumptions provide a concise and formal definition of AAM. At this point, it is worth mentioning that the vector notation used in equations 3.7 and 3.8 will be, in general, the preferred notation throughout this thesis.

3.2 Fitting AAM

AAM can be *fitted* to images [8, 10] in order to obtain the set of parameters that best describe specific instances of the object being modelled. The process of fitting an AAM to an image is a non-linear optimization problem that entails the minimization of a *global* error measure between the vectorized warped image and the AAM's linear texture model.

3.2.1 Sum of squared differences

One of the most natural choices of error measure is the *sum of squared differences* (SSD) between the intensity value at each pixel. Note that, this choice is naturally given by second main assumption behind AAM, equation 3.8. Consequently, AAM fitting can be defined by the following non-linear optimization problem² [10]:

$$\begin{aligned} \mathbf{p}^*, \mathbf{c}^* &= \arg \min_{\mathbf{p}, \mathbf{c}} \frac{1}{2} \sum_{\mathbf{x} \in \Omega} \left\| \underbrace{I(\mathcal{W}(\mathbf{x}; \mathbf{p})) - \bar{A}(\mathbf{x}) - \sum_{i=1}^m c_i A_i(\mathbf{x})}_{R(\mathbf{x})} \right\|^2 \\ &= \arg \min_{\mathbf{p}, \mathbf{c}} \frac{1}{2} \sum_{\mathbf{x} \in \Omega} \|R(\mathbf{x})\|^2 \end{aligned} \quad (3.9)$$

In the previous expression, the input image I is first back-warped onto the reference frame Ω using the deformation model and the current value of the shape parameters $\mathbf{p}$. Then, the intensity value at every pixel position $\mathbf{x} \in \Omega$ on the warped image $I(\mathcal{W}(\mathbf{x}, \mathbf{p}))$ is subtracted from the intensity value of the same pixel position $\mathbf{x}$ on the current instance of the texture model $A(\mathbf{x}) = \bar{A}(\mathbf{x}) + \sum_{i=1}^m c_i A_i(\mathbf{x})$. Finally, these differences are squared and summed together to produce the final error measure between both images.

Using the vector notation preferred in this thesis, equation 3.9 can be alternatively expressed as²:

$$\begin{aligned} \mathbf{p}^*, \mathbf{c}^* &= \arg \min_{\mathbf{p}, \mathbf{c}} \frac{1}{2} \left\| \underbrace{\mathbf{i}[\mathbf{p}] - (\bar{\mathbf{a}} + \mathbf{A}\mathbf{c})}_{\mathbf{r}} \right\|^2 \\ &= \arg \min_{\mathbf{p}, \mathbf{c}} \frac{1}{2} \mathbf{r}^T \mathbf{r} \end{aligned} \quad (3.10)$$

²Note that the residuals $R(\mathbf{x})$ and r in equations 3.9 3.10 are linear with respect to the texture parameters $\mathbf{c}$ but non-linear with respect to the shape parameters $\mathbf{p}$ through the warp $\mathcal{W}(\mathbf{x}; \mathbf{p})$

It is important to notice that:

- The shape parameters control the way in which the input image is warped onto the reference frame by the warp.
- The SSD between the vectorized warped image and its reconstruction by the linear texture model is the quantity to be minimized and, consequently, the optimization problem is driven by the object's texture.
- Equation 3.10 should be minimized when the shape parameters correctly place the object landmarks onto the image. When this happens, the image should be correctly warped onto the reference frame and its reconstruction by the linear texture model should be optimal.

Therefore, fitting AAM to images consists of finding the optimal shape $\mathbf{p}^*$ and texture $\mathbf{c}^*$ parameters for which the expression in equation 3.10 is minimized.

3.2.2 Fitting algorithm

Throughout the years, several techniques have been proposed to fit AAM to images [8, 30, 10, 31, 32, 33, 34, 35, 36, 37, 11, 38, 39, 40, 41]. Broadly speaking, they can be divided into two different groups:

- *Optimization* based [10, 32, 34, 37, 38, 40, 41]
- *Regression* based [8, 30, 31, 33, 36, 11, 39]

This thesis is mainly concerned about the study of optimization based algorithms for fitting AAM [10, 32, 34, 37, 38, 40, 41]. Consequently, regression

based approaches will not be reviewed in detailed and the interested reader is referred to the existent literature [8, 30, 31, 33, 36, 11, 39].

3.2.2.1 Compositional gradient descent

Optimization based methods for fitting AAM were first introduced in [8] and popularized by Matthews and Baker in [10]. These techniques are commonly known as *compositional gradient decent* (CGD) algorithms and are based on a direct analytical optimization of the problem defined by equation 3.10. These algorithms are built upon seminal work on CGD algorithms [68, 69] in the closely related field of *parametric image alignment*.

Assuming, for the time being, that the true texture parameters $\mathbf{c}^*$ are known, the problem defined by equation 3.10 reduces to a non-rigid image alignment problem [69, 70] between the particular instance of the object present in the image and its optimal texture reconstruction by the texture model:

$$\mathbf{p}^* = \arg \min_{\mathbf{p}} \frac{1}{2} \|\mathbf{i}[\mathbf{p}] - \mathbf{a}\|^2 \quad (3.11)$$

where $\mathbf{a} = \bar{\mathbf{a}} + \mathbf{A}\mathbf{c}^*$ is obtained by directly evaluating equation 3.6 given the true texture parameters $\mathbf{c}^*$.

In this simplified scenario, classic CGD algorithms iteratively solve the non-linear optimization problem with respect to the shape parameters $\mathbf{p}$ by:

1. Introducing an incremental warp $\mathcal{W}(\mathbf{x}; \Delta\mathbf{p})$ on the model side, in what is known as *inverse composition*, figure 3.5:

$$\epsilon = \frac{1}{2} \|\mathbf{i}[\mathbf{p}] - \mathbf{a}[\Delta\mathbf{p}]\|^2 \quad (3.12)$$

2. Performing a first order Taylor expansion of the residual around the identity incremental warp, $\mathcal{W}(\mathbf{x}; \Delta \mathbf{p}) = \mathcal{W}(\mathbf{x}; \mathbf{0}) = \mathbf{x}$:

$$\begin{aligned} \epsilon &= \frac{1}{2} \left\| \mathbf{i}[\mathbf{p}] - \underbrace{\mathbf{a}[\mathbf{0}]}_{\mathbf{a}} - \underbrace{\left. \frac{\partial \mathbf{r}[\Delta \mathbf{p}]}{\partial \Delta \mathbf{p}} \right|_{\Delta \mathbf{p}=\mathbf{0}}}_{\left. \frac{\partial \mathbf{a}[\Delta \mathbf{p}]}{\partial \Delta \mathbf{p}} \right|_{\Delta \mathbf{p}=\mathbf{0}}} \Delta \mathbf{p} \right\|^2 \\ &= \frac{1}{2} \left\| \mathbf{i}[\mathbf{p}] - \mathbf{a} - \mathbf{J}_a \Delta \mathbf{p} \right\|^2 \end{aligned} \quad (3.13)$$

where $\left. \frac{\partial \mathbf{r}[\Delta \mathbf{p}]}{\partial \Delta \mathbf{p}} \right|_{\Delta \mathbf{p}=\mathbf{0}} = \left. \frac{\partial \mathbf{a}[\Delta \mathbf{p}]}{\partial \Delta \mathbf{p}} \right|_{\Delta \mathbf{p}=\mathbf{0}} = \mathbf{J}_a$ denote the partial derivative of the residual with respect to the incremental parameters.

3. Inferring the optimal incremental parameters $\Delta \mathbf{p}^*$ by solving the following optimization problem:

$$\begin{aligned} \Delta \mathbf{p}^* &= \arg \min_{\Delta \mathbf{p}} \epsilon \\ &= \arg \min_{\Delta \mathbf{p}} \frac{1}{2} \left\| \mathbf{i}[\mathbf{p}] - \mathbf{a} - \mathbf{J}_a \Delta \mathbf{p} \right\|^2 \end{aligned} \quad (3.14)$$

for which the solution is given by:

$$\begin{aligned} \frac{\partial \epsilon}{\partial \Delta \mathbf{p}} &= \mathbf{0} \\ \frac{\partial \left(\frac{1}{2} \left\| \mathbf{i}[\mathbf{p}] - \mathbf{a} - \mathbf{J}_a \Delta \mathbf{p} \right\|^2 \right)}{\partial \Delta \mathbf{p}} &= \mathbf{0} \\ (-\mathbf{i}[\mathbf{p}] + \mathbf{a} + \mathbf{J}_a \Delta \mathbf{p}) \mathbf{J}_a^T &= \mathbf{0} \\ \Delta \mathbf{p}^* &= -(\mathbf{J}_a^T \mathbf{J}_a)^{-1} \mathbf{J}_a^T (\mathbf{i}[\mathbf{p}] - \mathbf{a}) \end{aligned} \quad (3.15)$$

4. Updating the current warp estimate by using the following update rule:

$$\mathcal{W}(\mathbf{x}; \mathbf{p}_k) \leftarrow \mathcal{W}(\mathbf{x}; \mathbf{p}_{k-1}) \circ \mathcal{W}(\mathbf{x}; \Delta \mathbf{p}^*)^{-1} \quad (3.16)$$

5. Going back to step 1 until a particular convergence criterion is met.

Typically $\|\mathbf{p}_k - \mathbf{p}_{k-1}\|^2 < \varepsilon$

Notice that, using the chain rule, the term $\mathbf{J}_\mathbf{a}$ in equations 3.13, 3.14 and 3.15 can be further expanded as:

$$\mathbf{J}_\mathbf{a} = \nabla \mathbf{a} \frac{\partial \mathcal{W}(\mathbf{x}; \Delta \mathbf{p})}{\partial \Delta \mathbf{p}} \quad (3.17)$$

where $\nabla \mathbf{a}$ is the image gradient of the linear texture model instance $\mathbf{a}$ and $\frac{\partial \mathcal{W}(\mathbf{x}; \Delta \mathbf{p})}{\partial \Delta \mathbf{p}}$ is the *Jacobian* of the incremental warp. A complete derivation of this last term as well as of *warp inversion*, $\mathcal{W}(\mathbf{x}; \Delta \mathbf{p})^{-1}$ in equation 3.16, are given in appendix A for the PWA and TPS deformation models typically used in AAM.

In general, the optimization of equation 3.10 with respect to the texture parameters $\mathbf{c}$ is dependent on the particular choice of CGD algorithm. The next sections introduce the *project-out inverse compositional* (PIC) and the *simultaneous inverse composition* (SIC) algorithms; two of the most well-known CGD algorithm for fitting AAM:

Project-out inverse compositional. Arguably, the most popular and fastest algorithm for solving equation 3.10 is the PIC algorithm [10] which was first proposed in [69] in the related field of parametric image alignment. This algorithm eliminates the need to solve for the texture parameters $\mathbf{c}$ by working on the orthogonal complement³ of the texture subspace, i.e. $\bar{\mathbf{A}} = \mathbf{I} - \mathbf{A}\mathbf{A}^T$. Consequently, the incremental warp $\Delta \mathbf{p}$ is estimated only

³A complete derivation of this procedure and the properties of this orthogonal texture subspace are given in section 5.2

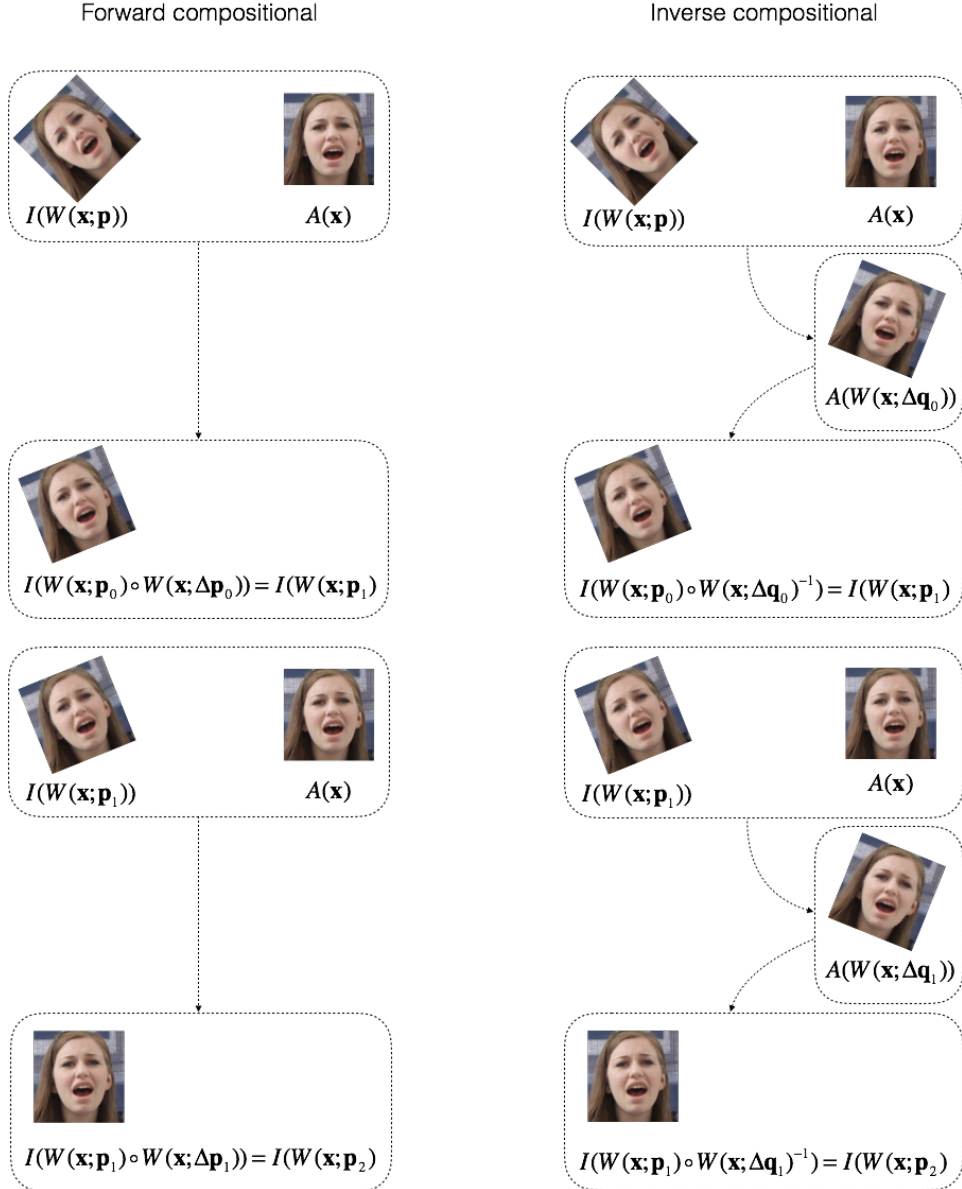


Figure 3.5: Schematic representation of the forward and inverse compositions. Forward composition solves the image alignment problem iteratively by introducing the incremental warp $\mathcal{W}(\mathbf{x}; \Delta \mathbf{p})$ on the image side, solving for the increments $\Delta \mathbf{p}$ and composing the resulting incremental warp with the current estimate of the warp $\mathcal{W}(\mathbf{x}; \mathbf{p})$. Inverse composition solves the same problem by introducing the incremental warp $\mathcal{W}(\mathbf{x}; \Delta \mathbf{q})$ on the model side, solving for the increments $\Delta \mathbf{q}$, inverting the incremental warp and composing it with the current estimate of the warp $\mathcal{W}(\mathbf{x}; \mathbf{p})$.

using the mean texture $\bar{\mathbf{a}}$:

$$\Delta \mathbf{p}^* = \arg \min_{\Delta \mathbf{p}} \frac{1}{2} \|\mathbf{i}[\mathbf{p}] - \bar{\mathbf{a}} - \mathbf{J}_{\bar{\mathbf{a}}} \Delta \mathbf{p}\|_{\bar{\mathbf{A}}}^2 \quad (3.18)$$

Defining $\hat{\mathbf{J}}_{\bar{\mathbf{a}}} = \bar{\mathbf{A}} \mathbf{J}_{\bar{\mathbf{a}}} = \bar{\mathbf{A}} \left(\nabla \bar{\mathbf{a}} \frac{\partial \mathcal{W}}{\partial \Delta \mathbf{p}} \right)$, the optimal solution for $\Delta \mathbf{p}$ is given by:

$$\Delta \mathbf{p}^* = - \left(\hat{\mathbf{J}}_{\bar{\mathbf{a}}}^T \hat{\mathbf{J}}_{\bar{\mathbf{a}}} \right)^{-1} \hat{\mathbf{J}}_{\bar{\mathbf{a}}}^T (\mathbf{i}[\mathbf{p}] - \bar{\mathbf{a}}) \quad (3.19)$$

The computation of the term $\left(\hat{\mathbf{J}}_{\bar{\mathbf{a}}}^T \hat{\mathbf{J}}_{\bar{\mathbf{a}}} \right)^{-1} \hat{\mathbf{J}}_{\bar{\mathbf{a}}}^T$ is performed offline and, hence, the computational complexity per iteration of the algorithm is reduced to $O(nF)$.

Simultaneous inverse compositional. The SIC algorithm [71], solves for shape and texture increments, $\Delta \mathbf{p}$ and $\Delta \mathbf{c}$, simultaneously by solving the following optimization problem:

$$\begin{aligned} \Delta \mathbf{p}^*, \Delta \mathbf{c}^* &= \arg \min_{\Delta \mathbf{p}, \Delta \mathbf{c}} \frac{1}{2} \|\mathbf{i}[\mathbf{p}] - \bar{\mathbf{a}} - \mathbf{A}(\mathbf{c} + \Delta \mathbf{c}) - \mathbf{J}_{\mathbf{A}} \Delta \mathbf{p}\|^2 \\ &= \arg \min_{\Delta \mathbf{p}, \Delta \mathbf{c}} \frac{1}{2} \|\mathbf{i}[\mathbf{p}] - \bar{\mathbf{a}} - \mathbf{A} \mathbf{c} - \mathbf{J}_{\mathbf{A}} \Delta \mathbf{p} - \mathbf{A} \Delta \mathbf{c}\|^2 \end{aligned} \quad (3.20)$$

Let $\Delta \mathbf{q} = (\Delta \mathbf{p}^T, \Delta \mathbf{c}^T)^T$ be the concatenation of the shape and texture parameters and let $\mathbf{J}_{\mathbf{A}} \approx (\mathbf{J}_{\bar{\mathbf{a}}}, \mathbf{J}_{\mathbf{a}_1}, \dots, \mathbf{J}_{\mathbf{a}_m})$ be the linearization of the bases $\mathbf{A}$ around $\Delta \mathbf{p} = \mathbf{0}$, where $\mathbf{J}_{\mathbf{a}_i}$ denotes the Jacobian of each texture component $\mathbf{a}_i$.

The solution to equation 3.20 is given by:

$$\Delta \mathbf{q} = - \left(\mathbf{J}_t^T \mathbf{J}_t \right)^{-1} \mathbf{J}_t^T (\mathbf{i}[\mathbf{p}] - \bar{\mathbf{a}} - \mathbf{A} \mathbf{c}) \quad (3.21)$$

where the total Jacobian $\mathbf{J}_t$ is defined as $\mathbf{J}_t = (\mathbf{J}_\mathbf{A}, \mathbf{A})$ and where $\mathbf{J}_\mathbf{A} = \mathbf{J}_{\bar{\mathbf{a}}} + \sum_{i=1}^m c_i \mathbf{J}_{\mathbf{a}_i}$ ⁴. Although the terms $\mathbf{J}_{\bar{\mathbf{a}}}$ and $\mathbf{J}_{\mathbf{a}_i}$ can be precomputed, the computation of $\mathbf{J}_t$ must be performed at each iteration due to its dependency on the current estimate of the texture parameters $\mathbf{c}$. Hence, the total cost per iteration of this algorithm is $O((n+m)F + (n+m)^2F + (n+m)^3)$.

⁴Notice that second order terms of the form $\Delta \mathbf{c}^T \mathbf{J}_\mathbf{A} \Delta \mathbf{p}$ have been neglected

Chapter 4

Active orientation models

This chapter introduces *active orientation models* (AOM); a new type of generative parametric model of shape and texture that generalizes well to unseen variations in identity, pose, expression and illumination, as well as occlusions. AOM are built upon the well-known paradigm of *active appearance models* (AAM). Like AAM, they are composed of shape, texture, and deformation models. Their shape and deformation models are the same ones used by AAM, but their texture model is defined using a robust subspace learning technique [43] that naturally leads to a robust formulation of the *compositional gradient descent* (CGD) algorithms described in section 3.2.2.1.

The content of this chapter is based on the following publications:

- J. Alabort-i-Medina, “Advances in generic facial model fitting,” tech. rep., Department of computing, Imperial College London, 2012.
- G. Tzimiropoulos, J. Alabort-i-Medina, S. Zafeiriou, and M. Pantic,

“Generic active appearance models revisited,” in *IEEE Asian conference on computer vision (ACCV)*, 2012.

- G. Tzimiropoulos, J. Alabort-i-Medina, S. Zafeiriou, and M. Pantic, “Active orientation models for face alignment in-the-wild,” *IEEE Transactions on information forensics and security (TIFS)*, 2014.

4.1 Motivation

Existing algorithms for fitting AAM [8, 10] have been criticized for their inability to correctly generalize to unseen variations in identity, pose, expression and illumination, as well as for their inability to correctly deal with occlusion. These unseen variations and occlusions create numerous undesirable local minima in the cost function which cause AAM fitting algorithms to diverge or converge to undesirable local solutions. Therefore, effective algorithms for fitting AAM must be able to mitigate the undesirable effects caused by local minima. Although several approaches have been proposed to solve this problem [72, 9, 73, 74, 75, 76, 77, 78, 21, 79] non of these techniques have been proven to work well on the problem of non-rigid face alignment under the presence of large variations in facial identity, pose and expression.

AOM address this problem by defining their texture model using a similarity measure that is robust to outliers. Outliers are defined as anything that the learned shape and texture models cannot reconstruct because:

1. It is not part of the training set, i.e. unseen texture variations in identity, expression and illumination.

2. Does not belong to the object's texture space, i.e. occlusions.
3. Is regarded as noise and excluded from the texture model.

4.2 Weighted non-linear least squares

Using the general framework for robust estimation proposed in [80], the *sum of squared differences* (SSD) between the vectorized warped image and the AAM's linear texture model defined in equations 3.9 and 3.10 can be expressed by the following *weighted non-linear least squares* (WNLS) problem:

$$\begin{aligned}
\mathbf{p}^*, \mathbf{c}^* &= \arg \min_{\mathbf{p}, \mathbf{c}} \frac{1}{2} \sum_{\mathbf{x} \in \Omega} \underbrace{\left\| I(\mathcal{W}(\mathbf{x}; \mathbf{p})) - \bar{A}(\mathbf{x}) - \sum_{i=1}^m c_i A_i(\mathbf{x}) \right\|_{Q(\mathbf{x})}^2}_{R(\mathbf{x})} \\
&= \arg \min_{\mathbf{p}, \mathbf{c}} \frac{1}{2} \sum_{\mathbf{x} \in \Omega} \left\| R(\mathbf{x}) \right\|_{Q(\mathbf{x})}^2 \\
&= \arg \min_{\mathbf{p}, \mathbf{c}} \frac{1}{2} \mathbf{r}^T \mathbf{Q} \mathbf{r}
\end{aligned} \tag{4.1}$$

where, as in equation 3.10, $\mathbf{r}$ is defined as:

$$\mathbf{r} = \mathbf{i}[\mathbf{p}] - (\bar{\mathbf{a}} + \mathbf{A}\mathbf{c}) \tag{4.2}$$

and where $\mathbf{Q} = \text{diag}(Q(\mathbf{x})) \in \mathbb{R}^{F \times F}$ is a diagonal weighting matrix that down-weights pixels corrupted by outliers. In the ideal case, $q_{i,i} = 0$ if pixel i is an outlier and $q_{i,i} = 1$ otherwise. The estimation of $\mathbf{Q}$ along with the optimal model parameters have been extensively studied in the robust statistics literature [80]. However, none of the existent out-of-the-box approaches has been proven successful for AAM fitting; mainly, because of the difficulty associated with defining an accurate noise model for outliers

in this case.

4.3 Image gradient orientations

In [43], Tzimiropoulos et al. proved that the combination of *image gradient orientations* (IGO) and the *cosine distance kernel* naturally leads to a robust measure of similarity. In this section, we introduce their approach and related back to the WNLS problem defined by equation 4.1

Let us start by first defining IGO. The complex gradient of an image I can be obtained by:

$$\begin{aligned} G(\mathbf{x}) &= \underbrace{f_x * I(\mathbf{x})}_{G_x(\mathbf{x})} + j \underbrace{f_y * I(\mathbf{x})}_{G_y(\mathbf{x})} \\ &= G_x(\mathbf{x}) + jG_y(\mathbf{x}) \end{aligned} \quad (4.3)$$

where $*$ denotes the convolution operator and f_x and f_y are linear filters used to approximate the ideal differentiation operator along the image horizontal and vertical axis respectively. Given the complex gradient G , the IGO of I are computed by:

$$\begin{aligned} O(\mathbf{x}) &= \angle G(\mathbf{x}) \\ &= \arctan \left(\frac{G_y(\mathbf{x})}{G_x(\mathbf{x})} \right) \end{aligned} \quad (4.4)$$

Using the this definition of IGO, we can measure the similarity between two images, I_1 and I_2 , using the following expression:

$$\begin{aligned} \varsigma &= \sum_{\mathbf{x} \in \Omega} \cos(\underbrace{O_1(\mathbf{x}) - O_2(\mathbf{x})}_{\Delta O(\mathbf{x})}) \\ &= \sum_{\mathbf{x} \in \Omega} \cos(\Delta O(\mathbf{x})) \end{aligned} \quad (4.5)$$

where O_1 and O_2 are the IGO of the previous two images and Ω and F denote the total image support region and the total number of pixels in this region respectively. Notice that, the value of ς is always between $[-F, F]$.

Let us denote by Ω_1 the image support region that is free from outliers and by Ω_2 the image support region corrupted by outliers where $\Omega = \Omega_1 \cup \Omega_2$. Then, under a number of mild assumptions, the authors of [43] empirically proved that, for natural images, the difference between IGO across the image support region corrupted by outliers can be well-described by a uniform distribution on the interval $[0, 2\pi)$, i.e.:

$$\Delta O(\mathbf{x}) \sim U[0, 2\pi) \quad \forall \mathbf{x} \in \Omega_2 \quad (4.6)$$

and given this result it follows that:

$$\sum_{\mathbf{x} \in \Omega_2} \cos(\Delta O(\mathbf{x})) \approx 0 \quad (4.7)$$

i.e. the local orientation mismatches caused by outliers are automatically *anceled out* by applying the cosine distance kernel¹.

Hence, we can rewrite equation 4.5 as:

$$\begin{aligned} \varsigma &= \sum_{\mathbf{x} \in \Omega_1} \cos(\Delta O(\mathbf{x})) + \sum_{\mathbf{x} \in \Omega_2} \cos(\Delta O(\mathbf{x})) \\ &\approx \sum_{\mathbf{x} \in \Omega_1} \cos(\Delta O(\mathbf{x})) \end{aligned} \quad (4.8)$$

Therefore, using equation 4.5, the similarity value between the images, I_1

¹It is important to emphasize that this property is a direct consequence of equation 4.6 rather than the specific use of the cosine kernel. For example, one could obtain the same *cancel out* property by adding $\frac{\pi}{2}$ to $\Delta O(\mathbf{x})$ and applying the *sine distance kernel*. However the specific use of the cosine distance kernel allows us to implicitly represent this robust measure of similarity in general framework for robust estimation proposed in [80], 4.10

and I_2 , is completely dominated by the contribution of the image support region that is free from outliers, Ω_1 .

As mention at the beginning of this section, the previous similarity measure can be elegantly expressed within the general framework for robust estimation proposed in [80]. To do so, let us turn the similarity measure ς into an error measure by:

$$\begin{aligned}\varepsilon &= F - \varsigma \\ &= F - \sum_{\mathbf{x} \in \Omega} \cos(\Delta O(\mathbf{x})) \\ &= \sum_{\mathbf{x} \in \Omega} (1 - \cos(\Delta O(\mathbf{x})))\end{aligned}\tag{4.9}$$

Then, making use of a few trigonometric identities one can express ε as:

$$\begin{aligned}\varepsilon &= \sum_{\mathbf{x} \in \Omega} (1 - \cos(\Delta O(\mathbf{x}))) \\ &= \sum_{\mathbf{x} \in \Omega} (1 - \cos(O_1(\mathbf{x}) - O_2(\mathbf{x}))) \\ &= \frac{1}{2} \sum_{\mathbf{x} \in \Omega} \underbrace{\sin^2(O_1(\mathbf{x})) + \cos^2(O_1(\mathbf{x}))}_1 + \underbrace{\sin^2(O_2(\mathbf{x})) + \cos^2(O_2(\mathbf{x}))}_1 \\ &\quad - 2 \underbrace{(\cos(O_1(\mathbf{x})) \cos(O_2(\mathbf{x})) + \sin(O_1(\mathbf{x})) \sin(O_2(\mathbf{x})))}_{\cos(O_1(\mathbf{x}) - O_2(\mathbf{x}))} \\ &= \frac{1}{2} \sum_{\mathbf{x} \in \Omega} \frac{(\cos(O_1(\mathbf{x})) - \cos(O_2(\mathbf{x})))^2}{\cos^2(O_1(\mathbf{x})) + \cos^2(O_2(\mathbf{x})) - \cos(O_1(\mathbf{x})) \cos(O_2(\mathbf{x}))} \\ &\quad + \frac{(\sin(O_1(\mathbf{x})) - \sin(O_2(\mathbf{x})))^2}{\sin^2(O_1(\mathbf{x})) + \sin^2(O_2(\mathbf{x})) - \sin(O_1(\mathbf{x})) \sin(O_2(\mathbf{x}))} \\ &= \frac{1}{2} \sum_{\mathbf{x} \in \Omega} \|(\cos(O_1(\mathbf{x})), \sin(O_1(\mathbf{x}))) - (\cos(O_2(\mathbf{x})), \sin(O_2(\mathbf{x})))\|^2 \\ &= \frac{1}{2} \mathbf{r}^T \mathbf{Q}_{ideal} \mathbf{r}\end{aligned}\tag{4.10}$$

where $\mathbf{Q}_{ideal}$ is an approximation to the *ideal* weighting matrix. Notice that,

in this case, the matrix $\mathbf{Q}_{ideal}$ is never explicitly computed and that we can write the expression in equation 4.10 because of the *cancel out* property defined by equation 4.7. Note that $\mathbf{r}$ is defined here as:

$$\mathbf{r} = \mathbf{z}_1 - \mathbf{z}_2 \quad (4.11)$$

where $\mathbf{z}_i$ denotes the vectorized version of the *normalized image gradient*:

$$\mathbf{z}_i = (\cos(\mathbf{o}_i)^T, \sin(\mathbf{o}_i)^T)^T \in \mathbb{R}^{2F \times 1} \quad (4.12)$$

and $\mathbf{o}_i$ is the vectorized version of the IGO of the image I_i , i.e. $\mathbf{o}_i = \text{vec}(O_i(\mathbf{x})) \in \mathbb{R}^{F \times 1}$.

4.4 IGO-based texture model

The robust texture model used by AOM is learned using a similar procedure to the one described in section 3.1.2. The main differences are:

1. The texture of the shape-free images is defined in terms of the *normalized image gradient* representation introduced in equation 4.12.
2. The application of standard linear PCA to the shape-free images results in a novel texture model that preserves the robust properties of IGO [43] described in section 4.3.

Following the argument presented in section 4.3, the robust texture model used by AOM can be learned by first computing the image gradient orien-

tation of the shape-free images, denoted here by $T_i(\mathbf{x})$:

$$\begin{aligned} O_i(\mathbf{x}) &= \angle (f_x * T_i(\mathbf{x}) + j f_y * T_i(\mathbf{x})) \\ &= \angle (G_{i,x}(\mathbf{x}) + j G_{i,y}(\mathbf{x})) \\ &= \arctan \frac{G_{i,y}(\mathbf{x})}{G_{i,x}(\mathbf{x})} \end{aligned} \quad (4.13)$$

and defining:

$$\mathbf{z}_i = (\cos(\mathbf{o}_i)^T, \sin(\mathbf{o}_i)^T)^T \in \mathbb{R}^{2F \times 1} \quad (4.14)$$

as the vectorized version of their normalized gradients. Then, given (4.10):

$$\frac{1}{2} \|\mathbf{z}_i - \mathbf{z}_j\|_{\mathbf{Q}_{ideal}} = \sum_{\mathbf{x} \in \Omega} (1 - \cos(\Delta O(\mathbf{x}))) \quad (4.15)$$

a robust texture model can be learned by directly applying standard linear PCA to the normalized gradient representation $\mathbf{z}_i$ of the shape-free images $T_i(\mathbf{x})$. Notice that, because of 4.15, the previous procedure implicitly defines a robust Kernel PCA that uses IGO together with a dissimilarity measure based on the *cosine distance kernel*.

The robust texture model used by AOM is defined as:

$$\mathbf{z} = \bar{\mathbf{z}} + \mathbf{Z}\mathbf{c} \quad (4.16)$$

where $\bar{\mathbf{z}} \in \mathbb{R}^{2F \times 1}$ is the robust mean texture and $\mathbf{Z} \in \mathbb{R}^{2F \times m}$ and $\mathbf{c} \in \mathbb{R}^{m \times 1}$ denote the robust texture bases and texture parameters, respectively. The output $\mathbf{z}$ of the texture model is defined in terms of the normalized image gradient representation in equation 4.14 and its original IGO representation

can be obtained as follows:

$$\phi = \arctan \left(\frac{\mathcal{Y}(\mathbf{z})}{\mathcal{X}(\mathbf{z})} \right) \quad (4.17)$$

where $\mathcal{X}$ and $\mathcal{Y}$ are operators returning the first and last F elements of the vector $\mathbf{z}$ (the horizontal and vertical components of the normalized gradient), respectively.

4.5 Fitting AOM

The SSD between the normalized gradient representation of two images is equivalent to the cosine dissimilarity measure of their IGO representation 4.15. Consequently, the SSD between the warped normalized gradient representation of an image and the robust IGO-based texture in equation 4.4 defines a robust error measure that can be used to define algorithms for AOM fitting. The use of this error measure makes it possible to adapt the *compositional gradient descent* (CGD) framework for fitting AAM introduced in section 3.2.2.1 to the fitting of AOM.

4.5.1 Project-out inverse compositional

The first strategy for fitting AOM is analogous to the one described in section 3.2.2.1. In this case, the orthogonal complement of the robust texture subspace is denoted as:

$$\bar{\mathbf{Z}} = \mathbf{I} - \mathbf{Z}\mathbf{Z}^T \quad (4.18)$$

where $\mathbf{I}$ is the identity matrix and $\mathbf{Z}$ are the robust texture bases in equation 4.16. In this case, denoting $\mathbf{z}$ as the vectorized representation of the warped

normalized gradients of the input image, the solution for the incremental warp $\Delta \mathbf{p}$ can be obtained by solving the following optimization problem:

$$\Delta \mathbf{p}^* = \arg \min_{\Delta \mathbf{p}} \frac{1}{2} \|\mathbf{z}[\mathbf{p}] - \bar{\mathbf{z}} - \mathbf{J}_{\bar{\mathbf{z}}} \Delta \mathbf{p}\|_{\bar{\mathbf{z}}}^2 \quad (4.19)$$

for which the solution is given by:

$$\Delta \mathbf{p}^* = -(\mathbf{J}_{\bar{\mathbf{z}}}^T \bar{\mathbf{Z}} \mathbf{J}_{\bar{\mathbf{z}}})^{-1} \mathbf{J}_{\bar{\mathbf{z}}}^T \bar{\mathbf{Z}} (\mathbf{z}[\mathbf{p}] - \bar{\mathbf{z}}) \quad (4.20)$$

Notice that the term $(\mathbf{J}^T \mathbf{P} \mathbf{J})^{-1} \mathbf{J}^T \bar{\mathbf{Z}}$ can be precomputed (it only depends on the robust mean texture $\bar{\mathbf{z}}$) and, consequently, this algorithm has the same *asymptotic* computational complexity as the original *project-out inverse compositional* (PIC) for fitting AAM, i.e $O(n2F)$ ².

4.5.2 Alternate inverse compositional algorithm

Using a similar strategy to the one used in section 4.5.1 one can define the AOM version of the *simultaneous inverse compositional* (SIC) algorithm for fitting AAM described in section 3.2.2.1. However, the large complexity per iteration of the resulting algorithm dramatically limits its applicability. Instead, this section derives a more efficient algorithm by solving a constrained version of the SIC optimization problem:

$$\begin{aligned} & \underset{\Delta \mathbf{p}, \mathbf{c}}{\text{minimize}} \quad \|\mathbf{z}[\mathbf{p}] - (\bar{\mathbf{z}} + \mathbf{Z}\mathbf{c})\|_{\Delta \mathbf{p}}^2 \\ & \text{subject to} \quad \mathbf{c} = \mathbf{Z}^T (\mathbf{z}[\mathbf{p}] - \bar{\mathbf{z}}) \end{aligned} \quad (4.21)$$

²The complexity of the original PIC for fitting AAM is $O(nF)$ where n is the number of shape parameters and F denotes the number of pixels in the error term. Notice that the AOM's error term has double the amount of pixels than the AAM one because it is defined in terms of normalized image gradients.

The expression in equation 4.21 can be elegantly expressed using the method of *Langrange multipliers* [48]. Defining the Lagrangian as:

$$\mathcal{L}(\Delta \mathbf{p}, \lambda) = \|\mathbf{z}[\mathbf{p}] - (\bar{\mathbf{z}} + \mathbf{Z}\mathbf{c})[\Delta \mathbf{p}]\|^2 + \lambda (\mathbf{Z}^T (\mathbf{z}[\mathbf{p}] - \bar{\mathbf{z}}) - \mathbf{c}) \quad (4.22)$$

by definition, solving for $\nabla \mathcal{L} = 0$ provides the solution of the initial constrained problem 4.21. Setting $\nabla \mathcal{L} = 0$ yields the following system of equations:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \lambda} &= \mathbf{Z}^T (\mathbf{z}[\mathbf{p}] - \bar{\mathbf{z}}) - \mathbf{c} = 0 \\ \frac{\partial \mathcal{L}}{\partial \Delta \mathbf{p}} &= \frac{\partial \|\mathbf{z}[\mathbf{p}] - (\bar{\mathbf{z}} + \mathbf{Z}\mathbf{c})[\Delta \mathbf{p}]\|^2}{\partial \Delta \mathbf{p}} = 0 \end{aligned} \quad (4.23)$$

The first of these equation is the initial constraint. Solving for $\mathbf{c}$:

$$\mathbf{c}^* = \mathbf{Z}^T (\mathbf{z}[\mathbf{p}] - \bar{\mathbf{z}}) \quad (4.24)$$

Notice that, $\bar{\mathbf{z}} + \mathbf{Z}\mathbf{c} = \bar{\mathbf{z}} + \mathbf{Z} \overbrace{\mathbf{Z}^T (\mathbf{z}[\mathbf{p}] - \bar{\mathbf{z}})}^{\mathbf{c}^*}$ is the estimated texture obtained by reconstructing the projection of $\mathbf{z}[\mathbf{p}]$ onto the robust texture subspace. Rewriting $\bar{\mathbf{z}} + \mathbf{Z}\mathbf{c}^*$ as $\tilde{\mathbf{z}}$ and substituting in the second equation:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \Delta \mathbf{p}} &= \frac{\partial \|\mathbf{z}[\mathbf{p}] - \tilde{\mathbf{z}}[\Delta \mathbf{p}]\|^2}{\partial \Delta \mathbf{p}} \\ &\approx \frac{\partial \|\mathbf{z}[\mathbf{p}] - \tilde{\mathbf{z}} - \mathbf{J}_{\tilde{\mathbf{z}}} \Delta \mathbf{p}\|^2}{\partial \Delta \mathbf{p}} = 0 \end{aligned} \quad (4.25)$$

where a first order Taylor expansion of the residual around the incremental warp $\Delta \mathbf{p}$ has been applied in the last step.

These set of equations leads to an iterative optimization algorithm that alternates between the optimization of the texture 4.24 and shape 4.25 parameters. At each iteration, the algorithm first estimates the texture parameters $\mathbf{c}$ using equation 4.24. Then, the increment on the shape parameters $\Delta \mathbf{p}$ is

obtained by minimizing equation 4.25. The solution is given by:

$$\Delta \mathbf{p}^* = -(\mathbf{J}_{\tilde{\mathbf{z}}}^T \mathbf{J}_{\tilde{\mathbf{z}}})^{-1} \mathbf{J}_{\tilde{\mathbf{z}}}^T (\mathbf{z}[\mathbf{p}] - \tilde{\mathbf{z}}) \quad (4.26)$$

Note that estimating the texture model reconstruction $\tilde{\mathbf{z}}_A$ comes at the price of having to recompute the Jacobian $\mathbf{J}_{\tilde{\mathbf{z}}}$ at each iteration. The complexity per iteration of this algorithm is, therefore, $O(2m2F + n2F + n^22F + n^3)$ which is significantly smaller than the complexity of an hypothetical AOM SIC algorithm $O((n+m)2F + (n+m)^22F + (n+m)^3)$ ³. This alternating algorithm is less efficient than the PIC one but it is still, in practice, very fast, probably allowing a close to real-time implementation, and leads to a dramatic increase in fitting accuracy.

4.6 Results

This section assesses the performance of the proposed AOM on the problem of non-rigid face alignment under controlled variations of identity, expression and pose. Their performance is compared to that of AAM and two state-of-the-art methods⁴: the *constrained local models* (CLM) proposed by Saragih et al. in [5] and the *parts-based tree model* of Zhu and Ramanan presented in [4]. Results for AOM and AAM were obtained using a preliminary Matlab implementation of these methods⁵; results for the other methods were obtained using the code provided by the original authors when possible.

³Note that the number of texture parameters m used for fitting AOM (as in AAM) is typically one order of magnitude bigger than the number of shape parameters n .

⁴[5] and [4] were state of the art methods at the time when AOM were originally proposed, since then, the state-of-art on the problem of non-rigid face alignment has advanced considerably.

⁵Note that, a more efficient implementation of these algorithms is now publicly available as part of the the Menpo Project (<http://www.menpo.org/>) [54].

For AOM, the fitting accuracy of the two CGD algorithms presented in sections 4.5.1 and 4.5.2 is reported (i.e. the *project-out inverse compositional* algorithm (PIC) for AOM denoted as AOM-PO and the *alternated inverse compositional* (AIC) algorithm for AOM denoted by AOM-A). Both algorithms are implemented using a standard *Gaussian pyramid* with 2 levels and are initialize by an in-house face detector. In some sense, the accuracy reported in this section could be treated as a baseline because AOM can directly benefit from more than 15 years of research on AAM. Sophisticated enhancements such as 3D constraints [81], densification [82] and other complex shape priors [83] can be directly incorporated onto the AOM framework but are not considered here because they fall outside of the scope of this comparison.

The performance of all methods is measured on the *XM2VTS* [84] and *Multi-PIE* [85] datasets; 68 points ground truth landmark annotations for these datasets were obtained from the XM2VTS website⁶ and kindly provided by the authors of [5] respectively.

4.6.1 XM2VTS

The XM2VTS dataset contains 2,360 frontal images of 295 different subjects displaying neutral expression. Experiments on this dataset are interesting mainly because of the large variation in facial shape and texture due to facial hair, glasses and ethnicity. To compare directly with [5] (the result for [5] has been taken directly from the paper) the original experiment in that paper is reproduced. Therefore, the dataset is divided into into four different sets with no identity overlap and a four-fold cross validation experiments

⁶http://personalpages.manchester.ac.uk/staff/timothy.f.cootes/data/xm2vts/xm2vts_markup.html

is performed, using three parts for training and one for testing in every trial. Results for this experiment are shown in figure 4.1a and table 4.1e. Unfortunately, because the trees provided in [4] were hard coded for Multi-PIE, a fair comparison with [4] is not feasible in this case. For the remaining methods, the graph in figure 4.1a shows the *cumulative error curves* (CED) obtained by computing the percentage of images for which the fitting error was less than a specific value. This experiment is reported using the same shape *root mean squared error* (RMSE) used in [5] which is defined as:

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{i=1}^N \sum_{j=1}^{2v} (\mathbf{s}_{i,j}^p - \mathbf{s}_{i,j}^g)^2} \quad (4.27)$$

where $\mathbf{s}_{i,j}^p$ and $\mathbf{s}_{i,j}^g$ denote the j -th coordinate of the i -th *predicted* and *ground truth* shapes, respectively; and N is the total number of test images. Table 4.1e states the exact proportion of images that were fitted with a RMSE smaller than 4 and 5 pixels. For these two cases of interest, both AOM algorithms outperform the CLM algorithm presented in [5] by around 40% and 30% in absolute fitting accuracy terms.

4.6.2 Multi-PIE

The Multi-PIE dataset contains around 750,000 images of 337 subjects under 15 view points, 19 illumination conditions and displaying 6 different facial expressions. Markup annotations for a small subset of the dataset (around 1500 images) were provided by the authors of [5]. Both AOM and AAM models were trained using 432 images from 54 different subjects. For each subject, 8 images in total were used as follows: 1 image for frontal (0 degrees) neutral expression, 2 images for 2 different viewpoints (-15 and 15

degrees) displaying neutral expression; and 5 frontal images (0 degrees) displaying the remaining 5 expressions. For testing, the remaining annotated images (more than 1000 images in total, all containing faces of subjects not present in the training set) were used. Note that, fitting accuracy is reported independently for three different cases: frontal-neutral, pose (-15 and 15 degrees) and expression as described above. Results for this experiment are reported by figures 4.1b, 4.1c and 4.1d and table 4.1f. For comparison purposes, CLM results were obtained with the original implementation kindly provided by the authors of [5]. This model was trained only for frontal images and the result for the pose experiment reported here are not representative of the full capabilities of the method. For [4] the publicly available code provided by the authors was used to train their so-called *independent* model with same training data used to trained AOM and AAM. Note that this independent model roughly corresponds to training different models for each of the variations being present in the training data set and requires large fitting times. On the contrary, AOM and AAM were trained using a single combined model using the entire training dataset. As before, for all methods considered, each graph shows the cumulative curve obtained by computing the percentage of images for which the fitting error was less than a specific value. For this experiment, results are reported using the same point-to-point error measure normalized by the *face size*⁷ proposed in [4] which is defined as:

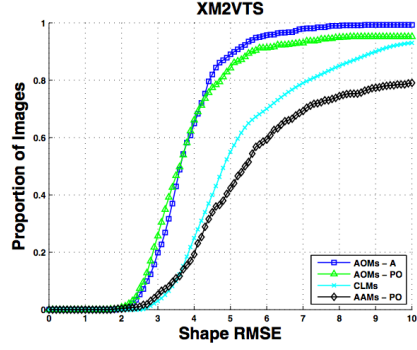
$$\text{Error} = \frac{1}{N} \sum_{i=1}^N \frac{\sum_{j=1}^v \sqrt{(x_{i,j}^p - x_{i,j}^g)^2 + (y_{i,j}^p - y_{i,j}^g)^2}}{f_i} \quad (4.28)$$

⁷The face size is computed as the mean of the height and width of the bounding box containing the face.

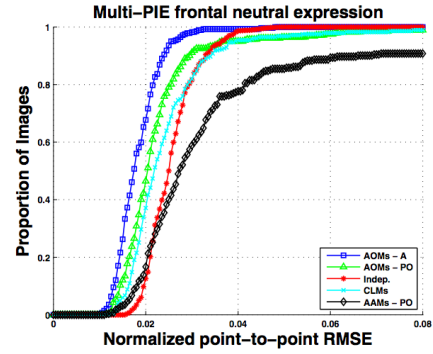
where $x_{i,j}^p, y_{i,j}^p$ and $x_{i,j}^g, y_{i,j}^g$ are the x and y coordinates of the j -th point of the i -th *predicted* and *ground truth* shapes, respectively; N is the total number of test images; and f_i denotes the face size of the i -th shape. Results for this experiment show that the AOM-A performs the best, surpassing all other methods by a considerable margin. AOM-PO outperforms [4] in the *frontal-neutral* experiment and is able to fit 60% of the expression images more accurately than [4], while both methods perform similarly in the pose experiment. Finally, CLM performs consistently worse than the other methods.

4.7 Conclusions

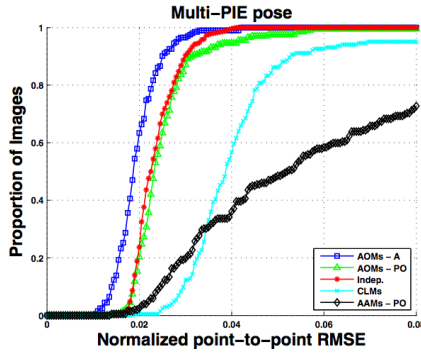
This chapter presents AOM, a new generative model for non-rigid object alignment that generalizes better than classic AAM to unseen object instances and variations. AOM employ a statistically robust texture model based on the principal components of image gradient orientations that naturally leads to a robust formulation of standard *compositional gradient descent* (CGD) algorithms for AAM fitting. AOM are shown to largely outperform AAM and compare favorably to other state-of-the-art techniques on the problem of non-rigid face alignment.



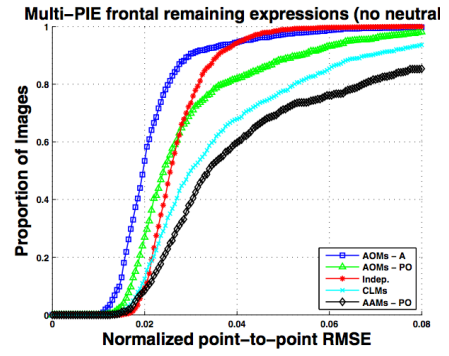
(a) Fitting accuracy for the XM2VTS experiment.



(b) Fitting accuracy for the frontal-neutral expression experiment on Multi-PIE.



(c) Fitting accuracy on the pose experiment on Multi-PIE.



(d) Fitting accuracy on the expression experiment on Multi-PIE.

Algorithm	< 3 pixels	< 4 pixels	< 5 pixels
AOM-A	0.19	0.64	0.89
AOM-PO	0.25	0.66	0.84
CLM [5]	0.03	0.25	0.55
AAM-PO [10]	0.05	0.19	0.42

(e) Table reporting the percentage of image for which the final RMSE obtained by each method was smaller than 3, 4 and 5 pixels on the XM2VTS dataset.

Algorithm	Frontal-neutral		Pose		Expression	
	< 0.02	< 0.03	< 0.02	< 0.03	< 0.02	< 0.03
AOM-A	0.67	0.98	0.53	0.90	0.63	0.96
AOM-PO	0.46	0.91	0.26	0.69	0.20	0.87
Indep. [4]	0.12	0.81	0.09	0.73	0.23	0.90
CLM [5]	0.37	0.81	0.12	0.50	0.00	0.12
AAM-PO [10]	0.16	0.58	0.08	0.38	0.02	0.19

(f) Table reporting the percentage of image for which the final normalized point-to-point RMSE obtained by each method was smaller than 0.02 and 0.03 for each of the three experiments on the Multi-PIE dataset.

Figure 4.1: Results showing the fitting accuracy of AOM, Indep. [4], CLM [5] and AAM on the XM2VTS and Multi-PIE datasets.

Chapter 5

Unifying compositional AAM fitting

This chapter presents a detailed study of *compositional gradient descent* (CGD) algorithms for fitting *active appearance models* (AAM). It proposes a unified and complete framework for classifying these type of algorithms with respect to three main characteristics: *i) cost function; ii) type of composition; and iii) optimization method*. Furthermore, it extends the this framework by: *a) introducing a probabilistic formulation of AAM; b) proposing a novel Bayesian cost function; c) introducing two novel types of composition, asymmetric and bidirectional; and d) providing new valuable insights into existent CGD algorithms by relating them to well-studied optimization techniques*.

The content of this chapter is based on the following two publications:

- J. Alabort-i-Medina and S. Zafeiriou, “Bayesian active appearance models,” in *IEEE Conference on computer vision and pattern recogni-*

tion (CVPR), 2014.

- J. Alabort-i-Medina and S. Zafeiriou, “A unified framework for compositional fitting of active appearance models,” *International journal of computer vision (IJCV)*, 2016.

5.1 Motivation and outline

The first optimization based method for fitting *active appearance models* (AAM) was proposed by Matthews and Baker in their seminal work [10]. Throughout the years, researchers have built upon the original *project-out inverse compositional* (PIC) algorithm of [10] and extended this family of algorithms with several new additions [10, 32, 34, 37, 38, 40, 41, 53, 56]. These methods are known as *compositional gradient decent* (CGD) algorithms and are based on direct analytical optimization of AAM cost functions.

The differences between CGD algorithms are subtle and have not always been correctly stated in the existing literature. The aim of this chapter is to correct this situation by providing a detailed study of CGD algorithms and by proposing a unified and complete framework in which these algorithms can be correctly classified with respect to three of their main characteristics:

1. *Cost function* (section 5.1);
2. *Type of composition* (section 5.3); and
3. *Optimization method* (section 5.4).

Furthermore, several contribution, with respect to each one of these three characteristics, are made across this chapter:

- Section 5.2 reviews the probabilistic formulation of AAM and proposes a novel *Bayesian formulation* of the fitting problem. This novel formulation is obtained by assuming a probabilistic model for texture generation with both Gaussian noise and a Gaussian prior over a latent texture space. A new cost function, that only depends on the shape parameters and that can be interpreted as a valid and more general probabilistic formulation of the well-known project-out cost function [10], can be derived by marginalizing out the latent texture space. This Bayesian formulation is motivated by seminal works on *probabilistic component analysis* (PPCA) [86, 87, 88] and object tracking [71].
- Section 5.3 proposes the use of two novel types of composition in AAM: *i) asymmetric*; and *ii) bidirectional*. These types of composition have been widely used in the related field of parametric image alignment [44, 45, 46, 47] and make use of the gradients of both the image and texture model to derive better convergent and more robust CGD algorithms.
- Section 5.4 provides valuable insights into existent strategies used to derive fast and exact simultaneous CGD algorithms [34, 37, 40] by reinterpreting them as direct applications of the *Schur complement* [48] and the *Wiberg method* [49, 50].

5.2 Cost Function

AAM fitting is typically formulated as a (regularized) search over the shape and texture parameters that minimize a global error measure between the

vectorized warped image and the texture model:

$$\mathbf{p}^*, \mathbf{c}^* = \arg \min_{\mathbf{p}, \mathbf{c}} \mathcal{R}(\mathbf{p}, \mathbf{c}) + \mathcal{D}(\mathbf{i}[\mathbf{p}], \mathbf{c}) \quad (5.1)$$

where $\mathcal{D}$ is a data term that quantifies the global error measure between the vectorized warped image and the texture model and $\mathcal{R}$ is an *optional* regularization term that penalizes complex shape and texture deformations.

5.2.1 Sum of squared differences

As explained in section 3.2.1, one of the most natural choices for the data term is the *Sum of Squared Differences* (SSD) between the vectorized warped image and the linear texture model. Consequently, the *classic* AAM fitting problem (which was first introduced in section 3.2.1, equations 3.9 and 3.10) is defined by the following non-linear optimization problem¹:

$$\begin{aligned} \mathbf{p}^*, \mathbf{c}^* &= \arg \min_{\mathbf{p}, \mathbf{c}} \frac{1}{2} \mathbf{r}^T \mathbf{r} \\ &= \arg \min_{\mathbf{p}, \mathbf{c}} \underbrace{\frac{1}{2} \|\mathbf{i}[\mathbf{p}] - (\bar{\mathbf{a}} + \mathbf{A}\mathbf{c})\|^2}_{\mathcal{D}(\mathbf{i}[\mathbf{p}], \mathbf{c})} \end{aligned} \quad (5.2)$$

On the other hand, considering regularization, the most natural choice for $\mathcal{R}$ is the sum of ℓ_2^2 -norms over the shape and texture parameters. In this case, the *regularized* AAM fitting problem is defined as follows:

$$\begin{aligned} \mathbf{p}^*, \mathbf{c}^* &= \arg \min_{\mathbf{p}, \mathbf{c}} \frac{1}{2} \|\mathbf{p}\|^2 + \frac{1}{2} \|\mathbf{c}\|^2 + \frac{1}{2} \mathbf{r}^T \mathbf{r} \\ &= \arg \min_{\mathbf{p}, \mathbf{c}} \underbrace{\frac{1}{2} \|\mathbf{p}\|^2 + \frac{1}{2} \|\mathbf{c}\|^2}_{\mathcal{R}(\mathbf{p}, \mathbf{c})} + \underbrace{\frac{1}{2} \|\mathbf{i}[\mathbf{p}] - (\bar{\mathbf{a}} + \mathbf{A}\mathbf{c})\|^2}_{\mathcal{D}(\mathbf{i}[\mathbf{p}], \mathbf{c})} \end{aligned} \quad (5.3)$$

¹Recall that, the residual $\mathbf{r}$ in Equation 5.2 is linear with respect to the texture parameters $\mathbf{c}$ and non-linear with respect to the shape parameters $\mathbf{p}$ through the warp $\mathcal{W}(\mathbf{x}; \mathbf{p})$

5.2.1.1 Probabilistic formulation

A probabilistic formulation of AAM can be obtained by rewriting the mathematical expressions, equations 3.7 and 3.8, defining the two main assumptions behind AAM assuming probabilistic models for shape and texture generation. Motivated by seminal works on *probabilistic component analysis* (PPCA) [86, 87, 88] and object tracking [71], one can assume probabilistic models for shape and texture generation with both Gaussian noise and Gaussian priors over the latent shape and texture spaces²:

$$\mathbf{s} = \bar{\mathbf{s}} + \mathbf{S}\mathbf{p} + \boldsymbol{\varepsilon} \quad \mathbf{p} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Lambda}) \quad \boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \zeta^2 \mathbf{I}) \quad (5.4)$$

$$\mathbf{i}[\mathbf{p}] = \bar{\mathbf{a}} + \mathbf{A}\mathbf{c} + \boldsymbol{\epsilon} \quad \mathbf{c} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}) \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}) \quad (5.5)$$

where the diagonal matrices $\boldsymbol{\Lambda} = \text{diag}(\lambda_{\mathbf{s}_1}, \dots, \lambda_{\mathbf{s}_m})$ and $\boldsymbol{\Sigma} = \text{diag}(\lambda_{\mathbf{a}_1}, \dots, \lambda_{\mathbf{a}_m})$ contain the eigenvalues associated with the shape and texture eigenvectors respectively and where ζ^2 and σ^2 denote the estimated shape and image noise³ respectively.

A probabilistic formulation of the SSD cost functions defined by equations 5.2 and 5.3 can be naturally derived using probabilistic generative models of shape and texture. Denoting the AAM parameters as $\Theta = \{\bar{\mathbf{s}}, \mathbf{S}, \boldsymbol{\Lambda}, \bar{\mathbf{a}}, \mathbf{A}, \boldsymbol{\Sigma}, \sigma^2\}$

²This formulation is generic and one could assume other probabilistic generative models [42, 89, 90, 91] to define novel probabilistic versions of AAM.

³Theoretically, the optimal value for ζ^2 and σ^2 is the average value of the eigenvalues associated to the discarded shape and texture eigenvectors respectively i.e. $\zeta^2 = \frac{1}{N-n} \sum_{i=n}^N \lambda_{\mathbf{s}_i}$ and $\sigma^2 = \frac{1}{M-m} \sum_{i=m}^M \lambda_{\mathbf{a}_i}$ [88].

a *maximum likelihood* (ML) formulation can be derived as follows:

$$\begin{aligned}
\mathbf{p}^*, \mathbf{c}^* &= \arg \max_{\mathbf{p}, \mathbf{c}} p(\mathbf{i}[\mathbf{p}] | \mathbf{p}, \mathbf{c}, \Theta) \\
&= \arg \max_{\mathbf{p}, \mathbf{c}} \ln p(\mathbf{i}[\mathbf{p}] | \mathbf{p}, \mathbf{c}, \Theta) \\
&= \arg \min_{\mathbf{p}, \mathbf{c}} \underbrace{\frac{1}{2\sigma^2} \|\mathbf{i}[\mathbf{p}] - (\bar{\mathbf{a}} + \mathbf{A}\mathbf{c})\|^2}_{\mathcal{D}(\mathbf{i}[\mathbf{p}], \mathbf{c})}
\end{aligned} \tag{5.6}$$

and a *maximum a posteriori* (MAP) formulation can be similarly derived by taking into account the prior distributions over the shape and texture parameters:

$$\begin{aligned}
\mathbf{p}^*, \mathbf{c}^* &= \arg \max_{\mathbf{p}, \mathbf{c}} p(\mathbf{p}, \mathbf{c}, \mathbf{i}[\mathbf{p}] | \Theta) \\
&= \arg \max_{\mathbf{p}, \mathbf{c}} p(\mathbf{p} | \mathbf{\Lambda}) p(\mathbf{c} | \mathbf{\Sigma}) p(\mathbf{i}[\mathbf{p}] | \mathbf{p}, \mathbf{c}, \Theta) \\
&= \arg \max_{\mathbf{p}, \mathbf{c}} \ln p(\mathbf{p} | \mathbf{\Lambda}) + \ln p(\mathbf{c} | \mathbf{\Sigma}) + \ln p(\mathbf{i}[\mathbf{p}] | \mathbf{p}, \mathbf{c}, \Theta) \\
&= \arg \min_{\mathbf{p}, \mathbf{c}} \underbrace{\frac{1}{2} \|\mathbf{p}\|_{\mathbf{\Lambda}^{-1}}^2 + \frac{1}{2} \|\mathbf{c}\|_{\mathbf{\Sigma}^{-1}}^2}_{\mathcal{R}(\mathbf{p}, \mathbf{c})} + \underbrace{\frac{1}{2\sigma^2} \|\mathbf{i}[\mathbf{p}] - (\bar{\mathbf{a}} + \mathbf{A}\mathbf{c})\|^2}_{\mathcal{D}(\mathbf{i}[\mathbf{p}], \mathbf{c})}
\end{aligned} \tag{5.7}$$

where we have assumed the shape and texture parameters to be independent⁴.

These ML and MAP formulations are weighted version of the optimization problem defined by equation 5.2 and 5.3. In both cases, the maximization of the conditional probability of the vectorized warped image given the shape, texture and model parameters leads to the minimization of the data term $\mathcal{D}$ and, in the MAP case, the maximization of the prior probability over the shape and texture parameters leads to the minimization of the regularization

⁴This is a common assumption in CGD algorithms [10], however, in reality, some degree of dependence between these parameters is to be expected [8].

term $\mathcal{R}$.

5.2.2 Project-out

Matthews and Baker showed in [10] that one could express the SSD between the vectorized warped image and the linear PCA-based⁵ texture model as the sum of two different terms:

$$\begin{aligned}
 \frac{1}{2} \mathbf{r}^T \mathbf{r} &= \frac{1}{2} \mathbf{r}^T (\mathbf{A} \mathbf{A}^T + \mathbf{I} - \mathbf{A} \mathbf{A}^T) \mathbf{r} \\
 &= \frac{1}{2} \mathbf{r}^T (\mathbf{A} \mathbf{A}^T) \mathbf{r} + \frac{1}{2} \mathbf{r}^T (\mathbf{I} - \mathbf{A} \mathbf{A}^T) \mathbf{r} \\
 &= \frac{1}{2} \|\mathbf{i}[\mathbf{p}] - (\bar{\mathbf{a}} + \mathbf{A} \mathbf{c})\|_{\mathbf{A} \mathbf{A}^T}^2 + \frac{1}{2} \|\mathbf{i}[\mathbf{p}] - (\bar{\mathbf{a}} + \mathbf{A} \mathbf{c})\|_{\mathbf{I} - \mathbf{A} \mathbf{A}^T}^2 \\
 &= f_1(\mathbf{p}, \mathbf{c}) + f_2(\mathbf{p}, \mathbf{c})
 \end{aligned} \tag{5.8}$$

The first term defines the distance *within* the texture subspace and it is always 0 regardless of the value of the shape parameters $\mathbf{p}$:

$$\begin{aligned}
 f_1(\mathbf{p}, \mathbf{c}) &= \frac{1}{2} \|\mathbf{i}[\mathbf{p}] - (\bar{\mathbf{a}} + \mathbf{A} \mathbf{c})\|_{\mathbf{A} \mathbf{A}^T}^2 \\
 &= \frac{1}{2} \left(\underbrace{\mathbf{i}[\mathbf{p}]^T \mathbf{A} \mathbf{A}^T \mathbf{i}[\mathbf{p}]}_{\mathbf{c}^T} - 2 \underbrace{\mathbf{i}[\mathbf{p}]^T \mathbf{A} \mathbf{A}^T \bar{\mathbf{a}}}_0 - 2 \underbrace{\mathbf{i}[\mathbf{p}]^T \mathbf{A} \mathbf{A}^T \mathbf{A} \mathbf{c}}_{\mathbf{c}^T} \right. \\
 &\quad \left. + \underbrace{\bar{\mathbf{a}}^T \mathbf{A} \mathbf{A}^T \bar{\mathbf{a}}}_0 + 2 \underbrace{\bar{\mathbf{a}}^T \mathbf{A} \mathbf{A}^T \mathbf{A} \mathbf{c}}_0 + \underbrace{\mathbf{c}^T \mathbf{A}^T \mathbf{A} \mathbf{A}^T \mathbf{A} \mathbf{c}}_{\mathbf{c}^T} \right) \\
 &= \frac{1}{2} (\mathbf{c}^T \mathbf{c} - 2 \mathbf{c}^T \mathbf{c} + \mathbf{c}^T \mathbf{c}) \\
 &= 0
 \end{aligned} \tag{5.9}$$

⁵The use of PCA ensures the orthonormality of the texture bases and, consequently, $\mathbf{A}^T \mathbf{A} = \mathbf{I}$ (where $\mathbf{I}$ denotes the identity matrix). Similarly, the use of PCA also ensures orthogonality between the texture mean and the texture bases and, hence, $\mathbf{A}^T \bar{\mathbf{a}} = \mathbf{0}$.

The second term measures the distance *to* the texture subspace i.e. the distance within its orthogonal complement. After some algebraic manipulation, one can show that this term reduces to a function that only depends on the shape parameters $\mathbf{p}$:

$$\begin{aligned}
f_2(\mathbf{p}, \mathbf{c}) &= \frac{1}{2} \|\mathbf{i}[\mathbf{p}] - (\bar{\mathbf{a}} + \mathbf{A}\mathbf{c})\|_{\bar{\mathbf{A}}}^2 \\
&= \frac{1}{2} \left(\mathbf{i}[\mathbf{p}]^T \bar{\mathbf{A}} \mathbf{i}[\mathbf{p}] - 2\mathbf{i}[\mathbf{p}]^T \bar{\mathbf{A}} \bar{\mathbf{a}} - \underbrace{2\mathbf{i}[\mathbf{p}]^T \bar{\mathbf{A}} \mathbf{A} \mathbf{c}}_0 \right. \\
&\quad \left. + \bar{\mathbf{a}}^T \bar{\mathbf{A}} \bar{\mathbf{a}} + 2 \underbrace{\bar{\mathbf{a}}^T \bar{\mathbf{A}} \mathbf{A} \mathbf{c}}_0 + \underbrace{\mathbf{c}^T \mathbf{A}^T \bar{\mathbf{A}} \mathbf{A} \mathbf{c}}_0 \right) \\
&= \frac{1}{2} (\mathbf{i}[\mathbf{p}]^T \bar{\mathbf{A}} \mathbf{i}[\mathbf{p}] - 2\mathbf{i}[\mathbf{p}]^T \bar{\mathbf{A}} \bar{\mathbf{a}} + \bar{\mathbf{a}}^T \bar{\mathbf{A}} \bar{\mathbf{a}}) \\
&= \frac{1}{2} \|\mathbf{i}[\mathbf{p}] - \bar{\mathbf{a}}\|_{\bar{\mathbf{A}}}^2
\end{aligned} \tag{5.10}$$

where, for convenience, we have defined the orthogonal complement to the texture subspace as $\bar{\mathbf{A}} = \mathbf{I} - \mathbf{A}\mathbf{A}^T$. Note that, as mentioned above, this term does not depend on the texture parameters $\mathbf{c}$:

$$f_2(\mathbf{p}, \mathbf{c}) = \hat{f}_2(\mathbf{p}) = \frac{1}{2} \|\mathbf{i}[\mathbf{p}] - \bar{\mathbf{a}}\|_{\bar{\mathbf{A}}}^2 \tag{5.11}$$

Therefore, using the *project-out* (PO) trick, the minimization problems defined by equations 5.2 and 5.3 reduce to:

$$\mathbf{p}^* = \arg \min_{\mathbf{p}} \underbrace{\frac{1}{2} \|\mathbf{i}[\mathbf{p}] - \bar{\mathbf{a}}\|_{\bar{\mathbf{A}}}^2}_{\mathcal{D}(\mathbf{i}[\mathbf{p}])} \tag{5.12}$$

and

$$\mathbf{p}^* = \arg \min_{\mathbf{p}} \underbrace{\frac{1}{2} \|\mathbf{p}\|^2}_{\mathcal{R}(\mathbf{p})} + \underbrace{\frac{1}{2} \|\mathbf{i}[\mathbf{p}] - \bar{\mathbf{a}}\|_{\bar{\mathbf{A}}}^2}_{\mathcal{D}(\mathbf{i}[\mathbf{p}])} \tag{5.13}$$

respectively.

5.2.2.1 Probabilistic formulation

Assuming the probabilistic models defined by equations 5.4 and 5.5, a *Bayesian* formulation of the project-out data term can be naturally derived by marginalizing over the texture parameters to obtain the following marginalized density:

$$\begin{aligned} p(\mathbf{i}[\mathbf{p}]|\mathbf{p}, \Theta) &= \int_{\mathbf{c}} p(\mathbf{i}[\mathbf{p}]|\mathbf{p}, \mathbf{c}, \Theta) p(\mathbf{c}|\Sigma) d\mathbf{c} \\ &= \mathcal{N}(\bar{\mathbf{a}}, \mathbf{A}\Sigma\mathbf{A}^T + \sigma^2\mathbf{I}) \end{aligned} \quad (5.14)$$

and applying the Woodbury formula⁶ [92] to decompose the natural logarithm of the previous density into the sum of two different terms:

$$\begin{aligned} \ln p(\mathbf{i}[\mathbf{p}]|\mathbf{p}, \Theta) &= \frac{1}{2} \|\mathbf{i}[\mathbf{p}] - \bar{\mathbf{a}}\|_{(\mathbf{A}\Sigma\mathbf{A}^T + \sigma^2\mathbf{I})^{-1}}^2 \\ &= \frac{1}{2} \|\mathbf{i}[\mathbf{p}] - \bar{\mathbf{a}}\|_{\mathbf{A}\mathbf{D}^{-1}\mathbf{A}^T}^2 + \frac{1}{2\sigma^2} \|\mathbf{i}[\mathbf{p}] - \bar{\mathbf{a}}\|_{\mathbf{A}}^2 \end{aligned} \quad (5.15)$$

where $\mathbf{D} = \text{diag}(\lambda_{\mathbf{a}_1} + \sigma^2, \dots, \lambda_{\mathbf{a}_m} + \sigma^2)$.

As depicted by figure 5.1, these two terms define respectively: *i*) the Mahalanobis distance *within* the linear texture subspace; and *ii*) the Euclidean distance *to* the linear texture subspace (i.e. the Euclidean distance within its orthogonal complement) weighted by the inverse of the estimated image noise. Note that when the variance Σ of the prior distribution over the latent texture space increases (and especially as $\Sigma \rightarrow \infty$) $\mathbf{c}$ becomes uniformly

⁶Using the Woodbury formula:

$$\begin{aligned} (\mathbf{A}\Sigma\mathbf{A}^T + \sigma^2\mathbf{I})^{-1} &= \frac{1}{\sigma^2}\mathbf{I} - \frac{1}{\sigma^4}\mathbf{A} \underbrace{(\Sigma^{-1} + \frac{1}{\sigma^2}\mathbf{I})^{-1}}_{\text{reapply Woodbury}} \mathbf{A}^T \\ &= \frac{1}{\sigma^2}\mathbf{I} - \frac{1}{\sigma^4}\mathbf{A}(\sigma^2\mathbf{I} - \sigma^4(\Sigma + \sigma^2\mathbf{I})^{-1})\mathbf{A}^T \\ &= \frac{1}{\sigma^2}\mathbf{I} - \frac{1}{\sigma^4}\mathbf{A}(\sigma^2\mathbf{I} - \sigma^4\mathbf{D}^{-1})\mathbf{A}^T \\ &= \mathbf{A}\mathbf{D}^{-1}\mathbf{A}^T + \frac{1}{\sigma^2}(\mathbf{I} - \mathbf{A}\mathbf{A}^T) \end{aligned}$$

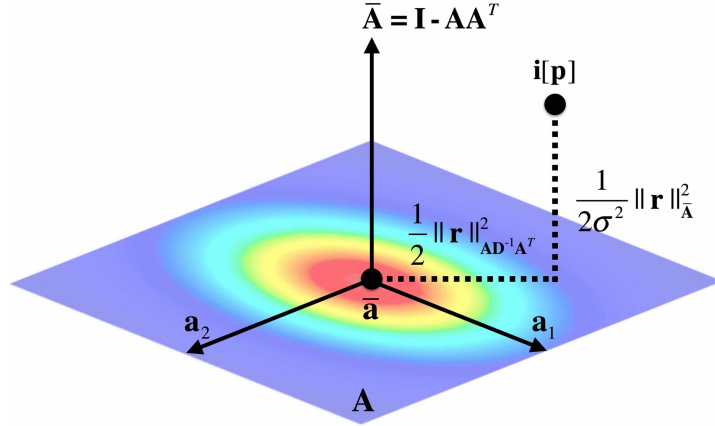


Figure 5.1: The Bayesian project-out formulation fits AAM by minimizing two different distances: *i*) the Mahalanobis distance *within* the linear texture subspace; and *ii*) the Euclidean distance *to* the linear texture subspace (i.e. the Euclidean distance within its orthogonal complement) weighted by the inverse of the estimated image noise.

distributed and the contribution of the first term $\frac{1}{2} \|\mathbf{i}[\mathbf{p}] - \bar{\mathbf{a}}\|_{\mathbf{A}\mathbf{D}^{-1}\mathbf{A}^T}^2$ vanishes; in this case, a weighted version of the project-out data term defined by equation 5.12 is obtained. Hence, given our Bayesian formulation, the project-out loss arises naturally by assuming a uniform prior over the latent texture space.

The probabilistic formulations of the minimization problems defined by equations 5.12 and 5.13 can be derived, from the *Bayesian project-out* (BPO) cost function, as

$$\begin{aligned}
 \mathbf{p}^* &= \arg \max_{\mathbf{p}} \ln p(\mathbf{i}[\mathbf{p}] | \mathbf{p}, \Theta) \\
 &= \arg \min_{\mathbf{p}} \underbrace{\frac{1}{2\sigma^2} \|\mathbf{i}[\mathbf{p}] - \bar{\mathbf{a}}\|_{\mathbf{Q}}^2}_{\mathcal{D}(\mathbf{i}[\mathbf{p}])}
 \end{aligned} \tag{5.16}$$

and

$$\begin{aligned}
\mathbf{p}^* &= \arg \max_{\mathbf{p}} p(\mathbf{p}, \mathbf{i}[\mathbf{p}] | \Theta) \\
&= \arg \max_{\mathbf{p}} p(\mathbf{p} | \mathbf{\Lambda}) p(\mathbf{i}[\mathbf{p}] | \mathbf{p}, \Theta) \\
&= \arg \max_{\mathbf{p}} \ln p(\mathbf{p} | \mathbf{\Lambda}) + \ln p(\mathbf{i}[\mathbf{p}] | \mathbf{p}, \Theta) \\
&= \arg \min_{\mathbf{p}} \underbrace{\frac{1}{2} \|\mathbf{p}\|_{\mathbf{\Lambda}^{-1}}^2}_{\mathcal{R}(\mathbf{p})} + \underbrace{\frac{1}{2\sigma^2} \|\mathbf{i}[\mathbf{p}] - \bar{\mathbf{a}}\|_{\mathbf{Q}}^2}_{\mathcal{D}(\mathbf{i}[\mathbf{p}])}
\end{aligned} \tag{5.17}$$

respectively. Where $\mathbf{Q} = \mathbf{I} - \mathbf{A}(\mathbf{I} - \sigma^2 \mathbf{D}^{-1})\mathbf{A}^T$.

5.3 Type of Composition

Assuming, for the time being, that the true texture parameters $\mathbf{c}^*$ are known, the problem defined by equation 5.2 reduces to a non-rigid image alignment problem [69, 70] between the particular instance of the object present in the image and its optimal texture reconstruction by the texture model:

$$\mathbf{p}^* = \arg \min_{\mathbf{p}} \frac{1}{2} \|\mathbf{i}[\mathbf{p}] - \mathbf{a}\|^2 \tag{5.18}$$

where $\mathbf{a} = \bar{\mathbf{a}} + \mathbf{A}\mathbf{c}^*$ is obtained by directly evaluating Equation 3.5 given the true texture parameters $\mathbf{c}^*$.

CGD algorithms iteratively solve the above non-linear optimization problem with respect to the shape parameters $\mathbf{p}$ by:

1. Introducing an incremental warp $\mathcal{W}(\mathbf{x}; \Delta\mathbf{p})$ according to the particular composition scheme being used.
2. Linearizing the incremental warp around the identity warp $\mathcal{W}(\mathbf{x}; \Delta\mathbf{p}) = \mathcal{W}(\mathbf{x}; \mathbf{0}) = \mathbf{x}$.

3. Solving for the parameters $\Delta \mathbf{p}$ of the incremental warp.
4. Updating the current warp estimate by using an appropriate compositional update rule.
5. Going back to Step 1 until a particular convergence criterion is met.

Existent CGD algorithms for fitting AAM have introduced the incremental warp either on the image or model sides in what are known as *forward* and *inverse* compositional frameworks [10, 32, 34, 37, 38, 40] respectively. However, drawing inspiration from related works in the field of image alignment [44, 45, 46, 47], one can derived novel CGD algorithms by introducing incremental warps on both the image and model sides simultaneously. Depending on the exact relationship between these incremental warps two novel types of composition can be defined: *asymmetric* and *bidirectional*.

The following subsections explain how to introduce the incremental warp into the cost function and how to update the current warp estimate for the four types of composition considered in this chapter: *i*) forward; *ii*) inverse; *iii*) asymmetric; and *iv*) bidirectional. These subsections will be derived using the non-regularized expression in equation 5.2 and the regularized expression in equation 5.7. Furthermore, to maintain consistency with the vector notation preferred in this thesis an abuse of notation will be made and the operations of warp composition⁷ $\mathcal{W}(\mathbf{x}; \mathbf{p}) \circ \mathcal{W}(\mathbf{x}; \Delta \mathbf{p})$ and inversion⁷ $\mathcal{W}(\mathbf{x}; \mathbf{q})^{-1}$ will be denoted as simply $\mathbf{p} \circ \Delta \mathbf{p}$ and $\mathbf{q}^{-1}$ respectively.

⁷Recall that, further details regarding composition, $\mathbf{p} \circ \Delta \mathbf{p}$, and inversion, $\Delta \mathbf{q}^{-1}$, of typical AAM deformation models such as PWA and TPS warps are given in appendix A.

5.3.1 Forward

In the forward compositional framework the incremental warp $\Delta \mathbf{p}$ is introduced on the image side at each iteration by composing it with the current warp estimate $\mathbf{p}$. For the non-regularized case in equation 5.2 this leads to:

$$\Delta \mathbf{p}^* = \arg \min_{\Delta \mathbf{p}} \frac{1}{2} \|\mathbf{i}[\mathbf{p} \circ \Delta \mathbf{p}] - (\mathbf{a} + \mathbf{A}\mathbf{c})\|^2 \quad (5.19)$$

Once the optimal values for the parameters of the incremental warp are obtained, the current warp estimate is updated according to the following compositional update rule:

$$\mathbf{p} \leftarrow \mathbf{p} \circ \Delta \mathbf{p} \quad (5.20)$$

On the other hand, using equation 5.7, forward composition can be expressed as:

$$\begin{aligned} \Delta \mathbf{p}^* = \arg \min_{\Delta \mathbf{p}} & \frac{1}{2\sigma^2} \|\mathbf{i}[\mathbf{p} \circ \Delta \mathbf{p}] - (\mathbf{a} + \mathbf{A}\mathbf{c})\|^2 \\ & + \frac{1}{2} \|\mathbf{p}\|_{\mathbf{\Lambda}^{-1}}^2 + \|\mathbf{c}\|_{\mathbf{\Sigma}^{-1}}^2 \end{aligned} \quad (5.21)$$

Because of the inclusion of the prior term over the shape parameters $\frac{1}{2} \|\mathbf{p}\|_{\mathbf{\Lambda}^{-1}}^2$, one cannot update the current warp estimate using the update rule in equation 5.20. Instead, as noted by Papandreou and Maragos in [34], one needs to compute the *forward compositional to forward additive* parameter update Jacobian matrix $J_{\mathbf{p}} \in \mathbb{R}^{(4+n) \times (4+n)}$ ⁸. This matrix is used to map the forward compositional increment $\Delta \mathbf{p}$ to its *first* order additive equivalent

⁸Note that, Papandreou and Maragos derived the *inverse compositional to forward additive* parameter update Jacobian matrix $J_{\mathbf{q}}$, however, it is straightforward to modify their original formulation to obtain $J_{\mathbf{p}}$. Further details regarding the computation of the parameter update Jacobian matrices can be found in [34], its appendix: http://www.stat.ucla.edu/~gpapan/pubs/confr/PapandreouMaragos_AAM_supmat-cvpr08.pdf and posterior correction http://www.stat.ucla.edu/~gpapan/pubs/confr/PapandreouMaragos_AAM_typo-cvpr08.pdf

$J_{\mathbf{p}}\Delta\mathbf{p}$. In this case, the current estimate of the warp is computed using the following update rule:

$$\mathbf{p} \leftarrow \left(\mathbf{\Lambda}^{-1} + (J_{\mathbf{p}}\mathbf{H}J_{\mathbf{p}})^{-1} \right)^{-1} \left((J_{\mathbf{p}}\mathbf{H}J_{\mathbf{p}})^{-1} (\mathbf{p} + J_{\mathbf{p}}\Delta\mathbf{p}) \right) \quad (5.22)$$

where $\mathbf{H}$ denotes the approximate or true *Hessians* of the residual $\|\mathbf{i}[\mathbf{p} \circ \Delta\mathbf{p}] - (\mathbf{a} + \mathbf{Ac})\|^2$ with respect to the incremental parameters $\Delta\mathbf{p}$ and $\Delta\mathbf{p}$ itself is the optimal solution of the non-regularized problem in equation 5.19. Note that, in section 5.4, $\mathbf{H}$ is explicitly derived for all the optimization methods studied in this chapter.

5.3.2 Inverse

The inverse compositional framework inverts the roles of the image and model by introducing the incremental warp on the model side. Using equation 5.2:

$$\Delta\mathbf{q}^* = \arg \min_{\Delta\mathbf{q}} \frac{1}{2} \|\mathbf{i}[\mathbf{p}] - (\mathbf{a} + \mathbf{Ac})[\Delta\mathbf{q}]\|^2 \quad (5.23)$$

Note that, in this case, the model is the one deformed by incremental warp.

Because the incremental warp is introduced on the model side, the solution $\Delta\mathbf{q}$ needs to be inverted before it is composed with the current warp estimate:

$$\mathbf{p} \leftarrow \mathbf{p} \circ \Delta\mathbf{q}^{-1} \quad (5.24)$$

Similarly, using the regularized expression in equation 5.7, inverse composition

is expressed as:

$$\begin{aligned} \Delta \mathbf{q}^* = \arg \min_{\Delta \mathbf{q}} & \frac{1}{2\sigma^2} \|\mathbf{i}[\mathbf{p}] - (\mathbf{a} + \mathbf{A}\mathbf{c})[\Delta \mathbf{q}]\|^2 \\ & + \frac{1}{2} \|\mathbf{p}\|_{\Lambda^{-1}}^2 + \|\mathbf{c}\|_{\Sigma^{-1}}^2 \end{aligned} \quad (5.25)$$

and the update of the current warp estimate is obtained using:

$$\mathbf{p} \leftarrow \left(\Lambda^{-1} + (J_{\mathbf{q}} \mathbf{H} J_{\mathbf{q}})^{-1} \right)^{-1} \left((J_{\mathbf{q}} \mathbf{H} J_{\mathbf{q}})^{-1} (\mathbf{p} + J_{\mathbf{q}} \Delta \mathbf{q}) \right) \quad (5.26)$$

where, in this case, $J_{\mathbf{q}}$ denotes the *inverse compositional* to *forward additive* parameter update Jacobian matrix $J_{\mathbf{q}} \in \mathbb{R}^{(4+n) \times (4+n)}$ originally derived by Papandreou and Maragos [34].

5.3.3 Asymmetric

Asymmetric composition introduces two related incremental warps onto the cost function; one on the image side (forward) and the other on the model side (inverse). Using equation 5.2 this is expressed as:

$$\Delta \mathbf{p}^* = \arg \min_{\Delta \mathbf{p}} \frac{1}{2} \|\mathbf{i}[\mathbf{p} \circ \alpha \Delta \mathbf{p}] - (\mathbf{a} + \mathbf{A}\mathbf{c})[\beta \Delta \mathbf{p}^{-1}]\|^2 \quad (5.27)$$

Note that the two incremental warps are defined to be each others inverse. Consequently, using the first order approximation to warp inversion for typical AAM warps $\Delta \mathbf{p}^{-1} = -\Delta \mathbf{p}$ defined in [10], one can rewrite the asymmetric cost function as:

$$\Delta \mathbf{p}^* = \arg \min_{\Delta \mathbf{p}} \frac{1}{2} \|\mathbf{i}[\mathbf{p} \circ \alpha \Delta \mathbf{p}] - (\mathbf{a} + \mathbf{A}\mathbf{c})[-\beta \Delta \mathbf{p}]\|^2 \quad (5.28)$$

Although this cost function needs to be linearized around both incremental warps, the parameters $\Delta \mathbf{p}$ controlling these warps are the same. Also, note that the parameters $\alpha \in [0, 1]$ and $\beta = (1 - \alpha)$ control the relative contribution of both incremental warps in the computation of the optimal value for $\Delta \mathbf{p}$.

In this case, the update rule for the current warp estimate is obtained by combining the forward and inverse compositional update rules into a single compositional update rule:

$$\mathbf{p} \leftarrow \mathbf{p} \circ \alpha \Delta \mathbf{p} \circ \beta \Delta \mathbf{p} \quad (5.29)$$

Using equation 5.7, asymmetric composition is expressed as:

$$\begin{aligned} \Delta \mathbf{p}^* = \arg \min_{\Delta \mathbf{q}} & \frac{1}{2\sigma^2} \|\mathbf{i}[\mathbf{p} \circ \alpha \Delta \mathbf{p}] - (\mathbf{a} + \mathbf{A}\mathbf{c})[-\beta \Delta \mathbf{p}]\|^2 \\ & + \frac{1}{2} \|\mathbf{p}\|_{\mathbf{\Lambda}^{-1}}^2 + \|\mathbf{c}\|_{\mathbf{\Sigma}^{-1}}^2 \end{aligned} \quad (5.30)$$

And the current warp estimate is updates using:

$$\begin{aligned} \mathbf{p} \leftarrow & \left(\mathbf{\Lambda}^{-1} + (J_{\mathbf{p}} \mathbf{H} J_{\mathbf{p}})^{-1} \right)^{-1} \\ & \left((J_{\mathbf{p}} \mathbf{H} J_{\mathbf{p}})^{-1} (\mathbf{p} + \alpha J_{\mathbf{p}} \Delta \mathbf{p} + \beta J_{\mathbf{p}} \Delta \mathbf{p}) \right) \end{aligned} \quad (5.31)$$

which reduces the forward update rule in equation 5.22 because $\alpha + \beta = 1$.

Note that, the special case in which $\alpha = \beta = 0.5$ is also referred to as *symmetric* composition [45, 46, 47] and that the forward and inverse compositions can also be obtained from asymmetric composition by setting $\alpha = 1$, $\beta = 0$ and $\alpha = 0$, $\beta = 1$ respectively.

5.3.4 Bidirectional

Similar to asymmetric composition, bidirectional composition also introduces incremental warps on both image and model sides. However, in this case, the two incremental warps are assumed to be independent from each other. Based on equation 5.2:

$$\Delta \mathbf{p}^*, \Delta \mathbf{q}^* = \arg \min_{\Delta \mathbf{p}, \Delta \mathbf{q}} \frac{1}{2} \|\mathbf{i}[\mathbf{p} \circ \Delta \mathbf{p}] - (\mathbf{a} + \mathbf{A}\mathbf{c})[\Delta \mathbf{q}]\|^2 \quad (5.32)$$

Consequently the cost function needs to be linearized around both incremental warps and solved with respect to the parameters controlling both warps, $\Delta \mathbf{p}$ and $\Delta \mathbf{q}$.

Once the optimal value for both sets of parameters is recovered, the current estimate of the warp is updated using:

$$\mathbf{p} \leftarrow \mathbf{p} \circ \Delta \mathbf{p} \circ \Delta \mathbf{q}^{-1} \quad (5.33)$$

For equation 5.7, bidirectional composition is written as:

$$\begin{aligned} \Delta \mathbf{p}^*, \Delta \mathbf{q}^* = \arg \min_{\Delta \mathbf{p}, \Delta \mathbf{q}} & \frac{1}{2\sigma^2} \|\mathbf{i}[\mathbf{p} \circ \Delta \mathbf{p}] - (\mathbf{a} + \mathbf{A}\mathbf{c})[\Delta \mathbf{q}]\|^2 \\ & + \frac{1}{2} \|\mathbf{p}\|_{\mathbf{\Lambda}^{-1}}^2 + \|\mathbf{c}\|_{\mathbf{\Sigma}^{-1}}^2 \end{aligned} \quad (5.34)$$

And, in this case, the update rule for the current warp estimate is:

$$\begin{aligned} \mathbf{p} \leftarrow & \left(\mathbf{\Lambda}^{-1} + (J_{\mathbf{p}}\mathbf{H}J_{\mathbf{p}} + J_{\mathbf{q}}\mathbf{H}J_{\mathbf{q}})^{-1} \right)^{-1} \\ & \left((J_{\mathbf{p}}\mathbf{H}J_{\mathbf{p}} + J_{\mathbf{q}}\mathbf{H}J_{\mathbf{q}})^{-1} (\mathbf{p} + J_{\mathbf{p}}\Delta \mathbf{p} + J_{\mathbf{q}}\Delta \mathbf{q}) \right) \end{aligned} \quad (5.35)$$

5.4 Optimization Method

Steps 2 and 3 in CGD algorithms, i.e. linearizing the cost and solving for the incremental warp respectively, depend on the specific optimization method used by the algorithm. This section presents a detailed study of the following optimization methods⁹: i) *Gauss-Newton* [48, 10, 32, 38, 34, 40]; ii) *Newton* [48, 41]; and iii) *Wiberg* [49, 50, 34, 40].

These methods can be used to iteratively solve the non-linear optimization problems defined by Equations 5.7 and 5.15. The main differences between them are:

1. The term being linearized. Gauss-Newton and Wiberg linearize the residual $\mathbf{r}$ while Newton linearizes the whole data term $\mathcal{D}$.
2. The way in which each method solves for the incremental parameters $\Delta\mathbf{c}$, $\Delta\mathbf{p}$ and $\Delta\mathbf{q}$. Gauss-Newton and Newton can either solve for them *simultaneously* or in an *alternated* manner while Wiberg defines its own procedure to solve for different sets of parameters¹⁰.

The following subsections thoroughly explain how these optimization methods are used in CGD algorithms. In order to simplify their comprehension full derivations will be given for all methods using the SSD data term (equation 5.2) with both asymmetric (section 5.3.3) and bidirectional (section 5.3.4) compositions¹¹ while only direct solutions will be given for the PO

⁹Amberg et al. proposed the use of the *Steepest Descent* method [48] in [37]. However, their approach requires a special formulation of the deformation model and it performs poorly using the standard independent AAM formulation [10] used in this thesis.

¹⁰Wiberg reduces to Gauss-Newton when only a single set of parameters needs to be inferred.

¹¹These represent the most general cases because the derivations for forward, inverse and symmetric compositions can be directly obtained from the asymmetric one; and they require solving for both shape and texture parameters.

data term (equation 5.12). Note that section 5.3 derives the update rules for the regularized expression in equation 5.7 and, consequently, there is no need to consider regularization throughout this section¹²

5.4.1 Gauss-Newton

When *asymmetric* composition is used, the optimization problem defined by the SSD data term is:

$$\Delta \mathbf{c}^*, \Delta \mathbf{p}^* = \arg \min_{\Delta \mathbf{c}, \Delta \mathbf{p}} \frac{1}{2} \mathbf{r}_a^T \mathbf{r}_a \quad (5.36)$$

with the asymmetric residual $\mathbf{r}_a$ defined as:

$$\mathbf{r}_a = \mathbf{i}[\mathbf{p} \circ \alpha \Delta \mathbf{p}] - (\mathbf{a} + \mathbf{A}(\mathbf{c} + \Delta \mathbf{c}))[\beta \Delta \mathbf{p}^{-1}] \quad (5.37)$$

and where the incremental texture parameters $\Delta \mathbf{c}$ ¹³ have been introduced. The Gauss-Newton method solves the optimization problem by performing a *first* order Taylor expansion of the residual:

$$\begin{aligned} \mathbf{r}_a(\Delta \ell) &\approx \hat{\mathbf{r}}_a(\Delta \ell) \\ &\approx \mathbf{r}_a + \frac{\partial \mathbf{r}_a}{\partial \Delta \ell} \Delta \ell \end{aligned} \quad (5.38)$$

and solving the following approximation of the original problem:

$$\Delta \ell^* = \arg \min_{\Delta \ell} \frac{1}{2} \hat{\mathbf{r}}_a^T \hat{\mathbf{r}}_a \quad (5.39)$$

¹²The derivation of regularized solutions with respect to the texture parameters $\Delta \mathbf{c}$ is straightforward and, hence, omitted throughout this section.

¹³The value of the current estimate of texture parameters is updated at each iteration using the following additive update rule: $\mathbf{c} \leftarrow \mathbf{c} + \Delta \mathbf{c}$

where $\Delta\ell = (\Delta\mathbf{c}^T, \Delta\mathbf{p}^T)^T$ and the partial derivative of the residual with respect to the parameters, i.e. the *Jacobian* of the residual, is defined as:

$$\begin{aligned}\frac{\partial \mathbf{r}_a}{\partial \Delta\ell} &= \begin{pmatrix} \frac{\partial \mathbf{r}_a}{\partial \Delta\mathbf{c}}, \frac{\partial \mathbf{r}_a}{\partial \Delta\mathbf{p}} \end{pmatrix} \\ &= \begin{pmatrix} -\mathbf{A}, \nabla \mathbf{t} \frac{\partial \mathcal{W}}{\partial \Delta\mathbf{p}} \end{pmatrix} \\ &= (-\mathbf{A}, \mathbf{J}_t)\end{aligned}\tag{5.40}$$

where $\nabla \mathbf{t} = (\alpha \nabla \mathbf{i}[\mathbf{p}] + \beta \nabla(\mathbf{a} + \mathbf{A}\mathbf{c}))$.

When *bidirectional* composition is used, the optimization problem is defined as:

$$\Delta\mathbf{c}^*, \Delta\mathbf{p}^*, \Delta\mathbf{q}^* = \arg \min_{\Delta\mathbf{c}, \Delta\mathbf{p}, \Delta\mathbf{q}} \frac{1}{2} \mathbf{r}_b^T \mathbf{r}_b \tag{5.41}$$

where the bidirectional residual $\mathbf{r}_b$ reduces to:

$$\mathbf{r}_b = \mathbf{i}[\mathbf{p} \circ \Delta\mathbf{p}] - (\mathbf{a} + \mathbf{A}(\mathbf{c} + \Delta\mathbf{c}))[\Delta\mathbf{q}] \tag{5.42}$$

The Gauss-Newton method proceeds in exactly the same manner as before, i.e. performing a first order Taylor expansion:

$$\begin{aligned}\mathbf{r}_b(\Delta\ell) &\approx \hat{\mathbf{r}}_b(\Delta\ell) \\ &\approx \mathbf{r}_b + \frac{\partial \mathbf{r}_b}{\partial \Delta\ell} \Delta\ell\end{aligned}\tag{5.43}$$

and solving the approximated problem:

$$\Delta\ell^* = \arg \min_{\Delta\ell} \frac{1}{2} \hat{\mathbf{r}}_b^T \hat{\mathbf{r}}_b \tag{5.44}$$

where, in this case, $\Delta\ell = (\Delta\mathbf{c}^T, \Delta\mathbf{p}^T, \Delta\mathbf{q}^T)^T$ and the Jacobian of the resid-

ual is defined as:

$$\begin{aligned}\frac{\partial \mathbf{r}_b}{\partial \Delta \ell} &= \left(\frac{\partial \mathbf{r}_b}{\partial \Delta \mathbf{c}}, \frac{\partial \mathbf{r}_b}{\partial \Delta \mathbf{p}}, \frac{\partial \mathbf{r}_b}{\partial \Delta \mathbf{q}} \right) \\ &= (-\mathbf{A}, \mathbf{J}_i, -\mathbf{J}_a)\end{aligned}\tag{5.45}$$

where $\mathbf{J}_i = \nabla \mathbf{i}[\mathbf{p}] \frac{\partial \mathcal{W}}{\partial \Delta \mathbf{p}}$ and $\mathbf{J}_a = \nabla (\mathbf{a} + \mathbf{A}\mathbf{c}) \frac{\partial \mathcal{W}}{\partial \Delta \mathbf{q}}$.

5.4.1.1 Simultaneous

The optimization problem defined by equations 5.39 and 5.44 can be solved with respect to all parameters simultaneously by simply equating their derivative to 0:

$$\begin{aligned}0 &= \frac{\partial \frac{1}{2} \hat{\mathbf{r}}^T \hat{\mathbf{r}}}{\partial \Delta \ell} \\ &= \frac{\partial \frac{1}{2} (\mathbf{r} + \frac{\partial \mathbf{r}}{\partial \Delta \ell} \Delta \ell)^T (\mathbf{r} + \frac{\partial \mathbf{r}}{\partial \Delta \ell} \Delta \ell)}{\partial \Delta \ell} \\ &= \left(\mathbf{r} + \frac{\partial \mathbf{r}}{\partial \Delta \ell} \Delta \ell \right) \frac{\partial \mathbf{r}^T}{\partial \Delta \ell}\end{aligned}\tag{5.46}$$

The solution is given by:

$$\Delta \ell^* = - \left(\frac{\partial \mathbf{r}^T}{\partial \Delta \ell} \frac{\partial \mathbf{r}}{\partial \Delta \ell} \right)^{-1} \frac{\partial \mathbf{r}^T}{\partial \Delta \ell} \mathbf{r}\tag{5.47}$$

where $\left(\frac{\partial \mathbf{r}^T}{\partial \Delta \ell} \frac{\partial \mathbf{r}}{\partial \Delta \ell} \right)$ is known as the Gauss-Newton approximation to the *Hessian* matrix.

Directly inverting $\left(\frac{\partial \mathbf{r}^T}{\partial \Delta \ell} \frac{\partial \mathbf{r}}{\partial \Delta \ell} \right)$ has complexity¹⁴ $O((n+m)^3)$ for asymmetric composition and $O((2n+m)^3)$ for bidirectional composition. However, one can take advantage of the problem structure and derive an algorithm with smaller complexity by using the *Schur complement*¹⁵ [48].

¹⁴ m and n denote the number of shape and texture parameters respectively while F denotes the number of pixels on the reference frame.

¹⁵Applying the Schur complement to the following system of equations:

$$\begin{aligned}\mathbf{Ax} + \mathbf{By} &= \mathbf{a} \\ \mathbf{Cx} + \mathbf{Dy} &= \mathbf{b}\end{aligned}$$

For *asymmetric* composition we have:

$$\begin{aligned}
 -\left(\frac{\partial \mathbf{r}_a}{\partial \Delta \ell} \frac{\partial \mathbf{r}_a}{\partial \Delta \ell}^T\right) \Delta \ell &= \frac{\partial \mathbf{r}_a}{\partial \Delta \ell}^T \mathbf{r} \\
 \begin{pmatrix} -\underbrace{\mathbf{A}^T \mathbf{A}}_{\mathbf{I}} & \mathbf{A}^T \mathbf{J}_t \\ \mathbf{J}_t^T \mathbf{A} & -\mathbf{J}_t^T \mathbf{J}_t \end{pmatrix} \begin{pmatrix} \Delta \mathbf{c} \\ \Delta \mathbf{p} \end{pmatrix} &= \begin{pmatrix} -\mathbf{A}^T \\ \mathbf{J}_t^T \end{pmatrix} \mathbf{r}_a
 \end{aligned} \tag{5.48}$$

Applying the Schur complement, the solution for $\Delta \mathbf{p}$ is given by:

$$\begin{aligned}
 -(\mathbf{J}_t^T \mathbf{J}_t + \mathbf{J}_t^T \mathbf{A} \mathbf{A}^T \mathbf{J}_t) \Delta \mathbf{p} &= \mathbf{J}_t^T \mathbf{r} - \mathbf{J}_t^T \mathbf{A} \mathbf{A}^T \mathbf{r}_a \\
 -\mathbf{J}_t^T (\mathbf{I} - \mathbf{A} \mathbf{A}^T) \mathbf{J}_t \Delta \mathbf{p} &= \mathbf{J}_t^T (\mathbf{I} - \mathbf{A} \mathbf{A}^T) \mathbf{r}_a \\
 -\mathbf{J}_t^T \bar{\mathbf{A}} \mathbf{J}_t \Delta \mathbf{p} &= \mathbf{J}_t^T \bar{\mathbf{A}} \mathbf{r}_a \\
 \Delta \mathbf{p}^* &= -(\mathbf{J}_t^T \bar{\mathbf{A}} \mathbf{J}_t)^{-1} \\
 &\quad \mathbf{J}_t^T \bar{\mathbf{A}} \mathbf{r}_a
 \end{aligned} \tag{5.49}$$

and plugging the solution for $\Delta \mathbf{p}$ into equation 5.48 the optimal value for $\Delta \mathbf{c}$ is obtained by:

$$\begin{aligned}
 -\Delta \mathbf{c} + \mathbf{A}^T \mathbf{J}_t \Delta \mathbf{p} &= -\mathbf{A}^T \mathbf{r}_a \\
 \Delta \mathbf{c}^* &= \mathbf{A}^T (\mathbf{r}_a + \mathbf{J}_t \Delta \mathbf{p})
 \end{aligned} \tag{5.50}$$

Using the above procedure the complexity¹⁴ of solving each Gauss-Newton step is reduced to:

$$O(\underbrace{nmF}_{\mathbf{J}_t^T \bar{\mathbf{A}}} + \underbrace{n^2F + n^3}_{(\mathbf{J}_t^T \bar{\mathbf{A}} \mathbf{J}_t)^{-1}}) \tag{5.51}$$

Using *bidirectional* composition, one can apply the Schur complement either

the solution for $\mathbf{x}$ is given by:

$$(\mathbf{A} - \mathbf{B} \mathbf{D}^{-1} \mathbf{C}) \mathbf{x} = \mathbf{a} - \mathbf{B} \mathbf{D}^{-1} \mathbf{b}$$

and the solution for $\mathbf{y}$ is obtained by substituting the value of $\mathbf{x}$ into the original system.

one or two times in order to take advantage of the 3×3 block structure of the matrix $\begin{pmatrix} \frac{\partial \mathbf{r}_b}{\partial \Delta \ell}^T & \frac{\partial \mathbf{r}_b}{\partial \Delta \ell} \end{pmatrix}$:

$$\begin{aligned} - \begin{pmatrix} \frac{\partial \mathbf{r}_b}{\partial \Delta \ell}^T & \frac{\partial \mathbf{r}_b}{\partial \Delta \ell} \end{pmatrix} \Delta \ell &= \frac{\partial \mathbf{r}_b}{\partial \Delta \ell}^T \mathbf{r}_b \\ - \begin{pmatrix} \frac{\partial \mathbf{r}_b}{\partial \Delta \ell}^T & \frac{\partial \mathbf{r}_b}{\partial \Delta \ell} \end{pmatrix} \begin{pmatrix} \Delta \mathbf{c} \\ \Delta \mathbf{p} \\ \Delta \mathbf{q} \end{pmatrix} &= \begin{pmatrix} -\mathbf{A}^T \\ \mathbf{J}_i^T \\ -\mathbf{J}_a^T \end{pmatrix} \mathbf{r}_b \end{aligned} \quad (5.52)$$

where

$$- \begin{pmatrix} \frac{\partial \mathbf{r}_b}{\partial \Delta \ell}^T & \frac{\partial \mathbf{r}_b}{\partial \Delta \ell} \end{pmatrix} = \left(\begin{array}{c|cc} \underbrace{-\mathbf{A}^T \mathbf{A}}_{\mathbf{I}} & \mathbf{A}^T \mathbf{J}_i & -\mathbf{A}^T \mathbf{J}_a \\ \hline \mathbf{J}_i^T \mathbf{A} & -\mathbf{J}_i^T \mathbf{J}_i & \mathbf{J}_i^T \mathbf{J}_a \\ -\mathbf{J}_a^T \mathbf{A} & \mathbf{J}_a^T \mathbf{J}_i & -\mathbf{J}_a^T \mathbf{J}_a \end{array} \right) \quad (5.53)$$

Applying the Schur complement once, the combined solution for $(\Delta \mathbf{p}^T, \Delta \mathbf{q}^T)^T$ is given by:

$$\begin{aligned} \begin{pmatrix} -\mathbf{J}_i^T \bar{\mathbf{A}} \mathbf{J}_i & \mathbf{J}_i^T \bar{\mathbf{A}} \mathbf{J}_a \\ \mathbf{J}_a^T \bar{\mathbf{A}} \mathbf{J}_i & -\mathbf{J}_a^T \bar{\mathbf{A}} \mathbf{J}_a \end{pmatrix} \begin{pmatrix} \Delta \mathbf{p} \\ \Delta \mathbf{q} \end{pmatrix} &= \begin{pmatrix} \mathbf{J}_i^T \bar{\mathbf{A}} \\ -\mathbf{J}_a^T \bar{\mathbf{A}} \end{pmatrix} \mathbf{r}_b \\ \begin{pmatrix} \Delta \mathbf{p}^* \\ \Delta \mathbf{q}^* \end{pmatrix} &= \begin{pmatrix} -\mathbf{J}_i^T \bar{\mathbf{A}} \mathbf{J}_i & \mathbf{J}_i^T \bar{\mathbf{A}} \mathbf{J}_a \\ \mathbf{J}_a^T \bar{\mathbf{A}} \mathbf{J}_i & -\mathbf{J}_a^T \bar{\mathbf{A}} \mathbf{J}_a \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{J}_i^T \bar{\mathbf{A}} \\ -\mathbf{J}_a^T \bar{\mathbf{A}} \end{pmatrix} \mathbf{r}_b \end{aligned} \quad (5.54)$$

Note that the complexity of inverting this new approximation to the Hessian matrix is $O((2n)^3)^{16}$. Similar to before, plugging the solutions for $\Delta \mathbf{p}$ and $\Delta \mathbf{q}$ into equation 5.53 one can infer the optimal value for $\Delta \mathbf{c}$ using:

$$\Delta \mathbf{c}^* = \mathbf{A}^T (\mathbf{r}_b - \mathbf{J}_i \Delta \mathbf{p} + \mathbf{J}_a \Delta \mathbf{q}) \quad (5.55)$$

¹⁶This is an important reduction in complexity because for CGD algorithms usually $m \gg n$.

The total complexity per iteration of this approach is:

$$O(\underbrace{2nmF}_{\begin{pmatrix} \mathbf{J}_i^T \bar{\mathbf{A}} \\ -\mathbf{J}_a^T \bar{\mathbf{A}} \end{pmatrix}} + \underbrace{(2n)^2 F + (2n)^3}_{\begin{pmatrix} -\mathbf{J}_i^T \bar{\mathbf{A}} \mathbf{J}_i & \mathbf{J}_i^T \bar{\mathbf{A}} \mathbf{J}_a \\ \mathbf{J}_a^T \bar{\mathbf{A}} \mathbf{J}_i & -\mathbf{J}_a^T \bar{\mathbf{A}} \mathbf{J}_a \end{pmatrix}^{-1}}) \quad (5.56)$$

The Schur complement can be re-applied to equation 5.54 to derive a solution for $\Delta \mathbf{q}$ that only requires inverting a Hessian approximation matrix of size $n \times n$:

$$\begin{aligned} (\mathbf{J}_a^T \mathbf{P} \mathbf{J}_a) \Delta \mathbf{q} &= \mathbf{J}_a^T \mathbf{P} \mathbf{r}_b \\ \Delta \mathbf{q}^* &= (\mathbf{J}_a^T \mathbf{P} \mathbf{J}_a)^{-1} \mathbf{J}_a^T \mathbf{P} \mathbf{r}_b \end{aligned} \quad (5.57)$$

where the projection matrix $\mathbf{P}$ is defined as:

$$\mathbf{P} = \bar{\mathbf{A}} - \bar{\mathbf{A}} \mathbf{J}_i (\mathbf{J}_i^T \bar{\mathbf{A}} \mathbf{J}_i)^{-1} \mathbf{J}_i^T \bar{\mathbf{A}} \quad (5.58)$$

and the solutions for $\Delta \mathbf{p}$ and $\Delta \mathbf{c}$ can be obtained by plugging the solutions for $\Delta \mathbf{q}$ into equation 5.54 and the solutions for $\Delta \mathbf{q}$ and $\Delta \mathbf{p}$ into equation 5.53 respectively:

$$\begin{aligned} \Delta \mathbf{p}^* &= -(\mathbf{J}_i^T \bar{\mathbf{A}} \mathbf{J}_i)^{-1} \mathbf{J}_i^T \bar{\mathbf{A}} (\mathbf{r}_b - \mathbf{J}_a \Delta \mathbf{q}) \\ \Delta \mathbf{c}^* &= \mathbf{A}^T (\mathbf{r}_b + \mathbf{J}_i \Delta \mathbf{p} - \mathbf{J}_a \Delta \mathbf{q}) \end{aligned} \quad (5.59)$$

The total complexity per iteration of this approach reduces to:

$$O(\underbrace{2nmF}_{\mathbf{J}_a^T \mathbf{P} \& \mathbf{J}_i^T \bar{\mathbf{A}}} + \underbrace{2n^2 F + 2n^3}_{(\mathbf{J}_a^T \mathbf{P} \mathbf{J}_a)^{-1} \& (\mathbf{J}_i^T \bar{\mathbf{A}} \mathbf{J}_i)^{-1}}) \quad (5.60)$$

Note that because of their reduced complexity, the solutions defined by equations 5.57 and 5.59 are preferred over the ones defined by equations

5.54 and 5.55.

Finally, the solutions using the PO cost function are:

- For *asymmetric* composition:

$$\Delta \mathbf{p}^* = - (\mathbf{J}_t^T \bar{\mathbf{A}} \mathbf{J}_t)^{-1} \mathbf{J}_t^T \bar{\mathbf{A}} \mathbf{r} \quad (5.61)$$

with complexity¹⁷ given by equation 5.51.

- For *bidirectional* composition:

$$\begin{aligned} \Delta \mathbf{q}^* &= (\mathbf{J}_{\bar{\mathbf{a}}}^T \mathbf{P} \mathbf{J}_{\bar{\mathbf{a}}})^{-1} \mathbf{J}_{\bar{\mathbf{a}}}^T \mathbf{P} \mathbf{r} \\ \Delta \mathbf{p}^* &= - (\mathbf{J}_i^T \bar{\mathbf{A}} \mathbf{J}_i)^{-1} \mathbf{J}_i^T \bar{\mathbf{A}} (\mathbf{r} - \mathbf{J}_a \Delta \mathbf{q}) \end{aligned} \quad (5.62)$$

with complexity¹⁷ given by equation 5.60.

where, in both cases, $\mathbf{r} = \mathbf{i}[\mathbf{p}] - \bar{\mathbf{a}}$.

5.4.1.2 Alternated

Another way of solving optimization problems with two or more sets of variables is to use alternated optimization [93]. Hence, instead of solving the problem simultaneously with respect to all parameters, one can update one set of parameters at a time while keeping the other sets fixed.

More specifically, using *asymmetric* composition one can alternate between updating $\Delta \mathbf{c}$ given the $\Delta \mathbf{p}$ and then update $\Delta \mathbf{p}$ given the updated $\Delta \mathbf{c}$ in an

¹⁷In practice, the solutions for the PO cost function can be computed slightly faster than those for the SSD because they do not need to explicitly solve for $\Delta \mathbf{c}$. This is specially important in the *inverse* compositional case because expressions of the form $(\mathbf{J}^T \mathbf{U} \mathbf{J})^{-1} \mathbf{J}^T \mathbf{U}$ can be completely precomputed and the computational cost per iteration reduces to $O(nF)$.

alternate manner. Taking advantage of the structure of the problem defined by equation 5.48, the following system of equations can be obtained:

$$\begin{aligned} -\Delta\mathbf{c} + \mathbf{A}^T \mathbf{J}_t \Delta\mathbf{p} &= -\mathbf{A}^T \mathbf{r}_a \\ \mathbf{J}_t^T \mathbf{A} \Delta\mathbf{c} - \mathbf{J}_t^T \mathbf{J}_t \Delta\mathbf{p} &= \mathbf{J}_t^T \mathbf{r}_a \end{aligned} \quad (5.63)$$

which can be rewritten as:

$$\begin{aligned} \Delta\mathbf{c}^* &= \mathbf{A}^T (\mathbf{r}_a + \mathbf{J}_t \Delta\mathbf{p}) \\ \Delta\mathbf{p}^* &= -(\mathbf{J}_t^T \mathbf{J}_t)^{-1} \mathbf{J}_t^T (\mathbf{r}_a - \mathbf{A} \Delta\mathbf{c}) \end{aligned} \quad (5.64)$$

in order to obtain the analytical expression for the alternated update rules. The complexity at each iteration is dominated by:

$$O(\underbrace{n^2 F + n^3}_{(\mathbf{J}_t^T \mathbf{J}_t)^{-1}}) \quad (5.65)$$

In the case of *bidirectional* composition one can proceed in two different ways: *a*) update $\Delta\mathbf{c}$ given $\Delta\mathbf{p}$ and $\Delta\mathbf{q}$ and then update $(\Delta\mathbf{p}^T, \Delta\mathbf{q}^T)^T$ from the updated $\Delta\mathbf{c}$; or *b*) update $\Delta\mathbf{c}$ given $\Delta\mathbf{p}$ and $\Delta\mathbf{q}$, then $\Delta\mathbf{p}$ given the updated $\Delta\mathbf{c}$ and $\Delta\mathbf{q}$ and, finally, $\Delta\mathbf{q}$ given the updated $\Delta\mathbf{c}$ and $\Delta\mathbf{p}$.

From equation 5.53, one can derive the following system of equations:

$$\begin{aligned} -\Delta\mathbf{c} + \mathbf{A}^T \mathbf{J}_i \Delta\mathbf{p} - \mathbf{A}^T \mathbf{J}_a \Delta\mathbf{q} &= -\mathbf{A}^T \mathbf{r}_b \\ \mathbf{J}_i^T \mathbf{A} \Delta\mathbf{c} - \mathbf{J}_i^T \mathbf{J}_i \Delta\mathbf{p} + \mathbf{J}_i^T \mathbf{J}_a \Delta\mathbf{q} &= \mathbf{J}_i^T \mathbf{r}_b \\ -\mathbf{J}_a^T \mathbf{A} \Delta\mathbf{c} + \mathbf{J}_a^T \mathbf{J}_i \Delta\mathbf{p} - \mathbf{J}_a^T \mathbf{J}_a \Delta\mathbf{q} &= -\mathbf{J}_a^T \mathbf{r}_b \end{aligned} \quad (5.66)$$

from which the alternated update rules for the first of the two options can

be defined:

$$\begin{aligned} \Delta \mathbf{c}^* &= \mathbf{A}^T (\mathbf{r}_b + \mathbf{J}_i \Delta \mathbf{p} - \mathbf{J}_a \Delta \mathbf{q}) \\ \begin{pmatrix} \Delta \mathbf{p}^* \\ \Delta \mathbf{q}^* \end{pmatrix} &= \begin{pmatrix} -\mathbf{J}_i^T \mathbf{J}_i & \mathbf{J}_i^T \mathbf{J}_a \\ \mathbf{J}_a^T \mathbf{J}_i & -\mathbf{J}_a^T \mathbf{J}_a \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{J}_i^T \\ -\mathbf{J}_a^T \end{pmatrix} (\mathbf{r}_b - \mathbf{A} \Delta \mathbf{c}) \end{aligned} \quad (5.67)$$

with complexity:

$$\begin{aligned} O(\underbrace{(2n)^2 F + (2n)^3}_{\substack{(2n)^2 F \\ (2n)^3}}) \\ \begin{pmatrix} -\mathbf{J}_i^T \mathbf{J}_i & \mathbf{J}_i^T \mathbf{J}_a \\ \mathbf{J}_a^T \mathbf{J}_i & -\mathbf{J}_a^T \mathbf{J}_a \end{pmatrix}^{-1} \end{aligned} \quad (5.68)$$

The rules for the second option are:

$$\begin{aligned} \Delta \mathbf{c}^* &= \mathbf{A}^T (\mathbf{r}_b + \mathbf{J}_i \Delta \mathbf{p} - \mathbf{J}_a \Delta \mathbf{q}) \\ \Delta \mathbf{p}^* &= -(\mathbf{J}_i^T \mathbf{J}_i)^{-1} \mathbf{J}_i^T (\mathbf{r}_b - \mathbf{A} \Delta \mathbf{c} - \mathbf{J}_a \Delta \mathbf{q}) \\ \Delta \mathbf{q}^* &= (\mathbf{J}_a^T \mathbf{J}_a)^{-1} \mathbf{J}_a^T (\mathbf{r}_b - \mathbf{A} \Delta \mathbf{c} + \mathbf{J}_i \Delta \mathbf{p}) \end{aligned} \quad (5.69)$$

and their complexity is dominated by:

$$\begin{aligned} O(\underbrace{2n^2 F + 2n^3}_{\substack{(2n)^2 F \\ (2n)^3}}) \\ (\mathbf{J}_i^T \mathbf{J}_i)^{-1} \text{ \& } (\mathbf{J}_a^T \mathbf{J}_a)^{-1} \end{aligned} \quad (5.70)$$

On the other hand, the alternated update rules using the PO cost function are:

- For *asymmetric* composition: There is no proper alternated rule because the PO cost function only depends on one set of parameters, $\Delta \mathbf{p}$.

- For *bidirectional* composition:

$$\begin{aligned}\Delta \mathbf{q}^* &= (\mathbf{J}_{\bar{\mathbf{a}}}^T \bar{\mathbf{A}} \mathbf{J}_{\bar{\mathbf{a}}})^{-1} \mathbf{J}_{\bar{\mathbf{a}}}^T \bar{\mathbf{A}} (\mathbf{r} + \mathbf{J}_{\mathbf{i}} \Delta \mathbf{p}) \\ \Delta \mathbf{p}^* &= -(\mathbf{J}_{\mathbf{i}}^T \bar{\mathbf{A}} \mathbf{J}_{\mathbf{i}})^{-1} \mathbf{J}_{\mathbf{i}}^T \bar{\mathbf{A}} (\mathbf{r} - \mathbf{J}_{\mathbf{a}} \Delta \mathbf{q})\end{aligned}\tag{5.71}$$

with equivalent complexity to the one given by equation 5.51 because, in this case, the term $(\mathbf{J}_{\bar{\mathbf{a}}}^T \bar{\mathbf{A}} \mathbf{J}_{\bar{\mathbf{a}}})^{-1} \mathbf{J}_{\bar{\mathbf{a}}}^T \bar{\mathbf{A}}$ can be completely precomputed.

Note that all alternated update rules, equations 5.64, 5.67, 5.69 and 5.71, are similar but slightly different from their simultaneous counterparts, equations 5.49 and 5.50, 5.54 and 5.55, 5.57 and 5.59, and 5.62.

5.4.2 Newton

The Newton method performs a *second* order Taylor expansion of the entire data term $\mathcal{D}$:

$$\begin{aligned}\mathcal{D}(\Delta \ell) &\approx \hat{\mathcal{D}}(\Delta \ell) \\ &\approx \mathcal{D} + \frac{\partial \mathcal{D}}{\partial \Delta \ell} \Delta \ell + \frac{1}{2} \Delta \ell^T \frac{\partial^2 \mathcal{D}}{\partial \Delta \ell^2} \Delta \ell\end{aligned}\tag{5.72}$$

and solves the approximate problem:

$$\Delta \ell^* = \arg \min_{\Delta \ell} \hat{\mathcal{D}}\tag{5.73}$$

Assuming *asymmetric* composition, the data term is defined as:

$$\mathcal{D}_a(\Delta \ell) = \frac{1}{2} \mathbf{r}_a^T \mathbf{r}_a\tag{5.74}$$

and the matrix containing the first order partial derivatives with respect to the parameters, i.e. the data term's *Jacobian*, can be written as:

$$\begin{aligned}\frac{\partial \mathcal{D}_a}{\partial \Delta \ell} &= \left(\frac{\partial \mathcal{D}_a}{\partial \Delta \mathbf{c}}, \frac{\partial \mathcal{D}_a}{\partial \Delta \mathbf{p}} \right) \\ &= (-\mathbf{A}^T \mathbf{r}_a, \mathbf{J}_t^T \mathbf{r}_a)\end{aligned}\tag{5.75}$$

On the other hand, the matrix $\frac{\partial^2 \mathcal{D}_a}{\partial \Delta \ell^2}$ of the second order partial derivatives, i.e. the *Hessian* of the data term, takes the following form:

$$\begin{aligned}\frac{\partial^2 \mathcal{D}_a}{\partial \Delta \ell^2} &= \begin{pmatrix} \frac{\partial^2 \mathcal{D}_a}{\partial \Delta \mathbf{c}^2} & \frac{\partial^2 \mathcal{D}_a}{\partial \Delta \mathbf{c} \partial \Delta \mathbf{p}} \\ \frac{\partial^2 \mathcal{D}_a}{\partial \Delta \mathbf{p} \partial \Delta \mathbf{c}} & \frac{\partial^2 \mathcal{D}_a}{\partial \Delta \mathbf{p}^2} \end{pmatrix} \\ &= \begin{pmatrix} \frac{\partial^2 \mathcal{D}_a}{\partial \Delta \mathbf{c}^2} & \frac{\partial^2 \mathcal{D}_a}{\partial \Delta \mathbf{c} \partial \Delta \mathbf{p}} \\ \left(\frac{\partial^2 \mathcal{D}_a}{\partial \Delta \mathbf{c} \partial \Delta \mathbf{p}} \right)^T & \frac{\partial^2 \mathcal{D}_a}{\partial \Delta \mathbf{p}^2} \end{pmatrix}\end{aligned}\tag{5.76}$$

Note that the Hessian matrix is, by definition, symmetric. The definition of its individual terms is provided in appendix C.1.

A similar derivation can be obtained for *bidirectional* composition where, as expected, the data term is defined as:

$$\mathcal{D}_b(\Delta \ell) = \frac{1}{2} \mathbf{r}_b^T \mathbf{r}_b\tag{5.77}$$

In this case, the Jacobian matrix becomes:

$$\begin{aligned}\frac{\partial \mathcal{D}_b}{\partial \Delta \ell} &= \left(\frac{\partial \mathcal{D}_b}{\partial \Delta \mathbf{c}}, \frac{\partial \mathcal{D}_b}{\partial \Delta \mathbf{p}}, \frac{\partial \mathcal{D}_b}{\partial \Delta \mathbf{q}} \right) \\ &= (-\mathbf{A}^T \mathbf{r}_a, \mathbf{J}_i^T \mathbf{r}_a, -\mathbf{J}_a^T \mathbf{r}_a)\end{aligned}\tag{5.78}$$

and the Hessian matrix takes the following form:

$$\begin{aligned}
 \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \ell^2} &= \begin{pmatrix} \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{c}^2} & \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{c} \partial \Delta \mathbf{p}} & \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{c} \partial \Delta \mathbf{q}} \\ \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{p} \partial \Delta \mathbf{c}} & \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{p}^2} & \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{p} \partial \Delta \mathbf{q}} \\ \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{q} \partial \Delta \mathbf{c}} & \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{q} \partial \Delta \mathbf{p}} & \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{q}^2} \end{pmatrix} \\
 &= \begin{pmatrix} \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{c}^2} & \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{c} \partial \Delta \mathbf{p}} & \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{c} \partial \Delta \mathbf{q}} \\ \left(\frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{c} \partial \Delta \mathbf{p}} \right)^T & \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{p}^2} & \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{p} \partial \Delta \mathbf{q}} \\ \left(\frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{c} \partial \Delta \mathbf{q}} \right)^T & \left(\frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{p} \partial \Delta \mathbf{q}} \right)^T & \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{q}^2} \end{pmatrix}
 \end{aligned} \tag{5.79}$$

Notice that this matrix is again symmetric. The definition of its individual terms is provided in appendix C.2.

5.4.2.1 Simultaneous

Using the Newton method one can solve for all parameters simultaneously by equating the partial derivative of equation 5.73 to 0:

$$\begin{aligned}
 0 &= \frac{\partial \hat{\mathcal{D}}}{\partial \Delta \ell} \\
 &= \frac{\partial \left(\mathcal{D} + \frac{\partial \mathcal{D}}{\partial \Delta \ell} \Delta \ell + \frac{1}{2} \Delta \ell^T \frac{\partial^2 \mathcal{D}}{\partial^2 \Delta \ell} \Delta \ell \right)}{\partial \Delta \ell} \\
 &= \frac{\partial \mathcal{D}}{\partial \Delta \ell} + \frac{\partial^2 \mathcal{D}}{\partial \Delta \ell^2} \Delta \ell
 \end{aligned} \tag{5.80}$$

with the solution given by:

$$\Delta \ell^* = - \frac{\partial^2 \mathcal{D}}{\partial \Delta \ell^2}^{-1} \frac{\partial \mathcal{D}}{\partial \Delta \ell} \tag{5.81}$$

Note that, similar to the Gauss-Newton method, the complexity of inverting the Hessian matrix $\frac{\partial^2 \mathcal{D}}{\partial \Delta \ell^2}$ is $O((n + m)^3)$ for asymmetric composition and $O((2n + m)^3)$ for bidirectional composition. As shown by Kossaifi et al.

[41]¹⁸, one can take advantage of the structure of the Hessian in Equations 5.76 and 5.79 and apply the Schur complement to obtain more efficient solutions.

The solutions for $\Delta \mathbf{p}$ and $\Delta \mathbf{c}$ using *asymmetric* composition are given by the following expressions:

$$\begin{aligned}\Delta \mathbf{p}^* &= \left(\frac{\partial^2 \mathcal{D}_a}{\partial \Delta \mathbf{p}^2} - \frac{\partial^2 \mathcal{D}_a}{\partial \Delta \mathbf{p} \partial \Delta \mathbf{c}} \frac{\partial^2 \mathcal{D}_a}{\partial \Delta \mathbf{c} \partial \Delta \mathbf{p}} \right)^{-1} \left(\frac{\partial \mathcal{D}_a}{\partial \Delta \mathbf{p}} - \frac{\partial^2 \mathcal{D}_a}{\partial \Delta \mathbf{p} \partial \Delta \mathbf{c}} \frac{\partial \mathcal{D}_a}{\partial \Delta \mathbf{c}} \right) \\ \Delta \mathbf{c}^* &= \frac{\partial \mathcal{D}_a}{\partial \Delta \mathbf{c}} - \frac{\partial^2 \mathcal{D}_a}{\partial \Delta \mathbf{c} \partial \Delta \mathbf{p}} \Delta \mathbf{p}^*\end{aligned}\quad (5.82)$$

with complexity:

$$O\left(\underbrace{nmF}_{\frac{\partial^2 \mathcal{D}_a}{\partial \Delta \mathbf{p} \partial \Delta \mathbf{c}}} + \underbrace{n^2 m}_{\frac{\partial^2 \mathcal{D}_a}{\partial \Delta \mathbf{p} \partial \Delta \mathbf{c}} \frac{\partial^2 \mathcal{D}_a}{\partial \Delta \mathbf{c} \partial \Delta \mathbf{p}}} + \underbrace{2n^2 F}_{\frac{\partial^2 \mathcal{D}_a}{\partial \Delta \mathbf{p}^2}} + \underbrace{n^3}_{\mathbf{H}^{-1}} \right) \quad (5.83)$$

where $\mathbf{H} = \left(\frac{\partial^2 \mathcal{D}_a}{\partial \Delta \mathbf{p}^2} - \frac{\partial^2 \mathcal{D}_a}{\partial \Delta \mathbf{p} \partial \Delta \mathbf{c}} \frac{\partial^2 \mathcal{D}_a}{\partial \Delta \mathbf{c} \partial \Delta \mathbf{p}} \right)^{-1}$.

On the other hand, the solutions for *bidirectional* composition are given either by:

$$\begin{aligned}\begin{pmatrix} \Delta \mathbf{p}^* \\ \Delta \mathbf{q}^* \end{pmatrix} &= \begin{pmatrix} \mathbf{V} & \mathbf{W}^T \\ \mathbf{W} & \mathbf{U} \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{v} \\ \mathbf{u} \end{pmatrix} \\ \Delta \mathbf{c}^* &= \frac{\partial \mathcal{D}_b}{\partial \Delta \mathbf{c}} - \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{c} \partial \Delta \mathbf{p}} \Delta \mathbf{p} - \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{c} \partial \Delta \mathbf{q}} \Delta \mathbf{q}\end{aligned}\quad (5.84)$$

¹⁸In [41], Kossaifi et al. applied the Schur complement to the Newton method using *only* inverse composition while here it is applied using the more general *asymmetric* (which includes *forward*, *inverse* and *symmetric*) and *bidirectional* compositions.

or

$$\begin{aligned}
\Delta \mathbf{q}^* &= (\mathbf{U} - \mathbf{WV}^{-1}\mathbf{W}^T)^{-1} (\mathbf{u} - \mathbf{WV}^{-1}\mathbf{v}) \\
\Delta \mathbf{p}^* &= \mathbf{V}^{-1} (\mathbf{v} - \mathbf{W}^T \Delta \mathbf{q}) \\
\Delta \mathbf{c}^* &= \frac{\partial \mathcal{D}_b}{\partial \Delta \mathbf{c}} - \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{c} \partial \Delta \mathbf{p}} \Delta \mathbf{p} - \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{c} \partial \Delta \mathbf{q}} \Delta \mathbf{q}
\end{aligned} \tag{5.85}$$

where following auxiliary matrices:

$$\begin{aligned}
\mathbf{V} &= \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{p}^2} - \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{p} \partial \Delta \mathbf{c}} \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{c} \partial \Delta \mathbf{p}} \\
\mathbf{W} &= \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{q} \partial \Delta \mathbf{p}} - \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{q} \partial \Delta \mathbf{c}} \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{c} \partial \Delta \mathbf{p}} \\
\mathbf{U} &= \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{q}^2} - \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{q} \partial \Delta \mathbf{c}} \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{c} \partial \Delta \mathbf{q}}
\end{aligned} \tag{5.86}$$

and vectors:

$$\begin{aligned}
\mathbf{v} &= \frac{\partial \mathcal{D}_b}{\partial \Delta \mathbf{p}} - \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{p} \partial \Delta \mathbf{c}} \frac{\partial \mathcal{D}_b}{\partial \Delta \mathbf{c}} \\
\mathbf{u} &= \frac{\partial \mathcal{D}_b}{\partial \Delta \mathbf{q}} - \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{q} \partial \Delta \mathbf{c}} \frac{\partial \mathcal{D}_b}{\partial \Delta \mathbf{c}}
\end{aligned} \tag{5.87}$$

have been defined.

The complexity of the solutions is of:

$$\begin{aligned}
O(\underbrace{nmF}_{\mathbf{v}} + \underbrace{2nmF}_{\mathbf{u}} + \underbrace{4n^2F + 2n^2m}_{\mathbf{U} \& \mathbf{V}} + \\
\underbrace{2n^2F + n^2m}_{\mathbf{W}} + \underbrace{(2n)^3}_{\left(\begin{smallmatrix} \mathbf{V} & \mathbf{W}^T \\ \mathbf{W} & \mathbf{U} \end{smallmatrix} \right)^{-1}})
\end{aligned} \tag{5.88}$$

and

$$\begin{aligned}
O(\underbrace{nmF}_{\mathbf{v}} + \underbrace{2nmF}_{\mathbf{u}} + \underbrace{4n^2F + 2n^2m}_{\mathbf{U} \& \mathbf{V}} \\
+ \underbrace{2n^2F + n^2m}_{\mathbf{W}} + \underbrace{4n^3}_{\mathbf{V}^{-1} \& (\mathbf{U} - \mathbf{WV}^{-1}\mathbf{W}^T)^{-1}})
\end{aligned} \tag{5.89}$$

respectively.

The solutions using the PO cost function are:

- For *asymmetric* composition:

$$\Delta \mathbf{p}^* = - \left(\frac{\partial \mathcal{W}^T}{\Delta \mathbf{p}} \nabla^2 \mathbf{t} \frac{\partial \mathcal{W}}{\Delta \mathbf{p}} \bar{\mathbf{A}} \mathbf{r} + \mathbf{J}_{\mathbf{t}}^T \bar{\mathbf{A}} \mathbf{J}_{\mathbf{t}} \right)^{-1} \mathbf{J}_{\mathbf{t}}^T \bar{\mathbf{A}} \mathbf{r} \quad (5.90)$$

with complexity¹⁹ given by equation 5.83.

- For *bidirectional* composition:

$$\Delta \mathbf{q}^* = \left(\frac{\partial \mathcal{W}^T}{\Delta \mathbf{p}} \nabla^2 \bar{\mathbf{a}} \frac{\partial \mathcal{W}}{\Delta \mathbf{p}} \bar{\mathbf{A}} \mathbf{r} + \mathbf{J}_{\bar{\mathbf{a}}}^T \tilde{\mathbf{P}} \mathbf{J}_{\bar{\mathbf{a}}} \right)^{-1} \mathbf{J}_{\bar{\mathbf{a}}}^T \tilde{\mathbf{P}} \mathbf{r} \quad (5.91)$$

$$\Delta \mathbf{p}^* = -\mathbf{H}_{\mathbf{i}}^{-1} \mathbf{J}_{\mathbf{i}}^T \bar{\mathbf{A}} (\mathbf{r} - \mathbf{J}_{\mathbf{a}} \Delta \mathbf{q})$$

where the projection operator $\tilde{\mathbf{P}}$ is defined as:

$$\tilde{\mathbf{P}} = \bar{\mathbf{A}} - \bar{\mathbf{A}} \mathbf{J}_{\mathbf{i}}^T \mathbf{H}_{\mathbf{i}}^{-1} \mathbf{J}_{\mathbf{i}} \bar{\mathbf{A}}^T \quad (5.92)$$

and $\mathbf{H}_{\mathbf{i}} = \left(\frac{\partial \mathcal{W}^T}{\Delta \mathbf{p}} \nabla^2 \mathbf{i}[\mathbf{p}] \frac{\partial \mathcal{W}}{\Delta \mathbf{p}} \bar{\mathbf{A}} \mathbf{r} + \mathbf{J}_{\mathbf{i}}^T \bar{\mathbf{A}} \mathbf{J}_{\mathbf{i}} \right)$.

The complexity per iteration¹⁹ is given by equation 5.89.

Note that, the derivations of the solutions, for both types of composition, are analogous to the ones shown in Section 5.4.1.1 for the Gauss-Newton method and, consequently, have been omitted here.

¹⁹In practice, the solutions for the PO cost function can also be computed slightly faster because they do not need to explicitly solve for $\Delta \mathbf{c}$. However, in this case, using *inverse* composition one can only precompute terms of the form $\mathbf{J}^T \mathbf{U}$ and $\mathbf{J}^T \mathbf{U} \mathbf{J}$ but not the entire $\mathbf{H}^{-1} \mathbf{J}^T \mathbf{U}$ because of the explicit dependence between $\mathbf{H}$ and the current residual $\mathbf{r}$.

5.4.2.2 Alternated

Alternated optimization rules can also be derived for the Newton method following the strategy shown in section 5.4.2.2 for the Gauss-Newton case. Again, only the update rules and computational complexity for both types of composition are provided and the details of their full derivation omitted.

For *asymmetric* composition the alternated rules are defined as:

$$\begin{aligned}\Delta \mathbf{c}^* &= \frac{\partial \mathcal{D}_a}{\partial \Delta \mathbf{c}} - \frac{\partial^2 \mathcal{D}_a}{\partial \Delta \mathbf{c} \partial \Delta \mathbf{p}} \Delta \mathbf{p} \\ \Delta \mathbf{p}^* &= \frac{\partial^2 \mathcal{D}_a}{\partial \Delta \mathbf{p}^2}^{-1} \left(\frac{\partial \mathcal{D}_a}{\partial \Delta \mathbf{p}} - \frac{\partial^2 \mathcal{D}_a}{\partial \Delta \mathbf{p} \partial \Delta \mathbf{c}} \Delta \mathbf{c} \right)\end{aligned}\tag{5.93}$$

with complexity:

$$O\left(\underbrace{nmF}_{\frac{\partial^2 \mathcal{D}_a}{\partial \Delta \mathbf{p} \partial \Delta \mathbf{c}}} + \underbrace{2n^2F + n^3}_{\frac{\partial^2 \mathcal{D}_a}{\partial \Delta \mathbf{p}^2}^{-1}}\right)\tag{5.94}$$

The alternated rules for *bidirectional* composition case are given either by:

$$\begin{aligned}\Delta \mathbf{c}^* &= \frac{\partial \mathcal{D}_b}{\partial \Delta \mathbf{c}} - \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{c} \partial \Delta \mathbf{p}} \Delta \mathbf{p} - \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{c} \partial \Delta \mathbf{q}} \Delta \mathbf{q} \\ \begin{pmatrix} \Delta \mathbf{p}^* \\ \Delta \mathbf{q}^* \end{pmatrix} &= \begin{pmatrix} \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{p}^2} & \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{p} \partial \Delta \mathbf{q}} \\ \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{q} \partial \Delta \mathbf{p}} & \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{q}^2} \end{pmatrix}^{-1} \begin{pmatrix} \frac{\partial \mathcal{D}_b}{\partial \Delta \mathbf{p}} - \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{p} \partial \Delta \mathbf{c}} \Delta \mathbf{c} \\ \frac{\partial \mathcal{D}_b}{\partial \Delta \mathbf{q}} - \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{q} \partial \Delta \mathbf{c}} \Delta \mathbf{c} \end{pmatrix}\end{aligned}\tag{5.95}$$

with complexity:

$$O\left(\underbrace{nmF}_{\frac{\partial^2 \mathcal{D}}{\partial \Delta \mathbf{p} \partial \Delta \mathbf{p}}} + \underbrace{4n^2F}_{\frac{\partial^2 \mathcal{D}}{\partial \Delta \mathbf{p}^2} \& \frac{\partial^2 \mathcal{D}}{\partial \Delta \mathbf{q}^2}} + \underbrace{(2n)^3}_{\begin{pmatrix} \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{p}^2} & \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{p} \partial \Delta \mathbf{q}} \\ \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{q} \partial \Delta \mathbf{p}} & \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{q}^2} \end{pmatrix}^{-1}}\right)\tag{5.96}$$

or:

$$\begin{aligned}
\Delta \mathbf{c}^* &= \frac{\partial \mathcal{D}_b}{\partial \Delta \mathbf{c}} - \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{c} \partial \Delta \mathbf{p}} \Delta \mathbf{p} - \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{c} \partial \Delta \mathbf{q}} \Delta \mathbf{q} \\
\Delta \mathbf{p}^* &= \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{p}^2}^{-1} \left(\frac{\partial \mathcal{D}_b}{\partial \Delta \mathbf{p}} - \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{p} \partial \Delta \mathbf{c}} \Delta \mathbf{c} - \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{p} \partial \Delta \mathbf{q}} \Delta \mathbf{q} \right) \\
\Delta \mathbf{q}^* &= \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{q}^2}^{-1} \left(\frac{\partial \mathcal{D}_b}{\partial \Delta \mathbf{q}} - \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{q} \partial \Delta \mathbf{c}} \Delta \mathbf{c} - \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{q} \partial \Delta \mathbf{p}} \Delta \mathbf{p} \right)
\end{aligned} \tag{5.97}$$

with complexity:

$$O \left(\underbrace{nmF}_{\frac{\partial^2 \mathcal{D}}{\partial \Delta \mathbf{p} \partial \Delta \mathbf{p}}} + \underbrace{4n^2 F}_{\frac{\partial^2 \mathcal{D}}{\partial \Delta \mathbf{p}^2} \& \frac{\partial^2 \mathcal{D}}{\partial \Delta \mathbf{q}^2}} + \underbrace{2n^3}_{\frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{p}^2}^{-1} \& \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{q}^2}^{-1}} \right) \tag{5.98}$$

On the other hand, the alternated update rules for the Newton method using the PO cost function are:

- For *asymmetric* composition: Again, there is no proper alternated rule because the project-out cost function only depends on one set of parameters, $\Delta \mathbf{p}$.
- For *bidirectional* composition:

$$\begin{aligned}
\Delta \mathbf{q}^* &= \mathbf{H}_{\mathbf{a}}^{-1} \mathbf{J}_{\mathbf{a}}^T \bar{\mathbf{A}} (\mathbf{r} + \mathbf{J}_{\mathbf{i}} \Delta \mathbf{p}) \\
\Delta \mathbf{p}^* &= -\mathbf{H}_{\mathbf{i}}^{-1} \mathbf{J}_{\mathbf{i}}^T \bar{\mathbf{A}} (\mathbf{r} - \mathbf{J}_{\mathbf{a}} \Delta \mathbf{q})
\end{aligned} \tag{5.99}$$

where $\mathbf{H}_{\mathbf{a}} = \left(\frac{\partial \mathcal{W}}{\partial \Delta \mathbf{p}}^T \nabla^2 \bar{\mathbf{a}} \frac{\partial \mathcal{W}}{\partial \Delta \mathbf{p}} \bar{\mathbf{A}} \mathbf{r} + \mathbf{J}_{\mathbf{a}}^T \bar{\mathbf{A}} \mathbf{J}_{\mathbf{a}} \right)$.

The complexity at every iteration is given by the following expression complexity:

$$O \left(\underbrace{nmF}_{\mathbf{J}_{\mathbf{i}}^T \bar{\mathbf{A}}} + \underbrace{3n^2 F + 2n^3}_{\mathbf{H}_{\mathbf{i}}^{-1} \& \mathbf{H}_{\mathbf{a}}^{-1}} \right) \tag{5.100}$$

Note that *Newton* algorithms are true *second* order optimizations algorithms

with respect to the incremental warps. However, as shown in this section, this property comes at expenses of a significant increase in computational complexity with respect to (*first order*) *Gauss-Newton* algorithms. Appendix B shows that some of the *Gauss-Newton* algorithms derived in section 5.4.1, i.e. the *asymmetric Gauss-Newton* algorithms, are, in fact, true *efficient second order minimization* (ESM) algorithms that effectively circumvent the increase in computational complexity.

5.4.3 Wiberg

The idea behind the Wiberg method is similar to the one used by the alternated Gauss-Newton method in section 5.4.2.2, i.e. solving for one set of parameters at a time while keeping the other sets fixed. However, Wiberg does so by rewriting the asymmetric $\mathbf{r}_a(\Delta\mathbf{c}, \Delta\mathbf{p})$ and bidirectional $\mathbf{r}_b(\Delta\mathbf{c}, \Delta\mathbf{p}, \Delta\mathbf{q})$ residuals as functions that only depend on $\Delta\mathbf{p}$ and $\Delta\mathbf{q}$ respectively.

For *asymmetric* composition, the residual $\bar{\mathbf{r}}_a(\Delta\mathbf{p})$ is defined as follows:

$$\begin{aligned}\bar{\mathbf{r}}_a(\Delta\mathbf{p}) &= \mathbf{r}_a(\bar{\Delta\mathbf{c}}, \Delta\mathbf{p}) \\ &= \mathbf{i}[\mathbf{p} \circ \alpha\Delta\mathbf{p}] - (\mathbf{a} + \mathbf{A}(\mathbf{c} + \bar{\Delta\mathbf{c}}_a))[\beta\Delta\mathbf{p}]\end{aligned}\tag{5.101}$$

where the function $\bar{\Delta\mathbf{c}}_a(\Delta\mathbf{p})$ is obtained by solving for $\Delta\mathbf{c}$ while keeping $\Delta\mathbf{p}$ fixed:

$$\bar{\Delta\mathbf{c}}_a(\Delta\mathbf{p}) = \mathbf{A}^T \mathbf{r}_a\tag{5.102}$$

Given this residual, the Wiberg method proceeds to define the following

optimization problem with respect to $\Delta \mathbf{p}$:

$$\Delta \mathbf{p}^* = \arg \min_{\Delta \mathbf{p}} \bar{\mathbf{r}}_a^T \bar{\mathbf{r}}_a \quad (5.103)$$

which then solves approximately by performing a first order Taylor of the residual around the incremental warp:

$$\Delta \mathbf{p}^* = \arg \min_{\Delta \mathbf{p}} \left\| \bar{\mathbf{r}}_a(\Delta \mathbf{p}) + \frac{\partial \bar{\mathbf{r}}_a}{\partial \Delta \mathbf{p}} \Delta \mathbf{p} \right\|^2 \quad (5.104)$$

In this case, the Jacobian $\frac{\partial \bar{\mathbf{r}}}{\partial \Delta \mathbf{p}}$ can be obtain by direct application of the *chain rule* and it is defined as follows:

$$\begin{aligned} \frac{d\bar{\mathbf{r}}_a}{d\Delta \mathbf{p}} &= \frac{\partial \bar{\mathbf{r}}_a}{\partial \Delta \mathbf{p}} + \frac{\partial \bar{\mathbf{r}}_a}{\partial \bar{\Delta \mathbf{c}}_a} \frac{\partial \bar{\Delta \mathbf{c}}_a}{\partial \Delta \mathbf{p}} \\ &= \mathbf{J}_t - \mathbf{A} \mathbf{A}^T \mathbf{J}_t \\ &= \bar{\mathbf{A}} \mathbf{J}_t \end{aligned} \quad (5.105)$$

The solution for $\Delta \mathbf{p}$ is obtained as usual by equating the derivative of 5.103 with respect to $\Delta \mathbf{p}$ to 0:

$$\begin{aligned} \Delta \mathbf{p}^* &= - \left((\bar{\mathbf{A}} \mathbf{J}_t)^T \bar{\mathbf{A}} \mathbf{J}_t \right)^{-1} (\bar{\mathbf{A}} \mathbf{J}_t)^T \bar{\mathbf{r}}_a \\ &= - (\mathbf{J}_t^T \bar{\mathbf{A}} \mathbf{J}_t)^{-1} \mathbf{J}_t^T \bar{\mathbf{A}} \bar{\mathbf{r}}_a \end{aligned} \quad (5.106)$$

where the fact that the matrix $\bar{\mathbf{A}}$ is idempotent has been used²⁰.

²⁰ $\bar{\mathbf{A}}$ is idempotent:

$$\begin{aligned} \bar{\mathbf{A}} \bar{\mathbf{A}} &= (\mathbf{I} - \mathbf{A} \mathbf{A}^T) (\mathbf{I} - \mathbf{A} \mathbf{A}^T) \\ &= \mathbf{I}^T \mathbf{I} - 2 \mathbf{A} \mathbf{A}^T + \underbrace{\mathbf{A} \mathbf{A}^T \mathbf{A} \mathbf{A}^T}_{\mathbf{I}} \\ &= \mathbf{I} - 2 \mathbf{A} \mathbf{A}^T + \mathbf{A} \mathbf{A}^T \\ &= \mathbf{I} - \mathbf{A} \mathbf{A}^T \\ &= \bar{\mathbf{A}} \end{aligned}$$

Therefore, the Wiberg method solves explicitly, at each iteration, for $\Delta \mathbf{p}$ using the previous expression and implicitly for $\Delta \mathbf{c}$ (through $\bar{\Delta} \mathbf{c}_a(\Delta \mathbf{p})$) using equation 5.102. The complexity per iteration of the Wiberg method is the same as the one of the Gauss-Newton method after applying the Schur complement, equation 5.51. In fact, note that the Wiberg solution for $\Delta \mathbf{p}$ (equation 5.106) is the same as the one of the Gauss-Newton method after applying the Schur complement, equation 5.49; and also note the similarity between the solutions for $\Delta \mathbf{c}$ of both methods, equations 5.102 and 5.50. Finally, note that, due to the close relation between the *Wiberg* and *Gauss-Newton* methods, *Asymmetric Wiberg* algorithms are also ESM algorithms for fitting AAM.

On the other hand, for *bidirectional* composition, the residual $\bar{\mathbf{r}}_b(\Delta \mathbf{p})$ is defined as:

$$\begin{aligned} \bar{\mathbf{r}}_b(\Delta \mathbf{q}) &= \mathbf{r}_b(\bar{\Delta} \mathbf{c}_b, \bar{\Delta} \mathbf{p}_b, \Delta \mathbf{q}) \\ &= \mathbf{i}[\mathbf{p} \circ \bar{\Delta} \mathbf{p}_b] - (\mathbf{a} - \mathbf{A}(\mathbf{c} + \bar{\Delta} \mathbf{c}_b))[\Delta \mathbf{q}] \end{aligned} \quad (5.107)$$

where, similarly as before, the function $\bar{\Delta} \mathbf{c}_b(\Delta \mathbf{p}, \Delta \mathbf{q})$ is obtained solving for $\Delta \mathbf{c}$ while keeping both $\Delta \mathbf{p}$ and $\Delta \mathbf{q}$ fixed:

$$\bar{\Delta} \mathbf{c}_b(\Delta \mathbf{p}, \Delta \mathbf{q}) = \mathbf{A}^T \mathbf{r}_b \quad (5.108)$$

and the function $\bar{\Delta} \mathbf{p}_b(\bar{\Delta} \mathbf{c}_b, \Delta \mathbf{q})$ is obtained by solving for $\Delta \mathbf{p}$ using the Wiberg method while keeping $\Delta \mathbf{q}$ fixed:

$$\bar{\Delta} \mathbf{p}_b(\bar{\Delta} \mathbf{c}_b, \Delta \mathbf{q}) = -(\mathbf{J}_i^T \bar{\mathbf{A}} \mathbf{J}_i)^{-1} \mathbf{J}_i^T \bar{\mathbf{A}} \bar{\mathbf{r}}_b \quad (5.109)$$

At this point, the Wiberg method proceeds to define the following optimiza-

tion problem with respect to $\Delta \mathbf{q}$:

$$\Delta \mathbf{q}^* = \arg \min_{\Delta \mathbf{q}} \bar{\mathbf{r}}_b^T \bar{\mathbf{r}}_b \quad (5.110)$$

which, as before, then solves approximately by performing a first order Taylor expansion around $\Delta \mathbf{q}$:

$$\Delta \mathbf{q}^* = \arg \min_{\Delta \mathbf{q}} \left\| \bar{\mathbf{r}}_b(\Delta \mathbf{q}) + \frac{\partial \bar{\mathbf{r}}_b}{\partial \Delta \mathbf{q}} \Delta \mathbf{q} \right\|^2 \quad (5.111)$$

In this case, the Jacobian of the residual can also be obtained by direct application of the chain rule and takes the following form:

$$\begin{aligned} \frac{d\bar{\mathbf{r}}_b}{d\Delta \mathbf{q}} &= \frac{\partial \bar{\mathbf{r}}_b}{\partial \Delta \mathbf{q}} + \frac{\partial \bar{\mathbf{r}}_b}{\partial \Delta \bar{\mathbf{p}}_b} \frac{\partial \Delta \bar{\mathbf{p}}_b}{\partial \Delta \mathbf{q}} + \left(\frac{\partial \bar{\mathbf{r}}_b}{\partial \Delta \bar{\mathbf{c}}_b} + \frac{\partial \bar{\mathbf{r}}_b}{\partial \Delta \bar{\mathbf{p}}_b} \frac{\partial \Delta \bar{\mathbf{p}}_b}{\partial \Delta \bar{\mathbf{c}}} \right) \frac{\partial \Delta \bar{\mathbf{c}}_b}{\partial \Delta \mathbf{q}} \\ &= -\mathbf{J}_a + \mathbf{J}_i (\mathbf{J}_i^T \bar{\mathbf{A}} \mathbf{J}_i)^{-1} \mathbf{J}_i^T \bar{\mathbf{A}} \mathbf{J}_a \\ &\quad + \left(\mathbf{A} - \mathbf{J}_i (\mathbf{J}_i^T \bar{\mathbf{A}} \mathbf{J}_i)^{-1} \mathbf{J}_i^T \bar{\mathbf{A}} \mathbf{A} \right) \mathbf{A}^T \mathbf{J}_a \\ &= -\mathbf{J}_a + \mathbf{A} \mathbf{A}^T \mathbf{J}_a + \mathbf{J}_i (\mathbf{J}_i^T \bar{\mathbf{A}} \mathbf{J}_i)^{-1} \mathbf{J}_i^T \bar{\mathbf{A}} \mathbf{J}_a - \\ &\quad \mathbf{J}_i (\mathbf{J}_i^T \bar{\mathbf{A}} \mathbf{J}_i)^{-1} \mathbf{J}_i^T \bar{\mathbf{A}} \mathbf{A} \mathbf{A}^T \mathbf{J}_a \\ &= -(\mathbf{I} - \mathbf{A} \mathbf{A}^T) \mathbf{J}_a + \mathbf{J}_i (\mathbf{J}_i^T \bar{\mathbf{A}} \mathbf{J}_i)^{-1} \mathbf{J}_i^T \bar{\mathbf{A}} (\mathbf{I} - \mathbf{A} \mathbf{A}^T) \mathbf{J}_a \\ &= -\bar{\mathbf{A}} \mathbf{J}_a + \mathbf{J}_i (\mathbf{J}_i^T \bar{\mathbf{A}} \mathbf{J}_i)^{-1} \mathbf{J}_i^T \bar{\mathbf{A}} \bar{\mathbf{A}} \mathbf{J}_a \\ &= \left(-\mathbf{I} + \mathbf{J}_i (\mathbf{J}_i^T \bar{\mathbf{A}} \mathbf{J}_i)^{-1} \mathbf{J}_i^T \bar{\mathbf{A}} \right) \bar{\mathbf{A}} \mathbf{J}_a \\ &= -\mathbf{P} \mathbf{J}_a \end{aligned} \quad (5.112)$$

And, again, the solution for $\Delta \mathbf{q}$ is obtained as usual by equating the derivative of equation 5.111 with respect to $\Delta \mathbf{q}$ to 0:

$$\Delta \mathbf{q}^* = \left((\mathbf{P} \mathbf{J}_t)^T \mathbf{P} \mathbf{J}_t \right)^{-1} (\mathbf{P} \mathbf{J}_t)^T \bar{\mathbf{r}}_a \quad (5.113)$$

In this case, the Wiberg method solves explicitly, at each iteration, for

$\Delta \mathbf{p}$ using the previous expression and implicitly for $\Delta \mathbf{p}$ and $\Delta \mathbf{c}$ (through $\bar{\Delta} \mathbf{p}_b(\bar{\Delta} \mathbf{c}_b, \Delta \mathbf{q})$ and $\bar{\Delta} \mathbf{c}_b(\Delta \mathbf{p}, \Delta \mathbf{q})$) using equations 5.109 and 5.108 respectively. Again, the complexity per iteration is the same as the one of the Gauss-Newton method after applying the Schur complement, equation 5.60; and the solutions for both methods are almost identical, equations 5.113, 5.109 and 5.108 and Equations 5.54, 5.55 and 5.57.

On the other hand, the Wiberg solutions for the PO cost function are:

- For *asymmetric* composition: Because the PO cost function only depends on one set of parameters, $\Delta \mathbf{p}$, in this case Wiberg reduces to Gauss-Newton.
- For *bidirectional* composition:

$$\begin{aligned}\Delta \mathbf{p}^* &= -(\mathbf{J}_i^T \bar{\mathbf{A}} \mathbf{J}_i)^{-1} \mathbf{J}_i^T \bar{\mathbf{A}} \mathbf{r} \\ \Delta \mathbf{q}^* &= (\mathbf{J}_a^T \mathbf{P} \mathbf{J}_a)^{-1} \mathbf{J}_a^T \mathbf{P} \mathbf{r}\end{aligned}\tag{5.114}$$

Again, in this case, the solutions obtained with the Wiberg method are almost identical to the ones obtained using Gauss-Newton after applying the Schur complement, equation 5.62.

5.5 Relation to prior work

In this section we relate relevant prior work on CGD algorithms for fitting AAM [10, 32, 34, 37, 38, 40, 41] to the unified and complete view introduced in the previous sections.

5.5.1 Project-out algorithms

In their seminal work [10], Matthews and Baker proposed the first CGD algorithm for fitting AAM, the so-called *project-out inverse compositional* (PIC) algorithm. This algorithm uses Gauss-Newton to solve the optimization problem posed by the project-out cost function using inverse composition. The use of the project-out norm removes the need to solve for the texture parameters and the use of inverse composition allows for the precomputation of the pseudo-inverse of the Jacobian with respect to $\Delta \mathbf{p}$, i.e. $(\mathbf{J}_{\bar{\mathbf{a}}}^T \bar{\mathbf{A}} \mathbf{J}_{\bar{\mathbf{a}}})^{-1} \mathbf{J}_{\bar{\mathbf{a}}} \bar{\mathbf{A}}$. The PIC algorithm is very efficient ($O(nF)$) but it has been shown to perform poorly in generic and unconstrained scenarios [32, 34]. Using the framework proposed in this chapter, this algorithm is referred to as the *PO inverse Gauss-Newton* algorithm.

The forward version of this algorithm, i.e. the *PO forward Gauss-Newton* algorithm, was proposed by Amberg et al. in [37]. In this case, the use of forward composition prevents the precomputation of the Jacobian pseudo-inverse and its complexity increases to $O(nmF + n^2F + n^3)$. However, this algorithm has been shown to largely outperform its inverse counterpart, and obtains good performance under generic and unconstrained conditions [37, 40]²¹

The rest of PO algorithms derived in this chapter, i.e.:

- *PO forward Newton*
- *PO inverse Newton*

²¹Notice that, in [37], Amberg et al. also introduced a hybrid forward/inverse algorithm, coined CoLiNe. This algorithm is a compromise between the previous two algorithms in terms of both complexity and accuracy. Due to its rather ad-hoc derivation, this algorithm was not considered in this paper.

- *PO asymmetric Gauss-Newton*
- *PO asymmetric Newton*
- *PO bidirectional Gauss-Newton Schur*
- *PO bidirectional Gauss-Newton Alternated*
- *PO bidirectional Newton Schur*
- *PO bidirectional Newton Alternated*
- *PO bidirectional Wiberg*

have never been published before and are a significant contribution of this thesis.

5.5.2 SSD algorithms

In [32] Gross et al. presented the *simultaneous inverse compositional* (SIC) algorithm and show that it largely outperforms the PIC algorithm in terms of fitting accuracy. This algorithm uses Gauss-Newton to solve the optimization problem posed by the SSD cost function using inverse composition. In this case, the Jacobian with respect to $\Delta \mathbf{p}$, depends on the current value of the texture parameters and needs to be recomputed at every iteration. Moreover, the inclusion of the Jacobian with respect to the texture increments $\Delta \mathbf{c}$, increases the size of the simultaneous Jacobian to $\frac{\partial \mathbf{r}}{\partial \Delta \ell} = (-\mathbf{A}, -\mathbf{J}_{\mathbf{a}}) \in \mathbb{R}^{F \times (m+n)}$ and, consequently, the computational cost per iteration of the algorithm is $O((m+n)^2 F + (m+n)^3)$.

As shown in sections 5.4.1.1, 5.4.2.2 and 5.4.3 this complexity can be dramatically reduced by taking advantage of the problem structure in order to

derive more efficient and exact algorithm by: *a)* applying the Schur complement; *b)* adopting an alternated optimization approach; or *c)* using the Wiberg method. Papandreou and Maragos [34] proposed an algorithm that is equivalent to the solution obtained by applying the Schur complement to the problem, as described in section 5.4.1.1. The same algorithm was reintroduced in [40] using a somehow ad-hoc derivation (reminiscent of the Wiberg method) under the name Fast-SIC. This algorithm has a computational cost per iteration of $O(nmF + n^2F + n^3)$. Following the unified view on CGD algorithm introduced in this chapter this algorithm is referred to as the *SSD inverse Gauss-Newton Schur* algorithm. The alternated optimization approach was used in [51] and [57] with complexity $O(n^2F + n^3)$ per iteration. In this chapter, this algorithm is referred to as the *SSD inverse Gauss-Newton Alternated* algorithm.

On the other hand, the forward version of this algorithm was first proposed by Martins et al. in [38]²². In this case, the Jacobian with respect to $\Delta \mathbf{p}$ depends on the current value of the shape parameters $\mathbf{p}$ through the warped image $\mathbf{i}[\mathbf{p}]$ and also needs to be recomputed at every iteration. Consequently, the complexity of the algorithm is the same as in the naive inverse approach of Gross et al. In this chapter, this algorithm is referred to as the *SSD forward Gauss-Newton* algorithm. It is important to notice that Tzimiropoulos and Pantic [40] derived a more efficient version of this algorithm ($O(nmF + n^2F + n^3)$), coined Fast-Forward, by applying the same derivation used to obtain their Fast-SIC algorithm. They showed that in the forward case their derivation removed the need to explicitly solve for the texture parameters. Their algorithm is equivalent to the *PO forward*

²²Note that Martins et al. used an additive update rule for the shape parameters, $\mathbf{p}^* = \mathbf{p} + \Delta \mathbf{p}$, so strictly speaking they derived an additive version of the algorithm i.e the *Simultaneous Forward Additive* (SFA) algorithm.

Gauss-Newton.

Finally, Kossaifi et al. derived the *SSD inverse Newton Schur* algorithm in [41]. This algorithm has a total complexity per iteration of $O(nmF + n^2m + 2n^2F + n^3)$ and was shown to slightly underperform its equivalent Gauss-Newton counterpart.

The remaining SSD algorithms derived in chapter, i.e.:

- *SSD Inverse Wiberg*
- *SSD Forward Gauss-Newton Alternated*
- *SSD Forward Newton Schur*
- *SSD Forward Newton Alternated*
- *SSD Forward Wiberg*
- *SSD Asymmetric Gauss-Newton Schur*
- *SSD Asymmetric Gauss-Newton Alternated*
- *SSD Asymmetric Newton Schur*
- *SSD Asymmetric Newton Alternated*
- *SSD Asymmetric Wiberg*
- *SSD Bidirectional Gauss-Newton Schur*
- *SSD Bidirectional Gauss-Newton Alternated*
- *SSD Bidirectional Newton Schur*
- *SSD Bidirectional Newton Alternated*

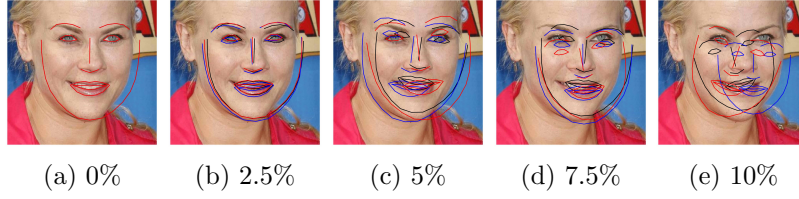


Figure 5.2: Exemplar initializations obtained by varying the percentage of uniform noise added to the similarity parameters. Note that, increasing the percentage of noise produces more challenging initialization.

- *SSD Bidirectional Wiberg*

have never been published before and are also a key contribution of this thesis.

Note that the iterative solutions of all CGD algorithms studied in this chapter are given in appendix [D](#).

5.6 Results

This section analyzes the performance of the CGD algorithms derived in this chapter on the specific problem of non-rigid face alignment in-the-wild. Results for five experiments are reported. The first experiment compares the fitting accuracy and convergence properties of all algorithms on the test set of the popular *labeled faces parts in-the-wild* (LFPW) [\[1\]](#) dataset. The second experiment quantifies the importance of the two terms in the Bayesian project-out cost function in relation to the fitting accuracy obtained by project-out algorithms. The effect that varying the value of the parameters α and β has on the performance of asymmetric algorithms is reported in the third experiment. The fourth experiment explores the effect of optimizing the cost functions using reduced subsets of the total number of

pixels and quantifies the impact that this has on the accuracy and computational efficiency of CGD algorithms. Finally, the fifth experiment reports the performance of the most accurate CGD algorithms on the test images of the *Helen* [14] dataset and on the entire *annotated faces in-the-wild* (AFW) [4] dataset.

Throughout this section, CGD algorithms are abbreviated using the following convention: CF_TC_OM(_OS) where: *a*) CF stands for *cost function* and can be either SSD or PO depending on whether the algorithm uses the *sum of squared differences* or the *project-out* cost function; *b*) TC stands for *type of composition* and can be For, Inv, Asy or Bid depending on whether the algorithm uses *forward*, *inverse*, *asymmetric* or *bidirectional* compositions; *c*) OM stands for *optimization method* and can be GN, N or W depending on whether the algorithm uses the *Gauss-Newton*, *Newton* or *Wiberg* optimization methods; and, finally, *d*) if Gauss-Newton or Newton methods are used, the optional field OS, which stands for *optimization strategy*, can be Sch or Alt depending on whether the algorithm solves for the parameters simultaneously using the *Schur complement* or using *alternated optimization*. For example, following this convention the *project out bidirectional Gauss-Newton Schur* algorithm is denoted by PO_Bid_GN_Sch.

Landmark annotations for all databases are provided by the iBUG group²³ [2, 3] and fitting accuracy is reported using the point-to-point error measure normalized by the *face size* proposed in [4], defined in equation 4.28, over the 49 interior points of the iBUG annotation scheme.

The same AAM, trained using the ~ 800 and ~ 2000 training images of the LFPW and Helen datasets is used for all experiments. Similar to [94],

²³<http://ibug.doc.ic.ac.uk/resources/300-W/>

a modified version of the *dense scale invariant feature transform* (DSIFT) [58, 95] is used to define its texture representation²⁴. In particular, each pixel is described with a reduced SIFT descriptor of length 8 using the public implementation provided by the authors of [96]. All algorithms are implemented in a coarse to fine manner using a *Gaussian pyramid* with 2 levels (face images are normalized to have a face size⁷ of roughly 150 pixels at the top level). In all experiments, 7 shape parameters (4 similarity transform and 3 non-rigid shape parameters) are optimized in the first pyramid level and 16 (4 similarity transform and 12 non-rigid shape parameters) in the second one. The dimensionality of the texture models is kept to represent 75% of the total variance in both levels; this results in 225 and 280 texture parameters at the first and second pyramid levels respectively. This choices were determined by testing on a small validation set of the training data.

In all experiments, algorithms are initialized by perturbing the similarity transform that perfectly aligns the model’s mean shape (all shape parameter controlling non-rigid deformations are set to 0) with the ground truth shape of each image. These transforms are perturbed by adding uniformly distributed random noise to their scale, rotation and translation parameters²⁵. Exemplar initializations obtained by this procedure for different amounts of noise are shown in figure 5.2. Note that, initializing using 5% uniform noise is approximately equivalent (in a statistical sense) to initializ-

²⁴Note that the work comprised in this thesis spans over a period of four years. The robust IGO texture representation presented in chapter 4 was state-of-the-art when it was originally proposed at the beginning of this period. However, texture representations based on highly engineer non-linear feature descriptors such as SIFT or HOG were latter shown to produce more accurate results in [57, 94] and that is why they are used in here.

²⁵The percentage of added noise is relative to a factor of 2 change in scale, 0.25 in rotation and 0.5 in translation (a factor of 2 change is equivalent to either doubling or halving the original face size; a factor of 0.25 change in rotation is equivalent to rotating the face by either 45 or -45 degrees; and a factor of 0.5 change in translation is equivalent to translating the face by half of its size).

ing with the popular OpenCV [97] implementation of the well-known Viola and Jones face detector [98] on the test images of the LFPW database.

Finally, unless stated otherwise: *i*) algorithms are initialized with 5% uniform noise; *ii*) test images are fitted three times using different random initializations (the same exact random initializations are used for all algorithms); *iii*) algorithms are left to run for 40 iterations (24 iterations at the first pyramid level and 16 at the second); *iv*) results for project-out algorithms are obtained using the Bayesian project-out cost function defined by equation 5.15; and *v*) results for asymmetric algorithms are reported for the special case of symmetric composition i.e. $\alpha = \beta = 0.5$ in equation 5.27.

The implementation of all the algorithms tested in this section is publicly available at <https://github.com/jalabort/alabortijcv2015>.

5.6.1 Comparison on LFPW

This experiment reports the fitting accuracy and convergence properties of all CGD algorithms studied in this paper. Results are reported on the ~ 220 test images of the LFPW database. In order to keep the information easily readable and interpretable, algorithms are grouped by cost function (i.e. SSD or PO), and optimization method (i.e. GN, N or W).

Results for this experiment are reported in figures 6.1, 5.5, 5.6, 5.7, 5.8 and 5.9. These figures have all the same structure and are composed of four figures and a table. Figures 5.4a, 5.5a, 5.6a, 5.7a, 5.8a and 5.9a report the *cumulative error distribution* (CED) curves, i.e the *proportion of images* vs *normalized point-to-point error* for each of the algorithms' groups. Tables 5.4e, 5.5e, 5.6e, 5.7e, 5.8e, and 5.9e summarize and complete the information

on the CED curves by stating the proportion of images fitted with a normalized point-to-point error smaller than 0.02, 0.03 and 0.04; and by stating the *mean*, *std* and *median* of the final normalized point-to-point error as well as the approximate *run-time*²⁶. The aim of these figures and tables is to help comparing the final fitting accuracy obtained by each algorithm. On the other hand, figures 5.4b, 5.5b, 5.6b, 5.7b, 5.8b and 5.9b report the mean normalized point-to-point error at each iteration while figures 5.4c, 5.4d, 5.5c, 5.5d, 5.6c, 5.6d, 5.7c, 5.7d, 5.8c, 5.8d and 5.9c, 5.9d report the mean normalized cost at each iteration²⁷. The aim of these figures is to help comparing the convergence properties of every algorithm.

5.6.1.1 SSD Gauss-Newton algorithms

Results for SSD Gauss-Newton algorithms are reported in figure 6.1. One can observe that inverse, asymmetric and bidirectional algorithms obtain a similar performance and significantly outperform forward algorithms in terms of fitting accuracy, figure 5.4a and table 5.4e. In absolute terms, bidirectional algorithms slightly outperform inverse and asymmetric algorithms. On the other hand, the difference in performance between the simultaneous Schur and alternated optimizations strategies are minimal for all algorithms and they were found to have no statistical significance.

Looking at figures 5.4b, 5.4c and 5.4d there seems to be a clear (and expected) correlation between the normalized point-to-point error and the normalized value of the cost function at each iteration. In terms of convergence, it can be seen that forward algorithms converge slower than inverse,

²⁶Run-time was measured on a laptop equipped with a 2.3GHz quad-core Intel Core i7 processor

²⁷These figures are produced by dividing the value of the cost function at each iteration by its initial value and averaging for all images.

asymmetric and bidirectional. Bidirectional algorithms converge slightly faster than inverse algorithms and these slightly faster than asymmetric algorithms. In this case, the simultaneous Schur optimization strategy seems to converge slightly faster than the alternated one for all SSD Gauss-Newton algorithms.

5.6.1.2 SSD Newton algorithms

Results for SSD Newton algorithms are reported on figure 5.5. In this case, we can observe that the fitting performance of all algorithms decreases with respect to their Gauss-Newton counterparts, figure 5.5a and table 5.5e. This is most noticeable in the case of forward algorithms for which there is $\sim 20\%$ drop in the proportion of images fitted below 0.02, 0.03 and 0.04 with respect to its Gauss-Newton equivalents. For these algorithms there is also a significant increase in the mean and median of the normalized point-to-point error. Asymmetric Newton algorithms also perform considerably worse, between 5% and 10%, than their Gauss-Newton versions. The drop in performance is reduced for inverse and bidirectional Newton algorithms for which accuracy is only reduced by around 3% with respect their Gauss-Newton equivalent.

Within Newton algorithms, there are clear differences in terms of speed of convergence, figures 5.5b, 5.5c and 5.5d. Bidirectional algorithms are the fastest to converge followed by inverse and asymmetric algorithms, in this order, and lastly forward algorithms. In this case, the simultaneous Schur optimization strategy seems to converge again slightly faster than the alternated one for all algorithms but bidirectional algorithms, for which the alternated strategy converges slightly faster. Overall, SSD Newton algorithms converge slower than SSD Gauss-Newton algorithms.

5.6.1.3 SSD Wiberg algorithms

Results for SSD Wiberg algorithms are reported on figure 5.6. Figure 5.6a and table 5.6e and figures 5.6b, 5.6c and 5.6d show that these results are (as one would expect) virtually equivalent to those obtained by their Gauss-Newton counterparts.

5.6.1.4 PO Gauss-Newton algorithms

Results for PO Gauss-Newton algorithms are reported on figure 5.7. One can observe that, there is significant drop in terms of fitting accuracy for inverse and bidirectional algorithms with respect to the SSD versions, figure 5.7a and table 5.7e. As expected, the forward algorithm achieves virtually the same results as its SSD counterpart. The asymmetric algorithm obtains similar accuracy to the best performing SSD algorithms.

Looking at figures 5.7b, 5.7c and 5.7d one can see that inverse and bidirectional algorithms converge slightly faster than the asymmetric algorithm. However, the asymmetric algorithm ends up descending to a significant lower value of the mean normalized cost which also translates to a lower value for the final mean normalized point-to-point error. Similar to SSD algorithms, the forward algorithm is the worst algorithm in terms of speed of convergence.

Finally, notice that, in this case, there is virtually no difference, in terms of both final fitting accuracy and speed of convergence, between the simultaneous Schur and alternated optimizations strategies used by the bidirectional algorithm.

5.6.1.5 PO Newton algorithms

Results for PO Newton algorithms are reported on figure 5.8. It can be clearly seen that PO Newton algorithms perform much worse than their Gauss-Newton and SSD counterparts. The final fitting accuracy obtained by these algorithms is very poor compared to the one obtained by the best SSD and PO Gauss-Newton algorithms, figures 5.8a and table 5.8e. In fact, by looking at figures 5.8b, 5.8c and 5.8d only the forward and asymmetric algorithms seem to be stable at the second level of the Gaussian pyramid with inverse and bidirectional algorithms completely diverging for some of the images as shown by the large mean and std of their final normalized point-to-point errors.

5.6.1.6 PO Wiberg algorithms

Results for the PO bidirectional Wiberg algorithm are reported on figure 5.9. As expected, the results are virtually identical to those of the obtained by PO bidirectional Gauss-Newton algorithms.

5.6.2 Weighted Bayesian project-out

This experiment quantifies the importance of each one of the two terms in the Bayesian project-out cost function defined by equation 5.15. To this end, the parameters, $\rho \in [0, 1]$ and $\gamma = 1 - \rho$ are introduced in order to weight up the relative contribution of both terms:

$$\rho ||\mathbf{i}[\mathbf{p}] - \bar{\mathbf{a}}||_{\mathbf{A}\mathbf{D}^{-1}\mathbf{A}^T}^2 + \frac{\gamma}{\sigma^2} ||\mathbf{i}[\mathbf{p}] - \bar{\mathbf{a}}||_{\mathbf{A}}^2 \quad (5.115)$$

Setting $\rho = 0$, $\gamma = 1$ reduces this cost function to the original project-out loss proposed in [10]; completely disregarding the contribution of the prior distribution over the texture parameters, i.e the Mahalanobis distance *within* the texture subspace. On the contrary, setting $\rho = 1$, $\gamma = 0$ reduces the cost function to the first term; completely disregarding the contribution of the project-out term i.e. the distance *to* the texture subspace. Finally setting $\rho = \gamma = 0.5$ leads to the standard Bayesian project-out cost function proposed in section 5.2.2.1.

In order to assess the impact that each term has on the fitting accuracy obtained by the PO algorithm the experimental set up of the first experiment is repeated and all PO Gauss-Newton algorithms are tested for different values of the parameters $\rho = 1 - \gamma$. Notice that, in this case, only the performance of Gauss-Newton algorithms is reported because they were shown to vastly outperform Newton algorithms and to be virtually equivalent to Wiberg algorithms in the first experiment.

Results for this experiment are reported by figure 5.10. One can see that, regardless of the type of composition, a weighted combination of the two terms always leads to a smaller mean normalized point-to-point error compared to either term on its own. Note that the final fitting accuracy obtained with the standard Bayesian project-out cost function is substantially better than the one obtained by the original project-out loss (this is specially noticeable for the inverse and bidirectional algorithms); fully justifying the inclusion of the first term, i.e the Mahalanobis distance *within* the texture subspace, into the cost function. Finally, in this particular experiment, the final fitting accuracy of all algorithms is maximized by setting $\rho = 0.1$, $\gamma = 0.9$, further highlighting the importance of the first term in the Bayesian formulation.

5.6.3 Optimal asymmetric composition

This experiment quantifies the effect that varying the value of the parameters $\alpha \in [0, 1]$ and $\beta = 1 - \alpha$ in equation 5.27 has in the fitting accuracy obtained by the *Asymmetric* algorithms. Note that for $\alpha = 1, \beta = 0$ and $\alpha = 0, \beta = 1$ these algorithms reduce to their forward and inverse versions respectively. Recall that, in previous experiments, the symmetric case $\alpha = \beta = 0.5$ was used to generate the results reported for asymmetric algorithms. Again, only the performance of Gauss-Newton algorithms is reported.

The experimental set up described in the first experiments is repeated again and the fitting accuracy obtained by the project out and SSD asymmetric Gauss-Newton algorithms reported for different values of the parameters $\alpha = 1 - \beta$. Results are shown in figure 5.12. For the PO asymmetric algorithm, the best results are obtain by setting $\alpha = 0.4, \beta = 0.6$, figures 5.12a (top) and 5.12b. These results slightly outperform those obtain by the default symmetric algorithm and this particular configuration of the PO asymmetric algorithm is the best performing one on the LFPW test dataset. For the SSD asymmetric Gauss-Newton algorithm the best results are obtained by setting $\alpha = 0.2, \beta = 0.8$, figures 5.12a (bottom) and 5.12c. In this case, the boost in performance with respect to the default symmetric algorithm is significant and, with this particular configuration, the SSD asymmetric Gauss-Newton algorithm is the best performing SSD algorithm on the LFPW test dataset, outperforming inverse and bidirectional algorithms.

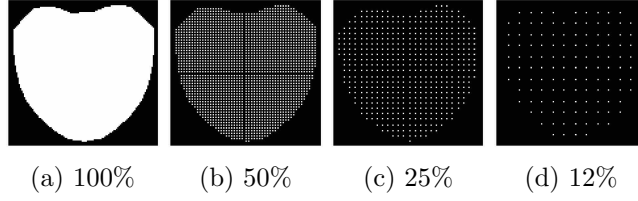


Figure 5.3: Subset of pixels on the reference frame used to optimize the SSD and PO cost functions for different sampling rates.

5.6.4 Sampling and number of iterations

In this experiment, we explore two different strategies to reduce the running time of the CGD algorithms.

The first one consists of optimizing the SSD and PO cost functions using only a subset of all pixels in the reference frame. In AAM the total number of pixels on the reference frame, F , is typically several orders of magnitude bigger than the number of shape, n , and texture, m , components i.e. $F \gg m \gg n$. Therefore, a significant reduction in the complexity (and running time) of CGD algorithms can be obtained by decreasing the number of pixels that are used to optimize the cost functions. To this end, this experiment compares the accuracy obtained by using 100%, 50%, 25% and 12% of the total number of pixels on the reference frame. Note that pixels are (approximately) evenly sampled across the reference frame in all cases, figure 5.3.

The second strategy consists of simply reducing the number of iterations that each algorithm is run. Based on the figures used to assess the convergence properties of CGD algorithms in previous experiments, the accuracy obtained by running the algorithms for 40 (24+16) and 20 (12+8) iterations is compared.

Note that, in order to further highlight the advantages and disadvantages of using these strategies, the fitting accuracy obtained by initializing the algorithms using different amounts of uniform noise is also reported.

Once more, the experimental set up of the first experiment is repeated and the fitting accuracy obtained by the PO and SSD asymmetric Gauss-Newton algorithms is reported. Results for this experiment are shown in figure 5.11. It can be seen that reducing the number of pixels up to 25% while maintaining the original number of iterations to 40 (24+16) has little impact on the fitting accuracy achieved by both algorithms while reducing them to 12% has a clear negative impact, figures 5.11a and 5.11b. Also, performance seems to be consistent along the amount of noise. In terms of run time, table 5.11e, reducing the number of pixels to 50%, 25% and 12% offers speed ups of $\sim 2.0x$, $\sim 2.9x$ and $\sim 3.7x$ for the PO algorithm and of $\sim 1.8x$, $\sim 2.6x$ and $\sim 2.8x$ for the SSD algorithm respectively.

On the other hand, reducing the number of iterations from 40 (24 + 16) to 20 (12 + 8) has no negative impact in performance for levels of noise smaller than 2% but has a noticeable negative impact for levels of noise bigger than 5%. Notice that remarkable speed ups, table 5.11e, can be obtain for both algorithms by combining the two strategies at expenses of a small but noticeable decreases in fitting accuracy.

5.6.5 Comparison on Helen and AFW

In order to facilitate comparisons with recent prior work on AAM [40, 57, 41] and with another state-of-the-art approach in non-rigid face alignment [20, 22], this experiment reports the fitting accuracy of the SSD and PO asymmetric Gauss-Newton algorithms on the widely used test set of the

Helen dataset and on the entire AFW dataset. Furthermore the performance of these two algorithms is compared with the one obtained by the recently proposed *Gauss-Newton deformable part models* (GN-DPM) proposed by Tzimiropoulos and Pantic in [94]; which was shown to achieve state-of-the-art results in the problem of non-rigid face alignment in-the-wild.

For SD and PO asymmetric Gauss-Newton algorithms, two different types of results are reported: *i*) sampling rate of 25% and 20 (12 + 8) iterations; and *ii*) sampling rate of 50% and 40 (24 + 16) iterations, . For GN-DPM the authors public implementation²⁸ is used to obtain the results. In this case, two different types of results are reported by letting the algorithm run for 20 and 40 iterations.

Result for this experiment are shown in figure 5.13. Looking at figure 5.13a one can see that both, SSD and PO asymmetric Gauss-Newton algorithms, obtain similar fitting accuracy on the Helen test dataset. Note that, in all cases, their accuracy is comparable to the one achieved by GN-DPM for normalized point-to-point errors < 0.2 and significantly better for < 0.3 , < 0.4 . As expected, the best results for both of these algorithms are obtained using 50% of the total amount of pixels and 40 (24+16) iterations. However, the results obtained by using only 25% of the total amount of pixels and 20 (12+8) iterations are comparable to the previous ones; specially for the PO asymmetric Gauss-Newton. In general, these results are consistent with the ones obtained on the LFPW test dataset, experiments 5.6.1 and 5.6.3.

On the other hand, the performance of both algorithms drops significantly on the AFW dataset, figure 5.13b. In this case, GN-DPM achieves slightly better results than the SSD and PO Asymmetric Gauss-Newton algorithms

²⁸http://www.cs.nott.ac.uk/~pszyt/code/gn_dpm_code_v1.zip

for normalized point-to-point errors < 0.2 and slightly worst for < 0.3 , < 0.4 . Again, both of these algorithms obtain better results by using 50% sampling rate and 40 ($24 + 16$) iterations and the difference in accuracy with respect to the versions using 25% sampling rate and 20 ($12 + 8$) iterations slightly widens when compared to the results obtained on the Helen test dataset. This drop in performance is consistent with other published results in the existent literature [94, 53, 57, 55] and it is attributed to large difference in terms of shape and texture statistics between the images of the AFW dataset and the ones of the LFPW and Helen training datasets where the AAM model was trained on.

5.6.6 Analysis of the results

Given the results obtained in the previous experiments the following conclusions can be drawn:

1. Overall, Gauss-Newton and Wiberg algorithms vastly outperform Newton algorithms in AAM fitting. Experiment 5.6.1 clearly shows that the former algorithms provide significantly higher levels of fitting accuracy at considerably lower computational complexities and run times. These findings are consistent with existent literature in the related field of parametric image alignment [10] and also, to certain extend, with prior work on Newton algorithms for AAM fitting [41]. The bad performance of Newton algorithms is attributed to the difficulty of accurately computing a (noiseless) estimate of the full Hessian matrix using finite differences.
2. Gauss-Newton and Wiberg algorithms are virtually equivalent in per-

- formance. The results in experiment 5.6.1 show that the difference in accuracy between both types of algorithms is minimal and the small differences in their respective solutions are, in practice, insignificant.
3. The novel Bayesian project-out formulation presented in this chapter leads to significant improvements in fitting accuracy without adding extra computational cost. Experiment 5.6.2 shows that a weighted combination of the two terms forming the Bayesian project-out loss always outperforms the classic project-out formulation.
 4. The asymmetric composition proposed in this chapter leads to CGD algorithms that are more accurate and that converge faster than previous algorithms. In particular, the SSD and PO asymmetric Gauss-Newton algorithms are shown to achieve significantly better performance than their forward and inverse counterparts in experiments 5.6.1 and 5.6.3.
 5. Finally, a significant reduction in the computational complexity and run-time can be obtained by limiting the number of pixels considered during optimization of the loss function and by adjusting the number of iterations that CGD algorithms are run, experiment 5.6.4.

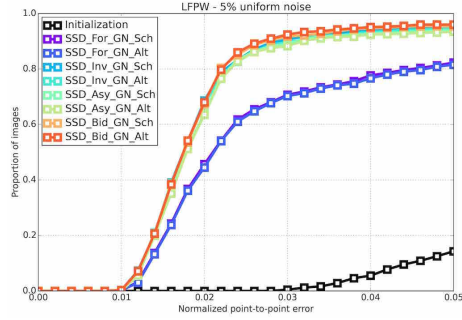
5.7 Conclusions

This chapter offers a detailed study of CGD algorithms for AAM fitting. It presents a unified and complete framework for these type of algorithms and classifies them with respect to three of their main characteristics: *i) cost function; ii) type of composition; and iii) optimization method.*

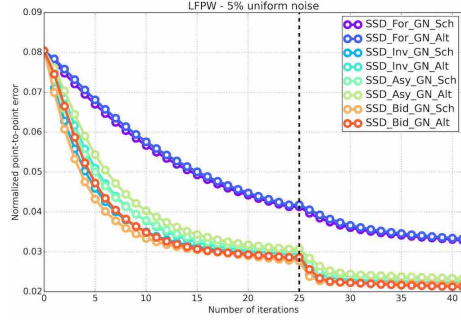
Furthermore, several important contribution are made across the chapter

with respect to each one of the previous three characteristics:

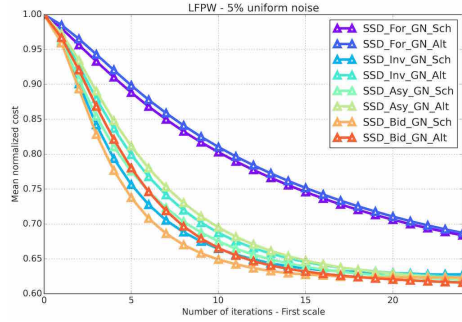
- A novel *Bayesian cost function* for fitting AAM. This cost function can be interpreted as as a valid and more general probabilistic formulation of the well-known project-out loss. The results reported in sections 5.6.1 and 5.6.2 show that this new Bayesian formulation considerably outperforms the original project-out cost function.
- Two new types of compositions for CGD algorithms, *asymmetric* and *bidirectional*. The results reported in section 5.6.1 and 5.6.3 experimentally prove that these two new types of composition lead to better convergent and more robust CGD algorithm for AAM fitting.
- New valuable insights into existent CGD algorithms are provided in sections 5.4.1 and 5.4.3 by reinterpreting them as direct applications of the *Schur complement* and the *Wiberg method*.



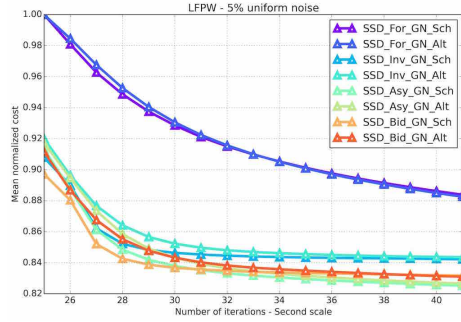
(a) Fitting accuracy for all SSD Gauss-Newton algorithms.



(b) Fitting convergence for all SSD Gauss-Newton algorithms.



(c) Cost convergence on the first pyramid level for all SSD Gauss-Newton algorithms.

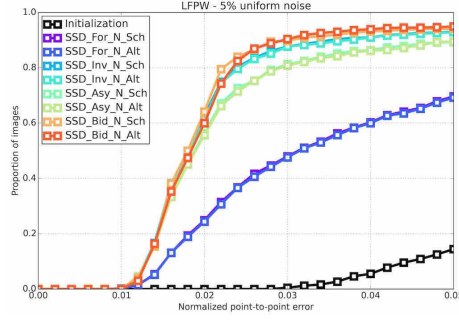


(d) Cost convergence on the second pyramid level for all SSD Gauss-Newton algorithms.

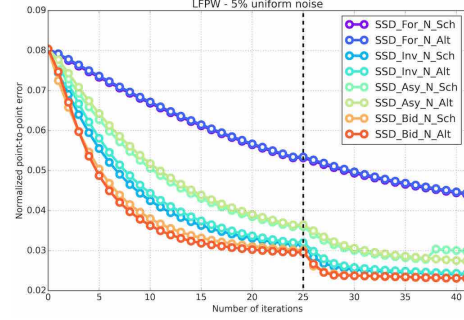
Algorithm	< 0.02	< 0.03	< 0.04	Mean	Std	Median	Run-Time
Initialization	0.000	0.004	0.055	0.080	0.028	0.078	-
SSD_For_GN_Sch	0.456	0.707	0.777	0.033	0.030	0.021	~ 1.6 s
SSD_For_GN_Alt	0.445	0.702	0.766	0.033	0.030	0.021	~ 1.4 s
SSD_Inv_GN_Sch	0.686	0.906	0.939	0.022	0.019	0.017	~ 1.6 s
SSD_Inv_GN_Alt	0.673	0.897	0.933	0.022	0.020	0.017	~ 1.4 s
SSD_Asy_GN_Sch	0.640	0.891	0.929	0.023	0.021	0.018	~ 1.7 s
SSD_Asy_GN_Alt	0.635	0.882	0.924	0.023	0.021	0.018	~ 1.4 s
SSD_Bid_GN_Sch	0.674	0.917	0.946	0.022	0.019	0.017	~ 1.9 s
SSD_Bid_GN_Alt	0.680	0.924	0.951	0.021	0.019	0.017	~ 1.6 s

(e) Table showing the proportion of images fitted with a normalized point-to-point error below 0.02, 0.03 and 0.04 together with the normalized point-to-point error mean, std and median for all SSD Gauss-Newton algorithms.

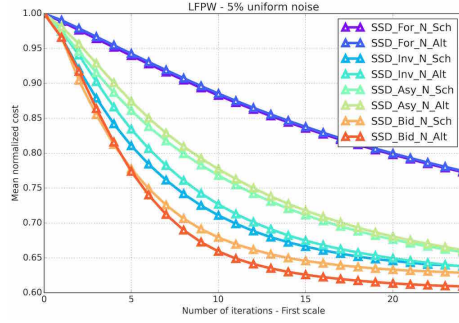
Figure 5.4: Results showing the fitting accuracy and fitting and cost convergence properties of the SSD Gauss-Newton algorithms on the LFPW test dataset initialized with 5% uniform noise.



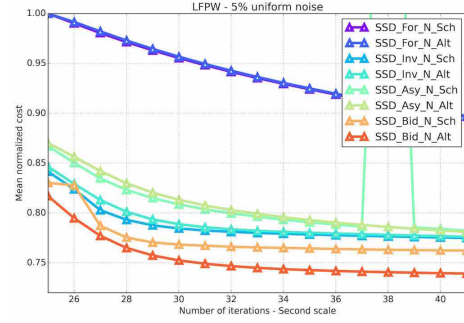
(a) Fitting accuracy for all SSD Newton algorithms.



(b) Fitting convergence for all SSD Newton algorithms.



(c) Cost convergence on the first pyramid level for all SSD Newton algorithms.

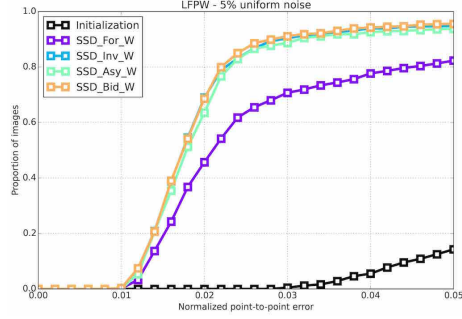


(d) Cost convergence on the second pyramid level for all SSD Newton algorithms.

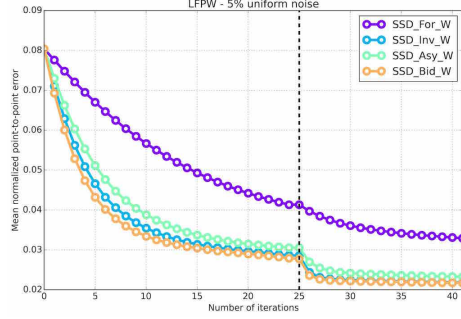
Algorithm	< 0.02	< 0.03	< 0.04	Mean	Std	Median	Run-Time
Initialization	0.000	0.004	0.055	0.080	0.028	0.078	-
SSD_For_N_Sch	0.249	0.479	0.603	0.044	0.033	0.031	~ 2.3 s
SSD_For_N_Alt	0.244	0.476	0.600	0.044	0.033	0.032	~ 2.1 s
SSD_Inv_N_Sch	0.626	0.876	0.909	0.024	0.022	0.018	~ 2.3 s
SSD_Inv_N_Alt	0.613	0.876	0.909	0.024	0.022	0.018	~ 2.1 s
SSD_Asy_N_Sch	0.562	0.812	0.863	0.030	0.076	0.019	~ 2.5 s
SSD_Asy_N_Alt	0.557	0.808	0.862	0.027	0.025	0.019	~ 2.1 s
SSD_Bid_N_Sch	0.641	0.897	0.932	0.023	0.022	0.018	~ 3.2 s
SSD_Bid_N_Alt	0.600	0.903	0.939	0.023	0.021	0.018	~ 2.7 s

(e) Table showing the proportion of images fitted with a normalized point-to-point error below 0.02, 0.03 and 0.04 together with the normalized point-to-point error mean, std and median for all SSD Newton algorithms.

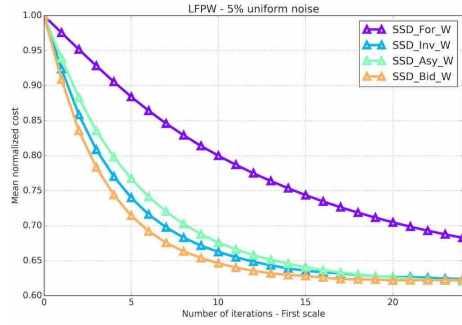
Figure 5.5: Results showing the fitting accuracy and fitting and cost convergence properties of the SSD Newton algorithms on the LFPW test dataset initialized with 5% uniform noise.



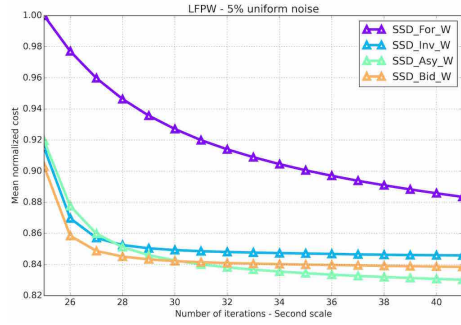
(a) Fitting accuracy for all SSD Wiberg algorithms.



(b) Fitting convergence for all SSD Wiberg algorithms.



(c) Cost convergence on the first pyramid level for all SSD Wiberg algorithms.

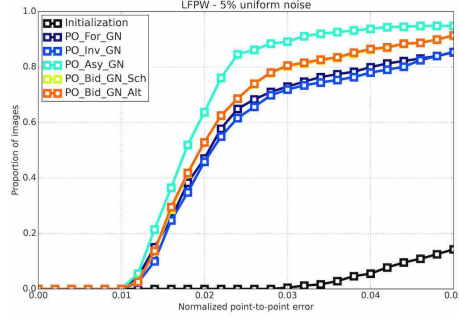


(d) Cost convergence on the second pyramid level for all SSD Wiberg algorithms.

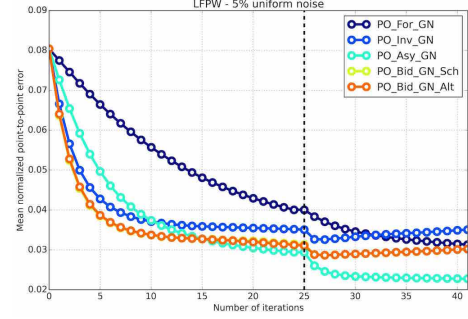
Algorithm	< 0.02	< 0.03	< 0.04	Mean	Std	Median	Run-Time
Initialization	0.000	0.004	0.055	0.080	0.028	0.078	-
SSD_For_W	0.457	0.707	0.777	0.33	0.030	0.021	~ 1.6 s
SSD_Inv_W	0.689	0.903	0.939	0.22	0.019	0.017	~ 1.6 s
SSD_Asy_W	0.635	0.887	0.926	0.23	0.021	0.018	~ 1.7 s
SSD_Bid_W	0.686	0.911	0.942	0.22	0.019	0.017	~ 1.9 s

(e) Table showing the proportion of images fitted with a normalized point-to-point error below 0.02, 0.03 and 0.04 together with the normalized point-to-point error mean, std and median for all SSD Wiberg algorithms.

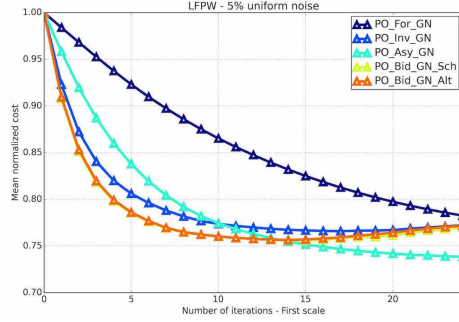
Figure 5.6: Results showing the fitting accuracy and fitting and cost convergence properties of the SSD Wiberg algorithms on the LFPW test dataset initialized with 5% uniform noise.



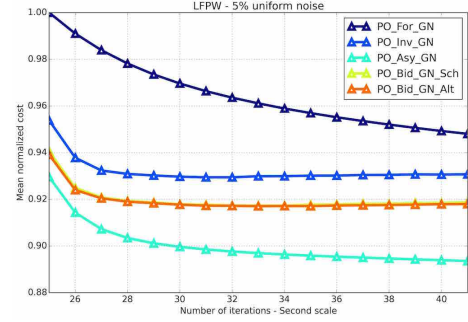
(a) Fitting accuracy for all PO Gauss-Newton algorithms.



(b) Fitting convergence for all PO Gauss-Newton algorithms.



(c) Cost convergence on the first pyramid level for all PO Gauss-Newton algorithms.

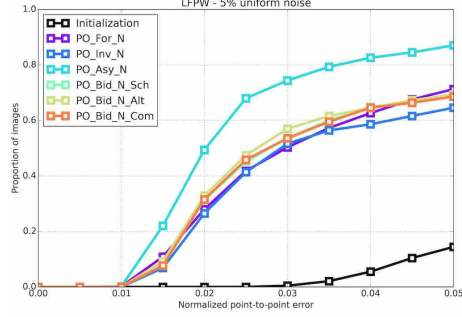


(d) Cost convergence on the second pyramid level for all PO Gauss-Newton algorithms.

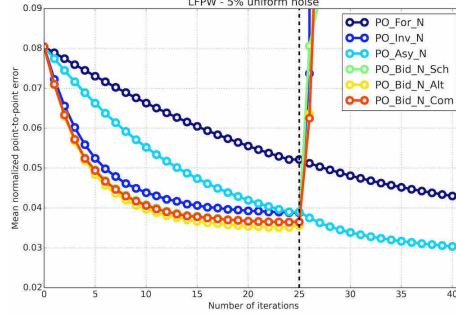
Algorithm	< 0.02	< 0.03	< 0.04	Mean	Std	Median	Run-Time
Initialization	0.000	0.004	0.055	0.080	0.028	0.078	-
PO_For_GN_Sch	0.470	0.729	0.799	0.031	0.029	0.021	~ 1.4 s
PO_Inv_GN_ASch	0.458	0.719	0.780	0.035	0.044	0.021	~ 0.6 s
PO_Asy_GN_Sch	0.637	0.891	0.938	0.023	0.021	0.018	~ 1.4 s
PO_Bid_GN_Sch	0.528	0.802	0.862	0.030	0.039	0.020	~ 1.6 s
PO_Bid_GN_Alt	0.528	0.805	0.865	0.030	0.040	0.019	~ 1.4 s

(e) Table showing the proportion of images fitted with a normalized point-to-point error below 0.02, 0.03 and 0.04 together with the normalized point-to-point error mean, std and median for all PO Gauss-Newton algorithms.

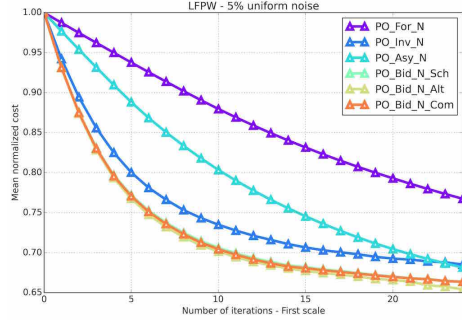
Figure 5.7: Results showing the fitting accuracy and fitting and cost convergence properties of the PO Gauss-Newton algorithms on the LFPW test dataset initialized with 5% uniform noise.



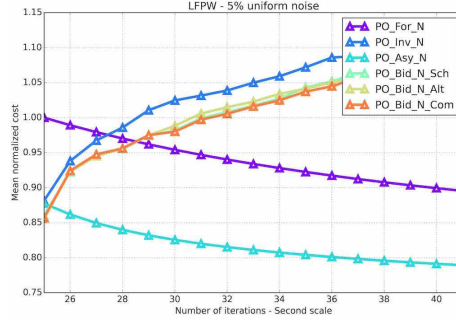
(a) Fitting accuracy for all PO Newton algorithms.



(b) Fitting convergence for all PO Newton algorithms.



(c) Cost convergence on the first pyramid level for all PO Newton algorithms.

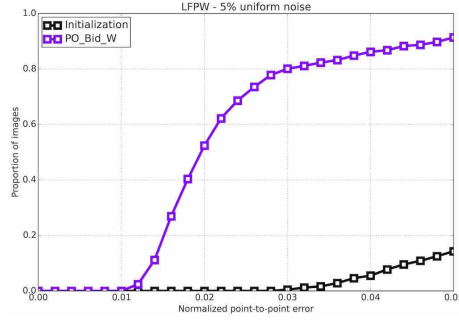


(d) Cost convergence on the second pyramid level for all PO Newton algorithms.

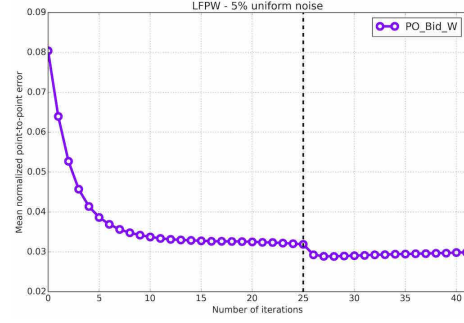
Algorithm	< 0.02	< 0.03	< 0.04	Mean	Std	Median	Run-Time
Initialization	0.000	0.004	0.055	0.080	0.028	0.078	-
PO_For_N_Sch	0.280	0.503	0.626	0.043	0.033	0.030	~ 2.1 s
PO_Inv_N_Alt	0.265	0.516	0.586	11.929	179.525	0.029	~ 1.6 s
PO_Asy_N_Sch	0.494	0.744	0.826	0.030	0.028	0.020	~ 2.1 s
PO_Bid_N_Sch	0.314	0.536	0.649	0.287	1.347	0.027	~ 2.8 s
PO_Bid_N_Alt	0.329	0.570	0.649	0.280	1.465	0.026	~ 2.1 s

(e) Table showing the proportion of images fitted with a normalized point-to-point error below 0.02, 0.03 and 0.04 together with the normalized point-to-point error mean, std and median for all PO Newton algorithms.

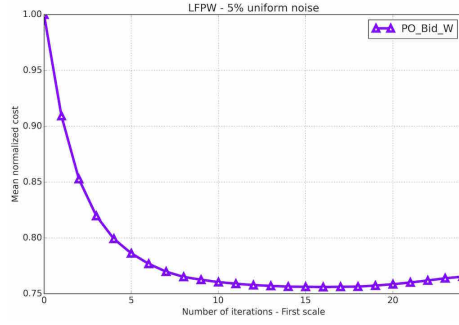
Figure 5.8: Results showing the fitting accuracy and fitting and cost convergence properties of the PO Newton algorithms on the LFPW test dataset initialized with 5% uniform noise.



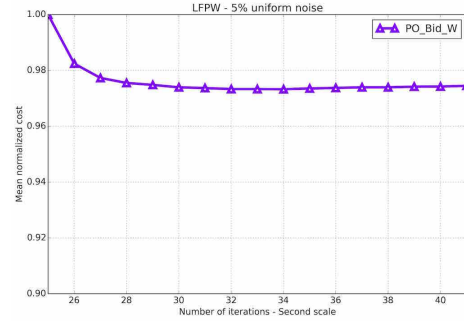
(a) Fitting accuracy for all PO Wiberg algorithms.



(b) Fitting convergence for all PO Wiberg algorithms.



(c) Cost convergence on the first pyramid level for for all PO Wiberg algorithms.

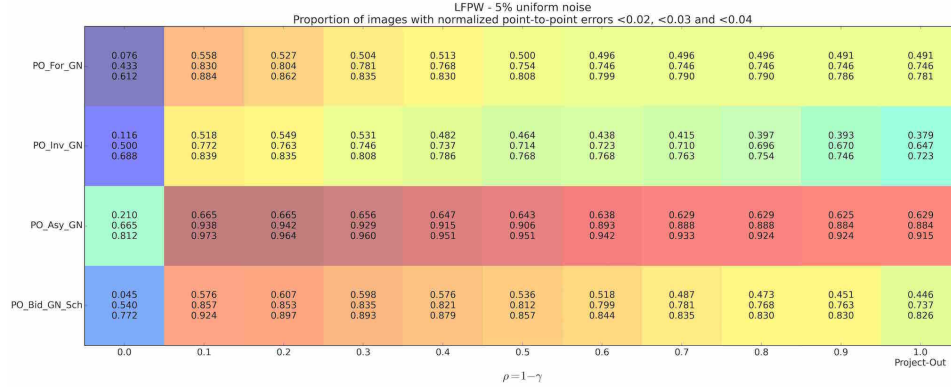


(d) Cost convergence on the second pyramid level for for all PO Wiberg algorithms.

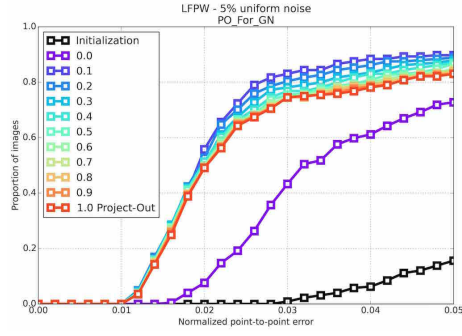
Algorithm	< 0.02	< 0.03	< 0.04	Mean	Std	Median	Run-Time
Initialization	0.000	0.004	0.055	0.080	0.028	0.078	-
PO_Bid_W_Sch	0.524	0.801	0.862	0.030	0.039	0.020	~ 1.4 s

(e) Table showing the proportion of images fitted with a normalized point-to-point error below 0.02, 0.03 and 0.04 together with the normalized point-to-point error mean, std and median for all PO Wiberg algorithms.

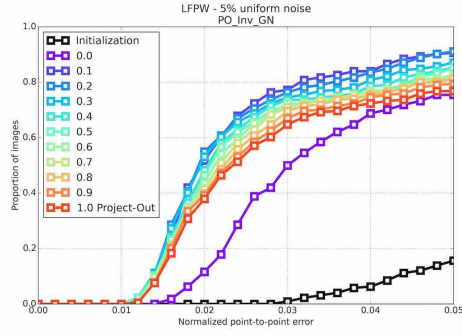
Figure 5.9: Results showing the fitting accuracy and fitting and cost convergence properties of the PO Wiberg algorithms on the LFPW test dataset initialized with 5% uniform noise



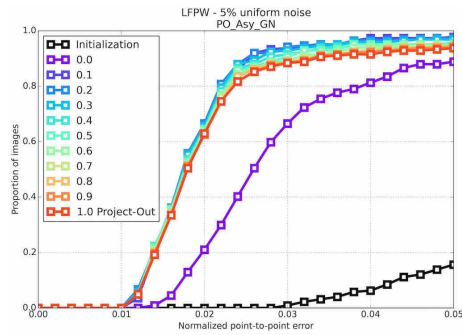
(a) Proportion of images with normalized point-to-point errors smaller than 0.02, 0.03 and 0.04 for the PO and SSD asymmetric Gauss-Newton algorithms for different values of $\rho = 1 - \gamma$. Colors encode overall fitting accuracy, from highest to lowest: red, orange, yellow, green, blue and purple.



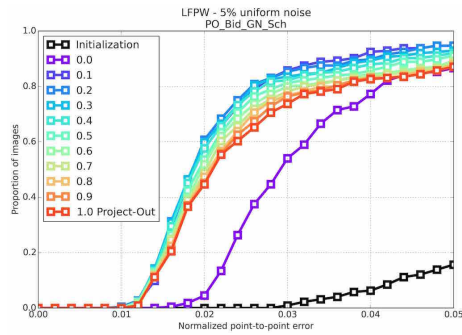
(b) Fitting accuracy for PO forward Gauss-Newton algorithms for different values of $\rho = 1 - \gamma$.



(c) Fitting accuracy for PO inverse Gauss-Newton algorithms for different values of $\rho = 1 - \gamma$.

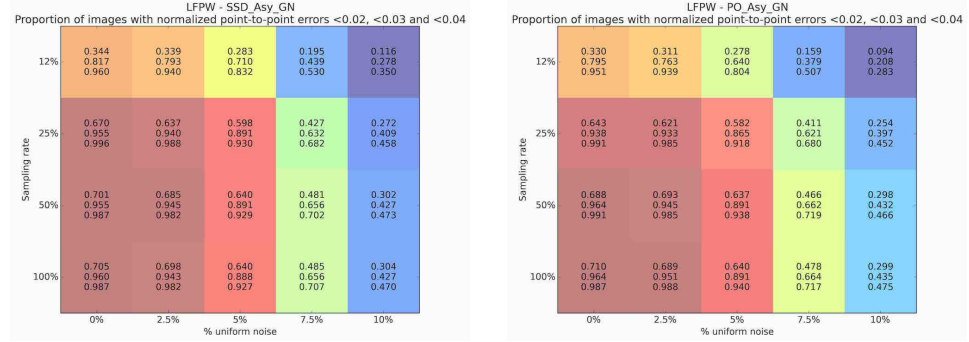


(d) Fitting accuracy for PO asymmetric Gauss-Newton algorithms for different values of $\rho = 1 - \gamma$.



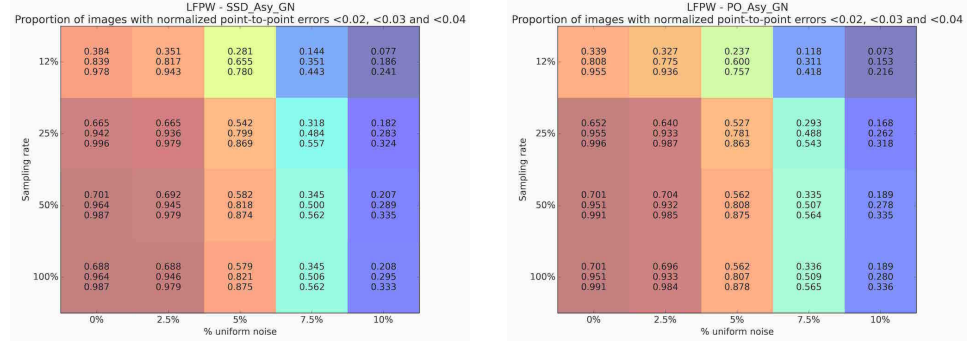
(e) Fitting accuracy for PO bidirectional Gauss-Newton algorithms for different values of $\rho = 1 - \gamma$.

Figure 5.10: Results quantifying the effect of varying the value of the parameters $\rho = 1 - \gamma$ in PO Gauss-Newton algorithms on the LFPW test dataset initialized with 5% noise.



(a) Proportion of images with normalized point-to-point errors smaller than 0.02, 0.03 and 0.04 for the SSD asymmetric Gauss-Newton algorithm using different sampling rates, 40 (24 + 16) iterations, and initialized with different amounts of noise.

(b) Proportion of images with normalized point-to-point errors smaller than 0.02, 0.03 and 0.04 for the PO asymmetric Gauss-Newton algorithm using different sampling rates, 40 (24 + 16) iterations, and initialized with different amounts of noise.



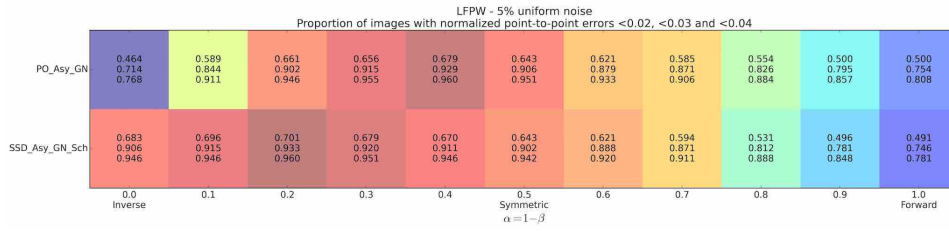
(c) Proportion of images with normalized point-to-point errors smaller than 0.02, 0.03 and 0.04 for the SSD asymmetric Gauss-Newton algorithm using different sampling rates, 20 (12 + 8) iterations, and initialized with different amounts of noise.

(d) Proportion of images with normalized point-to-point errors smaller than 0.02, 0.03 and 0.04 for the PO asymmetric Gauss-Newton algorithm using different sampling rates, 20 (12 + 8) iterations, and initialized with different amounts of noise.

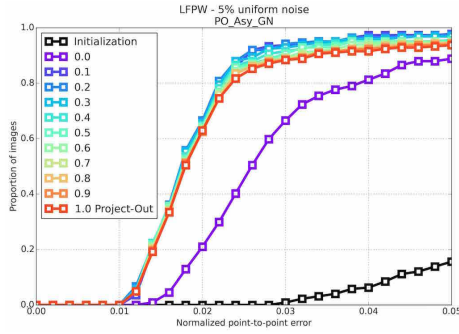
	Iterations	100%	$< 50\%$	$< 25\%$	$< 12\%$
SSD_Asy_GN_Sch	40 (24 + 16)	~ 1680 ms	~ 930 ms	~ 650 ms	~ 590 ms
PO_Asy_GN	40 (24 + 16)	~ 1400 ms	~ 680 ms	~ 480 ms	~ 380 ms
SSD_Asy_GN_Sch	20 (12 + 8)	~ 892 ms	~ 519 ms	~ 369 ms	~ 331 ms
PO_Asy_GN	20 (12 + 8)	~ 707 ms	~ 365 ms	~ 235 ms	~ 211 ms

(e) Table showing run time of each algorithm for different amounts of sampling and iterations.

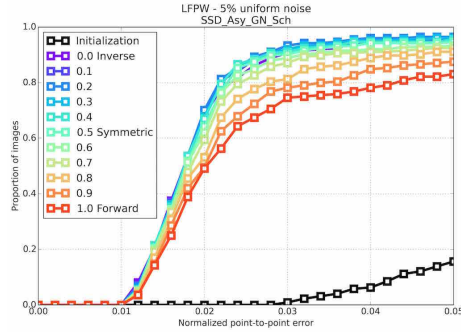
Figure 5.11: Results assessing the effectiveness of sampling for the best performing PO and SSD algorithms on the LFPW database. Colors encode overall fitting accuracy, from highest to lowest: red, orange, yellow, green, blue and purple.



(a) Proportion of images with normalized point-to-point errors smaller than 0.02, 0.03 and 0.04 for the PO and SSD asymmetric Gauss-Newton algorithms for different values of $\alpha = 1 - \beta$. Colors encode overall fitting accuracy, from highest to lowest: red, orange, yellow, green, blue and purple.

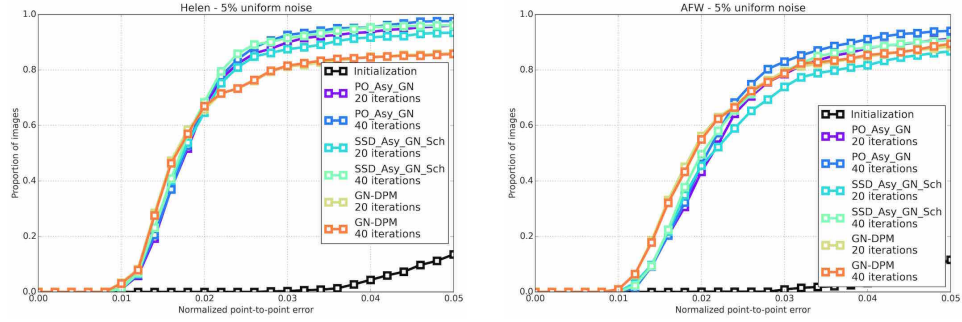


(b) Fitting accuracy for the PO asymmetric Gauss-Newton algorithm for different values of $\alpha = 1 - \beta$. Colors encode different values of $\alpha = 1 - \beta$.



(c) Fitting accuracy for the the SSD asymmetric Gauss-Newton algorithm for different values of $\alpha = 1 - \beta$. Colors encode different values of $\alpha = 1 - \beta$.

Figure 5.12: Results quantifying the effect of varying the value of the parameters $\alpha = 1 - \beta$ for asymmetric algorithms on the LFPW dataset initialized with 5% noise.



(a) Fitting accuracy on the Helen test dataset.

(b) Fitting accuracy on the entire AFW dataset.

Figure 5.13: Results showing the fitting accuracy for the SSD and PO asymmetric Gauss-Newton algorithms on the Helen test dataset and the entire AFW dataset both initialized with 5% noise.

Chapter 6

Combining AAM and CLM

This chapter proposes to combine two of the most popular techniques in deformable model fitting: *active appearance models* (AAM) and *constrained local models* (CLM). AAM fall into the category of *holistic deformable models* (HDM), i.e. models that represent the texture of objects as a whole. On the contrary, CLM are a particular type of *parts-based deformable models* (PBDM), i.e. models that represent object texture as a group of independent parts. Both of these models have been shown to have their own advantages. This chapter proposes to unify the fitting procedures of AAM and CLM by defining a novel probabilistic formulation of the deformable model fitting problem that combines the strengths of both methods.

The content of this chapter is based on the following publication:

- J. Alabort-i-Medina and S. Zafeiriou, “Unifying holistic and parts-based deformable model fitting,” in *IEEE Conference on computer vision and pattern recognition (CVPR)*, 2015.

6.1 Motivation

So far, this thesis has focused on the study of algorithms for fitting *active appearance models* (AAM) [8, 10]. However, deformable model fitting has been an incredibly active area of research in computer vision for over 20 years and, consequently, a large variety of deformable models and fitting strategies have been proposed throughout the years [6, 7, 8, 9, 10, 12, 11, 5, 13, 1, 4, 51, 15, 16, 22, 20, 21, 19, 17, 18, 25, 26, 27, 57, 23, 24, 56]. In general, research in this area can be divided into two¹ different groups:

1. *Holistic deformable models* (HDM) [7, 8, 9, 10, 12, 11, 51, 21, 57, 56]
2. *Parts-based deformable models* (PBDM) [6, 5, 13, 1, 4, 22, 23, 24]

The main difference between both groups is the approach used to model object texture.

AAM fall into the category of HDM. These models define texture globally; typically, by means of a generative representation [7, 8, 10]. As explained in chapter 5, AAM fitting strategies are generally posed as a regularized search for the optimal shape and texture parameters, $\mathbf{p}$ and $\mathbf{c}$ respectively, that minimize a *global* measure of misalignment that simultaneously depends on the position of all landmarks, i.e.:

$$\mathbf{p}^*, \mathbf{c}^* = \arg \min_{\mathbf{p}, \mathbf{c}} \mathcal{R}(\mathbf{p}, \mathbf{c}) + \mathcal{D}(\mathbf{s}, I) \quad (6.1)$$

¹Notice that, *direct regression techniques* (DRT) [15, 16, 20, 19, 17, 18, 25, 26, 27] can also be used to solve the same type of problems solved by deformable model fitting. These techniques attempt to solve these problems directly, without the use of an explicit deformable model, by learning a direct function mapping between the object's texture and the correct position of its landmarks.

where $\mathcal{R}$ is a regularization term that penalizes complex shape and texture deformations and $\mathcal{D}$ is a data term that quantifies the global measure of misalignment given the current position of all landmark points $\mathbf{x}_i = (x_i, y_i)^T$ defining the shape $\mathbf{s} = (\mathbf{x}_1^T, \dots, \mathbf{x}_v^T)^T$ of the object on the image I . AAM are capable of producing very accurate fitting results [51, 53, 57, 56]. However, the large dimensionality of their parameter space can make them difficult to optimize and, occasionally, make them converge to undesirable local minima for inaccurate initializations.

On the contrary, *constrained local models* (CLM) [6, 5], are categorized as PBDM. These models represent object texture locally as the combination of several independent local texture parts [6, 5, 22, 23, 24]. CLM fitting strategies are commonly formulated as a regularized search for the optimal shape parameters, $\mathbf{p}$, (local texture parts are usually learned discriminatively) that jointly minimize v *local* measures of misalignment dependent on each landmark $\mathbf{x}_i$ i.e:

$$\mathbf{p}^* = \arg \min_{\mathbf{p}} \mathcal{R}(\mathbf{p}) + \sum_{i=1}^v \mathcal{D}_i(\mathbf{x}_i, I) \quad (6.2)$$

where, as before, $\mathcal{R}$ is a regularization term that penalizes only complex shape deformations and, in this case, $\mathcal{D}_i$ are independent data terms quantifying the local misalignment measures given by the current position of each landmark $\mathbf{x}_i$ on the image I . CLM are generally easier to optimize than AAM, less dependent on the initialization and better suited to handle partial occlusions due to their local nature [6, 5, 22, 23, 24]. However, they have not been shown to match the accuracy obtained of optimally fitted AAM.

This chapter proposes to overcome the previous limitations by unifying AAM and CLM fitting. To this end, a novel a novel probabilistic formulation of

the fitting problem that unifies the fitting procedures of AAM and CLM is proposed. This approach explicitly and optimally combines two different statistically learned texture models, i.e. the holistic (the AAM generative texture model) and the parts-based (the CLM discriminative texture model) with a single statistically learned 2d shape model (the same 2d point distribution model used by both AAM and CLM) using a *maximum a posteriori* (MAP) estimation framework. The result is a unified cost function that can be iteratively solved using the Gauss-Newton method and in which the solution at each iteration is given by an optimal, in a MAP sense, weighted combination of the original AAM and CLM iterative solutions.

6.2 Constrained local models

Constrained local models (CLM) [6, 5] are a particular type of *parts-based deformable models* (PBDM) that define the texture of an object by independently modelling the local image region around the landmarks defining its shape. They represent the object's shape with a global 2d shape model like the one used by *active appearance models* (AAM) (section 3.1.1).

Although CLM can use generative approaches to model local image regions, the usual approach is discriminative. Hence, for each landmark a local patch expert that quantifies the probability of the landmark being correctly aligned $p(l_i = 1 | \mathbf{x}_i, I)$ is learned based on the support of its local image region. The previous density can be formally defined as:

$$p(l_i = 1 | \mathbf{x}_i, I) = \frac{1}{1 + \exp\{l_i \mathcal{C}_i(I, \mathbf{x}_i)\}} \quad (6.3)$$

where $\mathcal{C}_i$ is a (typically linear) classifier that discriminates between aligned

and misaligned locations i.e.:

$$\mathcal{C}_i(I, \mathbf{x}_i) = \mathbf{w}_i [I(\mathbf{y}_{i,1}), \dots, I(\mathbf{y}_{i,k})] + b_i \quad (6.4)$$

and $\{\mathbf{y}_{i,j}\}_{j=1}^k \in \Omega_{\mathbf{x}_i}$ denotes the image patch around the current landmark estimate $\mathbf{x}_i$.

Notice that different types of classifiers have been used to define $\mathcal{C}_i$ throughout the literature, e.g: *logistic regression* (LR) [5], *correlation filters* (CF) [23, 61], and *support vector machines* (SVM) [22] among others.

6.2.1 Fitting CLM

Fitting CLM involves solving the following optimization problem [5]:

$$\begin{aligned} \mathbf{p}^* &= \arg \min_{\mathbf{p}} \mathcal{R}(\mathbf{p}) + \sum_{i=1}^v \mathcal{D}_i(\mathbf{x}_i, I, \mathbf{p}) \\ &= \arg \min_{\mathbf{p}} \underbrace{\|\mathbf{p}\|_{\mathbf{\Lambda}^{-1}}^2}_{\mathcal{R}(\mathbf{p})} + \underbrace{\sum_{i=1}^v \sum_{j=1}^k \frac{w_j}{\rho^2} \|\mathbf{x}_i - \mathbf{y}_{i,j}\|^2}_{\sum_{i=1}^v \mathcal{D}_i(\mathbf{x}_i, I, \mathbf{p})} \end{aligned} \quad (6.5)$$

where $w_j = \frac{1}{1 + \exp\{l_i \mathcal{C}_j(\mathcal{I}, \mathbf{x}_j)\}}$, $\mathbf{\Lambda}$ is a diagonal matrix containing the eigenvalues associated to the shape eigenvectors $\mathbf{S}$ and ρ^2 is the estimated shape noise².

²The theoretically optimal value for ρ^2 can be computed as the average value of the eigenvalues associated to the discarded shape eigenvectors i.e. $\rho^2 = \frac{1}{N-n} \sum_{i=n}^N \lambda_{\mathbf{s},i}$ [88].

6.2.1.1 Probabilistic interpretation

A probabilistic interpretation of equation 6.5 can be obtained assuming the following probabilistic generative model of shape:

$$\mathbf{s} = \bar{\mathbf{s}} + \mathbf{S}\mathbf{p} + \boldsymbol{\varepsilon} \quad \mathbf{p} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Lambda}) \quad \boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \rho^2 \mathbf{I}) \quad (6.6)$$

Denoting the model parameters by $\Phi = \{\bar{\mathbf{s}}, \mathbf{S}, \boldsymbol{\Lambda}, \rho^2\}$, CLM fitting can be defined as a regularized search over the model shape parameters that jointly maximize the probability of all landmark being correctly aligned:

$$\begin{aligned} \mathbf{p}^* &= \arg \max_{\mathbf{p}} p(\mathbf{p}, \{l_i = 1\}_{i=1}^v | I, \mathbf{s}, \Phi) \\ &= \arg \max_{\mathbf{p}} p(\mathbf{p} | \boldsymbol{\Lambda}) p(\{l_i = 1\}_{i=1}^v | \mathbf{p}, I, \mathbf{s}, \Phi) \\ &= \arg \max_{\mathbf{p}} p(\mathbf{p} | \boldsymbol{\Lambda}) \prod_{i=1}^v p(l_i = 1 | \mathbf{p}, I, \mathbf{x}_i, \Phi) \\ &= \arg \max_{\mathbf{p}} \ln p(\mathbf{p} | \boldsymbol{\Lambda}) + \ln \left(\prod_{i=1}^v p(l_i = 1 | \mathbf{p}, I, \mathbf{x}_i, \Phi) \right) \end{aligned} \quad (6.7)$$

Several approaches [5] have been proposed to approximate the true response maps $p(l_i = 1 | I, \mathbf{x}_i)$. The most popular one is the non-parametric approach of [5], which approximates the true response maps as:

$$\sum_{\mathbf{y}_j \in \Psi_{\mathbf{x}_i}} p(l_i = 1 | I, \mathbf{y}_i) \mathcal{N}(\mathbf{x}_i, \rho^2 \mathbf{I}) \quad (6.8)$$

where the current landmark positions $\mathbf{x}_i$ are defined in terms of the probabilistic generative shape model defined in equation 6.6.

Substituting equation 6.8 into equation 6.7 leads to the following optimiza-

tion problem:

$$\begin{aligned}
\mathbf{p}^* &= \arg \min_{\mathbf{p}} \underbrace{-\ln p(\mathbf{p}|\Lambda)}_{\mathcal{R}(\mathbf{p})} + \underbrace{-\sum_{i=1}^v \ln p(l_i = 1|\mathbf{p}, I, \mathbf{x}_i, \Phi)}_{\sum_{i=1}^v \mathcal{D}_i(\mathbf{x}_i, I, \mathbf{p})} \\
&= \arg \min_{\mathbf{p}} \underbrace{\|\mathbf{p}\|_{\Lambda^{-1}}^2}_{\mathcal{R}(\mathbf{p})} + \underbrace{\sum_{\mathbf{x}_i \in \mathbf{s}} \sum_{\mathbf{y}_j \in \Psi_{\mathbf{x}_i}} \frac{w_{y_j}}{\rho^2} \|\mathbf{x}_i - \mathbf{y}_j\|^2}_{\sum_{i=1}^v \mathcal{D}_i(\mathbf{x}_i, I, \mathbf{p})}
\end{aligned} \tag{6.9}$$

which is equivalent to the optimization problem defined by equation 6.5, where the response maps are evaluated at all pixels positions $\{\mathbf{y}_j\}_{j=1}^k$ of the local image patches $\Psi_{\mathbf{x}_i}$.

Treating the true landmark positions $\mathbf{y}_i$ as latent variables, equation 6.9 can be solved iteratively using a specific variation of the *expectation maximization* (EM) algorithm commonly known as the *regularized landmarks mean-shift* (RLMS) algorithm [5]. The solution for the optimal $\mathbf{p}^*$ is given by the following expressions at each iteration:

$$\begin{aligned}
\boldsymbol{\mu}_i &= \sum_{\mathbf{y}_j \in \Psi_{\mathbf{x}_i}} \frac{\pi_{y_j} \mathcal{N}(\mathbf{y}_j, \mathbf{x}_i, \rho^2 \mathbf{I})}{\sum_{\mathbf{z}_j \in \Psi_{\mathbf{x}_i}} \pi_{z_j} \mathcal{N}(\mathbf{z}_j, \mathbf{x}_i, \rho^2 \mathbf{I})} \mathbf{y}_j \\
\Delta \mathbf{p}^* &= -(\Lambda^{-1} + \frac{1}{\rho^2} \mathbf{J}^T \mathbf{J})^{-1} \\
&\quad (\Lambda^{-1} \mathbf{p} - \frac{1}{\rho^2} \mathbf{J}^T (\boldsymbol{\mu} - \mathbf{s})) \\
\mathbf{p}^* &\leftarrow \mathbf{p} + \Delta \mathbf{p}^*
\end{aligned} \tag{6.10}$$

where $\boldsymbol{\mu} = (\boldsymbol{\mu}_1^T, \dots, \boldsymbol{\mu}_n^T)^T$ and $\mathbf{J}$ is the Jacobian of the shape model. For a detailed derivation of these equations the interested reader is referred to [5].

6.3 A unified approach for AAM and CLM fitting

In order to derive a fitting strategy that combines the CLM fitting algorithm with the *compositional gradient descent* (CGD) algorithms for AAM fitting presented in chapter 5, a novel probabilistic interpretation of the deformable model fitting problem is derived. In particular, the fitting problem is redefined as the maximization of the joint probability distribution of the shape $\mathbf{p}$ and texture parameters $\mathbf{c}$, the image I and the aligned landmarks $\{l_i = 1\}_{i=1}^v$ given the model parameters $\Delta = \{\Theta^3, \Phi\}$:

$$\begin{aligned}
\mathbf{p}^*, \mathbf{c}^* &= \arg \max_{\mathbf{p}, \mathbf{c}} p(\mathbf{p}, \mathbf{c}, \mathbf{i}[\mathbf{p}], \{l_i = 1\}_{i=1}^v | \Delta) \\
&= \arg \max_{\mathbf{p}, \mathbf{c}} p(\mathbf{p} | \Lambda) p(\mathbf{c} | \Sigma) p(\mathbf{i}[\mathbf{p}], \{l_i = 1\}_{i=1}^v | \mathbf{p}, \mathbf{c}, \Delta) \\
&= \arg \max_{\mathbf{p}, \mathbf{c}} p(\mathbf{p} | \Lambda) p(\mathbf{c} | \Sigma) \\
&\quad p(\mathbf{i}[\mathbf{p}] | \mathbf{p}, \mathbf{c}, \Theta) p(\{l_i = 1\}_{i=1}^v | I, \mathbf{p}, \Phi) \\
&= \arg \max_{\mathbf{p}, \mathbf{c}} \ln p(\mathbf{p} | \Lambda) + \ln p(\mathbf{c} | \Sigma) + \\
&\quad \ln p(\mathbf{i}[\mathbf{p}] | \mathbf{p}, \mathbf{c}, \Theta) + \ln p(\{l_i = 1\}_{i=1}^v | I, \mathbf{p}, \Phi)
\end{aligned} \tag{6.11}$$

This probabilistic formulation poses the problem of deformable fitting as a regularized search for the optimal shape $\mathbf{p}$ and texture $\mathbf{c}$ parameters that jointly minimize both a *global* misalignment measure that depends simultaneously on all landmarks and a set of v independent *local* measures of

³Recall that, Θ denotes the AAM parameters defined in section 5.2.1.1

misalignment associated to each landmark, i.e.:

$$\begin{aligned} \mathbf{p}^*, \mathbf{c}^* = \arg \min_{\mathbf{p}, \mathbf{c}} & \underbrace{-\ln p(\mathbf{p}|\mathbf{\Lambda}) - \ln p(\mathbf{c}|\mathbf{\Sigma})}_{\mathcal{R}(\mathbf{p}, \mathbf{c})} + \\ & \underbrace{-\ln p(\mathbf{i}[\mathbf{p}]|\mathbf{p}, \mathbf{c}, \Theta)}_{\mathcal{D}(I, \mathbf{p}, \mathbf{c})} + \underbrace{-\ln p(\{l_i = 1\}_{i=1}^v | I, \mathbf{p}, \Theta)}_{\sum_{i=1}^v \mathcal{D}_i(\mathbf{x}_i, I, \mathbf{p})} \end{aligned} \quad (6.12)$$

where $\mathcal{R}(\mathbf{p}, \mathbf{c})$ corresponds to a regularization term that penalizes complex shape and texture deformations, $\mathcal{D}(I, \mathbf{p}, \mathbf{c})$ denotes the global misalignment measure and corresponds to the data term used in AAM fitting and $\sum_{i=1}^n \mathcal{D}_i(\mathbf{x}_i, \mathcal{I}, \mathbf{p})$ denote the v local measures of misalignment which correspond to the data term used in CLM fitting.

Substituting the above terms for their expressions in equations 5.6⁴ and 6.9, the probabilistic formulation can be rewritten as the following optimization problem:

$$\begin{aligned} \mathbf{p}^*, \mathbf{c}^* = \arg \min_{\mathbf{p}, \mathbf{c}} & \underbrace{\|\mathbf{p}\|_{\mathbf{\Lambda}^{-1}}^2 + \|\mathbf{c}\|_{\mathbf{\Sigma}^{-1}}^2}_{\mathcal{R}(\mathbf{p}, \mathbf{c})} + \\ & \underbrace{\frac{1}{\sigma^2} \|\mathbf{i}[\mathbf{p}] - \bar{\mathbf{a}} + \mathbf{A}\mathbf{c}\|^2}_{\mathcal{D}(I, \mathbf{p}, \mathbf{c})} + \\ & \underbrace{\sum_{\mathbf{x}_i \in \mathbf{s}} \sum_{\mathbf{y}_j \in \Psi_{\mathbf{x}_i}} \frac{w_j}{\rho^2} \|\mathbf{x}_i - \mathbf{y}_j\|^2}_{\sum_{i=1}^v \mathcal{D}_i(\mathbf{x}_i, I, \mathbf{p})} \end{aligned} \quad (6.13)$$

Notice that, this unified formulation naturally accommodates for uncertainty (noise) with respect to both shape and texture by explicitly incorporating σ^2 and ρ^2 to the cost function. In fact, the ratio $\frac{\sigma^2}{\rho^2}$ determines the relative contribution of the AAM and CLM data terms to the final cost function⁵.

⁴Note that the probabilistic formulation of the SSD cost function has been used here.

⁵Although theoretically optimal values for σ^2 and ρ^2 can be used, in practice, a better performant algorithm can be obtained by computing the value of the ratio $\frac{\sigma^2}{\rho^2}$ experimentally by using cross-validation on a small validation set.

Equation 6.13 can be optimized by combining the RLMS algorithm for CLM fitting described in section 6.2.1.1 with the CGD algorithms for fitting AAM presented in chapter 5. For example, combining the *project-out inverse compositional* (PIC)⁶ and the RLMS algorithms, the optimal solution for the incremental shape parameters $\Delta \mathbf{p}^*$ is given by:

$$\Delta \mathbf{p}^* = -\mathbf{H}^{-1} \mathbf{b} \quad (6.14)$$

where:

$$\begin{aligned} \mathbf{H} &= \mathbf{\Lambda}^{-1} + \frac{1}{\sigma^2} \mathbf{J}_a^T \mathbf{P} \mathbf{J}_a + \frac{1}{\rho^2} \mathbf{J}_s^T \mathbf{J}_s \\ \mathbf{b} &= \mathbf{\Lambda}^{-1} \mathbf{p} - \frac{1}{\sigma^2} \mathbf{J}_a^T \mathbf{P} (\mathbf{i}[\mathbf{p}] - \bar{\mathbf{a}}) - \frac{1}{\rho^2} \mathbf{J}_s^T (\boldsymbol{\mu} - \mathbf{s}) \end{aligned} \quad (6.15)$$

and $\mathbf{J}_a$ and $\mathbf{J}_s$ are the Jacobians defined in 5.45 and 6.10 respectively.

Alternatively, by combining the *alternated inverse compositional* (AIC)⁷ with the RLMS algorithms the optimal value for $\Delta \mathbf{p}^*$ is again given by 6.14 where, in this case:

$$\begin{aligned} \mathbf{H} &= \mathbf{\Lambda}^{-1} + \frac{1}{\sigma^2} \mathbf{J}_a^T \mathbf{J}_a + \frac{1}{\rho^2} \mathbf{J}_s^T \mathbf{J}_s \\ \mathbf{b} &= \mathbf{\Lambda}^{-1} \mathbf{p} - \frac{1}{\sigma^2} \mathbf{J}_a^T (\mathbf{i}[\mathbf{p}] - \mathbf{a}_a) - \frac{1}{\rho^2} \mathbf{J}_s^T (\boldsymbol{\mu} - \mathbf{s}) \end{aligned} \quad (6.16)$$

and $\mathbf{J}_a$ is defined in 5.45. Note that the solution for the optimal texture parameters $\mathbf{c}^*$ is again given by 4.24 and that both algorithm still utilize the exact same update rules defined in 5.24 and 6.10.

⁶Note that the PIC algorithms is referred to as *PO Inverse Gauss-Newton* algorithm in the unified CGD framework proposed in chapter 5.

⁷Note that the AIC algorithms is referred to as *SSD Inverse Gauss-Newton* algorithm in the unified CGD framework proposed in chapter 5

6.4 Results

This section reports the performance of the *unified* fitting algorithms on the problem of non-rigid face alignment in-the-wild. Results for two different experiments are reported. The first experiment compares the accuracy and convergence properties of the proposed unified PIC-RLMS and AIC-RLMS algorithms with respect to the well known PIC [69], AIC [34] algorithms for fitting AAM and the RLMS [5] algorithm for CLM fitting, on the popular *labelled faces parts in-the-wild* (LFPW) [1] and *Helen* [14] datasets. The second experiment compares the performance of these algorithms against two state-of-the-art methods for non-rigid face alignment in-the-wild, i.e. the *supervised descent method* (SDM) of Xiong and De La Torre [20] and the *Gauss-Newton deformable parts model* (GNDPM) of Tzimiropoulos and Pantic [94], on the very challenging *annotated faces in-the-wild* (AFW) dataset.

In all experiments, the images of the LFPW training dataset (currently 813 images) are used to train the unified algorithms presented in this chapter (PIC-RLMS and AIC-RLMS) as well as the base-line algorithms for AAM (PIC and AIC) and CLM (RLMS).

These algorithms are implemented using a *Gaussian pyramid* with 2 levels (face images are normalized to have a face size of roughly 100 pixels on the top resolution level). Similar to [94] a reduced version of the *dense scale invariant feature transform* (DSIFT) [58, 95] is used to define the image representation of both holistic and parts-based texture models. For holistic generative texture models the number of texture components is set to 50 and kept constant throughout the optimization. *Multi-channel correlation*

filters (MCCF) [99] are used to learn parts-based discriminative texture models. The size of each local patch is set to 17×17 and kept constant throughout the optimization. The dimensionality of the 2d shape model is set to 7 (4 similarity parameters + 3 nonrigid shape components) at the low resolution level and to 16 (4 + 12) at the top resolution one. Finally, the noise ratio between texture and shape, σ/ρ , was experimentally obtained by cross-validation on a small part of Helen training dataset (which was not used in any of the experiments to either train or test the model).

Finally, a public implementations of all algorithms is available as part of the Menpo Project⁸ [54].

6.4.1 Comparison with AAM and CLM

Results for this experiment are reported over the 224 and 330 test images of the LFPW [1] and Helen[14] datasets. 66 points ground truth landmark annotations were obtained from the iBUG group website⁹. All methods were initialized by perturbing the ground truth scale and translation parameters with Gaussian noise (rotations were not considered) and applying the resulting transformation to mean of the shape model. (Notice that this procedure produces initializations that are considerably more challenging than those reported in the recent AAM literature [40, 94]). The *cumulative error distribution* (CED) curves for this experiment are shown in figures 6.1a and 6.1c. Figures 6.1b and 6.1d show the evolution of the mean normalized point-to-point error as a function of the number of iterations run by each algorithm. This experiment shows that the proposed unified AIC-RLMS approach considerably outperforms all other methods by a large margin on

⁸<http://www.menpo.org/>.

⁹<http://ibug.doc.ic.ac.uk/resources/300-W/>

both datasets. More specifically, AIC-RLMS achieves a constant improvement of between 10% to 20% over PIC-RLMS and AIC at the significant region $0.020 < error > 0.040$ (at which the results are generally considered adequate by visual inspection). Note that the fast unified PIC-RLMS algorithm is also the second most performant algorithm, surpassing both AIC and RLMS, on this particular experiment. Finally, the average run-time for each one of the algorithms (20 iteration per image), on a laptop equipped with a 2.3GHz quad-core Intel Core i7 processor is reported in table 6.1e.

6.4.2 Comparison with state-of-the-art

Results for this experiment are reported over the entire 337 images of the AFW [4] dataset. In this case, 49 points ground truth landmark annotations for this dataset were again provided by the iBUG group⁹. Results for [20] and [94] were directly obtained using the publicly available models and fitting code kindly provided by the authors^{10,11}. Note that, the provided models have been potentially trained using thousands of images in contrast to the only 813 images used to train the proposed methods. In this experiment, all algorithms were initialized using the bounding box provided by our own in-house implementation of the face detector of [4]. The CED curves for this experiment are shown in Figure 6.1f. The results show that the proposed unified AIC-RLMS algorithm achieves state-of-the-art results on the AFW dataset, considerably outperforming both the GNDPM¹² method of Tzimiropoulos and Pantic [94] (which can be extremely accurate but sensitive to inaccurate initializations) and the SDM method of Xiong and De la

¹⁰<http://www.humansensing.cs.cmu.edu/intraface/>

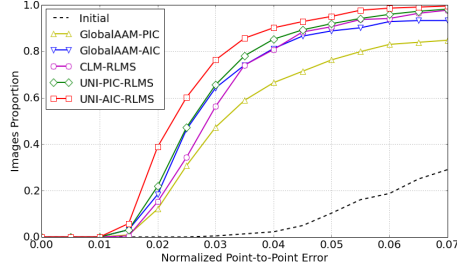
¹¹http://www.cs.nott.ac.uk/~pszyt/code/gn_dpm_code_v1.zip

¹²This model is essentially an AAM that uses the sampling-based deformation model proposed in [60] to sample local texture regions around each landmark.

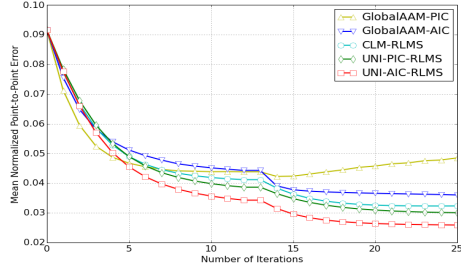
Torre [20] (which can deal with very noisy initializations but is significantly less accurate than our method).

6.5 Conclusions

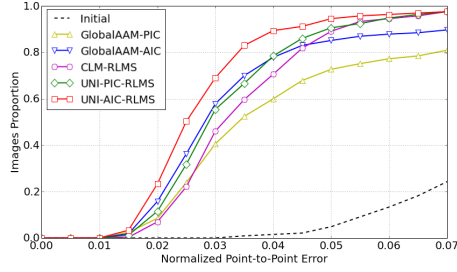
This chapter presents a novel approach to deformable model fitting that unifies the well-established frameworks of AAM and CLM fitting. This approach naturally arises from a new probabilistic formulation of the fitting problem from which two novel deformable model fitting algorithms are derived, PIC-RLMS and AIC-RLMS. The proposed algorithms compare favorably, in terms of fitting accuracy, against previous methods for AAM and CLM fitting and against two recently proposed state-of-the-art techniques on the challenging problem of non-rigid face alignment in-the-wild.



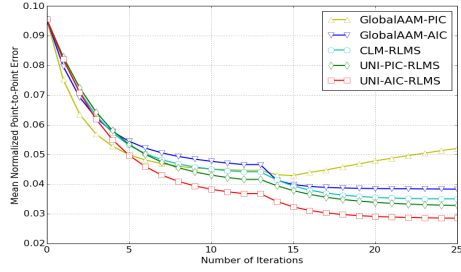
(a) Fitting accuracy on the LFPW dataset.



(b) Fitting convergence on the LFPW dataset.



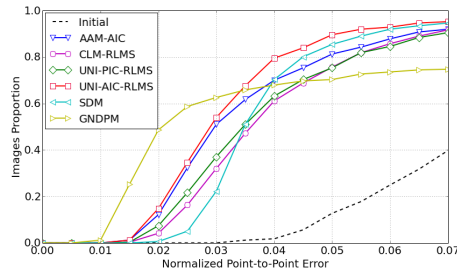
(c) Fitting accuracy on the Helen dataset.



(d) Fitting convergence on the Helen dataset.

Algorithm	PIC	AIC	RLMS	PIC-RLMS	AIC-RLMS
Run-time	80 ms	130 ms	100 ms	110 ms	140 ms

(e) Table reporting the average run-time for the unified algorithms presented in this chapter and the base-line algorithms for AAM and CLM using their public implementations available as part of the Menpo Project⁸. Each method was run for 20 iteration and run-time was measured on a laptop equipped with a 2.3GHz quad-core Intel Core i7 processor.



(f) Fitting accuracy on the AFW dataset.

Figure 6.1: Results showing the fitting accuracy and convergence properties together with average run-time for the unified, AAM and CLM algorithms on the LFPW and Helen test datasets; and the fitting accuracy for the previous algorithms as well as for SDM and GNDPM on the entire AFW dataset.

Chapter 7

Conclusion

This thesis presents a detailed and complete study of *compositional fitting algorithms* (CGD) for fitting *active appearance models* (AAM). In particular, three different parts of the AAM fitting problem are studied:

- *texture* representations.
- *optimization* algorithms.
- *combination* with complementary methods in object alignment.

Chapters 4 explores the use of a robust texture representation based on image gradient orientations [43] to define the texture model of a new kind of deformable model built upon AAM. As shown in the result sections of this chapters, the proposed *active orientation models* (AOM) generalize better than classic AAM to variations in identity, pose, expression, illumination and occlusions and lead to substantially better results in the problem of non-rigid face alignment.

Chapter 5 proposes a unified framework for compositional fitting of AAM built around three of the main characteristics of CGD algorithms: (i) the cost function defining the fitting problem; (ii) the type of composition used; and (iii) the optimization method employed to solve the non-linear optimization problem. This chapter also presents a novel *Bayesian formulation* of the AAM fitting problem that can be interpreted as a probabilistic generalization of the well-known *project-out inverse compositional* (PIC) algorithm [10]. Moreover, chapter 5 also introduces the use of *asymmetric* and *bidirectional* compositions in AAM and shows how these types of composition lead to better convergent and more robust CGD fitting algorithms. Lastly, a final contribution of this chapter is to relate existent ad-hoc strategies used in the derivation of fast and exact simultaneous CGD algorithms [34, 37, 40] to well-known techniques in the nonlinear optimization literature (*Schur complement* [48] and the *Wiberg method* [49, 50]).

Finally, chapter 6 combines the fitting procedures of AAM and *constrained local models* (CLM) [6, 5] in a unified fitting framework that combines the strengths of both models. The proposed approach explicitly and optimally combines the generative and discriminative texture models of AAM and CLM using a *maximum a posteriori* (MAP) estimation framework. This results in a unified cost function that is iteratively solved using the Gauss-Newton method and in which the solution at each iteration is given by an optimal, in a MAP sense, weighted combination of the original AAM and CLM iterative solutions. The results presented in this chapter show that the proposed combined approach compares favorably against previous methods for AAM and CLM fitting and, more importantly, against recently proposed discriminatively trained state-of-the-art methods in non-rigid face alignment in-the-wild.

7.1 Future Work

The area of *non-rigid object alignment* is one the most active areas of research in computer vision and machine learning. In recent years, the advances produced by *deep learning* have revolutionized the field (and computer vision and machine learning in general) and pushed the state-of-the-art forward. An obvious direction of future research would consist in replacing the engineered feature descriptors used in [57] to define the texture representation of AAM for the robust feature representation obtained from *deep convolutional neural networks*. An alternative, and possible, more interesting research direction would be to replace the PCA based models of shape and texture typically used in AAM by *deep generative models* such as *denoising autoencoders* or *deep Boltzmann machines* (DBM). Note that, preliminary steps in this direction were recently presented by Duong et al. in [28]. Finally, another interesting area for future research would be to incorporate more discriminative forms of probabilistic generative models [90, 91] into the probabilistic AAM framework presented in chapter 5. This could lead to AAM models with the potential of solving complementary tasks such as face and expression recognition via face alignment.

Bibliography

- [1] P. N. Belhumeur, D. W. Jacobs, D. J. Kriegman, and N. Kumar, “Localizing parts of faces using a consensus of exemplars,” in *IEEE Conference on computer vision and pattern recognition (CVPR)*, 2011. [xv](#), [1](#), [8](#), [11](#), [14](#), [89](#), [116](#), [125](#), [126](#)
- [2] C. Sagonas, G. Tzimiropoulos, S. Zafeiriou, and M. Pantic, “300 faces in-the-wild challenge: The first facial landmark localization challenge,” in *IEEE International conference on computer vision workshop (ICCV-W)*, 2013. [xv](#), [14](#), [90](#)
- [3] C. Sagonas, G. Tzimiropoulos, S. Zafeiriou, and M. Pantic, “A semi-automatic methodology for facial landmark annotation,” in *IEEE Conference on computer vision and pattern recognition workshop (CVPR-W)*, pp. 896–903, June 2013. [xv](#), [14](#), [90](#)
- [4] X. Zhu and D. Ramanan, “Face detection, pose estimation, and landmark localization in the wild,” in *IEEE Conference on computer vision and pattern recognition (CVPR)*, 2012. [xvi](#), [1](#), [8](#), [11](#), [39](#), [41](#), [42](#), [43](#), [44](#), [90](#), [116](#), [127](#)
- [5] J. M. Saragih, S. Lucey, and J. F. Cohn, “Deformable model fitting by regularized landmark mean-shift,” *International journal of computer*

- vision (IJCV)*, 2011. xvi, 1, 6, 8, 10, 11, 39, 40, 41, 42, 44, 116, 117, 118, 119, 120, 121, 125, 131
- [6] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham, “Active shape models: Their training and application,” *Computer vision and image understanding*, 1995. 1, 6, 8, 10, 116, 117, 118, 131
- [7] V. Blanz and T. Vetter, “A morphable model for the synthesis of 3d faces,” in *SIGGRAPH*, 1999. 1, 8, 9, 116
- [8] T. F. Cootes, G. J. Edwards, and C. J. Taylor, “Active appearance models,” *IEEE Transactions on pattern analysis and machine intelligence (TPAMI)*, 2001. 1, 2, 8, 9, 14, 19, 21, 22, 29, 50, 116
- [9] T. F. Cootes and C. J. Taylor, “On representing edge structure for model matching,” in *IEEE Conference on computer vision and pattern recognition (CVPR)*, 2001. 1, 8, 9, 29, 116
- [10] I. Matthews and S. Baker, “Active appearance models revisited,” *International journal of computer vision (IJCV)*, 2004. 1, 2, 3, 5, 8, 9, 14, 15, 18, 19, 20, 21, 22, 24, 29, 44, 46, 47, 50, 51, 56, 59, 62, 84, 85, 97, 102, 116, 131, 167
- [11] P. Tresadern, P. Sauer, and T. Cootes, “Additive update predictors in active appearance models,” in *British machine vision conference (BMVC)*, 2010. 1, 2, 8, 9, 21, 22, 116
- [12] J. Saragih and R. Goecke, “A nonlinear discriminative approach to aam fitting,” in *IEEE International conference on computer vision (ICCV)*, 2007. 1, 8, 9, 116
- [13] J. Saragih, “Principal regression analysis,” in *IEEE Conference on computer vision and pattern recognition (CVPR)*, 2011. 1, 8, 11, 116

- [14] V. Le, B. Jonathan, Z. Lin, L. Boudev, and T. S. Huang, “Interactive facial feature localization,” in *European conference on computer vision (ECCV)*, 2012. 1, 8, 11, 90, 125, 126
- [15] X. Cao, Y. Wei, F. Wen, and J. Sun, “Face alignment by explicit shape regression,” in *IEEE Conference on computer vision and pattern recognition (CVPR)*, 2012. 1, 8, 12, 116
- [16] S. Rivera and A. Martinez, “Learning deformable shape manifolds,” *Pattern recognition (PR)*, 2012. 1, 8, 12, 116
- [17] J. Yan, Z. Lei, D. Yi, and S. Li, “Learn to combine multiple hypotheses for accurate face alignment,” in *IEEE International conference on computer vision workshop (ICCV-W)*, 2013. 1, 8, 12, 116
- [18] Y. Sun, X. Wang, and X. Tang, “Deep convolutional network cascade for facial point detection,” in *IEEE Conference on computer vision and pattern recognition (CVPR)*, 2013. 1, 8, 12, 116
- [19] X. P. Burgos-Artizzu, P. Perona, and P. Dollr, “Robust face landmark estimation under occlusion,” in *IEEE International conference on computer vision (ICCV)*, 2013. 1, 8, 12, 116
- [20] X. Xiong and F. De la Torre, “Supervised descent method and its applications to face alignment,” in *IEEE Conference on computer vision and pattern recognition (CVPR)*, 2013. 1, 8, 12, 100, 116, 125, 127, 128
- [21] S. Lucey, R. Navarathna, A. B. Ashraf, and S. Sridharan, “Fourier lucas-kanade algorithm,” *IEEE Transactions on pattern analysis and machine intelligence (TPAMI)*, 2013. 1, 8, 10, 29, 116

- [22] A. Asthana, S. Zafeiriou, S. Cheng, and M. Pantic, “Robust discriminative response map fitting with constrained local models,” in *IEEE Conference on computer vision and pattern recognition (CVPR)*, 2013. 1, 8, 11, 100, 116, 117, 119
- [23] P. Martins, R. Caseiro, and J. Batista, “Non-parametric bayesian constrained local models,” in *IEEE Conference on computer vision and pattern recognition (CVPR)*, 2014. 1, 8, 11, 116, 117, 119
- [24] T. Baltruaitis, P. Robinson, and L.-P. Morency, “Continuous conditional neural fields for structured regression,” in *European conference on computer vision (ECCV)*, 2014. 1, 8, 11, 116, 117
- [25] A. Asthana, S. Zafeiriou, S. Cheng, and M. Pantic, “Incremental face alignment in the wild,” in *IEEE Conference on computer vision and pattern recognition (CVPR)*, 2014. 1, 8, 12, 116
- [26] V. Kazemi and J. Sullivan, “One millisecond face alignment with an ensemble of regression trees,” in *IEEE Conference on computer vision and pattern recognition (CVPR)*, 2014. 1, 8, 12, 116
- [27] S. Ren, X. Cao, Y. Wei, and J. Sun, “Face alignment at 3000 fps via regressing local binary features,” in *IEEE Conference on computer vision and pattern recognition (CVPR)*, 2014. 1, 8, 12, 116
- [28] C. N. Duong, K. Luu, K. G. Quach, and T. D., “Beyond principal components: Deep boltzmann machines for face modeling,” in *IEEE Conference on computer vision and pattern recognition (CVPR)*, 2015. 1, 8, 10, 132
- [29] D. Lee, H. Park, and C. D. Yoo, “Face alignment using cascade gaus-

- sian process regression trees,” in *IEEE Conference on computer vision and pattern recognition (CVPR)*, 2015. 1, 8, 12
- [30] X. Hou, S. Z. Li, H. Zhang, and Q. Cheng, “Direct appearance models,” in *IEEE Conference on computer vision and pattern recognition (CVPR)*, 2001. 2, 21, 22
- [31] A. Batur and M. Hayes, “Adaptive active appearance models,” *IEEE Transactions on image processing (TIP)*, 2005. 2, 21, 22
- [32] R. Gross, I. Matthews, and S. Baker, “Generic vs. person specific active appearance models,” *Image and vision computing*, 2005. 2, 3, 5, 9, 21, 46, 56, 62, 84, 85, 86
- [33] R. Donner, M. Reiter, G. Langs, P. Peloschek, and H. Bischof, “Fast active appearance model search using canonical correlation analysis,” *IEEE Transactions on pattern analysis and machine intelligence (TPAMI)*, 2006. 2, 21, 22
- [34] G. Papandreou and P. Maragos, “Adaptive and constrained algorithms for inverse compositional active appearance model fitting,” in *IEEE Conference on computer vision and pattern recognition (CVPR)*, 2008. 2, 3, 5, 6, 9, 18, 21, 46, 47, 56, 57, 59, 62, 84, 85, 87, 125, 131, 155
- [35] X. Liu, “Discriminative face alignment,” *IEEE Transactions on pattern analysis and machine intelligence (TPAMI)*, 2009. 2, 21
- [36] J. Saragih and R. Göcke, “Learning aam fitting through simulation,” *Pattern recognition*, 2009. 2, 21, 22
- [37] B. Amberg, A. Blake, and T. Vetter, “On compositional image alignment, with an application to active appearance models,” in *IEEE Con-*

- ference on computer vision and pattern recognition (CVPR)*, 2009. 2, 5, 6, 9, 21, 46, 47, 56, 62, 84, 85, 131, 167
- [38] P. Martins, J. Batista, and R. Caseiro, “Face alignment through 2.5d active appearance models,” in *British machine vision conference (BMVC)*, 2010. 2, 5, 21, 46, 56, 62, 84, 87
- [39] P. Sauer, T. Cootes, and C. Taylor, “Accurate regression procedures for active appearance models,” in *British machine vision conference (BMVC)*, 2011. 2, 21, 22
- [40] G. Tzimiropoulos and M. Pantic, “Optimization problems for fast aam fitting in-the-wild,” in *IEEE International conference on computer vision (ICCV)*, 2013. 2, 3, 5, 6, 9, 21, 46, 47, 56, 62, 84, 85, 87, 100, 126, 131, 167
- [41] J. Kossaifi, G. Tzimiropoulos, and M. Pantic, “Fast newton active appearance models,” in *IEEE International conference on image processing (ICIP)*, 2014. 2, 21, 46, 62, 75, 84, 88, 100, 102
- [42] L. van der Maaten and E. Hendriks, “Capturing appearance variation in active appearance models,” in *IEEE Conference on computer vision and pattern recognition workshop (CVPR-W)*, 2010. 3, 49
- [43] G. Tzimiropoulos, S. Zafeiriou, and M. Pantic, “Subspace learning from image gradient orientations,” *IEEE Transactions on pattern analysis and machine intelligence (TPAMI)*, 2012. 4, 28, 31, 32, 34, 130
- [44] E. Malis, “Improving vision-based control using efficient second-order minimization techniques,” in *International conference on robotics and automation (ICRA)*, 2004. 5, 47, 56, 158

- [45] R. Mégret, J.-B. Authesserre, and Y. Berthoumieu, “The bi-directional framework for unifying parametric image alignment approaches,” in *European conference on computer vision (ECCV)*, 2008. 5, 47, 56, 60, 158
- [46] J.-B. Authesserre, R. Mégret, and Y. Berthoumieu, “Asymmetric gradient-based image alignment,” in *IEEE International conference on acoustics, speech and signal processing (ICASSP)*, 2009. 5, 47, 56, 60
- [47] R. Mégret, J.-B. Authesserre, and Y. Berthoumieu, “Bidirectional composition on lie groups for gradient-based image alignment,” *IEEE Transactions on image processing (TIP)*, 2010. 5, 47, 56, 60, 158
- [48] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004. 6, 38, 47, 62, 65, 131
- [49] T. Okatani and K. Deguchi, “On the wiberg algorithm for matrix factorization in the presence of missing components,” *International journal of computer vision (IJCV)*, 2006. 6, 47, 62, 131
- [50] D. StreLOW, “General and nested wiberg minimization: L2 and maximum likelihood,” in *European conference on computer vision (ECCV)*, 2012. 6, 47, 62, 131
- [51] G. Tzimiropoulos, J. Alabort-i-Medina, S. Zafeiriou, and M. Pantic, “Generic active appearance models revisited,” in *IEEE Asian conference on computer vision (ACCV)*, 2012. 87, 116, 117
- [52] G. Tzimiropoulos, J. Alabort-i-Medina, S. Zafeiriou, and M. Pantic, “Active orientation models for face alignment in-the-wild,” *IEEE Transactions on information forensics and security (TIFS)*, 2014.

- [53] J. Alabort-i-Medina and S. Zafeiriou, “Bayesian active appearance models,” in *IEEE Conference on computer vision and pattern recognition (CVPR)*, 2014. 46, 102, 117
- [54] J. Alabort-i-Medina, E. Antonakos, J. Booth, P. Snape, and S. Zafeiriou, “Menpo: A comprehensive platform for parametric image alignment and visual deformable models,” in *ACM international conference on multimedia (ACMM)*, 2014. 39, 126
- [55] J. Alabort-i-Medina and S. Zafeiriou, “Unifying holistic and parts-based deformable model fitting,” in *IEEE Conference on computer vision and pattern recognition (CVPR)*, 2015. 102
- [56] J. Alabort-i-Medina and S. Zafeiriou, “A unified framework for compositional fitting of active appearance models,” *International journal of computer vision (IJCV)*, 2016. 46, 116, 117
- [57] E. Antonakos, J. Alabort-i-Medina, G. Tzimiropoulos, and S. Zafeiriou, “Feature-based lucas-kanade and active appearance models,” *IEEE Transactions on image processing (TIP)*, 2014. 10, 87, 91, 100, 102, 116, 117, 132
- [58] D. G. Lowe, “Object recognition from local scale-invariant features,” in *IEEE International conference on computer vision (ICCV)*, 1999. 10, 91, 125
- [59] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *IEEE Conference on computer vision and pattern recognition (CVPR)*, 2005. 10
- [60] D. Cristinacce and T. F. Cootes, “Feature detection and tracking

- with constrained local models,” in *British machine vision conference (BMVC)*, 2006. 11, 127
- [61] P. Martins, R. Caseiro, J. F. Henriques, and J. Batista, “Likelihood-enhanced bayesian constrained local models,” in *IEEE International conference on image processing (ICIP)*, 2014. 11, 119
- [62] J. Alabort-i-Medina, “Advances in generic facial model fitting,” tech. rep., Department of computing, Imperial College London, 2012.
- [63] T. F. Cootes, C. J. Taylor, D. Cooper, and J. Graham, “Training models of shape from sets of examples,” in *British machine vision conference (BMVC)*, 1992. 14, 15
- [64] T. F. Cootes and C. J. Taylor, “Active shape modelssmart snakes,” in *British machine vision conference (BMVC)*, 1992. 14
- [65] G. J. Edwards, C. J. Taylor, and T. Cootes, “Learning to identify and track facesin image sequences,” in *British machine vision conference (BMVC)*, 1997. 16
- [66] G. J. Edwards, C. J. Taylor, and T. F. Cootes, “Interpreting face images using active appearance models,” in *IEEE International conference on automatic face and gesture recognition*, 1998. 16
- [67] T. F. Cootes and C. J. Taylor, “Statistical models of appearance for computer vision,” tech. rep., Imaging Science and Biomedical Engineering, University of Manchester, 2004. 16, 18
- [68] B. D. Lucas and T. Kanade, “An iterative image registration technique with an application to stereo vision,” in *International Joint Conference on Artificial Intelligence (IJCAI)*, 1981. 22

- [69] S. Baker and I. Matthews, “Lucas-kanade 20 years on: A unifying framework,” *International journal of computer vision (IJCV)*, 2004. 22, 24, 55, 125
- [70] E. Muñoz, P. Márquez-Neila, and L. Baumela, “Rationalizing efficient compositional image alignment,” *International journal of computer vision (IJCV)*, 2014. 22, 55
- [71] D. Ross, J. Lim, R.-S. Lin, and M.-H. Yang, “Incremental learning for robust visual tracking,” *International journal of computer vision (IJCV)*, 2007. 26, 47, 49
- [72] C. Wolstenholme and C. J. Taylor, “Wavelet compression of active appearance models,” in *International conference on medical image computing and computer-assisted intervention (MICCAI)*, 1999. 29
- [73] I. M. Scott, T. F. Cootes, and C. J. Taylor, “Improving appearance model matching using local image structure,” *Information processing in medical Imaging (IPMI)*, 2003. 29
- [74] M. B. Stegmann and R. Larsen, “Multi-band modelling of appearance,” *Image and vision computing (IVC)*, 2003. 29
- [75] S. Darkner, R. Larsen, M. B. Stegmann, and B. Ersboll, “Wedgelet enhanced appearance models,” in *IEEE Conference on computer vision and pattern recognition workshop (CVPR-W)*, 2004. 29
- [76] P. Kittipanya-ngam and T. Cootes, “The effect of texture representations on aam performance,” in *International conference on pattern recognition (ICPR)*, 2006. 29
- [77] X. Gao, Y. Su, X. Li, and D. Tao, “Gabor texture in active appearance models,” *Journal of neurocomputing (JN)*, 2009. 29

- [78] Y. Su, D. Tao, X. Li, and X. Gao, "Texture representation in aam using gabor wavelet and local binary patterns," 2009. 29
- [79] Y. Ge, D. Yang, J. Lu, B. Li, and X. Zhang, "Active appearance models using statistical characteristics of gabor based texture representation," in *Journal of visual communication and image representation (VCIR)*, 2013. 29
- [80] F. De la Torre and M. J. Black, "A framework for robust subspace learning," *International journal of computer vision (IJCV)*, 2003. 30, 32, 33
- [81] J. Xiao, S. Baker, I. Matthews, and T. Kanade, "Real-time combined 2d+3d active appearance models," in *IEEE Conference on computer vision and pattern recognition (CVPR)*, 2004. 40
- [82] K. Ramnath, S. Baker, I. Matthews, and D. Ramanan, "Increasing the density of active appearance models," in *IEEE Conference on computer vision and pattern recognition (CVPR)*, 2008. 40
- [83] A. Patel and W. Smith, "3d morphable face models revisited," in *IEEE Conference on computer vision and pattern recognition (CVPR)*, 2009. 40
- [84] K. Messer, J. Matas, J. Kittler, J. Luetttin, and G. Maitre, "Xm2vtsdb: The extended m2vts database," in *International Conference on Audio and Video-Based Biometric Person Authentication*, 2009. 40
- [85] R. Gross, I. Matthews, J. Cohn, and T. Kanade, "Multi-pie," *Image and vision computing*, 2010. 40
- [86] M. E. Tipping and C. M. Bishop, "Probabilistic principal component

- analysis,” *Journal of the royal statistical society: series B (statistical methodology)*, 1999. 47, 49
- [87] S. Roweis, “Em algorithms for pca and spca,” *Advances in neural information processing systems (NIPS)*, 1998. 47, 49
- [88] B. Moghaddam and A. Pentland, “Probabilistic visual learning for object representation,” *IEEE Transactions on pattern analysis and machine intelligence (TPAMI)*, 1997. 47, 49, 119
- [89] F. Bach and M. Jordan, “A probabilistic interpretation of canonical correlation analysis,” tech. rep., Department of Statistics, University of California, Berkeley, 2005. 49
- [90] S. Prince, P. Li, Y. Fu, U. Mohammed, and J. H. Elder, “Probabilistic models for inference about identity,” *IEEE Transactions on pattern analysis and machine intelligence (TPAMI)*, 2012. 49, 132
- [91] M. A. Nicolaou, S. Zafeiriou, and P. Pantic, “A unified framework for probabilistic component analysis,” in *Machine learning and knowledge discovery in databases (ECML PKDD)*, 2014. 49, 132
- [92] M. A. Woodbury, *Inverting Modified Matrices*. Princeton University, 1950. 53
- [93] F. De la Torre, “A least-squares framework for component analysis,” *IEEE Transactions on pattern analysis and machine intelligence (TPAMI)*, 2012. 69
- [94] G. Tzimiropoulos and M. Pantic, “Gauss-newton deformable part models for face alignment in-the-wild,” in *IEEE Conference on computer vision and pattern recognition (CVPR)*, 2014. 90, 91, 101, 102, 125, 126, 127

- [95] C. Liu, J. Yuen, and A. Torralba, “Sift flow: Dense correspondence across scenes and its applications,” *IEEE Transactions on pattern analysis and machine intelligence (TPAMI)*, 2011. 91, 125
- [96] A. Vedaldi and B. Fulkerson, “VLFeat: An open and portable library of computer vision algorithms,” in *ACM international conference on multimedia (ACMM)*, 2010. 91
- [97] G. Bradski, “The opencv library,” *Dr. Dobb’s journal of software tools*, 2000. 92
- [98] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” in *IEEE Conference on computer vision and pattern recognition (CVPR)*, 2001. 92
- [99] H. Kiani Galoogahi, T. Sim, and S. Lucey, “Multi-channel correlation filters,” in *IEEE International conference on computer vision (ICCV)*, 2013. 126
- [100] F. J. Bookstein, “Principal warps: Thin-plate splines and the decomposition of deformations,” *IEEE Transactions on pattern analysis and machine intelligence (TPAMI)*, 1989. 149
- [101] S. Benhimane and E. Malis, “Real-time image-based tracking of planes using efficient second-order minimization,” in *IEEE International conference on intelligent robots and systems (IROS)*, 2004. 158

Appendices

Appendix A

Deformation models

This chapter provides a detailed explanation of the two most common deformation models used in AAM, i.e. *piece-wise affine* (PWA) and *thin plate spline* (TPS) warps.

A.1 Warp

Remember that, as described in section 3.1.3, the role of the deformation model is to map the position of all pixel positions $\mathbf{x} \in \Omega$ on the reference frame from one shape instance $\mathbf{s}_i$ to another $\mathbf{s}_j$ based on the relative position of the sparse set of landmarks defining both shapes (for which direct correspondences are always known).

A.1.1 Piece-wise affine

Arguably, the easiest way to define such a mapping is by using a PWA warp. Given two shapes and a shared triangulation of their vertices, PWA

describes the differences between the positions of their vertices as local affine transformations between corresponding pairs of triangles.

More specifically, assuming that a particular triangle t is defined by the vertices $(x_i^0, y_i^0)^T$, $(x_j^0, y_j^0)^T$ and $(x_k^0, y_k^0)^T$ in the reference frame Ω , then every point $\mathbf{x} = (x, y)^T$ inside t can be expressed as a linear combination of the previous vertices:

$$\mathbf{x} = (x, y)^T = (x_i^0, y_i^0)^T + \alpha[(x_j^0, y_j^0)^T - (x_i^0, y_i^0)^T] + \beta[(x_k^0, y_k^0)^T - (x_i^0, y_i^0)^T] \quad (\text{A.1})$$

where:

$$\alpha = \frac{(x - x_i^0)(y_k^0 - y_i^0) - (y - y_i^0)(x_k^0 - x_i^0)}{(x_j^0 - x_i^0)(y_k^0 - y_i^0) - (y_j^0 - y_i^0)(x_k^0 - x_i^0)} \quad (\text{A.2})$$

and

$$\beta = \frac{(y - y_i^0)(x_j^0 - x_i^0) - (x - x_i^0)(y_j^0 - y_i^0)}{(x_j^0 - x_i^0)(y_k^0 - y_i^0) - (y_j^0 - y_i^0)(x_k^0 - x_i^0)} \quad (\text{A.3})$$

The scalars α and β uniquely define the relative position of $\mathbf{x}$ on this triangle and they can be used to map $\mathbf{x}$ onto a particular shape $\mathbf{s}$ by using:

$$\mathcal{W}(\mathbf{x}, \mathbf{p}) = (x_i, y_i)^T + \alpha[(x_j, y_j)^T - (x_i, y_i)^T] + \beta[(x_k, y_k)^T - (x_i, y_i)^T] \quad (\text{A.4})$$

where $(x_i, y_i)^T$, $(x_j, y_j)^T$ and $(x_k, y_k)^T$ are the vertices defining the previous triangle t in $\mathbf{s}$.

The previous equations defines a simple affine warp between each pair of corresponding triangles in $\mathbf{s}_0$ and $\mathbf{s}$ and, therefore, can also be expressed as:

$$\mathcal{W}(\mathbf{x}, \mathbf{p}) = (a_1 + a_2 \cdot x + a_3 \cdot y, a_4 + a_5 \cdot x + a_6 \cdot y)^T \quad (\text{A.5})$$

A.1.2 Thin plate splines

Another popular technique that leads to a globally smooth warp is TPS [100]. Mathematically, a TPS warp between two shapes, $\bar{\mathbf{s}}$ and $\mathbf{s}$, can be written in the form of a generalized linear model:

$$\mathcal{W}(\mathbf{x}; \mathbf{p}) = \underbrace{\boldsymbol{\Psi}(\mathbf{p})}_{2 \times (v+3)} \underbrace{\mathbf{k}(\mathbf{x})}_{(v+3) \times 1} \quad (\text{A.6})$$

where $\mathbf{k}(\mathbf{x})$ is given by:

$$\mathbf{k}(\mathbf{x}) = [U(|\mathbf{x} - \mathbf{x}_1|), U(|\mathbf{x} - \mathbf{x}_2|), \dots, U(|\mathbf{x} - \mathbf{x}_v|), 1, x, y]^T \quad (\text{A.7})$$

In the previous equation $U(r) = r^2 \ln r^2$ is the spline kernel and $\mathbf{x}_i$ denotes the vertices (x_i, y_i) of the *source* shape $\bar{\mathbf{s}}$ (which in this case is set to the mean shape). The matrix of weights $\boldsymbol{\Psi}(\mathbf{p})$ defines the warp and it is determined by requiring that the source landmarks $\bar{\mathbf{s}}$ map to the *target* landmarks $\mathbf{s}$.

The weighting matrix $\boldsymbol{\Psi}(\mathbf{p})$ can be obtained following the approach described by Bookstein in [100]. Writing:

$$r_{ij} = |\bar{\mathbf{s}}^{\mathbf{x}_i} - \bar{\mathbf{s}}^{\mathbf{x}_j}| \quad (\text{A.8})$$

where $\bar{\mathbf{s}}^{\mathbf{x}_i}$ and $\bar{\mathbf{s}}^{\mathbf{x}_j}$ denote the *i*th and *j*th vertices of the mean shape $\bar{\mathbf{s}}$

respectively. And defining:

$$\begin{aligned}
 \mathbf{K} &= \begin{bmatrix} 0 & U(r_{12}) & \dots & U(r_{1v}) \\ U(r_{21}) & 0 & \dots & U(r_{1v}) \\ \vdots & \vdots & \ddots & \vdots \\ U(r_{v1}) & U(r_{v2}) & \dots & 0 \end{bmatrix} \\
 \mathbf{S} &= \begin{bmatrix} 1 & x_1^0 & y_1^0 \\ 1 & x_2^0 & y_2^0 \\ \vdots & \vdots & \vdots \\ 1 & x_v^0 & y_v^0 \end{bmatrix} \\
 \mathbf{T} &= \begin{bmatrix} x_1 & x_2 & \dots & 0 & 0 & 0 \\ y_1 & y_2 & \dots & 0 & 0 & 0 \end{bmatrix} \\
 \mathbf{L} &= \begin{bmatrix} \mathbf{K} & \mathbf{S} \\ \mathbf{S}^T & \mathbf{O} \end{bmatrix}
 \end{aligned} \tag{A.9}$$

where $\mathbf{O}$ is a 3×3 matrix of zeros and (x_i^0, y_i^0) and (x_i, y_i) denote the i th vertex of the shapes $\bar{\mathbf{s}}$ and $\mathbf{s}$ respectively. The previous problem can be elegantly expressed in vector notation as:

$$\mathbf{T} = \boldsymbol{\Psi}(\mathbf{p})\mathbf{L} \tag{A.10}$$

and $\boldsymbol{\Psi}(\mathbf{p})$ obtained by:

$$\boldsymbol{\Psi}(\mathbf{p}) = \mathbf{T}\mathbf{L}^{-1} \tag{A.11}$$

A.2 Warp Jacobian

In both families of warps, PWA and TPS, the warp *Jacobian* can be mathematically expressed as:

$$\frac{\partial \mathcal{W}}{\partial \mathbf{p}} = \sum_{i=0}^v \frac{\partial \mathcal{W}}{\partial x_i} \frac{\partial x_i}{\partial \mathbf{p}} + \frac{\partial \mathcal{W}}{\partial y_i} \frac{\partial y_i}{\partial \mathbf{p}} \quad (\text{A.12})$$

where the x_i and y_i refer to the vertices of a particular shape $\mathbf{s} = (x_1, y_1, x_2, y_2, \dots, x_v, y_v)^T$.

The terms $\frac{\partial \mathcal{W}}{\partial x_i}$ and $\frac{\partial \mathcal{W}}{\partial y_i}$ describe the way in which the warp $\mathcal{W}(\mathbf{x}; \mathbf{p})$ changes with respect to the vertices x_i and y_i . These two terms completely depend on the family of warps being used and will have different analytical expressions for both PWA and TPS .

On the other hand, the terms $\frac{\partial x_i}{\partial \mathbf{p}}$ and $\frac{\partial y_i}{\partial \mathbf{p}}$ are independent of the family of warps being used and can be easily obtained by differentiating the expression in equation 3.4 with respect to the shape parameters $\mathbf{p}$:

$$\frac{\partial x_i}{\partial \mathbf{p}} = (s_1^{x_i}, s_2^{x_i}, \dots, s_m^{x_i}) \quad \frac{\partial y_i}{\partial \mathbf{p}} = (s_1^{y_i}, s_2^{y_i}, \dots, s_m^{y_i}) \quad (\text{A.13})$$

where the terms $s_j^{x_i}$ and $s_j^{y_i}$ denote the x and y components of the i th vertex of the j th shape eigenvector. That is, $\frac{\partial x_i}{\partial \mathbf{p}}$ and $\frac{\partial y_i}{\partial \mathbf{p}}$ are simply the shape eigenvectors $\mathbf{s}_i$ appropriately rearranged.

A.2.1 Piece-wise affine

In the case of PWA, the Jacobian of the warp with respect to the vertices of the shape $\mathbf{s}$ can be obtained by differentiating the expression in equation

A.4 with respect to x_i and y_i :

$$\frac{\partial \mathcal{W}}{\partial x_i} = (1 - \alpha - \beta, 0) \quad \frac{\partial \mathcal{W}}{\partial y_i} = (0, 1 - \alpha - \beta) \quad (\text{A.14})$$

and evaluating the previous expressions at every pixel position $\mathbf{x} \in \Omega$. The expressions in equation A.14 produces 4v images denoting the rate of change of the previous $\mathbf{x}$ with respect to the x and y components of every vertex in the $\bar{\mathbf{s}}$ under a PWA warp . Notice that, due to the locality of PWA these images are only non-zero for those triangles around the particular vertex i for which the Jacobian is being specifically derived for.

A.2.2 Thin plate splines

Taking the derivative of the expression in equation A.6 with respect to x_i leads to:

$$\frac{\partial \mathcal{W}}{\partial x_i} = \frac{\partial \Psi(\mathbf{p})}{\partial x_i} \mathbf{k}(\mathbf{x}) + \Psi(\mathbf{p}) \frac{\partial \mathbf{k}(\mathbf{x})}{\partial x_i} \quad (\text{A.15})$$

Notice that, using a compositional framework, the previous expression is only evaluated at $\mathbf{p} = 0$, i.e. the identity warp. Consequently, the source and target shapes of the TPS warp are the same, i.e. $\bar{\mathbf{s}}$, and the right-hand side term in equation A.15 banishes.

The derivative $\frac{\partial \Psi(\mathbf{p})}{\partial x_i}$ on the left-hand side term can be further expanded as:

$$\frac{\partial \Psi(\mathbf{p})}{\partial x_i} = \frac{\partial \mathbf{T}}{\partial x_i} \mathbf{L}^{-1} + \mathbf{T} \frac{\partial \mathbf{L}^{-1}}{\partial x_i} \quad (\text{A.16})$$

$\frac{\partial \mathbf{T}}{\partial x_i}$ is only related to the the target shape and does not depend on x_i . This causes the left-hand side term in A.16 to banish.

$\frac{\partial \mathbf{L}^{-1}}{\partial x_i}$ is the derivative of an inverse which writes:

$$\frac{\partial \mathbf{L}^{-1}}{\partial x_i} = -\mathbf{L}^{-1} \frac{\partial \mathbf{L}}{\partial x_i} \mathbf{L}^{-1} \quad (\text{A.17})$$

Expanding $\frac{\partial \mathbf{L}}{\partial x_i}$ one has:

$$\frac{\partial \mathbf{L}}{\partial x_i} = \begin{bmatrix} \frac{\partial \mathbf{K}}{\partial x_i} & \frac{\partial \mathbf{S}}{\partial x_i} \\ \frac{\partial \mathbf{S}^T}{\partial x_i} & \mathbf{O} \end{bmatrix} \quad (\text{A.18})$$

The term $\frac{\partial \mathbf{S}}{\partial x_i}$ is zero everywhere but at the particular position x_i where it is equal to 1. Similarly, the term $\frac{\partial \mathbf{K}}{\partial x_i}$ is zero everywhere but at the i th row and i th column where the derivatives of the individual $U(r_{i\cdot})$ and $U(r_{\cdot i})$ elements have to be computed. As an example, the derivative of $\mathbf{K}$ with respect to x_1 is provided:

$$\frac{\partial \mathbf{K}}{\partial x_1} = \begin{bmatrix} 0 & \frac{\partial U(r_{12})}{\partial x_1} & \dots & \frac{\partial U(r_{1v})}{\partial x_1} \\ \frac{\partial U(r_{21})}{\partial x_1} & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial U(r_{v1})}{\partial x_1} & 0 & \dots & 0 \end{bmatrix} \quad (\text{A.19})$$

where

$$\frac{\partial U(r_{ij})}{\partial x_i} = 2(1 + \ln r_{ij}^2)(x_i - \mathbf{x}_x) \quad (\text{A.20})$$

and

$$\frac{\partial U(r_{ji})}{\partial x_i} = -2(1 + \ln r_{ij}^2)(\mathbf{x}_x - x_i) \quad (\text{A.21})$$

Given the previous derivation it is straight forward to derive $\frac{\partial \mathcal{W}}{\partial y_i}$.

Notice that, evaluating the previous expression at every pixel position $\mathbf{x} \in \Omega$ produces, as before, $4v$ images explaining the rate of change of the previous $\mathbf{x}$ with respect to the x and y components of every vertex in the mean shape $\bar{\mathbf{s}}$ under a TPS warp.

A.3 Warp inversion

This section details the computation of the warp inverse $\mathcal{W}(\mathbf{x}; \Delta \mathbf{q})^{-1}$.

Notice that, $\mathcal{W}(\mathbf{x}; \Delta \mathbf{p})$ can be expressed as:

$$\mathcal{W}(\mathbf{x}; \Delta \mathbf{p}) = \frac{\partial \mathcal{W}}{\partial \mathbf{0}} + \frac{\partial \mathcal{W}}{\partial \mathbf{p}} = \mathbf{x} + \frac{\partial \mathcal{W}}{\partial \mathbf{p}} + O(\Delta \mathbf{p}^2) \quad (\text{A.22})$$

consequently:

$$\mathcal{W}(\mathbf{x}; \Delta \mathbf{p}) \circ \mathcal{W}(\mathbf{x}; -\Delta \mathbf{p}) = \mathbf{x} - \frac{\partial \mathcal{W}}{\partial \mathbf{p}} + \frac{\partial \mathcal{W}}{\partial \mathbf{p}} = \mathbf{x} + O(\Delta \mathbf{p}^2) \quad (\text{A.23})$$

and then, up to first order in $\Delta \mathbf{p}$, the following expression holds:

$$\mathcal{W}(\mathbf{x}; \Delta \mathbf{p})^{-1} = \mathcal{W}(\mathbf{x}; -\Delta \mathbf{p}) \quad (\text{A.24})$$

The previous derivation does not depend on the family of warps being used and, therefore, holds for both PWA and TP.

A.4 Warp composition

The approach used to approximate the composition operation $\mathcal{W}(\mathbf{x}; \Delta \mathbf{p}) \circ \mathcal{W}(\mathbf{x}; -\Delta \mathbf{p})$ is different PWA and TPS.

A.4.1 Piece-wise affine

In PWA the approach is algorithmic. Consider

$$\begin{aligned} \mathbf{s}^{-1} &= \bar{\mathbf{s}} - \sum_{i=1}^n \Delta p_i \mathbf{s}_i \\ &= \bar{\mathbf{s}} - \mathbf{S} \mathbf{p} \end{aligned} \tag{A.25}$$

as the shape obtained by warping the mean shape $\bar{\mathbf{s}}$ with the previous approximation of $\mathcal{W}(\mathbf{x}; \Delta \mathbf{p})^{-1}$. For every triangle containing a particular vertex (x_i^{-1}, y_i^{-1}) in the previous shape, there exist a different affine warp relating the triangle to its corresponding triangle in the mean shape $\bar{\mathbf{s}}$. One way to compute the destination of the vertex (x_i, y_i) of the current shape $\mathbf{s}$ under $\mathcal{W}(\mathbf{x}; \Delta \mathbf{p}) \circ \mathcal{W}(\mathbf{x}; -\Delta \mathbf{p})$ is to apply all the affine warps defined by the previous triangles to the vertex (x_i, y_i) and then average the result. Repeating this process for all vertices i in the shapes $\mathbf{s}^{-1}$ and $\mathbf{s}$ is one way to compute $\mathcal{W}(\mathbf{x}; \Delta \mathbf{p}) \circ \mathcal{W}(\mathbf{x}; -\Delta \mathbf{p})$. Denoting the shape obtained by the previous process as $\mathbf{s}_n$, the new parameters $\mathbf{p}$ of the warp can be obtained solving equation 3.4:

$$\mathbf{p} = \mathbf{S}^T (\mathbf{s}_n - \bar{\mathbf{s}}) \tag{A.26}$$

A.4.2 Thin plate splines

Using TPS, the strategy is the one proposed in [34] consisting of deriving the parameter Jacobian matrix $J_{\mathbf{p}}$ that converts the inverse compositional parameter update to its forward additive first order equivalent.

The starting point is:

$$\mathcal{W}(\mathbf{x}; \mathbf{p} + J_{\mathbf{p}} \Delta \mathbf{p}) \approx \mathcal{W}(\mathcal{W}(\mathbf{x}; -\Delta \mathbf{p}); \mathbf{p}) \quad (\text{A.27})$$

Taking the derivative of the previous expression with respect to $\Delta \mathbf{p}$ yields:

$$\underbrace{\left. \frac{\partial \mathcal{W}}{\partial \mathbf{p}} \right|_{(\mathbf{x}; \mathbf{p})}}_{2 \times (n)} \underbrace{J_{\mathbf{p}}}_{n \times n} \approx - \underbrace{\left. \frac{\partial \mathcal{W}}{\partial \mathbf{x}} \right|_{(\mathbf{x}; \mathbf{p})}}_{2 \times 2} \underbrace{\left. \frac{\partial \mathcal{W}}{\partial \mathbf{p}} \right|_{(\mathbf{x}; \mathbf{0})}}_{2 \times n} \quad (\text{A.28})$$

$$\underbrace{\begin{bmatrix} \left. \frac{\partial \mathcal{W}}{\partial \mathbf{p}} \right|_{(\mathbf{x}_1; \mathbf{p})} \\ \vdots \\ \left. \frac{\partial \mathcal{W}}{\partial \mathbf{p}} \right|_{(\mathbf{x}_v; \mathbf{p})} \end{bmatrix}}_{2v \times n} \underbrace{J_{\mathbf{p}}}_{n \times n} \approx - \underbrace{\begin{bmatrix} \left. \frac{\partial \mathcal{W}}{\partial \mathbf{x}} \right|_{(\mathbf{x}_1; \mathbf{p})} & \left. \frac{\partial \mathcal{W}}{\partial \mathbf{p}} \right|_{(\mathbf{x}_1; \mathbf{p})} \\ \vdots \\ \left. \frac{\partial \mathcal{W}}{\partial \mathbf{x}} \right|_{(\mathbf{x}_v; \mathbf{p})} & \left. \frac{\partial \mathcal{W}}{\partial \mathbf{p}} \right|_{(\mathbf{x}_v; \mathbf{p})} \end{bmatrix}}_{2v \times n} \quad (\text{A.29})$$

$$\underbrace{\left. \frac{\partial \mathcal{W}}{\partial \mathbf{p}} \right|_{(\mathbf{x}_{1:v}; \mathbf{p})}}_{2v \times (n)} \underbrace{J_{\mathbf{p}}}_{n \times n} \approx - \underbrace{\left. \frac{\partial \mathcal{W}}{\partial \mathbf{x}} \right|_{(\mathbf{x}_{1:v}; \mathbf{p})}}_{2v \times n} \odot \underbrace{\left. \frac{\partial \mathcal{W}}{\partial \mathbf{p}} \right|_{(\mathbf{x}_{1:v}; \mathbf{0})}}_{2v \times n} \quad (\text{A.30})$$

where $\odot$ denotes the appropriate stacked block-by-block matrix product.

Solving the previous expression for $J_{\mathbf{p}}$ yields:

$$J_{\mathbf{p}} \approx - \left(\left. \frac{\partial \mathcal{W}}{\partial \mathbf{p}} \right|_{(\mathbf{x}_{1:v}; \mathbf{p})}^T \left. \frac{\partial \mathcal{W}}{\partial \mathbf{p}} \right|_{(\mathbf{x}_{1:v}; \mathbf{p})} \right)^{-1} \left. \frac{\partial \mathcal{W}}{\partial \mathbf{p}} \right|_{(\mathbf{x}_{1:v}; \mathbf{p})}^T \left(\left. \frac{\partial \mathcal{W}}{\partial \mathbf{x}} \right|_{(\mathbf{x}_{1:v}; \mathbf{p})} \odot \left. \frac{\partial \mathcal{W}}{\partial \mathbf{p}} \right|_{(\mathbf{x}_{1:v}; \mathbf{0})} \right) \quad (\text{A.31})$$

Note that one is not interested in evaluating the previous expression at every pixel position $\mathbf{x} \in \Omega$ but only at the v landmarks points $\mathbf{x}_{1:v}$ in the mean shape $\bar{\mathbf{s}}$. Consequently, the terms $\left. \frac{\partial \mathcal{W}}{\partial \mathbf{p}} \right|_{(\mathbf{x}_{1:v}; \mathbf{p})}$ and $\left. \frac{\partial \mathcal{W}}{\partial \mathbf{p}} \right|_{(\mathbf{x}_{1:v}; \mathbf{0})}$ reduce both to

the matrix $\mathbf{S}$ in equation 3.4 whose columns are the shape eigenvectors $\mathbf{s}_i$.

The interesting term here is $\frac{\partial \mathcal{W}}{\partial \mathbf{x}} \Big|_{(\mathbf{x}_{1:v}; \mathbf{p})}$. Notice that this is different from the expression in equation A.15, which can be re-written as $\frac{\partial \mathcal{W}}{\partial \mathbf{x}_{1:v}} \Big|_{(\mathbf{x}_{1:v}; \mathbf{0})}$. In this case, the derivative is taken with respect to $\mathbf{x}$ instead of $\mathbf{x}_{1:v}$ and evaluated using the current warp parameters $\mathbf{p}$ instead of the identity warp. Therefore:

$$\frac{\partial \mathcal{W}}{\partial \mathbf{x}} \Big|_{(\mathbf{x}_{1:v}; \mathbf{p})} = \mathbf{\Psi}(\mathbf{p}) \frac{\partial \mathbf{k}(\mathbf{x})}{\partial \mathbf{x}} \Big|_{\mathbf{x}} \quad (\text{A.32})$$

And taking the derivative of the expression in equation A.7 with respect to $\mathbf{x}$ yields:

$$\frac{\partial \mathbf{k}(\mathbf{x})}{\partial \mathbf{x}} \Big|_{\mathbf{x}} = \begin{bmatrix} 2(1 + \ln r_{\mathbf{x},1}^2)(\mathbf{x} - \mathbf{x}_1)^T \\ \vdots \\ 2(1 + \ln r_{\mathbf{x},v}^2)(\mathbf{x} - \mathbf{x}_v)^T \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \quad (\text{A.33})$$

Appendix B

Efficient second-order minimization

This chapter shows that the asymmetric Gauss-Newton algorithms derived in section 5.4.1 are, in fact, true *second* order optimization algorithms with respect to the incremental warp $\Delta\mathbf{p}$.

The use of *asymmetric* composition together with the Gauss-Newton method has been proven to naturally lead to *efficient second order minimization* (ESM) algorithms in the related field of parametric image alignment [44, 101, 45, 47]. Following a similar line of reasoning, one can show that asymmetric Gauss-Newton algorithms for fitting AAM can also be interpreted as ESM algorithms.

The previous relationship will be shown by making use of the simplified data term¹ introduced by equation 5.18.

¹Notice that similar derivations can also be obtained using the SSD and PO data terms, but the simplified one is used here for clarity.

Using *forward* composition, the optimization problem defined by:

$$\Delta \mathbf{p}^* = \arg \min_{\Delta \mathbf{p}} \frac{1}{2} \mathbf{r}_f^T \bar{\mathbf{A}} \mathbf{r}_f \quad (\text{B.1})$$

where the forward residual $\mathbf{r}_f$ is defined as:

$$\mathbf{r}_f = \mathbf{i}[\mathbf{p} \circ \Delta \mathbf{p}] - \bar{\mathbf{a}} \quad (\text{B.2})$$

As seen section 5.4.1, Gauss-Newton solves the previous optimization problem by performing a *first* order Taylor expansion of the residual around $\Delta \mathbf{p}$:

$$\begin{aligned} \hat{\mathbf{r}}_f(\Delta \mathbf{p}) &= \mathbf{r}_f + \frac{\partial \mathbf{r}_f}{\partial \Delta \mathbf{p}} \Delta \mathbf{p} + \underbrace{O_{\mathbf{r}_f}(\Delta \mathbf{p}^2)}_{\text{remainder}} \\ &= \mathbf{i}[\mathbf{p}] - \mathbf{a} + \mathbf{J}_i \Delta \mathbf{p} + O_{\mathbf{r}_f}(\Delta \mathbf{p}^2) \end{aligned} \quad (\text{B.3})$$

and solving the following approximation of the original problem:

$$\Delta \mathbf{p}^* = \arg \min_{\Delta \mathbf{p}} \frac{1}{2} \hat{\mathbf{r}}_f^T \hat{\mathbf{r}}_f \quad (\text{B.4})$$

However, note that, instead of performing a first order Taylor expansion, one can also perform a *second* order Taylor expansion of the residual:

$$\begin{aligned} \check{\mathbf{r}}_f(\Delta \mathbf{p}) &= \mathbf{r}_f + \frac{\partial \mathbf{r}_f}{\partial \Delta \mathbf{p}} \Delta \mathbf{p} + \frac{1}{2} \Delta \mathbf{p}^T \frac{\partial^2 \mathbf{r}_f}{\partial^2 \Delta \mathbf{p}} \Delta \mathbf{p} + O_{\mathbf{r}_f}(\Delta \mathbf{p}^3) \\ &= \mathbf{i}[\mathbf{p}] - \mathbf{a} + \mathbf{J}_i \Delta \mathbf{p} + \frac{1}{2} \Delta \mathbf{p}^T \mathbf{H}_i \Delta \mathbf{p} + O_{\mathbf{r}_f}(\Delta \mathbf{p}^3) \end{aligned} \quad (\text{B.5})$$

Then, given the second main assumption behind AAM (equation 3.8) the

following approximation must hold:

$$\begin{aligned}\nabla \mathbf{i}[\mathbf{p}] \frac{\partial \mathcal{W}}{\partial \Delta \mathbf{p}} &\approx \nabla \mathbf{a} \frac{\partial \mathcal{W}}{\partial \Delta \mathbf{p}} \\ \mathbf{J}_{\mathbf{i}} &\approx \mathbf{J}_{\mathbf{a}}\end{aligned}\tag{B.6}$$

and, because the previous $\mathbf{J}_{\mathbf{i}}$ and $\mathbf{J}_{\mathbf{a}}$ are functions of $\Delta \mathbf{p}$, one can perform a *first* order Taylor expansion of $\mathbf{J}_{\mathbf{i}}$ to obtain:

$$\begin{aligned}\mathbf{J}_{\mathbf{i}}(\Delta \mathbf{p}) &\approx \mathbf{J}_{\mathbf{i}} + \Delta \mathbf{p}^T \frac{\partial \mathbf{J}_{\mathbf{i}}}{\partial \Delta \mathbf{p}} + \underbrace{O_{\mathbf{J}_{\mathbf{i}}}(\Delta \mathbf{p}^2)}_{\text{remainder}} \\ &\approx \mathbf{J}_{\mathbf{i}} + \Delta \mathbf{p}^T \mathbf{H}_{\mathbf{i}} + O_{\mathbf{J}_{\mathbf{i}}}(\Delta \mathbf{p}^2) \\ \mathbf{J}_{\mathbf{a}} &\approx \mathbf{J}_{\mathbf{i}} + \Delta \mathbf{p}^T \mathbf{H}_{\mathbf{i}} + O_{\mathbf{J}_{\mathbf{i}}}(\Delta \mathbf{p}^2) \\ \Delta \mathbf{p}^T \mathbf{H}_{\mathbf{i}} &\approx \mathbf{J}_{\mathbf{a}} - \mathbf{J}_{\mathbf{i}} - O_{\mathbf{J}_{\mathbf{i}}}(\Delta \mathbf{p}^2)\end{aligned}\tag{B.7}$$

Finally, substituting the previous approximation for $\Delta \mathbf{p}^T \mathbf{H}_{\mathbf{i}}$ into equation [B.5](#) one arrives at:

$$\begin{aligned}\tilde{\mathbf{r}}_f(\Delta \mathbf{p}) &= \mathbf{i}[\mathbf{p}] - \mathbf{a} + \mathbf{J}_{\mathbf{i}} \Delta \mathbf{p} + \frac{1}{2} \Delta \mathbf{p}^T \mathbf{H}_{\mathbf{i}} \Delta \mathbf{p} + O_{\mathbf{r}_f}(\Delta \mathbf{p}^3) \\ &= \mathbf{i}[\mathbf{p}] - \mathbf{a} + \mathbf{J}_{\mathbf{i}} \Delta \mathbf{p} + \frac{1}{2} (\mathbf{J}_{\mathbf{a}} - \mathbf{J}_{\mathbf{i}} - O_{\mathbf{J}_{\mathbf{i}}}(\Delta \mathbf{p}^2)) \Delta \mathbf{p} \\ &\quad + O_{\mathbf{r}_f}(\Delta \mathbf{p}^3) \\ &= \mathbf{i}[\mathbf{p}] - \mathbf{a} + \frac{1}{2} (\mathbf{J}_{\mathbf{i}} + \mathbf{J}_{\mathbf{a}}) \Delta \mathbf{p} + O_{\text{total}}(\Delta \mathbf{p}^3)\end{aligned}\tag{B.8}$$

where the total remainder is cubic with respect to $\Delta \mathbf{p}$:

$$O_{\text{total}}(\Delta \mathbf{p}^3) = O_{\mathbf{r}_f}(\Delta \mathbf{p}^3) - O_{\mathbf{J}_{\mathbf{i}}}(\Delta \mathbf{p}^2) \Delta \mathbf{p}\tag{B.9}$$

The expression in equation [B.8](#) constitutes a true *second* order approximation of the forward residual $\mathbf{r}_f$ where the term $\frac{1}{2} (\mathbf{J}_{\mathbf{i}} + \mathbf{J}_{\mathbf{a}})$ is equivalent to

the asymmetric Jacobian in equation 5.40 when $\alpha = \beta = 0.5$:

$$\begin{aligned}
 \frac{1}{2}(\mathbf{J}_i + \mathbf{J}_a) &= \left(\frac{1}{2}\mathbf{J}_i + \frac{1}{2}\mathbf{J}_a \right) \\
 &= \left(\frac{1}{2}\nabla \mathbf{i}[\mathbf{p}] \frac{\partial \mathcal{W}}{\partial \Delta \mathbf{p}} + \frac{1}{2}\nabla \mathbf{a} \frac{\partial \mathcal{W}}{\partial \Delta \mathbf{p}} \right) \\
 &= \left(\frac{1}{2}\nabla \mathbf{i}[\mathbf{p}] + \frac{1}{2}\nabla \mathbf{a} \right) \frac{\partial \mathcal{W}}{\partial \Delta \mathbf{p}} \\
 &= (\nabla \mathbf{t}) \frac{\partial \mathcal{W}}{\partial \Delta \mathbf{p}} \\
 &= \mathbf{J}_t
 \end{aligned} \tag{B.10}$$

and, consequently, asymmetric Gauss-Newton algorithms for fitting AAM are ESM algorithms that only require *first* order partial derivatives of the residual and that have the same computational complexity as *first* order algorithms.

Appendix C

Newton Hessian terms

This chapter defines the individual terms of the *Hessian* matrices used by the SSD asymmetric and bidirectional Newton optimization algorithms derived in section 5.4.2.

C.1 Asymmetric

The individual terms forming the Hessian matrix of the SSD asymmetric Newton algorithm defined by equation 5.76 are defined as follows:

$$\begin{aligned}\frac{\partial^2 \mathcal{D}_a}{\partial \Delta \mathbf{c}^2} &= \frac{\partial -\mathbf{A}^T \mathbf{r}_a}{\partial \Delta \mathbf{c}} \\ &= -\mathbf{A}^T \frac{\partial \mathbf{r}_a}{\partial \Delta \mathbf{c}} \\ &= \underbrace{\mathbf{A}^T \mathbf{A}}_{\mathbf{I}}\end{aligned}\tag{C.1}$$

$$\begin{aligned}
\frac{\partial^2 \mathcal{D}_a}{\partial \Delta \mathbf{c} \partial \Delta \mathbf{p}} &= \frac{\partial - \mathbf{A}^T \mathbf{r}_a}{\partial \Delta \mathbf{p}} \\
&= \frac{\partial - \mathbf{A}^T}{\partial \Delta \mathbf{p}} \mathbf{r}_a - \mathbf{A}^T \frac{\partial \mathbf{r}_a}{\partial \Delta \mathbf{p}} \\
&= -\beta \mathbf{J}_{\mathbf{A}}^T \mathbf{r}_a - \mathbf{A}^T \mathbf{J}_{\mathbf{t}}
\end{aligned} \tag{C.2}$$

where $\mathbf{J}_{\mathbf{A}} = [\nabla \mathbf{a}_1, \dots, \nabla \mathbf{a}_m]^T \frac{\partial \mathcal{W}}{\partial \Delta \mathbf{p}}$.

$$\begin{aligned}
\frac{\partial^2 \mathcal{D}_a}{\partial \Delta \mathbf{p}^2} &= \frac{\partial \mathbf{J}_{\mathbf{t}}^T \mathbf{r}_a}{\partial \Delta \mathbf{p}} \\
&= \frac{\partial \mathbf{J}_{\mathbf{t}}^T}{\partial \Delta \mathbf{p}} \mathbf{r}_a + \mathbf{J}_{\mathbf{t}}^T \frac{\partial \mathbf{r}_a}{\partial \Delta \mathbf{p}} \\
&= \left(\frac{\partial \mathcal{W}}{\partial \Delta \mathbf{p}}^T \nabla^2 \mathbf{t} \frac{\partial \mathcal{W}}{\partial \Delta \mathbf{p}} + \underbrace{\nabla \mathbf{t} \frac{\partial^2 \mathcal{W}}{\partial^2 \mathbf{p}}}_{\mathbf{0}} \right) \mathbf{r}_a + \mathbf{J}_{\mathbf{t}}^T \mathbf{J}_{\mathbf{t}} \\
&= \left(\frac{\partial \mathcal{W}}{\partial \Delta \mathbf{p}}^T \nabla^2 \mathbf{t} \frac{\partial \mathcal{W}}{\partial \Delta \mathbf{p}} \right) \mathbf{r}_a + \mathbf{J}_{\mathbf{t}}^T \mathbf{J}_{\mathbf{t}}
\end{aligned} \tag{C.3}$$

C.2 Bidirectional

The individual terms forming the Hessian matrix of the SSD bidirectional Newton algorithm defined by Equation 5.79 are defined as follows:

$$\begin{aligned}
\frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{c}^2} &= \frac{\partial - \mathbf{A}^T \mathbf{r}_b}{\partial \Delta \mathbf{c}} \\
&= -\mathbf{A}^T \frac{\partial \mathbf{r}_b}{\partial \Delta \mathbf{c}} \\
&= \underbrace{\mathbf{A}^T \mathbf{A}}_{\mathbf{I}}
\end{aligned} \tag{C.4}$$

$$\begin{aligned}
\frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{c} \partial \Delta \mathbf{p}} &= \frac{\partial - \mathbf{A}^T \mathbf{r}_b}{\partial \Delta \mathbf{p}} \\
&= -\mathbf{A}^T \frac{\partial \mathbf{r}_b}{\partial \Delta \mathbf{p}} \\
&= -\mathbf{A}^T \mathbf{J}_{\mathbf{i}}
\end{aligned} \tag{C.5}$$

$$\begin{aligned}
\frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{c} \partial \Delta \mathbf{q}} &= \frac{\partial - \mathbf{A}^T \mathbf{r}_b}{\partial \Delta \mathbf{q}} \\
&= \frac{\partial - \mathbf{A}^T}{\partial \Delta \mathbf{q}} \mathbf{r}_b - \mathbf{A}^T \frac{\partial \mathbf{r}_b}{\partial \Delta \mathbf{q}} \\
&= -\mathbf{J}_{\mathbf{A}}^T \mathbf{r}_b + \mathbf{A}^T \mathbf{J}_{\mathbf{a}}
\end{aligned} \tag{C.6}$$

$$\begin{aligned}
\frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{p}^2} &= \frac{\partial \mathbf{J}_{\mathbf{i}}^T \mathbf{r}_b}{\partial \Delta \mathbf{p}} \\
&= \frac{\partial \mathbf{J}_{\mathbf{i}}^T}{\partial \Delta \mathbf{p}} \mathbf{r}_b + \mathbf{J}_{\mathbf{i}}^T \frac{\partial \mathbf{r}_b}{\partial \Delta \mathbf{p}} \\
&= \left(\frac{\partial \mathcal{W}}{\partial \Delta \mathbf{p}}^T \nabla^2 \mathbf{i}[\mathbf{p}] \frac{\partial \mathcal{W}}{\partial \Delta \mathbf{p}} \right) \mathbf{r}_b + \mathbf{J}_{\mathbf{i}}^T \mathbf{J}_{\mathbf{i}}
\end{aligned} \tag{C.7}$$

$$\begin{aligned}
\frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{p} \partial \Delta \mathbf{q}} &= \frac{\partial \mathbf{J}_{\mathbf{i}}^T \mathbf{r}_b}{\partial \Delta \mathbf{q}} \\
&= -\mathbf{J}_{\mathbf{i}}^T \mathbf{J}_{\mathbf{a}}
\end{aligned} \tag{C.8}$$

$$\begin{aligned}
\frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{q}^2} &= \frac{\partial - \mathbf{J}_{\mathbf{a}}^T \mathbf{r}_b}{\partial \Delta \mathbf{q}} \\
&= \frac{\partial - \mathbf{J}_{\mathbf{a}}^T}{\partial \Delta \mathbf{q}} \mathbf{r}_b - \mathbf{J}_{\mathbf{a}}^T \frac{\partial \mathbf{r}_b}{\partial \Delta \mathbf{q}} \\
&= - \left(\frac{\partial \mathcal{W}}{\partial \Delta \mathbf{q}}^T \nabla^2 (\mathbf{a} + \mathbf{A} \mathbf{c}) \frac{\partial \mathcal{W}}{\partial \Delta \mathbf{q}} \right) \mathbf{r}_b + \mathbf{J}_{\mathbf{a}}^T \mathbf{J}_{\mathbf{a}}
\end{aligned} \tag{C.9}$$

Appendix D

Iterative solutions

This chapter states the iterative solutions for all CGD algorithms studied in chapter 5. Algorithms are grouped by cost function; iterative solutions for SSD and PO algorithms are given in tables D.1 and D.2 respectively.

SSD algorithms	$\Delta \mathbf{p}$	Iterative solutions $\Delta \mathbf{q}$	$\Delta \mathbf{c}$
SSD_For_GN_Sch	$\Delta \mathbf{p} = -\hat{\mathbf{H}}_i^{-1} \mathbf{J}_i^T \bar{\mathbf{A}} \mathbf{r}$ $\hat{\mathbf{H}}_i = \mathbf{J}_i^T \bar{\mathbf{A}} \mathbf{J}_i$		$\Delta \mathbf{c} = \mathbf{A} (\mathbf{r} + \mathbf{J}_i \Delta \mathbf{p})$
SSD_For_GN_Alt	$\Delta \mathbf{p} = -\mathbf{H}_i^{-1} \mathbf{J}_i^T (\mathbf{r} - \mathbf{A} \Delta \mathbf{c})$ $\mathbf{H}_i = \mathbf{J}_i^T \mathbf{J}_i$		$\Delta \mathbf{c} = \mathbf{A} (\mathbf{r} + \mathbf{J}_i \Delta \mathbf{p})$
SSD_For_N_Sch	$\Delta \mathbf{p} = -\left(\hat{\mathbf{H}}_i^N\right)^{-1} \mathbf{J}_i^T \bar{\mathbf{A}} \mathbf{r}$ $\hat{\mathbf{H}}_i^N = \frac{\partial \mathcal{W}}{\Delta \mathbf{p}}^T \nabla^2 \mathbf{i} \frac{\partial \mathcal{W}}{\Delta \mathbf{p}} \mathbf{r} + \hat{\mathbf{H}}_i$		$\Delta \mathbf{c} = \mathbf{A} (\mathbf{r} + \mathbf{J}_i \Delta \mathbf{p})$
SSD_For_N_Alt	$\Delta \mathbf{p} = -\left(\hat{\mathbf{H}}_i^N\right)^{-1} \mathbf{J}_i^T \bar{\mathbf{A}} (\mathbf{r} - \mathbf{A} \Delta \mathbf{c})$ $\hat{\mathbf{H}}_i^N = \frac{\partial \mathcal{W}}{\Delta \mathbf{p}}^T \nabla^2 \mathbf{i} \frac{\partial \mathcal{W}}{\Delta \mathbf{p}} \mathbf{r} + \hat{\mathbf{H}}_i$		$\Delta \mathbf{c} = \mathbf{A} (\mathbf{r} + \mathbf{J}_i \Delta \mathbf{p})$
SSD_For_W	$\Delta \mathbf{p} = -\hat{\mathbf{H}}_i^{-1} \mathbf{J}_i^T \bar{\mathbf{A}} \mathbf{r}$		$\Delta \mathbf{c} = \mathbf{A} \mathbf{r}$
SSD_Inv_GN_Sch	$\Delta \mathbf{p} = \hat{\mathbf{H}}_a^{-1} \mathbf{J}_a^T \bar{\mathbf{A}} \mathbf{r}$ $\hat{\mathbf{H}}_a = \mathbf{J}_a^T \bar{\mathbf{A}} \mathbf{J}_a$		$\Delta \mathbf{c} = \mathbf{A} (\mathbf{r} - \mathbf{J}_a \Delta \mathbf{p})$
SSD_Inv_GN_Alt	$\Delta \mathbf{p} = \mathbf{H}_a^{-1} \mathbf{J}_a^T (\mathbf{r} - \mathbf{A} \Delta \mathbf{c})$ $\mathbf{H}_a = \mathbf{J}_a^T \mathbf{J}_a$		$\Delta \mathbf{c} = \mathbf{A} (\mathbf{r} - \mathbf{J}_a \Delta \mathbf{p})$
SSD_Inv_N_Sch	$\Delta \mathbf{p} = \left(\hat{\mathbf{H}}_a^N\right)^{-1} \mathbf{J}_a^T \bar{\mathbf{A}} \mathbf{r}$ $\hat{\mathbf{H}}_a^N = \frac{\partial \mathcal{W}}{\Delta \mathbf{p}}^T \nabla^2 \mathbf{a} \frac{\partial \mathcal{W}}{\Delta \mathbf{p}} \mathbf{r} + \hat{\mathbf{H}}_a$		$\Delta \mathbf{c} = \mathbf{A} (\mathbf{r} - \mathbf{J}_a \Delta \mathbf{p})$
SSD_Inv_N_Alt	$\Delta \mathbf{p} = \left(\hat{\mathbf{H}}_a^N\right)^{-1} \mathbf{J}_a^T \bar{\mathbf{A}} (\mathbf{r} - \mathbf{A} \Delta \mathbf{c})$ $\hat{\mathbf{H}}_a^N = \frac{\partial \mathcal{W}}{\Delta \mathbf{p}}^T \nabla^2 \mathbf{a} \frac{\partial \mathcal{W}}{\Delta \mathbf{p}} \mathbf{r} + \hat{\mathbf{H}}_a$		$\Delta \mathbf{c} = \mathbf{A} (\mathbf{r} - \mathbf{J}_a \Delta \mathbf{p})$
SSD_Inv_W	$\Delta \mathbf{p} = \hat{\mathbf{H}}_a^{-1} \mathbf{J}_a^T \bar{\mathbf{A}} \mathbf{r}$		$\Delta \mathbf{c} = \mathbf{A} \mathbf{r}$
SSD_Asy_GN_Sch	$\Delta \mathbf{p} = -\hat{\mathbf{H}}_t^{-1} \mathbf{J}_t^T \bar{\mathbf{A}} \mathbf{r}$ $\hat{\mathbf{H}}_t = \mathbf{J}_t^T \bar{\mathbf{A}} \mathbf{J}_t$		$\Delta \mathbf{c} = \mathbf{A} (\mathbf{r} + \mathbf{J}_t \Delta \mathbf{p})$
SSD_Asy_GN_Alt	$\Delta \mathbf{p} = -\mathbf{H}_t^{-1} \mathbf{J}_t^T (\mathbf{r} - \mathbf{A} \Delta \mathbf{c})$ $\mathbf{H}_t = \mathbf{J}_t^T \mathbf{J}_t$		$\Delta \mathbf{c} = \mathbf{A} (\mathbf{r} + \mathbf{J}_t \Delta \mathbf{p})$
SSD_Asy_N_Sch	$\Delta \mathbf{p} = -\left(\hat{\mathbf{H}}_t^N\right)^{-1} \mathbf{J}_t^T \bar{\mathbf{A}} \mathbf{r}$ $\hat{\mathbf{H}}_t^N = \frac{\partial \mathcal{W}}{\Delta \mathbf{p}}^T \nabla^2 \mathbf{t} \frac{\partial \mathcal{W}}{\Delta \mathbf{p}} \mathbf{r} + \hat{\mathbf{H}}_t$		$\Delta \mathbf{c} = \mathbf{A} (\mathbf{r} + \mathbf{J}_t \Delta \mathbf{p})$
SSD_Asy_N_Alt	$\Delta \mathbf{p} = -\left(\hat{\mathbf{H}}_t^N\right)^{-1} \mathbf{J}_t^T \bar{\mathbf{A}} (\mathbf{r} - \mathbf{A} \Delta \mathbf{c})$ $\hat{\mathbf{H}}_t^N = \frac{\partial \mathcal{W}}{\Delta \mathbf{p}}^T \nabla^2 \mathbf{t} \frac{\partial \mathcal{W}}{\Delta \mathbf{p}} \mathbf{r} + \hat{\mathbf{H}}_t$		$\Delta \mathbf{c} = \mathbf{A} (\mathbf{r} + \mathbf{J}_t \Delta \mathbf{p})$
SSD_Asy_W	$\Delta \mathbf{p} = -\hat{\mathbf{H}}_t^{-1} \mathbf{J}_t^T \bar{\mathbf{A}} \mathbf{r}$		$\Delta \mathbf{c} = \mathbf{A} \mathbf{r}$
SSD_Bid_GN_Sch	$\Delta \mathbf{p} = -\hat{\mathbf{H}}_i^{-1} \mathbf{J}_i^T \bar{\mathbf{A}} \mathbf{r}_1$ $\mathbf{r}_1 = (\mathbf{r} - \mathbf{J}_a \Delta \mathbf{q})$	$\Delta \mathbf{q} = \hat{\mathbf{H}}_a^{-1} \mathbf{J}_a^T \mathbf{P} \mathbf{r}$ $\hat{\mathbf{H}}_a = \mathbf{J}_a^T \mathbf{P} \mathbf{J}_a$ $\mathbf{P} = \bar{\mathbf{A}} - \bar{\mathbf{A}} \mathbf{J}_i \hat{\mathbf{H}}_i^{-1} \mathbf{J}_i^T \bar{\mathbf{A}}$	$\Delta \mathbf{c} = \mathbf{A} \mathbf{r}_2$ $\mathbf{r}_2 = (\mathbf{r} + \mathbf{J}_i \Delta \mathbf{p} - \mathbf{J}_a \Delta \mathbf{q})$
SSD_Bid_GN_Alt	$\Delta \mathbf{p} = -\mathbf{H}_i^{-1} \mathbf{J}_i^T \mathbf{r}_3$ $\mathbf{r}_3 = (\mathbf{r} - \mathbf{A} \Delta \mathbf{c} - \mathbf{J}_a \Delta \mathbf{q})$	$\Delta \mathbf{q} = \mathbf{H}_a^{-1} \mathbf{J}_a^T \mathbf{r}_4$ $\mathbf{r}_4 = (\mathbf{r} - \mathbf{A} \Delta \mathbf{c} + \mathbf{J}_i \Delta \mathbf{p})$	$\Delta \mathbf{c} = \mathbf{A} \mathbf{r}_2$
SSD_Bid_N_Sch	$\Delta \mathbf{p} = -\left(\hat{\mathbf{H}}_i^N\right)^{-1} \mathbf{J}_i^T \bar{\mathbf{A}} \mathbf{r}_1$	$\Delta \mathbf{q} = \left(\hat{\mathbf{H}}_a^N\right)^{-1} \mathbf{J}_a^T \mathbf{P}^N \mathbf{r}$ $\hat{\mathbf{H}}_a^N = \frac{\partial \mathcal{W}}{\Delta \mathbf{p}}^T \nabla^2 \mathbf{t} \frac{\partial \mathcal{W}}{\Delta \mathbf{p}} \mathbf{r} + \hat{\mathbf{H}}_a$ $\mathbf{P}^N = \bar{\mathbf{A}} - \bar{\mathbf{A}} \mathbf{J}_i \left(\hat{\mathbf{H}}_i^N\right)^{-1} \mathbf{J}_i^T \bar{\mathbf{A}}$	$\Delta \mathbf{c} = \mathbf{A} \mathbf{r}_2$
SSD_Bid_N_Alt	$\Delta \mathbf{p} = -\left(\hat{\mathbf{H}}_i^N\right)^{-1} \mathbf{J}_i^T \mathbf{r}_3$	$\Delta \mathbf{q} = \left(\hat{\mathbf{H}}_a^N\right)^{-1} \mathbf{J}_a^T \mathbf{r}_4$	$\Delta \mathbf{c} = \mathbf{A} \mathbf{r}_2$
SSD_Bid_W	$\Delta \mathbf{p} = -\hat{\mathbf{H}}_i^{-1} \mathbf{J}_i^T \bar{\mathbf{A}} \mathbf{r}$	$\Delta \mathbf{q} = \hat{\mathbf{H}}_a^{-1} \mathbf{J}_a^T \mathbf{P} \mathbf{r}$	$\Delta \mathbf{c} = \mathbf{A} \mathbf{r}$

Table D.1: Iterative solutions for all SSD algorithms studied in chapter 5.

Project-Out algorithms	Iterative solutions	
	$\Delta \mathbf{p}$	$\Delta \mathbf{q}$
PO_For_GN [37, 40]	$\Delta \mathbf{p} = -\hat{\mathbf{H}}_i^{-1} \mathbf{J}_i^T \bar{\mathbf{A}} \mathbf{r}$ $\hat{\mathbf{H}}_i = \mathbf{J}_i^T \bar{\mathbf{A}} \mathbf{J}_i$	
PO_For_N	$\Delta \mathbf{p} = -\left(\hat{\mathbf{H}}_i^N\right)^{-1} \mathbf{J}_i^T \bar{\mathbf{A}} \mathbf{r}$ $\hat{\mathbf{H}}_i^N = \frac{\partial \mathcal{W}}{\partial \Delta \mathbf{p}}^T \nabla^2 \mathbf{i} \frac{\partial \mathcal{W}}{\partial \Delta \mathbf{p}} \bar{\mathbf{A}} \mathbf{r} + \hat{\mathbf{H}}_i$	
PO_Inv_GN [10]	$\Delta \mathbf{p} = \hat{\mathbf{H}}_a^{-1} \mathbf{J}_a^T \bar{\mathbf{A}} \mathbf{r}$ $\hat{\mathbf{H}}_a = \mathbf{J}_a^T \bar{\mathbf{A}} \mathbf{J}_a$	
PO_Inv_N	$\Delta \mathbf{p} = \left(\hat{\mathbf{H}}_a^N\right)^{-1} \mathbf{J}_a^T \bar{\mathbf{A}} \mathbf{r}$ $\hat{\mathbf{H}}_a^N = \frac{\partial \mathcal{W}}{\partial \Delta \mathbf{p}}^T \nabla^2 \bar{\mathbf{a}} \frac{\partial \mathcal{W}}{\partial \Delta \mathbf{p}} \bar{\mathbf{A}} \mathbf{r} + \hat{\mathbf{H}}_a$	
PO_Asy_GN	$\Delta \mathbf{p} = -\hat{\mathbf{H}}_t^{-1} \mathbf{J}_t^T \bar{\mathbf{A}} \mathbf{r}$ $\hat{\mathbf{H}}_t = \mathbf{J}_t^T \bar{\mathbf{A}} \mathbf{J}_t$	
PO_Asy_N	$\Delta \mathbf{p} = -\left(\hat{\mathbf{H}}_t^N\right)^{-1} \mathbf{J}_t^T \bar{\mathbf{A}} \mathbf{r}$ $\hat{\mathbf{H}}_t^N = \frac{\partial \mathcal{W}}{\partial \Delta \mathbf{p}}^T \nabla^2 \mathbf{t} \frac{\partial \mathcal{W}}{\partial \Delta \mathbf{p}} \bar{\mathbf{A}} \mathbf{r} + \hat{\mathbf{H}}_t$	
PO_Bid_GN_Sch	$\Delta \mathbf{p} = -\hat{\mathbf{H}}_i^{-1} \mathbf{J}_i^T \bar{\mathbf{A}} (\mathbf{r} - \mathbf{J}_a \Delta \mathbf{q})$	$\Delta \mathbf{q} = \hat{\mathbf{H}}_a^{-1} \mathbf{J}_a^T \mathbf{P} \mathbf{r}$ $\hat{\mathbf{H}}_a = \mathbf{J}_a^T \mathbf{P} \mathbf{J}_a$ $\mathbf{P} = \bar{\mathbf{A}} - \bar{\mathbf{A}} \mathbf{J}_i \hat{\mathbf{H}}_i^{-1} \mathbf{J}_i^T \bar{\mathbf{A}}$
PO_Bid_GN_Alt	$\Delta \mathbf{p} = -\hat{\mathbf{H}}_i^{-1} \mathbf{J}_i^T \bar{\mathbf{A}} (\mathbf{r} - \mathbf{J}_a \Delta \mathbf{q})$	$\Delta \mathbf{q} = \hat{\mathbf{H}}_a^{-1} \mathbf{J}_a^T \bar{\mathbf{A}} (\mathbf{r} + \mathbf{J}_i \Delta \mathbf{p})$
PO_Bid_N_Sch	$\Delta \mathbf{p} = -\left(\hat{\mathbf{H}}_i^N\right)^{-1} \mathbf{J}_i^T \bar{\mathbf{A}} (\mathbf{r} - \mathbf{J}_a \Delta \mathbf{q})$	$\Delta \mathbf{q} = \left(\hat{\mathbf{H}}_a^N\right)^{-1} \mathbf{J}_a^T \mathbf{P}^N \mathbf{r}$ $\hat{\mathbf{H}}_a^N = \frac{\partial \mathcal{W}}{\partial \Delta \mathbf{p}}^T \nabla^2 \bar{\mathbf{a}} \frac{\partial \mathcal{W}}{\partial \Delta \mathbf{p}} \bar{\mathbf{A}} \mathbf{r} + \hat{\mathbf{H}}_a$ $\mathbf{P}^N = \bar{\mathbf{A}} - \bar{\mathbf{A}} \mathbf{J}_i \left(\hat{\mathbf{H}}_i^N\right)^{-1} \mathbf{J}_i^T \bar{\mathbf{A}}$
PO_Bid_N_Alt	$\Delta \mathbf{p} = -\left(\hat{\mathbf{H}}_i^N\right)^{-1} \mathbf{J}_i^T \bar{\mathbf{A}} (\mathbf{r} - \mathbf{J}_a \Delta \mathbf{q})$	$\Delta \mathbf{q} = \left(\hat{\mathbf{H}}_a^N\right)^{-1} \mathbf{J}_a^T \bar{\mathbf{A}} (\mathbf{r} + \mathbf{J}_i \Delta \mathbf{p})$
PO_Bid_W	$\Delta \mathbf{p} = -\hat{\mathbf{H}}_i^{-1} \mathbf{J}_i^T \bar{\mathbf{A}} \mathbf{r}$	$\Delta \mathbf{q} = \hat{\mathbf{H}}_a^{-1} \mathbf{J}_a^T \mathbf{P} \mathbf{r}$

Table D.2: Iterative solutions of all PO algorithms studied in chapter 5.