Int J Comput Vis
DOI 10.1007/s11263-016-0916-3

A Unified Framework for Compositional Fitting of Active Appearance Models

Joan Alabort-i-Medina[1] · **Stefanos Zafeiriou**[1]

Received: 5 October 2015 / Accepted: 18 May 2016

Abstract Active appearance models (AAMs) are one of the most popular and well-established techniques for modeling deformable objects in computer vision. In this paper, we study the problem of fitting AAMs using compositional gradient descent (CGD) algorithms. We present a unified and complete view of these algorithms and classify them with respect to three main characteristics: (i) *cost function*; (ii) type of *composition*; and (iii) *optimization method*. Furthermore, we extend the previous view by: (a) *proposing a novel Bayesian cost function* that can be interpreted as a general probabilistic formulation of the well-known project-out loss; (b) introducing two new types of composition, *asymmetric* and *bidirectional*, that combine the gradients of both image and appearance model to derive better convergent and more robust CGD algorithms; and (c) providing new valuable insights into existent CGD algorithms by reinterpreting them as direct applications of the *Schur complement* and the *Wiberg method*. Finally, in order to encourage open research and facilitate future comparisons with our work, we make the implementation of the algorithms studied in this paper publicly available as part of the Menpo Project (http://www.menpo.org).

Keywords Active appearance models · Non-linear optimization · Compositional gradient descent · Bayesian inference · Asymmetric and bidirectional composition · Schur complement · Wiberg algorithm

1 Introduction

Active appearance models (AAMs) (Cootes et al. 2001; Matthews and Baker 2004) are one of the most popular and well-established techniques for modeling and segmenting deformable objects in computer vision. AAMs are generative parametric models of shape and appearance that can be *fitted* to images to recover the set of model parameters that best describe a particular instance of the object being modeled.

Fitting AAMs is a non-linear optimization problem that requires the minimization (maximization) of a global error (similarity) measure between the input image and the appearance model. Several approaches (Cootes et al. 2001; Hou et al. 2001; Matthews and Baker 2004; Batur and Hayes 2005; Gross et al. 2005; Donner et al. 2006; Papandreou and Maragos 2008; Liu 2009; Saragih and Göcke 2009; Amberg et al. 2009; Tresadern et al. 2010; Martins et al. 2010; Sauer et al. 2011; Tzimiropoulos and Pantic 2013; Kossaifi et al. 2014; Antonakos et al. 2014) have been proposed to define and solve the previous optimization problem. Broadly speaking, they can be divided into two different groups:

- *Regression* based (Cootes et al. 2001; Hou et al. 2001; Batur and Hayes 2005; Donner et al. 2006; Saragih and Göcke 2009; Tresadern et al. 2010; Sauer et al. 2011)
- *Optimization* based (Matthews and Baker 2004; Gross et al. 2005; Papandreou and Maragos 2008; Amberg et al. 2009; Martins et al. 2010; Tzimiropoulos and Pantic 2013; Kossaifi et al. 2014)

Communicated by Sven J. Dickinson.

✉ Joan Alabort-i-Medina
ja310@imperial.ac.uk

Stefanos Zafeiriou
s.zafeiriou@imperial.ac.uk

[1] Department of Computing, Imperial College London, 180 Queen's Gate, London SW7 2AZ, UK

Published online: 09 June 2016

Regression based techniques attempt to solve the problem by learning a direct function mapping between the error measure and the optimal values of the parameters. Most notable approaches include variations on the original (Cootes et al. 2001) fixed linear regression approach of Hou et al. (2001), Donner et al. (2006), the adaptive linear regression approach of Batur and Hayes (2005), and the works of Saragih and Göcke (2009) and Tresadern et al. (2010) which considerably improved upon previous techniques by using boosted regression. Also, Cootes and Taylor (2001) and Tresadern et al. (2010) showed that the use of non-linear gradient-based and Haar-like appearance representations, respectively, lead to better fitting accuracy in regression based AAMs.

Optimization based methods for fitting AAMs were proposed by Matthews and Baker in Matthews and Baker (2004). 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 (Matthews and Baker 2004), the accurate but costly Simultaneous Inverse Compositional (SIC) algorithm (Gross et al. 2005), and the more efficient versions of SIC presented in Papandreou and Maragos (2008) and Tzimiropoulos and Pantic (2013). Lucey et al. (2013) extended these algorithms to the Fourier domain to efficiently enable convolution with Gabor filters, increasing their robustness; and the authors of Antonakos et al. (2014) showed that optimization based AAMs using non-linear feature based (e.g. SIFT Lowe 1999 and HOG Dalal and Triggs 2005) appearance models were competitive with modern state-of-the-art techniques in non-rigid face alignment (Xiong and De la Torre 2013; Asthana et al. 2013) in terms of fitting accuracy.

AAMs have often been criticized for several reasons: (i) the limited representational power of their linear appearance model; (ii) the difficulty of optimizing shape and appearance parameters simultaneously; and (iii) the complexity involved in handling occlusions. However, recent works in this area (Papandreou and Maragos 2008; Saragih and Göcke 2009; Tresadern et al. 2010; Lucey et al. 2013; Tzimiropoulos and Pantic 2013; Antonakos et al. 2014) suggest that these limitations might have been over-stressed in the literature and that AAMs can produce highly accurate results if appropriate training data (Tzimiropoulos and Pantic 2013), appearance representations (Tresadern et al. 2010; Lucey et al. 2013; Antonakos et al. 2014) and fitting strategies (Papandreou and Maragos 2008; Saragih and Göcke 2009; Tresadern et al. 2010; Tzimiropoulos and Pantic 2013) are employed.

In this paper, we study the problem of fitting AAMs using CGD algorithms thoroughly. Summarizing, our main contributions are:

- To present a unified and complete overview of the most relevant and recently published CGD algorithms for fitting AAMs (Matthews and Baker 2004; Gross et al. 2005; Papandreou and Maragos 2008; Amberg et al. 2009; Martins et al. 2010; Tzimiropoulos et al. 2012; Tzimiropoulos and Pantic 2013; Kossaifi et al. 2014). To this end, we classify CGD algorithms with respect to three main characteristics: (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.
- To review the probabilistic interpretation of AAMs and propose a novel *Bayesian formulation*[1] of the fitting problem. We assume a probabilistic model for appearance generation with both Gaussian noise and a Gaussian prior over a latent appearance space. Marginalizing out the latent appearance space, we derive a novel cost function that only depends on shape parameters and that can be interpreted as a valid and more general probabilistic formulation of the well-known project-out cost function (Matthews and Baker 2004). Our Bayesian formulation is motivated by seminal works on probabilistic component analysis and object tracking (Moghaddam and Pentland 1997; Roweis 1998; Tipping and Bishop 1999).
- To propose the use of two novel types of composition for AAMs: (i) *asymmetric*; and (ii) *bidirectional*. These types of composition have been widely used in the related field of parametric image alignment (Malis 2004; Mégret et al. 2008; Autheserre et al. 2009; Mégret et al. 2010) and use the gradients of both image and appearance model to derive better convergent and more robust CGD algorithms.
- To provide valuable insights into existent strategies used to derive fast and exact simultaneous algorithms for fitting AAMs by reinterpreting them as direct applications of the *Schur complement* (Boyd and Vandenberghe 2004) and the *Wiberg method* (Okatani and Deguchi 2006; Strelow 2012).

The remainder of the paper is structured as follows. Section 2 introduces AAMs and reviews their probabilistic interpretation. Section 3 constitutes the main section of the paper and contains the discussion and derivations related to the cost functions Sect. 3.1; composition types Sect. 3.2; and optimization methods Sect. 3.3. Implementation details and experimental results are reported in Sect. 5. Finally, conclusions are drawn in Sect. 6.

[1] A preliminary version of this work (Alabort-i-Medina and Zafeiriou 2014) was presented at CVPR 2014.

2 Active Appearance Models

AAMs (Cootes et al. 2001; Matthews and Baker 2004) are generative parametric models that explain visual variations, in terms of shape and appearance, within a particular object class. AAMs are built from a collection of images (Fig. 1) 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 $\mathbf{s} = (x_1, y_1, \ldots, x_v, y_v)^T \in \mathbb{R}^{2v \times 1}$ of the object being modeled have been manually defined a priori.

AAMs are themselves composed of three different models: (i) shape model; (ii) appearance model; and (iii) motion model.

The shape model, which is 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:

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

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

$$\mathbf{x}_i = s\mathbf{R}\left(\bar{\mathbf{x}}_i + \mathbf{X}_i \mathbf{p}\right) + \mathbf{t} \tag{2}$$

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 Matthews and Baker (2004) the final expression for the shape model can be compactly written as the linear combination of a set of bases:

$$\begin{aligned}\mathbf{s} &= \bar{\mathbf{s}} + \sum_{i=1}^{4} p_i^* \mathbf{s}_i^* + \sum_{i=1}^{n} p_i \mathbf{s}_i \\ &= \bar{\mathbf{s}} + \mathbf{S}\mathbf{p}\end{aligned} \tag{3}$$

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

The appearance model is obtained by warping the original images onto a common reference frame (typically defined in terms of the mean shape $\bar{\mathbf{s}}$) and applying PCA to the obtained warped images. Mathematically, the appearance model is defined by the following expression:

$$A(\mathbf{x}) = \bar{A}(\mathbf{x}) + \sum_{i=1}^{m} c_i A_i(\mathbf{x}) \tag{4}$$

where $\mathbf{x} \in \Omega$ denote all pixel positions on the reference frame, and $\bar{A}(\mathbf{x})$, $A_i(\mathbf{x})$ and c_i denote the mean texture, the appearance bases and appearance parameters, respectively. Denoting $\mathbf{a} = \text{vec}(A(\mathbf{x}))$ as the vectorized version of the previous appearance instance, Eq. 4 can be concisely written in vector form as:

$$\mathbf{a} = \bar{\mathbf{a}} + \mathbf{A}\mathbf{c} \tag{5}$$

where $\mathbf{a} \in \mathbb{R}^{F \times 1}$ is the mean appearance, and $\mathbf{A} \in \mathbb{R}^{F \times m}$ and $\mathbf{c} \in \mathbb{R}^{m \times 1}$ denote the appearance bases and appearance parameters, respectively.

The role of the motion model, denoted by $\mathcal{W}(\mathbf{x}; \mathbf{p})$, is to extrapolate the position of all pixel positions $\mathbf{x} \in \Omega$ from the reference frame to a particular shape instance $\mathbf{s}$ (and vice-versa) based on their relative position with respect to the sparse set of landmarks defining the shape model (for which direct correspondences are always known). Classic motion models for AAMs are PieceWise Affine (PWA) (Cootes and Taylor 2004; Matthews and Baker 2004) and thin plate splines (TPS) (Cootes and Taylor 2004; Papandreou and Maragos 2008) warps.

Given an image I containing the object of interest, its manually annotated ground truth shape $\mathbf{s}$, and a particular

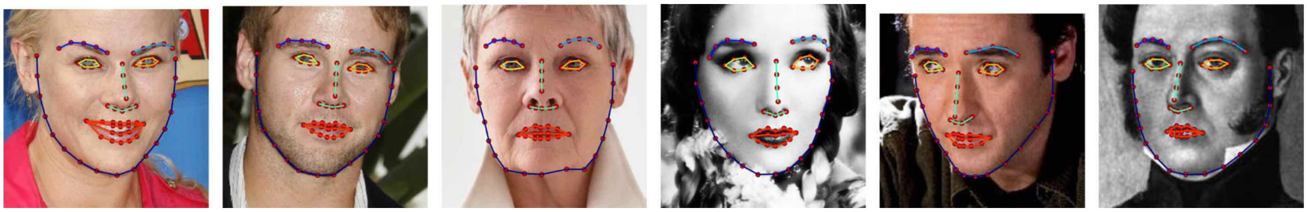


Fig. 1 Exemplar images from the Labelled Faces Parts in-the-Wild (LFPW) dataset (Belhumeur et al. 2011) for which a consistent set of sparse landmarks representing the shape of the object being model (*human face*) has been manually defined (Sagonas et al. 2013a, b)

motion model $\mathcal{W}(\mathbf{x}, \mathbf{p})$; the two main assumptions behind AAMs are:

1. The ground truth shape of the object can be well approximated by the shape model
$$\mathbf{s} \approx \bar{\mathbf{s}} + \mathbf{S}\mathbf{p} \tag{6}$$
2. The object's appearance can be well approximated by the appearance model after the image is warped, using the motion model and the previous shape approximation, onto the reference frame:
$$\mathbf{i}[\mathbf{p}] \approx \bar{\mathbf{a}} + \mathbf{A}\mathbf{c} \tag{7}$$
where $\mathbf{i}[\mathbf{p}] = \text{vec}(I(\mathcal{W}(\mathbf{x}; \mathbf{p})))$ 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 appearance models and is a central part of the AAMs formulation.

Because of the explicit use of the motion model, the two previous assumptions provide a concise definition of AAMs. At this point, it is worth mentioning that the vector notation of Eqs. 6 and 7 will be, in general, the preferred notation in this paper.

2.1 Probabilistic Formulation

A probabilistic interpretation of AAMs can be obtained by rewriting Eqs. 6 and 7 assuming probabilistic models for shape and appearance generation. In this paper, motivated by seminal works on Probabilistic Component Analysis (PPCA) and object tracking (Tipping and Bishop 1999; Roweis 1998; Moghaddam and Pentland 1997), we will assume probabilistic models for shape and appearance generation with both Gaussian noise and Gaussian priors over the latent shape and appearance spaces[2]:

$$\begin{aligned} \mathbf{s} &= \bar{\mathbf{s}} + \mathbf{S}\mathbf{p} + \boldsymbol{\varepsilon} \\ \mathbf{p} &\sim \mathcal{N}(\mathbf{0}, \Lambda) \\ \boldsymbol{\varepsilon} &\sim \mathcal{N}\left(\mathbf{0}, \varsigma^2 \mathbf{I}\right) \end{aligned} \tag{8}$$

$$\begin{aligned} \mathbf{i}[\mathbf{p}] &= \bar{\mathbf{a}} + \mathbf{A}\mathbf{c} + \boldsymbol{\epsilon} \\ \mathbf{c} &\sim \mathcal{N}(\mathbf{0}, \Sigma) \\ \boldsymbol{\epsilon} &\sim \mathcal{N}\left(\mathbf{0}, \sigma^2 \mathbf{I}\right) \end{aligned} \tag{9}$$

where the diagonal matrices $\Lambda = \text{diag}(\lambda_{\mathbf{s}_1}, \ldots, \lambda_{\mathbf{s}_m})$ and $\Sigma = \text{diag}(\lambda_{\mathbf{a}_1}, \ldots, \lambda_{\mathbf{a}_m})$ contain the eigenvalues associated to shape and appearance eigenvectors respectively and where ς^2 and σ^2 denote the estimated shape and image noise[3] respectively.

This probabilistic formulation will be used to derive Maximum-Likelihood (ML), Maximum A Posteriori (MAP) and Bayesian cost functions for fitting AAMs in Sects. 3.1.1 and 3.1.2.

3 Fitting Active Appearance Models

Several techniques have been proposed to fit AAMs to images (Cootes et al. 2001; Hou et al. 2001; Matthews and Baker 2004; Batur and Hayes 2005; Gross et al. 2005; Donner et al. 2006; Papandreou and Maragos 2008; Liu 2009; Saragih and Göcke 2009; Amberg et al. 2009; Tresadern et al. 2010; Martins et al. 2010; Sauer et al. 2011; Tzimiropoulos and Pantic 2013; Kossaifi et al. 2014; Antonakos et al. 2014). In this paper, we will center the discussion around compositional gradient descent (CGD) algorithms (Matthews and Baker 2004; Gross et al. 2005; Papandreou and Maragos 2008; Amberg et al. 2009; Martins et al. 2010; Tzimiropoulos and Pantic 2013; Kossaifi et al. 2014) for fitting AAMs. Consequently, we will not review regression based approaches. For more details on these type of methods the interested reader is referred to the existent literature (Cootes et al. 2001; Hou et al. 2001; Batur and Hayes 2005; Donner et al. 2006; Liu 2009; Saragih and Göcke 2009; Tresadern et al. 2010; Sauer et al. 2011.

The following subsections present a unified and complete view of CGD algorithms by classifying them with respect to their three main characteristics: (a) *cost function* (Sect. 3.1); (b) type of *composition* (Sect. 3.2); and (c) *optimization method* (Sect. 3.3).

3.1 Cost Function

AAM fitting is typically formulated as the (regularized) search over the shape and appearance parameters that minimize a global error measure between the vectorized warped image and the appearance model:

$$\mathbf{p}^*, \mathbf{c}^* = \underset{\mathbf{p},\mathbf{c}}{\arg\min}\; \mathcal{R}(\mathbf{p}, \mathbf{c}) + \mathcal{D}(\mathbf{i}[\mathbf{p}], \mathbf{c}) \tag{10}$$

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

[2] This formulation is generic and one could assume other probabilistic generative models (van der Maaten and Hendriks 2010; Bach and Jordan 2005; Prince et al. 2012; Nicolaou et al. 2014) to define novel probabilistic versions of AAMs.

[3] Theoretically, the optimal value for ς^2 and σ^2 is the average value of the eigenvalues associated to the discarded shape and appearance eigenvectors respectively i.e. $\varsigma^2 = \frac{1}{N-n} \sum_{i=n}^{N} \lambda_{\mathbf{s}_i}$ and $\sigma^2 = \frac{1}{M-m} \sum_{i=m}^{M} \lambda_{\mathbf{a}_i}$ (Moghaddam and Pentland 1997)

3.1.1 Sum of Squared Differences

Arguably, the most natural choice for the previous data term is the *Sum of Squared Differences* (SSD) between the vectorized warped image and the linear appearance model[4]. Consequently, the *classic* AAM fitting problem is defined by the following non-linear optimization problem[5]:

$$\begin{aligned} \mathbf{p}^*, \mathbf{c}^* &= \underset{\mathbf{p},\mathbf{c}}{\arg\min}\ \frac{1}{2}\mathbf{r}^T\mathbf{r} \\ &= \underset{\mathbf{p},\mathbf{c}}{\arg\min}\ \underbrace{\frac{1}{2}\|\mathbf{i}[\mathbf{p}] - (\bar{\mathbf{a}} + \mathbf{A}\mathbf{c})\|^2}_{\mathcal{D}(\mathbf{i}[\mathbf{p}],\mathbf{c})} \end{aligned} \tag{11}$$

On the other hand, considering regularization, the most natural choice for $\mathcal{R}$ is the sum of $\ell_2{}^2$-norms over the shape and appearance parameters. In this case, the *regularized* AAM fitting problem is defined as follows:

$$\begin{aligned} \mathbf{p}^*, \mathbf{c}^* &= \underset{\mathbf{p},\mathbf{c}}{\arg\min}\ \frac{1}{2}||\mathbf{p}||^2 + \frac{1}{2}||\mathbf{c}||^2 + \frac{1}{2}\mathbf{r}^T\mathbf{r} \\ &= \underset{\mathbf{p},\mathbf{c}}{\arg\min}\ \underbrace{\frac{1}{2}||\mathbf{p}||^2 + \frac{1}{2}||\mathbf{c}||^2}_{\mathcal{R}(\mathbf{p},\mathbf{c})} \\ &\quad + \underbrace{\frac{1}{2}||\mathbf{i}[\mathbf{p}] - (\bar{\mathbf{a}} + \mathbf{A}\mathbf{c})||^2}_{\mathcal{D}(\mathbf{i}[\mathbf{p}],\mathbf{c})} \end{aligned} \tag{12}$$

Probabilistic Formulation

A probabilistic formulation of the previous cost function can be naturally derived using the probabilistic generative models introduced in Sect. 2.1. Denoting the models' parameters as $\Theta = \{\bar{\mathbf{s}}, \mathbf{S}, \Lambda, \bar{\mathbf{a}}, \mathbf{A}, \Sigma, \sigma^2\}$ a ML formulation can be derived as follows:

$$\begin{aligned} \mathbf{p}^*, \mathbf{c}^* &= \underset{\mathbf{p},\mathbf{c}}{\arg\max}\ p(\mathbf{i}[\mathbf{p}]|\mathbf{p}, \mathbf{c}, \Theta) \\ &= \underset{\mathbf{p},\mathbf{c}}{\arg\max}\ \ln p(\mathbf{i}[\mathbf{p}]|\mathbf{p}, \mathbf{c}, \Theta) \\ &= \underset{\mathbf{p},\mathbf{c}}{\arg\min}\ \underbrace{\frac{1}{2\sigma^2}||\mathbf{i}[\mathbf{p}] - (\bar{\mathbf{a}} + \mathbf{A}\mathbf{c})||^2}_{\mathcal{D}(\mathbf{i}[\mathbf{p}],\mathbf{c})} \end{aligned} \tag{13}$$

and a MAP formulation can be similarly derived by taking into account the prior distributions over the shape and appearance parameters:

$$\begin{aligned} \mathbf{p}^*, \mathbf{c}^* &= \underset{\mathbf{p},\mathbf{c}}{\arg\max}\ p(\mathbf{p}, \mathbf{c}, \mathbf{i}[\mathbf{p}]|\Theta) \\ &= \underset{\mathbf{p},\mathbf{c}}{\arg\max}\ p(\mathbf{p}|\Lambda)p(\mathbf{c}|\Sigma)p(\mathbf{i}[\mathbf{p}]|\mathbf{p}, \mathbf{c}, \Theta) \\ &= \underset{\mathbf{p},\mathbf{c}}{\arg\max}\ \ln p(\mathbf{p}|\Lambda) + \ln p(\mathbf{c}|\Sigma) \\ &\quad + \ln p(\mathbf{i}[\mathbf{p}]|\mathbf{p}, \mathbf{c}, \Theta) \\ &= \underset{\mathbf{p},\mathbf{c}}{\arg\min}\ \underbrace{\frac{1}{2}||\mathbf{p}||^2_{\Lambda^{-1}} + \frac{1}{2}||\mathbf{c}||^2_{\Sigma^{-1}}}_{\mathcal{R}(\mathbf{p},\mathbf{c})} \\ &\quad + \underbrace{\frac{1}{2\sigma^2}||\mathbf{i}[\mathbf{p}] - (\bar{\mathbf{a}} + \mathbf{A}\mathbf{c})||^2}_{\mathcal{D}(\mathbf{i}[\mathbf{p}],\mathbf{c})} \end{aligned} \tag{14}$$

where we have assumed the shape and appearance parameters to be independent[6].

The previous ML and MAP formulations are weighted version of the optimization problem defined by Eqs. 11 and 12. In both cases, the maximization of the conditional probability of the vectorized warped image given the shape, appearance 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 appearance parameters leads to the minimization of the regularization term $\mathcal{R}$.

3.1.2 Project-Out

Matthews and Baker showed in Matthews and Baker (2004) that one could express the SSD between the vectorized warped image and the linear PCA-based[7] appearance model as the sum of two different terms:

$$\begin{aligned} \frac{1}{2}\mathbf{r}^T\mathbf{r} &= \frac{1}{2}\mathbf{r}^T(\mathbf{A}\mathbf{A}^T + \mathbf{I} - \mathbf{A}\mathbf{A}^T)\mathbf{r} \\ &= \frac{1}{2}\mathbf{r}^T(\mathbf{A}\mathbf{A}^T)\mathbf{r} + \frac{1}{2}\mathbf{r}^T(\mathbf{I} - \mathbf{A}\mathbf{A}^T)\mathbf{r} \\ &= \frac{1}{2}\ \|\mathbf{i}[\mathbf{p}] - (\bar{\mathbf{a}} + \mathbf{A}\mathbf{c})\|^2_{\mathbf{A}\mathbf{A}^T} \\ &\quad + \frac{1}{2}\ \|\mathbf{i}[\mathbf{p}] - (\bar{\mathbf{a}} + \mathbf{A}\mathbf{c})\|^2_{\mathbf{I}-\mathbf{A}\mathbf{A}^T} \\ &= f_1(\mathbf{p}, \mathbf{c}) + f_2(\mathbf{p}, \mathbf{c}) \end{aligned} \tag{15}$$

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

[4] This choice of $\mathcal{D}$ is naturally given by second main assumption behind AAMs, Eq. 7 and by the linear generative model of appearance defined by Eq. 9.

[5] The residual $\mathbf{r}$ in Eq. 11 is linear with respect to the appearance parameters $\mathbf{c}$ and non-linear with respect to the shape parameters $\mathbf{p}$ through the warp $\mathcal{W}(\mathbf{x}; \mathbf{p})$

[6] This is a common assumption in CGD algorithms (Matthews and Baker 2004), however, in reality, some degree of dependence between these parameters is to be expected (Cootes et al. 2001).

[7] The use of PCA ensures the orthonormality of the appearance 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 appearance mean and the appearance bases and, hence, $\mathbf{A}^T\bar{\mathbf{a}} = \mathbf{0}$.

$$\begin{aligned} f_1(\mathbf{p},\mathbf{c}) &= \frac{1}{2}\,\|\mathbf{i}[\mathbf{p}] - (\bar{\mathbf{a}} + \mathbf{A}\mathbf{c})\|^2_{\mathbf{A}\mathbf{A}^T} \\ &= \frac{1}{2}\Bigg(\underbrace{\mathbf{i}[\mathbf{p}]^T\mathbf{A}}_{\mathbf{c}^T}\underbrace{\mathbf{A}^T\mathbf{i}[\mathbf{p}]}_{\mathbf{c}} - \underbrace{2\,\overbrace{\mathbf{i}[\mathbf{p}]^T\mathbf{A}}^{\mathbf{c}^T}\overbrace{\mathbf{A}^T\bar{\mathbf{a}}}^{\mathbf{0}}}_{0} \\ &\quad - 2\,\underbrace{\mathbf{i}[\mathbf{p}]^T\mathbf{A}}_{\mathbf{c}^T}\underbrace{\overbrace{\mathbf{A}^T\mathbf{A}}^{\mathbf{I}}\mathbf{c}}_{\mathbf{c}} + \underbrace{\overbrace{\bar{\mathbf{a}}^T\mathbf{A}}^{\mathbf{0}^T}\overbrace{\mathbf{A}^T\bar{\mathbf{a}}}^{\mathbf{0}}}_{0} \\ &\quad + 2\,\underbrace{\overbrace{\bar{\mathbf{a}}^T\mathbf{A}}^{\mathbf{0}^T}\overbrace{\mathbf{A}^T\mathbf{A}}^{\mathbf{I}}\mathbf{c}}_{0} + \underbrace{\mathbf{c}^T\,\overbrace{\mathbf{A}^T\mathbf{A}}^{\mathbf{I}}}_{\mathbf{c}^T}\underbrace{\overbrace{\mathbf{A}^T\mathbf{A}}^{\mathbf{I}}\mathbf{c}}_{\mathbf{c}}\Bigg) \\ &= \frac{1}{2}(\mathbf{c}^T\mathbf{c} - 2\mathbf{c}^T\mathbf{c} + \mathbf{c}^T\mathbf{c}) \\ &= 0 \end{aligned} \tag{16}$$

The second term measures the distance *to* the appearance 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 **p**:

$$\begin{aligned} f_2(\mathbf{p},\mathbf{c}) &= \frac{1}{2}\,\|\mathbf{i}[\mathbf{p}] - (\bar{\mathbf{a}} + \mathbf{A}\mathbf{c})\|^2_{\bar{\mathbf{A}}} \\ &= \frac{1}{2}\Big(\mathbf{i}[\mathbf{p}]^T\bar{\mathbf{A}}\mathbf{i}[\mathbf{p}] - 2\mathbf{i}[\mathbf{p}]^T\bar{\mathbf{A}}\bar{\mathbf{a}} \\ &\quad - \underbrace{2\mathbf{i}[\mathbf{p}]^T\bar{\mathbf{A}}\mathbf{A}\mathbf{c}}_{0} + \bar{\mathbf{a}}^T\bar{\mathbf{A}}\bar{\mathbf{a}} \\ &\quad + 2\,\underbrace{\bar{\mathbf{a}}^T\bar{\mathbf{A}}\mathbf{A}\mathbf{c}}_{0} + \underbrace{\mathbf{c}^T\mathbf{A}^T\bar{\mathbf{A}}\mathbf{A}\mathbf{c}}_{0}\Big) \\ &= \frac{1}{2}(\mathbf{i}[\mathbf{p}]^T\bar{\mathbf{A}}\mathbf{i}[\mathbf{p}] - 2\mathbf{i}[\mathbf{p}]^T\bar{\mathbf{A}}\bar{\mathbf{a}} + \bar{\mathbf{a}}^T\bar{\mathbf{A}}\bar{\mathbf{a}}) \\ &= \frac{1}{2}\,\|\mathbf{i}[\mathbf{p}] - \bar{\mathbf{a}}\|^2_{\bar{\mathbf{A}}} \end{aligned} \tag{17}$$

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

$$f_2(\mathbf{p},\mathbf{c}) = \hat{f}_2(\mathbf{p}) = \frac{1}{2}\,\|\mathbf{i}[\mathbf{p}] - \bar{\mathbf{a}}\|^2_{\bar{\mathbf{A}}} \tag{18}$$

Therefore, using the previous *project-out trick*, the minimization problems defined by Eqs. 11 and 12 reduce to:

$$\mathbf{p}^* = \arg\min_{\mathbf{p}}\ \underbrace{\frac{1}{2}||\mathbf{i}[\mathbf{p}] - \bar{\mathbf{a}}||^2_{\bar{\mathbf{A}}}}_{\mathcal{D}(\mathbf{i}[\mathbf{p}])} \tag{19}$$

and

$$\mathbf{p}^* = \arg\min_{\mathbf{p}}\ \underbrace{\frac{1}{2}||\mathbf{p}||^2}_{\mathcal{R}(\mathbf{p})} + \underbrace{\frac{1}{2}||\mathbf{i}[\mathbf{p}] - \bar{\mathbf{a}}||^2_{\bar{\mathbf{A}}}}_{\mathcal{D}(\mathbf{i}[\mathbf{p}])} \tag{20}$$

respectively.

Probabilistic Formulation

Assuming the probabilistic models defined in Sect. 2.1, a *Bayesian* formulation of the previous project-out data term can be naturally derived by marginalizing over the appearance parameters to obtain the following marginalized density:

$$\begin{aligned} p(\mathbf{i}[\mathbf{p}]|\mathbf{p},\Theta) &= \int_c p(\mathbf{i}[\mathbf{p}]|\mathbf{p},\mathbf{c},\Theta)\,p(\mathbf{c}|\Sigma)d\mathbf{c} \\ &= \mathcal{N}(\bar{\mathbf{a}}, \mathbf{A}\Sigma\mathbf{A}^T + \sigma^2\mathbf{I}) \end{aligned} \tag{21}$$

and applying the Woodbury formula[8] Woodbury (1950) to decompose the natural logarithm of the previous density into the sum of two different terms:

$$\begin{aligned} \ln p(\mathbf{i}[\mathbf{p}]|\mathbf{p},\Theta) &= \frac{1}{2}||\mathbf{i}[\mathbf{p}] - \bar{\mathbf{a}}||^2_{(\mathbf{A}\Sigma\mathbf{A}^T + \sigma^2\mathbf{I})^{-1}} \\ &= \frac{1}{2}||\mathbf{i}[\mathbf{p}] - \bar{\mathbf{a}}||^2_{\mathbf{A}\mathbf{D}^{-1}\mathbf{A}^T} \\ &\quad + \frac{1}{2\sigma^2}||\mathbf{i}[\mathbf{p}] - \bar{\mathbf{a}}||^2_{\bar{\mathbf{A}}} \end{aligned} \tag{22}$$

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

As depicted by Fig. 2, the previous two terms define respectively: (i) the Mahalanobis distance *within* the linear appearance subspace; and (ii) the Euclidean distance *to* the linear appearance 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 appearance space increases (and especially as $\Sigma \to \infty$) **c** becomes uniformly distributed and the contribution of the first term $\frac{1}{2}||\mathbf{i}[\mathbf{p}] - \bar{\mathbf{a}}||^2_{\mathbf{A}\mathbf{D}^{-1}\mathbf{A}^T}$ vanishes; in this case, we obtain a

[8] Using the Woodbury formula:

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

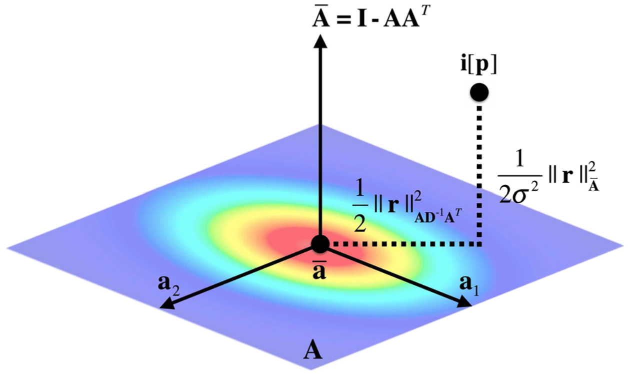

$\bar{\mathbf{A}} = \mathbf{I} - \mathbf{A}\mathbf{A}^T$
$\mathbf{i}[\mathbf{p}]$
$\frac{1}{2\sigma^2} \| \mathbf{r} \|^2_{\bar{\mathbf{A}}}$
$\frac{1}{2} \| \mathbf{r} \|^2_{\mathbf{A}\mathbf{D}^{-1}\mathbf{A}^T}$
$\bar{\mathbf{a}}$
$\mathbf{a}_2$
$\mathbf{a}_1$
$\mathbf{A}$

Fig. 2 The fits AAMs by minimizing two different distances: (*i*) the Mahalanobis distance *within* the linear appearance subspace; and (*ii*) the Euclidean distance *to* the linear appearance subspace (i.e. the Euclidean distance within its orthogonal complement) weighted by the inverse of the estimated image noise

weighted version of the project-out data term defined by Eq. 19. Hence, given our Bayesian formulation, the project-out loss arises naturally by assuming a uniform prior over the latent appearance space.

The probabilistic formulations of the minimization problems defined by Eqs. 19 and 20 can be derived, from the previous Bayesian Project-Out (BPO) cost function, as

$$\begin{aligned} \mathbf{p}^* &= \arg\max_{\mathbf{p}} \; \ln p(\mathbf{i}[\mathbf{p}] | \mathbf{p}, \Theta) \\ &= \arg\min_{\mathbf{p}} \; \underbrace{\frac{1}{2\sigma^2} ||\mathbf{i}[\mathbf{p}] - \bar{\mathbf{a}}||^2_{\mathbf{Q}}}_{\mathcal{D}(\mathbf{i}[\mathbf{p}])} \end{aligned} \tag{23}$$

and

$$\begin{aligned} \mathbf{p}^* &= \arg\max_{\mathbf{p}} \; p(\mathbf{p}, \mathbf{i}[\mathbf{p}] | \Theta) \\ &= \arg\max_{\mathbf{p}} \; p(\mathbf{p}|\Lambda) p(\mathbf{i}[\mathbf{p}] | \mathbf{p}, \Theta) \\ &= \arg\max_{\mathbf{p}} \; \ln p(\mathbf{p}|\Lambda) + \ln p(\mathbf{i}[\mathbf{p}] | \mathbf{p}, \Theta) \\ &= \arg\min_{\mathbf{p}} \; \underbrace{\frac{1}{2} ||\mathbf{p}||^2_{\Lambda^{-1}}}_{\mathcal{R}(\mathbf{p})} + \underbrace{\frac{1}{2\sigma^2} ||\mathbf{i}[\mathbf{p}] - \bar{\mathbf{a}}||^2_{\mathbf{Q}}}_{\mathcal{D}(\mathbf{i}[\mathbf{p}])} \end{aligned} \tag{24}$$

respectively. Where we have defined the BPO operator as $\mathbf{Q} = \mathbf{I} - \mathbf{A}(\mathbf{I} - \sigma^2 \mathbf{D}^{-1})\mathbf{A}^T$.

3.2 Type of Composition

Assuming, for the time being, that the true appearance parameters $\mathbf{c}^*$ are known, the problem defined by Eq. 11 reduces to a non-rigid image alignment problem (Baker and Matthews 2004; Muñoz et al. 2014) between the particular instance of the object present in the image and its optimal appearance reconstruction by the appearance model:

$$\mathbf{p}^* = \arg\min_{\mathbf{p}} \; \frac{1}{2} \|\mathbf{i}[\mathbf{p}] - \mathbf{a}\|^2 \tag{25}$$

where $\mathbf{a} = \bar{\mathbf{a}} + \mathbf{A}\mathbf{c}^*$ is obtained by directly evaluating Eq. 4 given the true appearance parameters $\mathbf{c}^*$.

CGD algorithms iteratively solve the previous 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 previous 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 AAMs have introduced the incremental warp either on the image or the model sides in what are known as *forward* and *inverse* compositional frameworks (Matthews and Baker 2004; Gross et al. 2005; Papandreou and Maragos 2008; Amberg et al. 2009; Martins et al. 2010; Tzimiropoulos and Pantic 2013) respectively. Inspired by related works in field of image alignment (Malis 2004; Mégret et al. 2008; Autheserre et al. 2009; Mégret et al. 2010), we notice that novel CGD algorithms can be derived by introducing incremental warps on both image and model sides simultaneously. Depending on the exact relationship between these incremental warps we define two novel types of composition: *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 paper: (i) forward; (ii) inverse; (iii) asymmetric; and (v) bidirectional. These subsections will be derived using the non-regularized expression in Eq. 11 and the regularized expression in Eq. 14. Furthermore, to maintain consistency with the vector notation used through out the paper, we will abuse the notation and write the operations of warp composition[9] $\mathcal{W}(\mathbf{x}; \mathbf{p}) \circ \mathcal{W}(\mathbf{x}; \Delta\mathbf{p})$ and inversion[1]. $\mathcal{W}(\mathbf{x}; \mathbf{q})^{-1}$ as simply $\mathbf{p} \circ \Delta\mathbf{p}$ and $\mathbf{q}^{-1}$ respectively.

[9] Further details regarding composition, $\mathbf{p} \circ \Delta\mathbf{p}$, and inversion, $\Delta\mathbf{q}^{-1}$, of typical AAMs' motion models such as PWA and TPS warps can be found in Matthews and Baker (2004), Papandreou and Maragos (2008).

3.2.1 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 Eq. 11 this leads to:

$$\Delta\mathbf{p}^* = \arg\min_{\Delta\mathbf{p}} \frac{1}{2} \|\mathbf{i}[\mathbf{p} \circ \Delta\mathbf{p}] - (\bar{\mathbf{a}} + \mathbf{A}\mathbf{c})\|^2 \tag{26}$$

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:

$$\mathbf{p} \leftarrow \mathbf{p} \circ \Delta\mathbf{p} \tag{27}$$

On the other hand, using Eq. 14, forward composition can be expressed as:

$$\begin{aligned}\Delta\mathbf{p}^* = \arg\min_{\Delta\mathbf{p}} & \frac{1}{2\sigma^2}||\mathbf{i}[\mathbf{p} \circ \Delta\mathbf{p}] - (\bar{\mathbf{a}} + \mathbf{A}\mathbf{c})||^2 \\ & + \frac{1}{2}||\mathbf{p}||^2_{\Lambda^{-1}} + ||\mathbf{c}||^2_{\Sigma^{-1}}\end{aligned} \tag{28}$$

Because of the inclusion of the prior term over the shape parameters $\frac{1}{2}||\mathbf{p}||^2_{\Lambda^{-1}}$, we cannot update the current warp estimate using the update rule in Eq. 27. Instead, as noted by Papandreou and Maragos in Papandreou and Maragos (2008), we need to compute the *forward compositional* to *forward additive* parameter update Jacobian matrix $J_{\mathbf{p}} \in \mathbb{R}^{(4+n)\times(4+n)}$[10]. This matrix is used to map the forward compositional increment $\Delta\mathbf{p}$ to its *first* order additive equivalent $J_{\mathbf{p}}\Delta\mathbf{p}$. In this case, the current estimate of the warp is computed using the following update rule:

$$\begin{aligned}\mathbf{p} \leftarrow & \left(\Lambda^{-1} + \left(J_{\mathbf{p}}\mathbf{H}J_{\mathbf{p}}\right)^{-1}\right)^{-1} \\ & \left(\left(J_{\mathbf{p}}\mathbf{H}J_{\mathbf{p}}\right)^{-1}\left(\mathbf{p} + J_{\mathbf{p}}\Delta\mathbf{p}\right)\right)\end{aligned} \tag{29}$$

where $\mathbf{H}$ denotes the approximate or true *Hessians* of the residual $||\mathbf{i}[\mathbf{p} \circ \Delta\mathbf{p}] - (\mathbf{a} + \mathbf{A}\mathbf{c})||^2$ with respect to the incremental parameters $\Delta\mathbf{p}$ and $\Delta\mathbf{p}$ itself is the optimal solution of the non-regularized problem in Eq. 26. Note that, in Sect. 3.3, we derive $\mathbf{H}$ for all the optimization methods studied in this paper.

3.2.2 Inverse

On the other hand, the inverse compositional framework inverts the roles of the image and the model by introducing the incremental warp on the model side. Using Eq. 11:

$$\Delta\mathbf{q}^* = \arg\min_{\Delta\mathbf{q}} \frac{1}{2}||\mathbf{i}[\mathbf{p}] - (\bar{\mathbf{a}} + \mathbf{A}\mathbf{c})[\Delta\mathbf{q}]||^2 \tag{30}$$

Note that, in this case, the model is the one we seek to deform using the 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:

$$\mathbf{p} \leftarrow \mathbf{p} \circ \Delta\mathbf{q}^{-1} \tag{31}$$

Simarly, using the regularized expression in Eq. 14, inverse compositon is expressed as:

$$\begin{aligned}\Delta\mathbf{q}^* = \arg\min_{\Delta\mathbf{q}} & \frac{1}{2\sigma^2}||\mathbf{i}[\mathbf{p}] - (\bar{\mathbf{a}} + \mathbf{A}\mathbf{c})[\Delta\mathbf{q}]||^2 \\ & + \frac{1}{2}||\mathbf{p}||^2_{\Lambda^{-1}} + ||\mathbf{c}||^2_{\Sigma^{-1}}\end{aligned} \tag{32}$$

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

$$\begin{aligned}\mathbf{p} \leftarrow & \left(\Lambda^{-1} + \left(J_{\mathbf{q}}\mathbf{H}J_{\mathbf{q}}\right)^{-1}\right)^{-1} \\ & \left(\left(J_{\mathbf{q}}\mathbf{H}J_{\mathbf{q}}\right)^{-1}\left(\mathbf{p} + J_{\mathbf{q}}\Delta\mathbf{q}\right)\right)\end{aligned} \tag{33}$$

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 (2008).

3.2.3 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 Eq. 11 this is expressed as:

$$\begin{aligned}\Delta\mathbf{p}^* = \arg\min_{\Delta\mathbf{p}} & \frac{1}{2}||\mathbf{i}[\mathbf{p} \circ \alpha\Delta\mathbf{p}] \\ & - (\bar{\mathbf{a}} + \mathbf{A}\mathbf{c})[\beta\Delta\mathbf{p}^{-1}]||^2\end{aligned} \tag{34}$$

Note that the previous two incremental warps are defined to be each others inverse. Consequently, using the first order

[10] Note 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 previous parameter update Jacobian matrices can be found in Papandreou and Maragos (2008), 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

approximation to warp inversion for typical AAMs warps $\Delta\mathbf{p}^{-1} = -\Delta\mathbf{p}$ defined in Matthews and Baker (2004), we can rewrite the previous asymmetric cost function as:

$$\Delta\mathbf{p}^* = \arg\min_{\Delta\mathbf{p}} \frac{1}{2}||\mathbf{i}[\mathbf{p} \circ \alpha\Delta\mathbf{p}] - (\bar{\mathbf{a}} + \mathbf{A}\mathbf{c})[-\beta\Delta\mathbf{p}||^2 \tag{35}$$

Although this cost function will need to be linearized around both incremental warps, the parameters $\Delta\mathbf{p}$ controlling both 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 previous forward and inverse compositional update rules into a single compositional update rule:

$$\mathbf{p} \leftarrow \mathbf{p} \circ \alpha\Delta\mathbf{p} \circ \beta\Delta\mathbf{p} \tag{36}$$

In this case, using Eq. 14, asymmetric compositon is expressed as:

$$\Delta\mathbf{p}^* = \arg\min_{\Delta\mathbf{q}} \frac{1}{2\sigma^2}||\mathbf{i}[\mathbf{p} \circ \alpha\Delta\mathbf{p}] - (\bar{\mathbf{a}} + \mathbf{A}\mathbf{c})[-\beta\Delta\mathbf{p}]||^2 + \frac{1}{2}||\mathbf{p}||^2_{\Lambda^{-1}} + ||\mathbf{c}||^2_{\Sigma^{-1}} \tag{37}$$

And the current warp estimate is updates using:

$$\mathbf{p} \leftarrow \left(\Lambda^{-1} + \left(J_\mathbf{p}\mathbf{H}J_\mathbf{p}\right)^{-1}\right)^{-1} \left(\left(J_\mathbf{p}\mathbf{H}J_\mathbf{p}\right)^{-1}\left(\mathbf{p} + \alpha J_\mathbf{p}\Delta\mathbf{p} + \beta J_\mathbf{p}\Delta\mathbf{p}\right)\right) \tag{38}$$

which reduces the forward update rule in Eq. 29 because $\alpha + \beta = 1$.

Note that, the special case in which $\alpha = \beta = 0.5$ is also referred to as *symmetric* composition (Mégret et al. 2008; Autheserre et al. 2009; Mégret et al. 2010) and that the previous forward and inverse compositions can also be obtained from asymmetric composition by setting $\alpha = 1$, $\beta = 0$ and $\alpha = 0$, $\beta = 1$ respectively.

3.2.4 Bidirectional

Similar to the previous 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 Eq. 11:

$$\Delta\mathbf{p}^*, \Delta\mathbf{q}^* = \arg\min_{\Delta\mathbf{p},\Delta\mathbf{q}} \frac{1}{2}||\mathbf{i}[\mathbf{p} \circ \Delta\mathbf{p}] - (\bar{\mathbf{a}} + \mathbf{A}\mathbf{c})[\Delta\mathbf{q}]||^2 \tag{39}$$

Consequently, in Step 4, 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:

$$\mathbf{p} \leftarrow \mathbf{p} \circ \Delta\mathbf{p} \circ \Delta\mathbf{q}^{-1} \tag{40}$$

For Eq. 14, bidirectional compositon is written as:

$$\Delta\mathbf{p}^*, \Delta\mathbf{q}^* = \arg\min_{\Delta\mathbf{p},\Delta\mathbf{q}} \frac{1}{2\sigma^2}||\mathbf{i}[\mathbf{p} \circ \Delta\mathbf{p}] - (\bar{\mathbf{a}} + \mathbf{A}\mathbf{c})[\Delta\mathbf{q}]||^2 + \frac{1}{2}||\mathbf{p}||^2_{\Lambda^{-1}} + ||\mathbf{c}||^2_{\Sigma^{-1}} \tag{41}$$

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

$$\mathbf{p} \leftarrow \left(\Lambda^{-1} + \left(J_\mathbf{p}\mathbf{H}J_\mathbf{p} + J_\mathbf{q}\mathbf{H}J_\mathbf{q}\right)^{-1}\right)^{-1} \left(\left(J_\mathbf{p}\mathbf{H}J_\mathbf{p} + J_\mathbf{q}\mathbf{H}J_\mathbf{q}\right)^{-1}\left(\mathbf{p} + J_\mathbf{p}\Delta\mathbf{p} + J_\mathbf{q}\Delta\mathbf{q}\right)\right) \tag{42}$$

which reduces the forward update rule in Eq. 29 because $\alpha + \beta = 1$.

3.3 Optimization Method

Step 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. In this paper, we distinguish between three main optimization methods[11]: (i) *Gauss-Newton* (Boyd and Vandenberghe 2004; Matthews and Baker 2004; Gross et al. 2005; Martins et al. 2010; Papandreou and Maragos 2008; Tzimiropoulos and Pantic 2013); (ii) *Newton* (Boyd and Vandenberghe 2004; Kossaifi et al. 2014); and (iii) *Wiberg* (Okatani and Deguchi 2006; Strelow 2012; Papandreou and Maragos 2008; Tzimiropoulos and Pantic 2013).

These methods can be used to iteratively solve the nonlinear optimization problems defined by Eqs. 14 and 22. The main differences between them are:

[11] Amberg et al. proposed the use of the *Steepest Descent* method (Boyd and Vandenberghe 2004) in Amberg et al. (2009). However, their approach requires a special formulation of the motion model and it performs poorly using the standard independent AAM formulation (Matthews and Baker 2004) used in this work.

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* fashion while Wiberg defines its own procedure to solve for different sets of parameters[12].

The following subsections thoroughly explain how the previous 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 (Eq. 11) with both asymmetric (Sect. 3.2.3) and bidirectional (Sect. 3.2.4) compositions[13] while only direct solutions will be given for the Project-Out data term (Eq. 19). Note that, in Sect. 3.2, we already derived update rules for the regularized expression in Eq. 14 and, consequently, there is no need to consider regularization throughout this section.[14]

3.3.1 Gauss-Newton

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

$$\Delta\mathbf{c}^*, \Delta\mathbf{p}^* = \underset{\Delta\mathbf{c}, \Delta\mathbf{p}}{\arg\min} \frac{1}{2}\mathbf{r}_a^T\mathbf{r}_a \tag{43}$$

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

$$\mathbf{r}_a = \mathbf{i}[\mathbf{p} \circ \alpha\Delta\mathbf{p}] - (\bar{\mathbf{a}} + \mathbf{A}(\mathbf{c} + \Delta\mathbf{c}))[\beta\Delta\mathbf{p}^{-1}] \tag{44}$$

and where we have introduced the incremental appearance parameters $\Delta\mathbf{c}$[15]. The Gauss-Newton method solves the previous optimization problem by performing a *first* order Taylor expansion of the residual:

$$\begin{aligned}\mathbf{r}_a(\Delta\boldsymbol{\ell}) &\approx \hat{\mathbf{r}}_a(\Delta\boldsymbol{\ell}) \\ &\approx \mathbf{r}_a + \frac{\partial\mathbf{r}_a}{\partial\Delta\boldsymbol{\ell}}\Delta\boldsymbol{\ell}\end{aligned} \tag{45}$$

and solving the following approximation of the original problem:

$$\Delta\boldsymbol{\ell}^* = \underset{\Delta\boldsymbol{\ell}}{\arg\min} \frac{1}{2}\hat{\mathbf{r}}_a^T\hat{\mathbf{r}}_a \tag{46}$$

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

$$\begin{aligned}\frac{\partial\mathbf{r}_a}{\partial\Delta\boldsymbol{\ell}} &= \left(\frac{\partial\mathbf{r}_a}{\partial\Delta\mathbf{c}}, \frac{\partial\mathbf{r}_a}{\partial\Delta\mathbf{p}}\right) \\ &= \left(-\mathbf{A}, \nabla\mathbf{t}\frac{\partial\mathcal{W}}{\partial\Delta\mathbf{p}}\right) \\ &= (-\mathbf{A}, \mathbf{J_t})\end{aligned} \tag{47}$$

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

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

$$\Delta\mathbf{c}^*, \Delta\mathbf{p}^*, \Delta\mathbf{q}^* = \underset{\Delta\mathbf{c}, \Delta\mathbf{p}, \Delta\mathbf{q}}{\arg\min} \frac{1}{2}\mathbf{r}_b^T\mathbf{r}_b \tag{48}$$

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

$$\mathbf{r}_b = \mathbf{i}[\mathbf{p} \circ \Delta\mathbf{p}] - (\bar{\mathbf{a}} + \mathbf{A}(\mathbf{c} + \Delta\mathbf{c}))[\Delta\mathbf{q}] \tag{49}$$

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

$$\begin{aligned}\mathbf{r}_b(\Delta\boldsymbol{\ell}) &\approx \hat{\mathbf{r}}_b(\Delta\boldsymbol{\ell}) \\ &\approx \mathbf{r}_b + \frac{\partial\mathbf{r}_b}{\partial\Delta\boldsymbol{\ell}}\Delta\boldsymbol{\ell}\end{aligned} \tag{50}$$

and solving the approximated problem:

$$\Delta\boldsymbol{\ell}^* = \underset{\Delta\boldsymbol{\ell}}{\arg\min} \frac{1}{2}\hat{\mathbf{r}}_b^T\hat{\mathbf{r}}_b \tag{51}$$

where, in this case, $\Delta\boldsymbol{\ell} = (\Delta\mathbf{c}^T, \Delta\mathbf{p}^T, \Delta\mathbf{q}^T)^T$ and the Jacobian of the residual is defined as:

$$\begin{aligned}\frac{\partial\mathbf{r}_b}{\partial\Delta\boldsymbol{\ell}} &= \left(\frac{\partial\mathbf{r}_b}{\partial\Delta\mathbf{c}}, \frac{\partial\mathbf{r}_b}{\partial\Delta\mathbf{p}}, \frac{\partial\mathbf{r}_b}{\partial\Delta\mathbf{q}}\right) \\ &= (-\mathbf{A}, \mathbf{J_i}, -\mathbf{J_a})\end{aligned} \tag{52}$$

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

Simultaneous

The optimization problem defined by Eqs. 46 and 51 can be solved with respect to all parameters simultaneously by simply equating their derivative to 0:

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

[13] These 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 appearance parameters.

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

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

$$
\begin{aligned}
0 &= \frac{\partial \frac{1}{2}\hat{\mathbf{r}}^T\hat{\mathbf{r}}}{\partial \Delta \boldsymbol{\ell}} \\
&= \frac{\partial \frac{1}{2}(\mathbf{r} + \frac{\partial \mathbf{r}}{\partial \Delta \boldsymbol{\ell}}\Delta \boldsymbol{\ell})^T(\mathbf{r} + \frac{\partial \mathbf{r}}{\partial \Delta \boldsymbol{\ell}}\Delta \boldsymbol{\ell})}{\partial \Delta \boldsymbol{\ell}} \\
&= \left(\mathbf{r} + \frac{\partial \mathbf{r}}{\partial \Delta \boldsymbol{\ell}}\Delta \boldsymbol{\ell}\right)\left(\frac{\partial \mathbf{r}}{\partial \Delta \boldsymbol{\ell}}\right)^T
\end{aligned}
\tag{53}
$$

The solution is given by:

$$
\Delta \boldsymbol{\ell}^* = -\left(\left(\frac{\partial \mathbf{r}}{\partial \Delta \boldsymbol{\ell}}\right)^T \frac{\partial \mathbf{r}}{\partial \Delta \boldsymbol{\ell}}\right)^{-1}\left(\frac{\partial \mathbf{r}}{\partial \Delta \boldsymbol{\ell}}\right)^T \mathbf{r} \tag{54}
$$

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

Directly inverting $\left((\frac{\partial \mathbf{r}}{\partial \Delta \boldsymbol{\ell}})^T \frac{\partial \mathbf{r}}{\partial \Delta \boldsymbol{\ell}}\right)$ has complexity[16] $O((n+m)^3)$ 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*[17] (Boyd and Vandenberghe 2004).

For *asymmetric* composition we have:

$$
\begin{aligned}
-\left(\left(\frac{\partial \mathbf{r}_a}{\partial \Delta \boldsymbol{\ell}}\right)^T \frac{\partial \mathbf{r}_a}{\partial \Delta \boldsymbol{\ell}}\right)\Delta \boldsymbol{\ell} &= \left(\frac{\partial \mathbf{r}_a}{\partial \Delta \boldsymbol{\ell}}\right)^T \mathbf{r} \\
\begin{pmatrix} -\underbrace{\mathbf{A}^T\mathbf{A}}_{\mathbf{I}} & \mathbf{A}^T\mathbf{J_t} \\ \mathbf{J_t}^T\mathbf{A} & -\mathbf{J_t}^T\mathbf{J_t} \end{pmatrix}\begin{pmatrix} \Delta\mathbf{c} \\ \Delta\mathbf{p} \end{pmatrix} &= \begin{pmatrix} -\mathbf{A}^T \\ \mathbf{J_t}^T \end{pmatrix}\mathbf{r}_a
\end{aligned}
\tag{55}
$$

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

$$
\begin{aligned}
-(\mathbf{J_t}^T\mathbf{J_t} + \mathbf{J_t}^T\mathbf{A}\mathbf{A}^T\mathbf{J_t}^T)\Delta\mathbf{p} &= \mathbf{J_t}^T\mathbf{r} - \mathbf{J_t}^T\mathbf{A}\mathbf{A}^T\mathbf{r}_a \\
-\mathbf{J_t}^T(\mathbf{I} - \mathbf{A}\mathbf{A}^T)\mathbf{J_t}\Delta\mathbf{p} &= \mathbf{J_t}^T(\mathbf{I} - \mathbf{A}\mathbf{A}^T)\mathbf{r}_a \\
-\mathbf{J_t}^T\bar{\mathbf{A}}\mathbf{J_t}\Delta\mathbf{p} &= \mathbf{J_t}^T\bar{\mathbf{A}}\mathbf{r}_a \\
\Delta\mathbf{p}^* &= -\left(\mathbf{J_t}^T\bar{\mathbf{A}}\mathbf{J_t}\right)^{-1} \\
&\quad \mathbf{J_t}^T\bar{\mathbf{A}}\mathbf{r}_a
\end{aligned}
\tag{56}
$$

and plugging the solution for $\Delta\mathbf{p}$ into Eq. 55 the optimal value for $\Delta\mathbf{c}$ is obtained by:

$$
\begin{aligned}
-\Delta\mathbf{c} + \mathbf{A}^T\mathbf{J_t}\Delta\mathbf{p} &= -\mathbf{A}^T\mathbf{r}_a \\
\Delta\mathbf{c}^* &= \mathbf{A}^T\,(\mathbf{r}_a + \mathbf{J_t}\Delta\mathbf{p})
\end{aligned}
\tag{57}
$$

Using the above procedure the complexity[17] of solving each Gauss-Newton step is reduced to:

$$
O(\underbrace{nmF}_{\mathbf{J_t}^T\bar{\mathbf{A}}} + \underbrace{n^2F + n^3}_{(\mathbf{J_t}^T\bar{\mathbf{A}}\mathbf{J_t})^{-1}}) \tag{58}
$$

Using *bidirectional* composition, we can apply the Schur complement either one or two times in order to take advantage of the 3×3 block structure of the matrix $\left((\frac{\partial \mathbf{r}_b}{\partial \Delta \boldsymbol{\ell}})^T \frac{\partial \mathbf{r}_b}{\partial \Delta \boldsymbol{\ell}}\right)$:

$$
\begin{aligned}
-\left(\left(\frac{\partial \mathbf{r}_b}{\partial \Delta \boldsymbol{\ell}}\right)^T \frac{\partial \mathbf{r}_b}{\partial \Delta \boldsymbol{\ell}}\right)\Delta \boldsymbol{\ell} &= \left(\frac{\partial \mathbf{r}_b}{\partial \Delta \boldsymbol{\ell}}\right)^T \mathbf{r}_b \\
-\left(\left(\frac{\partial \mathbf{r}_b}{\partial \Delta \boldsymbol{\ell}}\right)^T \frac{\partial \mathbf{r}_b}{\partial \Delta \boldsymbol{\ell}}\right)\begin{pmatrix} \Delta\mathbf{c} \\ \Delta\mathbf{p} \\ \Delta\mathbf{q} \end{pmatrix} &= \begin{pmatrix} -\mathbf{A}^T \\ \mathbf{J_i}^T \\ -\mathbf{J_a}^T \end{pmatrix}\mathbf{r}_b
\end{aligned}
\tag{59}
$$

where

$$
-\left(\left(\frac{\partial \mathbf{r}_b}{\partial \Delta \boldsymbol{\ell}}\right)^T \frac{\partial \mathbf{r}_b}{\partial \Delta \boldsymbol{\ell}}\right) = \left(\begin{array}{c|cc} -\underbrace{\mathbf{A}^T\mathbf{A}}_{\mathbf{I}} & \mathbf{A}^T\mathbf{J_i} & -\mathbf{A}^T\mathbf{J_a} \\ \hline \mathbf{J_i}^T\mathbf{A} & -\mathbf{J_i}^T\mathbf{J_i} & \mathbf{J_i}^T\mathbf{J_a} \\ -\mathbf{J_a}^T\mathbf{A} & \mathbf{J_a}^T\mathbf{J_i} & -\mathbf{J_a}^T\mathbf{J_a} \end{array}\right) \tag{60}
$$

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

$$
\begin{aligned}
\begin{pmatrix} -\mathbf{J_i}^T\bar{\mathbf{A}}\mathbf{J_i} & \mathbf{J_i}^T\bar{\mathbf{A}}\mathbf{J_a} \\ \mathbf{J_a}^T\bar{\mathbf{A}}\mathbf{J_i} & -\mathbf{J_a}^T\bar{\mathbf{A}}\mathbf{J_a} \end{pmatrix}\begin{pmatrix} \Delta\mathbf{p} \\ \Delta\mathbf{q} \end{pmatrix} &= \begin{pmatrix} \mathbf{J_i}^T\bar{\mathbf{A}} \\ -\mathbf{J_a}^T\bar{\mathbf{A}} \end{pmatrix}\mathbf{r}_b \\
\begin{pmatrix} \Delta\mathbf{p}^* \\ \Delta\mathbf{q}^* \end{pmatrix} = \begin{pmatrix} -\mathbf{J_i}^T\bar{\mathbf{A}}\mathbf{J_i} & \mathbf{J_i}^T\bar{\mathbf{A}}\mathbf{J_a} \\ \mathbf{J_a}^T\bar{\mathbf{A}}\mathbf{J_i} & -\mathbf{J_a}^T\bar{\mathbf{A}}\mathbf{J_a} \end{pmatrix}^{-1} &\begin{pmatrix} \mathbf{J_i}^T\bar{\mathbf{A}} \\ -\mathbf{J_a}^T\bar{\mathbf{A}} \end{pmatrix}\mathbf{r}_b
\end{aligned}
\tag{61}
$$

Note that the complexity of inverting this new approximation to the Hessian matrix is $O((2n)^3)$.[18] Similar to before, plugging the solutions for $\Delta\mathbf{p}$ and $\Delta\mathbf{q}$ into Eq. 60 we can infer the optimal value for $\Delta\mathbf{c}$ using:

$$
\Delta\mathbf{c}^* = \mathbf{A}^T\,(\mathbf{r}_b - \mathbf{J_i}\Delta\mathbf{p} + \mathbf{J_a}\Delta\mathbf{q}) \tag{62}
$$

The total complexity per iteration of the previous approach is:

$$
O(\underbrace{2nmF}_{\begin{pmatrix} \mathbf{J_i}^T\bar{\mathbf{A}} \\ -\mathbf{J_a}^T\bar{\mathbf{A}} \end{pmatrix}} + \underbrace{(2n)^2F + (2n)^3}_{\begin{pmatrix} -\mathbf{J_i}^T\bar{\mathbf{A}}\mathbf{J_i} & \mathbf{J_i}^T\bar{\mathbf{A}}\mathbf{J_a} \\ \mathbf{J_a}^T\bar{\mathbf{A}}\mathbf{J_i} & -\mathbf{J_a}^T\bar{\mathbf{A}}\mathbf{J_a} \end{pmatrix}^{-1}}) \tag{63}
$$

[16] m and n denote the number of shape and appearance parameters respectively while F denotes the number of pixels on the reference frame.

[17] Applying the Schur complement to the following system of equations:

$$
\begin{aligned}
\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} &= \mathbf{a} \\
\mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{y} &= \mathbf{b}
\end{aligned}
$$

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.

[18] This is an important reduction in complexity because usually $m >> n$ in CGD algorithms.

The Schur complement can be re-applied to Eq. 61 to derive a solution for $\Delta\mathbf{q}$ that only requires inverting a Hessian approximation matrix of size $n \times n$:

$$\begin{aligned}\left(\mathbf{J}_{\mathbf{a}}^T\mathbf{P}\mathbf{J}_{\mathbf{a}}\right)\Delta\mathbf{q} &= \mathbf{J}_{\mathbf{a}}^T\mathbf{P}\mathbf{r}_b\\ \Delta\mathbf{q}^* &= \left(\mathbf{J}_{\mathbf{a}}^T\mathbf{P}\mathbf{J}_{\mathbf{a}}\right)^{-1}\mathbf{J}_{\mathbf{a}}^T\mathbf{P}\mathbf{r}_b\end{aligned} \tag{64}$$

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

$$\mathbf{P} = \bar{\mathbf{A}} - \bar{\mathbf{A}}\mathbf{J}_{\mathbf{i}}\left(\mathbf{J}_{\mathbf{i}}^T\bar{\mathbf{A}}\mathbf{J}_{\mathbf{i}}\right)^{-1}\mathbf{J}_{\mathbf{i}}^T\bar{\mathbf{A}} \tag{65}$$

and the solutions for $\Delta\mathbf{p}$ and $\Delta\mathbf{c}$ can be obtained by plugging the solutions for $\Delta\mathbf{q}$ into Eq. 61 and the solutions for $\Delta\mathbf{q}$ and $\Delta\mathbf{p}$ into Eq. 60 respectively:

$$\begin{aligned}\Delta\mathbf{p}^* &= -\left(\mathbf{J}_{\mathbf{i}}^T\bar{\mathbf{A}}\mathbf{J}_{\mathbf{i}}\right)^{-1}\mathbf{J}_{\mathbf{i}}^T\bar{\mathbf{A}}\left(\mathbf{r}_b - \mathbf{J}_{\mathbf{a}}\Delta\mathbf{q}\right)\\ \Delta\mathbf{c}^* &= \mathbf{A}^T\left(\mathbf{r}_b + \mathbf{J}_{\mathbf{i}}\Delta\mathbf{p} - \mathbf{J}_{\mathbf{a}}\Delta\mathbf{q}\right)\end{aligned} \tag{66}$$

The total complexity per iteration of the previous approach reduces to:

$$O(\underbrace{2nmF}_{\mathbf{J}_{\mathbf{a}}^T\mathbf{P}\,\&\,\mathbf{J}_{\mathbf{i}}^T\bar{\mathbf{A}}} + \underbrace{2n^2F + 2n^3}_{(\mathbf{J}_{\mathbf{a}}^T\mathbf{P}\mathbf{J}_{\mathbf{a}})^{-1}\,\&\,(\mathbf{J}_{\mathbf{i}}^T\bar{\mathbf{A}}\mathbf{J}_{\mathbf{i}})^{-1}}) \tag{67}$$

Note that because of their reduced complexity, the solutions defined by Eqs. 64 and 66 are preferred over the ones defined by Eqs. 61 and 62.

Finally, the solutions using the Project-Out cost function are:

- For *asymmetric* composition:

$$\Delta\mathbf{p}^* = -\left(\mathbf{J}_{\mathbf{t}}^T\bar{\mathbf{A}}\mathbf{J}_{\mathbf{t}}\right)^{-1}\mathbf{J}_{\mathbf{t}}^T\bar{\mathbf{A}}\mathbf{r} \tag{68}$$

  with complexity[19] given by Eq. 58.
- For *bidirectional* composition:

$$\begin{aligned}\Delta\mathbf{q}^* &= \left(\mathbf{J}_{\bar{\mathbf{a}}}^T\mathbf{P}\mathbf{J}_{\bar{\mathbf{a}}}\right)^{-1}\mathbf{J}_{\bar{\mathbf{a}}}^T\mathbf{P}\mathbf{r}\\ \Delta\mathbf{p}^* &= -\left(\mathbf{J}_{\mathbf{i}}^T\bar{\mathbf{A}}\mathbf{J}_{\mathbf{i}}\right)^{-1}\mathbf{J}_{\mathbf{i}}^T\bar{\mathbf{A}}\left(\mathbf{r} - \mathbf{J}_{\mathbf{a}}\Delta\mathbf{q}\right)\end{aligned} \tag{69}$$

  with complexity[20] given by Eq. 67.

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

Alternated

Another way of solving optimization problems with two or more sets of variables is to use alternated optimization (De la Torre 2012). Hence, instead of solving the previous problem simultaneously with respect to all parameters, we can update one set of parameters at a time while keeping the other sets fixed.

More specifically, using *asymmetric* composition we can alternate between updating $\Delta\mathbf{c}$ given the previous $\Delta\mathbf{p}$ and then update $\Delta\mathbf{p}$ given the updated $\Delta\mathbf{c}$ in an alternate manner. Taking advantage of the structure of the problem defined by Eq. 55, we can obtain the following system of equations:

$$\begin{aligned}-\Delta\mathbf{c} + \mathbf{A}^T\mathbf{J}_{\mathbf{t}}\Delta\mathbf{p} &= -\mathbf{A}^T\mathbf{r}_a\\ \mathbf{J}_{\mathbf{t}}^T\mathbf{A}\Delta\mathbf{c} - \mathbf{J}_{\mathbf{t}}^T\mathbf{J}_{\mathbf{t}}\Delta\mathbf{p} &= \mathbf{J}_{\mathbf{t}}^T\mathbf{r}_a\end{aligned} \tag{70}$$

which we can rewrite as:

$$\begin{aligned}\Delta\mathbf{c}^* &= \mathbf{A}^T\left(\mathbf{r}_a + \mathbf{J}_{\mathbf{t}}\Delta\mathbf{p}\right)\\ \Delta\mathbf{p}^* &= -\left(\mathbf{J}_{\mathbf{t}}^T\mathbf{J}_{\mathbf{t}}\right)^{-1}\mathbf{J}_{\mathbf{t}}^T\left(\mathbf{r}_a - \mathbf{A}\Delta\mathbf{c}\right)\end{aligned} \tag{71}$$

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

$$O(\underbrace{n^2F + n^3}_{(\mathbf{J}_{\mathbf{t}}^T\mathbf{J}_{\mathbf{t}})^{-1}}) \tag{72}$$

In the case of *bidirectional* composition we can proceed in two different ways: (a) update $\Delta\mathbf{c}$ given the previous $\Delta\mathbf{p}$ 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 the previous $\Delta\mathbf{p}$ and $\Delta\mathbf{q}$, then $\Delta\mathbf{p}$ given the updated $\Delta\mathbf{c}$ and the previous $\Delta\mathbf{q}$ and, finally, $\Delta\mathbf{q}$ given the updated $\Delta\mathbf{c}$ and $\Delta\mathbf{p}$.

From Eq. 60, we can derive the following system of equations:

$$\begin{aligned}-\Delta\mathbf{c} + \mathbf{A}^T\mathbf{J}_{\mathbf{i}}\Delta\mathbf{p} - \mathbf{A}^T\mathbf{J}_{\mathbf{a}}\Delta\mathbf{q} &= -\mathbf{A}^T\mathbf{r}_b\\ \mathbf{J}_{\mathbf{i}}^T\mathbf{A}\Delta\mathbf{c} - \mathbf{J}_{\mathbf{i}}^T\mathbf{J}_{\mathbf{i}}\Delta\mathbf{p} + \mathbf{J}_{\mathbf{i}}^T\mathbf{J}_{\mathbf{a}}\Delta\mathbf{q} &= \mathbf{J}_{\mathbf{i}}^T\mathbf{r}_b\\ -\mathbf{J}_{\mathbf{a}}^T\mathbf{A}\Delta\mathbf{c} + \mathbf{J}_{\mathbf{a}}^T\mathbf{J}_{\mathbf{i}}\Delta\mathbf{p} - \mathbf{J}_{\mathbf{a}}^T\mathbf{J}_{\mathbf{a}}\Delta\mathbf{q} &= -\mathbf{J}_{\mathbf{a}}^T\mathbf{r}_b\end{aligned} \tag{73}$$

from which we can define the alternated update rules for the first of the previous two options:

$$\begin{aligned}\Delta\mathbf{c}^* &= \mathbf{A}^T\left(\mathbf{r}_b + \mathbf{J}_{\mathbf{i}}\Delta\mathbf{p} - \mathbf{J}_{\mathbf{a}}\Delta\mathbf{q}\right)\\ \begin{pmatrix}\Delta\mathbf{p}^*\\ \Delta\mathbf{q}^*\end{pmatrix} &= \begin{pmatrix}-\mathbf{J}_{\mathbf{i}}^T\mathbf{J}_{\mathbf{i}} & \mathbf{J}_{\mathbf{i}}^T\mathbf{J}_{\mathbf{a}}\\ \mathbf{J}_{\mathbf{a}}^T\mathbf{J}_{\mathbf{i}} & -\mathbf{J}_{\mathbf{a}}^T\mathbf{J}_{\mathbf{a}}\end{pmatrix}^{-1}\\ &\quad\begin{pmatrix}\mathbf{J}_{\mathbf{i}}^T\\ -\mathbf{J}_{\mathbf{a}}^T\end{pmatrix}\left(\mathbf{r}_b - \mathbf{A}\Delta\mathbf{c}\right)\end{aligned} \tag{74}$$

[19] In practice, the solutions for the Project-Out 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 $O(nF)$.

with complexity:

$$O(\underbrace{(2n)^2F+(2n)^3}_{\begin{pmatrix}-\mathbf{J}_{\mathbf{i}}^T\mathbf{J}_{\mathbf{i}} & \mathbf{J}_{\mathbf{i}}^T\mathbf{J}_{\mathbf{a}}\\ \mathbf{J}_{\mathbf{a}}^T\mathbf{J}_{\mathbf{i}} & -\mathbf{J}_{\mathbf{a}}^T\mathbf{J}_{\mathbf{a}}\end{pmatrix}^{-1}}) \tag{75}$$

The rules for the second option are:

$$\begin{aligned}\Delta\mathbf{c}^* &= \mathbf{A}^T\,(\mathbf{r}_b+\mathbf{J}_{\mathbf{i}}\Delta\mathbf{p}-\mathbf{J}_{\mathbf{a}}\Delta\mathbf{q})\\ \Delta\mathbf{p}^* &= -(\mathbf{J}_{\mathbf{i}}^T\mathbf{J}_{\mathbf{i}})^{-1}\mathbf{J}_{\mathbf{i}}^T\,(\mathbf{r}_b-\mathbf{A}\Delta\mathbf{c}-\mathbf{J}_{\mathbf{a}}\Delta\mathbf{q})\\ \Delta\mathbf{q}^* &= (\mathbf{J}_{\mathbf{a}}^T\mathbf{J}_{\mathbf{a}})^{-1}\mathbf{J}_{\mathbf{a}}^T\,(\mathbf{r}_b-\mathbf{A}\Delta\mathbf{c}+\mathbf{J}_{\mathbf{i}}\Delta\mathbf{p})\end{aligned} \tag{76}$$

and their complexity is dominated by:

$$O(\underbrace{2n^2F+2n^3}_{(\mathbf{J}_{\mathbf{i}}^T\mathbf{J}_{\mathbf{i}})^{-1}\,\&\,(\mathbf{J}_{\mathbf{a}}^T\mathbf{J}_{\mathbf{a}})^{-1}}) \tag{77}$$

On the other hand, the alternated update rules using the Project-Out cost function are:

- For *asymmetric* composition: 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:

  $$\begin{aligned}\Delta\mathbf{q}^* &= \left(\mathbf{J}_{\bar{\mathbf{a}}}^T\bar{\mathbf{A}}\mathbf{J}_{\bar{\mathbf{a}}}\right)^{-1}\mathbf{J}_{\bar{\mathbf{a}}}^T\bar{\mathbf{A}}\,(\mathbf{r}+\mathbf{J}_{\mathbf{i}}\Delta\mathbf{p})\\ \Delta\mathbf{p}^* &= -\left(\mathbf{J}_{\mathbf{i}}^T\bar{\mathbf{A}}\mathbf{J}_{\mathbf{i}}\right)^{-1}\mathbf{J}_{\mathbf{i}}^T\bar{\mathbf{A}}\,(\mathbf{r}-\mathbf{J}_{\mathbf{a}}\Delta\mathbf{q})\end{aligned} \tag{78}$$

  with equivalent complexity to the one given by Eq. 58 because, in this case, the term $\left(\mathbf{J}_{\bar{\mathbf{a}}}^T\bar{\mathbf{A}}\mathbf{J}_{\bar{\mathbf{a}}}\right)^{-1}\mathbf{J}_{\bar{\mathbf{a}}}^T\bar{\mathbf{A}}$ can be completely precomputed.

Note that all previous alternated update rules, Eqs. 71, 74, 76 and 107, are similar but slightly different from their simultaneous counterparts, Eqs. 56 and 57, 61 and 62, 64 and 66, and 69.

3.3.2 Newton

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

$$\begin{aligned}\mathcal{D}(\Delta\boldsymbol{\ell}) &\approx \hat{\mathcal{D}}(\Delta\boldsymbol{\ell})\\ &\approx \mathcal{D}+\frac{\partial\mathcal{D}}{\partial\Delta\boldsymbol{\ell}}\Delta\boldsymbol{\ell}+\frac{1}{2}\Delta\boldsymbol{\ell}^T\frac{\partial^2\mathcal{D}}{\partial\Delta\boldsymbol{\ell}^2}\Delta\boldsymbol{\ell}\end{aligned} \tag{79}$$

and solves the approximate problem:

$$\Delta\boldsymbol{\ell}^* = \arg\min_{\Delta\boldsymbol{\ell}}\hat{\mathcal{D}} \tag{80}$$

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

$$\mathcal{D}_a(\Delta\boldsymbol{\ell}) = \frac{1}{2}\mathbf{r}_a^T\mathbf{r}_a \tag{81}$$

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:

$$\begin{aligned}\frac{\partial\mathcal{D}_a}{\partial\Delta\boldsymbol{\ell}} &= \left(\frac{\partial\mathcal{D}_a}{\partial\Delta\mathbf{c}},\frac{\partial\mathcal{D}_a}{\partial\Delta\mathbf{p}}\right)\\ &= \left(-\mathbf{A}^T\mathbf{r}_a,\mathbf{J}_{\mathbf{t}}^T\mathbf{r}_a\right)\end{aligned} \tag{82}$$

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

$$\begin{aligned}\frac{\partial^2\mathcal{D}_a}{\partial\Delta\boldsymbol{\ell}^2} &= \begin{pmatrix}\frac{\partial^2\mathcal{D}_a}{\partial\Delta\mathbf{c}^2} & \frac{\partial^2\mathcal{D}_a}{\partial\Delta\mathbf{c}\partial\Delta\mathbf{p}}\\ \frac{\partial^2\mathcal{D}_a}{\partial\Delta\mathbf{p}\partial\Delta\mathbf{c}} & \frac{\partial^2\mathcal{D}_a}{\partial\Delta\mathbf{p}^2}\end{pmatrix}\\ &= \begin{pmatrix}\frac{\partial^2\mathcal{D}_a}{\partial\Delta\mathbf{c}^2} & \frac{\partial^2\mathcal{D}_a}{\partial\Delta\mathbf{c}\partial\Delta\mathbf{p}}\\ \left(\frac{\partial^2\mathcal{D}_a}{\partial\Delta\mathbf{c}\partial\Delta\mathbf{p}}\right)^T & \frac{\partial^2\mathcal{D}_a}{\partial\Delta\mathbf{p}^2}\end{pmatrix}\end{aligned} \tag{83}$$

Note that the Hessian matrix is, by definition, symmetric. The definition of its individual terms is provided in Appendix 2(a).

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

$$\mathcal{D}_b(\Delta\boldsymbol{\ell}) = \frac{1}{2}\mathbf{r}_b^T\mathbf{r}_b \tag{84}$$

In this case, the Jacobian matrix becomes:

$$\begin{aligned}\frac{\partial\mathcal{D}_b}{\partial\Delta\boldsymbol{\ell}} &= \left(\frac{\partial\mathcal{D}_b}{\partial\Delta\mathbf{c}},\frac{\partial\mathcal{D}_b}{\partial\Delta\mathbf{p}},\frac{\partial\mathcal{D}_b}{\partial\Delta\mathbf{q}}\right)\\ &= \left(-\mathbf{A}^T\mathbf{r}_a,\mathbf{J}_{\mathbf{i}}^T\mathbf{r}_a,-\mathbf{J}_{\mathbf{a}}^T\mathbf{r}_a\right)\end{aligned} \tag{85}$$

and the Hessian matrix takes the following form:

$$\begin{aligned}\frac{\partial^2\mathcal{D}_b}{\partial\Delta\boldsymbol{\ell}^2} &= \begin{pmatrix}\frac{\partial^2\mathcal{D}_b}{\partial\Delta\mathbf{c}^2} & \frac{\partial^2\mathcal{D}_b}{\partial\Delta\mathbf{c}\partial\Delta\mathbf{p}} & \frac{\partial^2\mathcal{D}_b}{\partial\Delta\mathbf{c}\partial\Delta\mathbf{q}}\\ \frac{\partial^2\mathcal{D}_b}{\partial\Delta\mathbf{p}\partial\Delta\mathbf{c}} & \frac{\partial^2\mathcal{D}_b}{\partial\Delta\mathbf{p}^2} & \frac{\partial^2\mathcal{D}_b}{\partial\Delta\mathbf{p}\partial\Delta\mathbf{q}}\\ \frac{\partial^2\mathcal{D}_b}{\partial\Delta\mathbf{q}\partial\Delta\mathbf{c}} & \frac{\partial^2\mathcal{D}_b}{\partial\Delta\mathbf{q}\partial\Delta\mathbf{p}} & \frac{\partial^2\mathcal{D}_b}{\partial\Delta\mathbf{q}^2}\end{pmatrix}\\ &= \begin{pmatrix}\frac{\partial^2\mathcal{D}_b}{\partial\Delta\mathbf{c}^2} & \frac{\partial^2\mathcal{D}_b}{\partial\Delta\mathbf{c}\partial\Delta\mathbf{p}} & \frac{\partial^2\mathcal{D}_b}{\partial\Delta\mathbf{c}\partial\Delta\mathbf{q}}\\ \left(\frac{\partial^2\mathcal{D}_b}{\partial\Delta\mathbf{c}\partial\Delta\mathbf{p}}\right)^T & \frac{\partial^2\mathcal{D}_b}{\partial\Delta\mathbf{p}^2} & \frac{\partial^2\mathcal{D}_b}{\partial\Delta\mathbf{p}\partial\Delta\mathbf{q}}\\ \left(\frac{\partial^2\mathcal{D}_b}{\partial\Delta\mathbf{c}\partial\Delta\mathbf{q}}\right)^T & \left(\frac{\partial^2\mathcal{D}_b}{\partial\Delta\mathbf{p}\partial\Delta\mathbf{q}}\right)^T & \frac{\partial^2\mathcal{D}_b}{\partial\Delta\mathbf{q}^2}\end{pmatrix}\end{aligned} \tag{86}$$

Notice that the previous matrix is again symmetric. The definition of its individual terms is provided in Appendix 2(a).

Simultaneous

Using the Newton method we can solve for all parameters simultaneously by equating the partial derivative of Eq. 80 to 0:

$$\begin{aligned} 0 &= \frac{\partial \hat{\mathcal{D}}}{\partial \Delta \boldsymbol{\ell}} \\ &= \frac{\partial \left(\mathcal{D} + \frac{\partial \mathcal{D}}{\partial \Delta \boldsymbol{\ell}} \Delta \boldsymbol{\ell} + \frac{1}{2} \Delta \boldsymbol{\ell}^T \frac{\partial^2 \mathcal{D}}{\partial^2 \Delta \boldsymbol{\ell}} \Delta \boldsymbol{\ell} \right)}{\partial \Delta \boldsymbol{\ell}} \\ &= \frac{\partial \mathcal{D}}{\partial \Delta \boldsymbol{\ell}} + \frac{\partial^2 \mathcal{D}}{\partial \Delta \boldsymbol{\ell}^2} \Delta \boldsymbol{\ell} \end{aligned} \tag{87}$$

with the solution given by:

$$\Delta \boldsymbol{\ell}^* = - \frac{\partial^2 \mathcal{D}}{\partial \Delta \boldsymbol{\ell}^2}^{-1} \frac{\partial \mathcal{D}}{\partial \Delta \boldsymbol{\ell}} \tag{88}$$

Note that, similar to the Gauss-Newton method, the complexity of inverting the Hessian matrix $\frac{\partial^2 \mathcal{D}}{\partial \Delta \boldsymbol{\ell}^2}$ is $O((n+m)^3)$ for asymmetric composition and $O((2n+m)^3)$ for bidirectional composition. As shown by Kossaifi et al. (2014)[20], we can take advantage of the structure of the Hessian in Eqs. 83 and 86 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:

$$\begin{aligned} \Delta \mathbf{p}^* &= \left(\frac{\partial^2 \mathcal{D}_a}{\partial \Delta \mathbf{p}^2} - \frac{\partial^2 \mathcal{D}_a}{\partial \Delta \mathbf{p} \Delta \mathbf{c}} \frac{\partial^2 \mathcal{D}_a}{\partial \Delta \mathbf{c} \Delta \mathbf{p}} \right)^{-1} \\ &\quad \left(\frac{\partial \mathcal{D}_a}{\partial \Delta \mathbf{p}} - \frac{\partial^2 \mathcal{D}_a}{\partial \Delta \mathbf{p} \Delta \mathbf{c}} \frac{\partial \mathcal{D}_a}{\partial \Delta \mathbf{c}} \right) \\ \Delta \mathbf{c}^* &= \frac{\partial \mathcal{D}_a}{\partial \Delta \mathbf{c}} - \frac{\partial^2 \mathcal{D}_a}{\partial \Delta \mathbf{c} \partial \Delta \mathbf{p}} \Delta \mathbf{p}^* \end{aligned} \tag{89}$$

with complexity:

$$O(\underbrace{nmF}_{\frac{\partial^2 \mathcal{D}_a}{\partial \Delta \mathbf{p} \partial \Delta \mathbf{c}}} + \underbrace{n^2 m}_{\frac{\partial^2 \mathcal{D}_a}{\partial \Delta \mathbf{p} \partial \Delta \mathbf{c}} \frac{\partial^2 \mathcal{D}_a}{\partial \Delta \mathbf{c} \partial \Delta \mathbf{p}}} + \underbrace{2n^2 F}_{\frac{\partial^2 \mathcal{D}_a}{\partial \Delta \mathbf{p}^2}} + \underbrace{n^3}_{\mathbf{H}^{-1}}) \tag{90}$$

where we have defined $\mathbf{H} = \left(\frac{\partial^2 \mathcal{D}_a}{\partial \Delta \mathbf{p}^2} - \frac{\partial^2 \mathcal{D}_a}{\partial \Delta \mathbf{p} \Delta \mathbf{c}} \frac{\partial^2 \mathcal{D}_a}{\partial \Delta \mathbf{c} \Delta \mathbf{p}} \right)^{-1}$ in order to unclutter the notation.

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

$$\begin{aligned} \begin{pmatrix} \Delta \mathbf{p}^* \\ \Delta \mathbf{q}^* \end{pmatrix} &= \begin{pmatrix} \mathbf{V} & \mathbf{W}^T \\ \mathbf{W} & \mathbf{U} \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{v} \\ \mathbf{u} \end{pmatrix} \\ \Delta \mathbf{c}^* &= \frac{\partial \mathcal{D}_b}{\partial \Delta \mathbf{c}} - \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{c} \partial \Delta \mathbf{p}} \Delta \mathbf{p} - \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{c} \partial \Delta \mathbf{q}} \Delta \mathbf{q} \end{aligned} \tag{91}$$

or

$$\begin{aligned} \Delta \mathbf{q}^* &= \left(\mathbf{U} - \mathbf{W} \mathbf{V}^{-1} \mathbf{W}^T \right)^{-1} \left(\mathbf{u} - \mathbf{W} \mathbf{V}^{-1} \mathbf{v} \right) \\ \Delta \mathbf{p}^* &= \mathbf{V}^{-1} \left(\mathbf{v} - \mathbf{W}^T \Delta \mathbf{q} \right) \\ \Delta \mathbf{c}^* &= \frac{\partial \mathcal{D}_b}{\partial \Delta \mathbf{c}} - \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{c} \partial \Delta \mathbf{p}} \Delta \mathbf{p} - \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{c} \partial \Delta \mathbf{q}} \Delta \mathbf{q} \end{aligned} \tag{92}$$

where we have defined the following auxiliary matrices

$$\begin{aligned} \mathbf{V} &= \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{p}^2} - \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{p} \Delta \mathbf{c}} \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{c} \Delta \mathbf{p}} \\ \mathbf{W} &= \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{q} \partial \Delta \mathbf{p}} - \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{q} \Delta \mathbf{c}} \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{c} \Delta \mathbf{p}} \\ \mathbf{U} &= \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{q}^2} - \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{q} \Delta \mathbf{c}} \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{c} \Delta \mathbf{q}} \end{aligned} \tag{93}$$

and vectors

$$\begin{aligned} \mathbf{v} &= \frac{\partial \mathcal{D}_b}{\partial \Delta \mathbf{p}} - \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{p} \Delta \mathbf{c}} \frac{\partial \mathcal{D}_b}{\partial \Delta \mathbf{c}} \\ \mathbf{u} &= \frac{\partial \mathcal{D}_b}{\partial \Delta \mathbf{q}} - \frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{q} \Delta \mathbf{c}} \frac{\partial \mathcal{D}_b}{\partial \Delta \mathbf{c}} \end{aligned} \tag{94}$$

The complexity of the previous solutions is of:

$$\begin{aligned} O(&\underbrace{nmF}_{\mathbf{v}} + \underbrace{2nmF}_{\mathbf{u}} + \underbrace{4n^2F + 2n^2m}_{\mathbf{U}\,\&\,\mathbf{V}} \\ &+ \underbrace{2n^2F + n^2m}_{\mathbf{W}} + \underbrace{(2n)^3}_{\begin{pmatrix} \mathbf{V} & \mathbf{W}^T \\ \mathbf{W} & \mathbf{U} \end{pmatrix}^{-1}}) \end{aligned} \tag{95}$$

and

$$\begin{aligned} O(&\underbrace{nmF}_{\mathbf{v}} + \underbrace{2nmF}_{\mathbf{u}} + \\ &\underbrace{4n^2F + 2n^2m}_{\mathbf{U}\,\&\,\mathbf{V}} + \underbrace{2n^2F + n^2m}_{\mathbf{W}} \\ &+ \underbrace{4n^3}_{\mathbf{V}^{-1}\,\&\,(\mathbf{U} - \mathbf{W}\mathbf{V}^{-1}\mathbf{W}^T)^{-1}}) \end{aligned} \tag{96}$$

respectively.

The solutions using the Project-Out cost function are:

[20] In (2014), Kossaifi et al. applied the Schur complement to the Newton method using *only* inverse composition while we apply it here using the more general *asymmetric* (which includes *forward*, *inverse* and *symmetric*) and *bidirectional* compositions.

- For *asymmetric* composition:

$$\Delta\mathbf{p}^* = -\left(\frac{\partial\mathcal{W}}{\Delta\mathbf{p}}^T \nabla^2\mathbf{t}\frac{\partial\mathcal{W}}{\Delta\mathbf{p}}\bar{\mathbf{A}}\mathbf{r} + \mathbf{J}_{\mathbf{t}}^T\bar{\mathbf{A}}\mathbf{J}_{\mathbf{t}}\right)^{-1} \\ \mathbf{J}_{\mathbf{t}}^T\bar{\mathbf{A}}\mathbf{r} \tag{97}$$

  with complexity[21] given by Eq. 90.
- For *bidirectional* composition:

$$\begin{aligned}\Delta\mathbf{q}^* &= \left(\frac{\partial\mathcal{W}}{\Delta\mathbf{p}}^T \nabla^2\bar{\mathbf{a}}\frac{\partial\mathcal{W}}{\Delta\mathbf{p}}\bar{\mathbf{A}}\mathbf{r} + \mathbf{J}_{\bar{\mathbf{a}}}^T\tilde{\mathbf{P}}\mathbf{J}_{\bar{\mathbf{a}}}\right)^{-1} \\ &\quad \mathbf{J}_{\bar{\mathbf{a}}}^T\tilde{\mathbf{P}}\mathbf{r} \\ \Delta\mathbf{p}^* &= -\mathbf{H}_{\mathbf{i}}^{-1}\mathbf{J}_{\mathbf{i}}^T\bar{\mathbf{A}}\left(\mathbf{r} - \mathbf{J}_{\mathbf{a}}\Delta\mathbf{q}\right)\end{aligned} \tag{98}$$

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

$$\tilde{\mathbf{P}} = \bar{\mathbf{A}} - \bar{\mathbf{A}}\mathbf{J}_{\mathbf{i}}^T\mathbf{H}_{\mathbf{i}}^{-1}\mathbf{J}_{\mathbf{i}}\bar{\mathbf{A}}^T \tag{99}$$

  and where we have defined:

$$\mathbf{H}_{\mathbf{i}} = \left(\frac{\partial\mathcal{W}}{\Delta\mathbf{p}}^T \nabla^2\mathbf{i}[\mathbf{p}]\frac{\partial\mathcal{W}}{\Delta\mathbf{p}}\bar{\mathbf{A}}\mathbf{r} + \mathbf{J}_{\mathbf{i}}^T\bar{\mathbf{A}}\mathbf{J}_{\mathbf{i}}\right) \tag{100}$$

  to unclutter the notation. The complexity per iteration[22] is given by Eq. 96.

Note that, the derivations of the previous solutions, for both types of composition, are analogous to the ones shown in Sect. 3.3.1 for the Gauss-Newton method and, consequently, have been omitted here.

Alternated

Alternated optimization rules can also be derived for the Newton method following the strategy shown in Sect. 3.3.1 for the Gauss-Newton case. Again, we will simply provide update rules and computational complexity for both types of composition and will omit the details of their full derivation.

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

$$\begin{aligned}\Delta\mathbf{c}^* &= \frac{\partial\mathcal{D}_a}{\partial\Delta\mathbf{c}} - \frac{\partial^2\mathcal{D}_a}{\partial\Delta\mathbf{c}\partial\Delta\mathbf{p}}\Delta\mathbf{p} \\ \Delta\mathbf{p}^* &= \frac{\partial^2\mathcal{D}_a}{\partial\Delta\mathbf{p}^2}^{-1}\left(\frac{\partial\mathcal{D}_a}{\partial\Delta\mathbf{p}} - \frac{\partial^2\mathcal{D}_a}{\partial\Delta\mathbf{p}\partial\Delta\mathbf{c}}\Delta\mathbf{c}\right)\end{aligned} \tag{101}$$

with complexity:

$$O(\underbrace{nmF}_{\frac{\partial^2\mathcal{D}_a}{\partial\Delta\mathbf{p}\partial\Delta\mathbf{c}}} + \underbrace{2n^2F + n^3}_{\frac{\partial^2\mathcal{D}_a}{\partial\Delta\mathbf{p}^2}^{-1}}) \tag{102}$$

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

$$\begin{aligned}\Delta\mathbf{c}^* &= \frac{\partial\mathcal{D}_b}{\partial\Delta\mathbf{c}} - \frac{\partial^2\mathcal{D}_b}{\partial\Delta\mathbf{c}\partial\Delta\mathbf{p}}\Delta\mathbf{p} \\ &\quad - \frac{\partial^2\mathcal{D}_b}{\partial\Delta\mathbf{c}\partial\Delta\mathbf{q}}\Delta\mathbf{q} \\ \begin{pmatrix}\Delta\mathbf{p}^* \\ \Delta\mathbf{q}^*\end{pmatrix} &= \begin{pmatrix}\frac{\partial^2\mathcal{D}_b}{\partial\Delta\mathbf{p}^2} & \frac{\partial^2\mathcal{D}_b}{\partial\Delta\mathbf{p}\partial\Delta\mathbf{q}} \\ \frac{\partial^2\mathcal{D}_b}{\partial\Delta\mathbf{q}\partial\Delta\mathbf{p}} & \frac{\partial^2\mathcal{D}_b}{\partial\Delta\mathbf{p}^2}\end{pmatrix}^{-1} \\ &\quad \begin{pmatrix}\frac{\partial\mathcal{D}_b}{\partial\Delta\mathbf{p}} - \frac{\partial^2\mathcal{D}_b}{\partial\Delta\mathbf{p}\partial\Delta\mathbf{c}}\Delta\mathbf{c} \\ \frac{\partial\mathcal{D}_b}{\partial\Delta\mathbf{q}} - \frac{\partial^2\mathcal{D}_b}{\partial\Delta\mathbf{q}\partial\Delta\mathbf{c}}\Delta\mathbf{c}\end{pmatrix}\end{aligned} \tag{103}$$

with complexity:

$$O(\underbrace{nmF}_{\frac{\partial^2\mathcal{D}}{\partial\Delta\mathbf{p}\partial\Delta\mathbf{p}}} + \underbrace{4n^2F}_{\frac{\partial^2\mathcal{D}}{\partial\Delta\mathbf{p}^2}\,\&\,\frac{\partial^2\mathcal{D}}{\partial\Delta\mathbf{q}^2}} + \\ \underbrace{(2n)^3}_{\begin{pmatrix}\frac{\partial^2\mathcal{D}_b}{\partial\Delta\mathbf{p}^2} & \frac{\partial^2\mathcal{D}_b}{\partial\Delta\mathbf{p}\partial\Delta\mathbf{q}} \\ \frac{\partial^2\mathcal{D}_b}{\partial\Delta\mathbf{q}\partial\Delta\mathbf{p}} & \frac{\partial^2\mathcal{D}_b}{\partial\Delta\mathbf{p}^2}\end{pmatrix}^{-1}}) \tag{104}$$

or:

$$\begin{aligned}\Delta\mathbf{c}^* &= \frac{\partial\mathcal{D}_b}{\partial\Delta\mathbf{c}} - \frac{\partial^2\mathcal{D}_b}{\partial\Delta\mathbf{c}\partial\Delta\mathbf{p}}\Delta\mathbf{p} - \frac{\partial^2\mathcal{D}_b}{\partial\Delta\mathbf{c}\partial\Delta\mathbf{q}}\Delta\mathbf{q} \\ \Delta\mathbf{p}^* &= \frac{\partial^2\mathcal{D}_b}{\partial\Delta\mathbf{p}^2}^{-1} \\ &\quad \left(\frac{\partial\mathcal{D}_b}{\partial\Delta\mathbf{p}} - \frac{\partial^2\mathcal{D}_b}{\partial\Delta\mathbf{p}\partial\Delta\mathbf{c}}\Delta\mathbf{c} - \frac{\partial^2\mathcal{D}_b}{\partial\Delta\mathbf{p}\partial\Delta\mathbf{q}}\Delta\mathbf{q}\right) \\ \Delta\mathbf{q}^* &= \frac{\partial^2\mathcal{D}_b}{\partial\Delta\mathbf{q}^2}^{-1} \\ &\quad \left(\frac{\partial\mathcal{D}_b}{\partial\Delta\mathbf{q}} - \frac{\partial^2\mathcal{D}_b}{\partial\Delta\mathbf{q}\partial\Delta\mathbf{c}}\Delta\mathbf{c} - \frac{\partial^2\mathcal{D}_b}{\partial\Delta\mathbf{q}\partial\Delta\mathbf{p}}\Delta\mathbf{p}\right)\end{aligned} \tag{105}$$

with complexity:

$$O(\underbrace{nmF}_{\frac{\partial^2\mathcal{D}}{\partial\Delta\mathbf{p}\partial\Delta\mathbf{p}}} + \underbrace{4n^2F}_{\frac{\partial^2\mathcal{D}}{\partial\Delta\mathbf{p}^2}\,\&\,\frac{\partial^2\mathcal{D}}{\partial\Delta\mathbf{q}^2}} + \underbrace{2n^3}_{\frac{\partial^2\mathcal{D}_b}{\partial\Delta\mathbf{p}^2}^{-1}\,\&\,\frac{\partial^2\mathcal{D}_b}{\partial\Delta\mathbf{q}^2}^{-1}}) \tag{106}$$

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

[21] In practice, the solutions for the project-out 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 we 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 $\mathbf{r}$.

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

$$\begin{aligned}\varDelta\mathbf{q}^* &= \mathbf{H}_{\mathbf{a}}^{-1}\mathbf{J}_{\bar{\mathbf{a}}}^T\bar{\mathbf{A}}\,(\mathbf{r} + \mathbf{J}_{\mathbf{i}}\varDelta\mathbf{p})\\ \varDelta\mathbf{p}^* &= -\mathbf{H}_{\mathbf{i}}^{-1}\mathbf{J}_{\mathbf{i}}^T\bar{\mathbf{A}}\,(\mathbf{r} - \mathbf{J}_{\mathbf{a}}\varDelta\mathbf{q})\end{aligned} \tag{107}$$

where we have defined:

$$\mathbf{H}_{\mathbf{a}} = \left(\frac{\partial\mathcal{W}}{\varDelta\mathbf{p}}^T\nabla^2\bar{\mathbf{a}}\frac{\partial\mathcal{W}}{\varDelta\mathbf{p}}\bar{\mathbf{A}}\mathbf{r} + \mathbf{J}_{\bar{\mathbf{a}}}^T\bar{\mathbf{A}}\mathbf{J}_{\bar{\mathbf{a}}}\right) \tag{108}$$

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

$$O(\underbrace{nmF}_{\mathbf{J}_{\mathbf{i}}^T\bar{\mathbf{A}}} + \underbrace{3n^2F + 2n^3}_{\mathbf{H}_{\mathbf{i}}^{-1}\,\&\,\mathbf{H}_{\mathbf{a}}^{-1}}) \tag{109}$$

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. In Appendix 1, we show that some of the *Gauss-Newton* algorithms derived in Sect. 3.3.1, i.e. the *Asymmetric Gauss-Newton* algorithms, are, in fact, true *Efficient Second order Minimization (ESM)* algorithms that effectively circumvent thie previous increase in computational complexity.

3.3.3 Wiberg

The idea behind the Wiberg method is similar to the one used by the alternated Gauss-Newton method in Sect. 3.3.1, 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 $\mathbf{r}_a(\varDelta\mathbf{c}, \varDelta\mathbf{p})$ and bidirectional $\mathbf{r}_b(\varDelta\mathbf{c}, \varDelta\mathbf{p}, \varDelta\mathbf{q})$ residuals as functions that only depend on $\varDelta\mathbf{p}$ and $\varDelta\mathbf{q}$ respectively.

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

$$\begin{aligned}\bar{\mathbf{r}}_a(\varDelta\mathbf{p}) &= \mathbf{r}_a(\bar{\varDelta\mathbf{c}}, \varDelta\mathbf{p})\\ &= \mathbf{i}[\mathbf{p}\circ\alpha\varDelta\mathbf{p}] - (\bar{\mathbf{a}} + \mathbf{A}(\mathbf{c} + \bar{\varDelta\mathbf{c}}_a))[\beta\varDelta\mathbf{p}]\end{aligned} \tag{110}$$

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

$$\bar{\varDelta\mathbf{c}}_a(\varDelta\mathbf{p}) = \mathbf{A}^T\mathbf{r}_a \tag{111}$$

Given the previous residual, the Wiberg method proceeds to define the following optimization problem with respect to $\varDelta\mathbf{p}$:

$$\varDelta\mathbf{p}^* = \arg\min_{\varDelta\mathbf{p}}\ \bar{\mathbf{r}}_a^T\bar{\mathbf{r}}_a \tag{112}$$

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

$$\varDelta\mathbf{p}^* = \arg\min_{\varDelta\mathbf{p}}\left\|\bar{\mathbf{r}}_a(\varDelta\mathbf{p}) + \frac{\partial\bar{\mathbf{r}}_a}{\partial\varDelta\mathbf{p}}\varDelta\mathbf{p}\right\|^2 \tag{113}$$

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

$$\begin{aligned}\frac{d\bar{\mathbf{r}}_a}{d\varDelta\mathbf{p}} &= \frac{\partial\bar{\mathbf{r}}_a}{\partial\varDelta\mathbf{p}} + \frac{\partial\bar{\mathbf{r}}_a}{\partial\bar{\varDelta\mathbf{c}}_a}\frac{\partial\bar{\varDelta\mathbf{c}}_a}{\partial\varDelta\mathbf{p}}\\ &= \mathbf{J}_{\mathbf{t}} - \mathbf{A}\mathbf{A}^T\mathbf{J}_{\mathbf{t}}\\ &= \bar{\mathbf{A}}\mathbf{J}_{\mathbf{t}}\end{aligned} \tag{114}$$

The solution for $\varDelta\mathbf{p}$ is obtained as usual by equating the derivative of 112 with respect to $\varDelta\mathbf{p}$ to 0:

$$\begin{aligned}\varDelta\mathbf{p}^* &= -\left(\left(\bar{\mathbf{A}}\mathbf{J}_{\mathbf{t}}\right)^T\bar{\mathbf{A}}\mathbf{J}_{\mathbf{t}}\right)^{-1}\left(\bar{\mathbf{A}}\mathbf{J}_{\mathbf{t}}\right)^T\bar{\mathbf{r}}_a\\ &= -\left(\mathbf{J}_{\mathbf{t}}^T\bar{\mathbf{A}}\mathbf{J}_{\mathbf{t}}\right)^{-1}\mathbf{J}_{\mathbf{t}}^T\bar{\mathbf{A}}\bar{\mathbf{r}}_a\end{aligned} \tag{115}$$

where we have used the fact that the matrix $\bar{\mathbf{A}}$ is idempotent[22].

Therefore, the Wiberg method solves explicitly, at each iteration, for $\varDelta\mathbf{p}$ using the previous expression and implicitly for $\varDelta\mathbf{c}$ (through $\bar{\varDelta\mathbf{c}}_a(\varDelta\mathbf{p})$) using Eq. 111. The complexity per iteration of the Wiberg method is the same as the one of the Gauss-Newton method after applying the Schur complement, Eq. 58. In fact, note that the Wiberg solution for $\varDelta\mathbf{p}$ (Eq. 115) is the same as the one of the Gauss-Newton method after applying the Schur complement, Eq. 56; and also note the similarity between the solutions for $\varDelta\mathbf{c}$ of both methods, Eqs. 111 and 57. Finally, note that, due to the close relation between the *Wiberg* and *Gauss-Newton* methods, *Asymmetric Wiberg* algorithms are also ESM algorithms for fitting AAMs.

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

[22] $\bar{\mathbf{A}}$ is idempotent:

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

$$\begin{aligned}\bar{\mathbf{r}}_b(\Delta\mathbf{q}) &= \mathbf{r}_b(\bar{\Delta}\mathbf{c}_b, \bar{\Delta}\mathbf{p}_b, \Delta\mathbf{q}) \\ &= \mathbf{i}[\mathbf{p} \circ \bar{\Delta}\mathbf{p}_b] - (\bar{\mathbf{a}} - \mathbf{A}(\mathbf{c} + \bar{\Delta}\mathbf{c}_b))[\Delta\mathbf{q}]\end{aligned} \tag{116}$$

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:

$$\bar{\Delta}\mathbf{c}_b(\Delta\mathbf{p}, \Delta\mathbf{q}) = \mathbf{A}^T\mathbf{r}_b \tag{117}$$

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:

$$\bar{\Delta}\mathbf{p}_b(\bar{\Delta}\mathbf{c}_b, \Delta\mathbf{q}) = -\left(\mathbf{J}_{\mathbf{i}}^T\bar{\mathbf{A}}\mathbf{J}_{\mathbf{i}}\right)^{-1}\mathbf{J}_{\mathbf{i}}^T\bar{\mathbf{A}}\bar{\mathbf{r}}_b \tag{118}$$

At this point, the Wiberg method proceeds to define the following optimization problem with respect to $\Delta\mathbf{q}$:

$$\Delta\mathbf{q}^* = \underset{\Delta\mathbf{q}}{\arg\min}\; \bar{\mathbf{r}}_b^T\bar{\mathbf{r}}_b \tag{119}$$

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

$$\Delta\mathbf{q}^* = \underset{\Delta\mathbf{q}}{\arg\min}\; \left\|\bar{\mathbf{r}}_b(\Delta\mathbf{q}) + \frac{\partial\bar{\mathbf{r}}_b}{\partial\Delta\mathbf{q}}\Delta\mathbf{q}\right\|^2 \tag{120}$$

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\bar{\Delta}\mathbf{p}_b}\frac{\partial\bar{\Delta}\mathbf{p}_b}{\partial\Delta\mathbf{q}} \\ &\quad + \left(\frac{\partial\bar{\mathbf{r}}_b}{\partial\bar{\Delta}\mathbf{c}_b} + \frac{\partial\bar{\mathbf{r}}_b}{\partial\bar{\Delta}\mathbf{p}_b}\frac{\partial\bar{\Delta}\mathbf{p}_b}{\partial\bar{\Delta}\mathbf{c}}\right)\frac{\partial\bar{\Delta}\mathbf{c}_b}{\partial\Delta\mathbf{q}} \\ &= -\mathbf{J}_{\mathbf{a}} + \mathbf{J}_{\mathbf{i}}\left(\mathbf{J}_i^T\bar{\mathbf{A}}\mathbf{J}_i\right)^{-1}\mathbf{J}_i^T\bar{\mathbf{A}}\mathbf{J}_{\mathbf{a}} \\ &\quad + \left(\mathbf{A} - \mathbf{J}_{\mathbf{i}}\left(\mathbf{J}_i^T\bar{\mathbf{A}}\mathbf{J}_i\right)^{-1}\mathbf{J}_i^T\bar{\mathbf{A}}\mathbf{A}\right)\mathbf{A}^T\mathbf{J}_{\mathbf{a}} \\ &= -\mathbf{J}_{\mathbf{a}} + \mathbf{A}\mathbf{A}^T\mathbf{J}_{\mathbf{a}} \\ &\quad + \mathbf{J}_{\mathbf{i}}\left(\mathbf{J}_i^T\bar{\mathbf{A}}\mathbf{J}_i\right)^{-1}\mathbf{J}_i^T\bar{\mathbf{A}}\mathbf{J}_{\mathbf{a}} \\ &\quad - \mathbf{J}_{\mathbf{i}}\left(\mathbf{J}_i^T\bar{\mathbf{A}}\mathbf{J}_i\right)^{-1}\mathbf{J}_i^T\bar{\mathbf{A}}\mathbf{A}\mathbf{A}^T\mathbf{J}_{\mathbf{a}} \\ &= -\left(\mathbf{I} - \mathbf{A}\mathbf{A}^T\right)\mathbf{J}_{\mathbf{a}} \\ &\quad + \mathbf{J}_{\mathbf{i}}\left(\mathbf{J}_i^T\bar{\mathbf{A}}\mathbf{J}_i\right)^{-1}\mathbf{J}_i^T\bar{\mathbf{A}}\left(\mathbf{I} - \mathbf{A}\mathbf{A}^T\right)\mathbf{J}_{\mathbf{a}} \\ &= -\bar{\mathbf{A}}\mathbf{J}_{\mathbf{a}} + \mathbf{J}_{\mathbf{i}}\left(\mathbf{J}_i^T\bar{\mathbf{A}}\mathbf{J}_i\right)^{-1}\mathbf{J}_i^T\bar{\mathbf{A}}\bar{\mathbf{A}}\mathbf{J}_{\mathbf{a}} \\ &= \left(-\mathbf{I} + \mathbf{J}_{\mathbf{i}}\left(\mathbf{J}_i^T\bar{\mathbf{A}}\mathbf{J}_i\right)^{-1}\mathbf{J}_i^T\bar{\mathbf{A}}\right)\bar{\mathbf{A}}\mathbf{J}_{\mathbf{a}} \\ &= -\mathbf{P}\mathbf{J}_{\mathbf{a}}\end{aligned} \tag{121}$$

And, again, the solution for $\Delta\mathbf{q}$ is obtained as usual by equating the derivative of 120 with respect to $\Delta\mathbf{q}$ to 0:

$$\Delta\mathbf{q}^* = \left((\mathbf{P}\mathbf{J}_{\mathbf{t}})^T\mathbf{P}\mathbf{J}_{\mathbf{t}}\right)^{-1}(\mathbf{P}\mathbf{J}_{\mathbf{t}})^T\bar{\mathbf{r}}_a \tag{122}$$

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 Eqs. 118 and 117 respectively. Again, the complexity per iteration is the same as the one of the Gauss-Newton method after applying the Schur complement, Eq. 67; and the solutions for both methods are almost identical, Eqs. 122, 118 and 117 and Eqs. 61, 62 and 64.

On the other hand, the Wiberg solutions for the project-out cost function are:

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

$$\begin{aligned}\Delta\mathbf{p}^* &= -\left(\mathbf{J}_{\mathbf{i}}^T\bar{\mathbf{A}}\mathbf{J}_{\mathbf{i}}\right)^{-1}\mathbf{J}_{\mathbf{i}}^T\bar{\mathbf{A}}\mathbf{r} \\ \Delta\mathbf{q}^* &= \left(\mathbf{J}_{\bar{\mathbf{a}}}^T\mathbf{P}\mathbf{J}_{\bar{\mathbf{a}}}\right)^{-1}\mathbf{J}_{\bar{\mathbf{a}}}^T\mathbf{P}\mathbf{r}\end{aligned} \tag{123}$$

  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, Eq. 69.

4 Relation to Prior Work

In this section we relate relevant prior work on CGD algorithms for fitting AAMs (Matthews and Baker 2004; Gross et al. 2005; Papandreou and Maragos 2008; Amberg et al. 2009; Martins et al. 2010; Tzimiropoulos and Pantic 2013; Kossaifi et al. 2014) to the unified and complete view introduced in the previous section.

4.1 Project-Out algorithms

In their seminal work (2004), Matthews and Baker proposed the first CGD algorithm for fitting AAMs, 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 appearance 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. $\left(\mathbf{J}_{\bar{\mathbf{a}}}^T\bar{\mathbf{A}}\mathbf{J}_{\bar{\mathbf{a}}}\right)^{-1}\mathbf{J}_{\bar{\mathbf{a}}}\bar{\mathbf{A}}$. The PIC algorithm is very efficient

($O(nF)$) but it has been shown to perform poorly in generic and unconstrained scenarios (Gross et al. 2005; Papandreou and Maragos 2008). In this paper, we refer to this algorithm as the *Project-Out Inverse Gauss-Newton* algorithm.

The forward version of the previous algorithm, i.e. the *Project-Out Forward Gauss-Newton* algorithm, was proposed by Amberg et al. in 2009. 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 (Amberg et al. 2009; Tzimiropoulos and Pantic 2013).[23]

To the best of our knowledge, the rest of Project-Out algorithms derived in Sect. 3, i.e.:

- *Project-Out Forward Newton*
- *Project-Out Inverse Newton*
- *Project-Out Asymmetric Gauss-Newton*
- *Project-Out Asymmetric Newton*
- *Project-Out Bidirectional Gauss-Newton Schur*
- *Project-Out Bidirectional Gauss-Newton Alternated*
- *Project-Out Bidirectional Newton Schur*
- *Project-Out Bidirectional Newton Alternated*
- *Project-Out Bidirectional Wiberg*

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

4.2 SSD algorithms

In Gross et al. (2005) Gross et al. presented the Simultaneous Inverse Compositional (SIC) algorithm and show that it largely outperforms the *Project-Out Inverse Gauss-Newton* 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 appearance parameters and needs to be recomputed at every iteration. Moreover, the inclusion of the Jacobian with respect to the appearance increments $\delta\mathbf{c}$, increases the size of the simultaneous Jacobian to $\frac{\partial \mathbf{r}}{\partial \Delta \ell} = (-\mathbf{A}, -\mathbf{J_a}) \in \mathbb{R}^{F\times(m+n)}$ and, consequently, the computational cost per iteration of the algorithm is $O((m+n)^2F + (m+n)^3)$.

As we shown in Sections 3.3.1, 3.3.1 and 3.3.3 the previous 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) or using the Wiberg method. Papandreou and Maragos (2008) proposed an algorithm that is equivalent to the solution obtained by applying the Schur complement to the problem, as described in Sect. 3.3.1. The same algorithm was reintroduced in Tzimiropoulos and Pantic (2013) 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 $O(nmF + n^2F + n^3)$. In this paper, following our unified view on CGD algorithm, we refer to the previous algorithm as the *SSD Inverse Gauss-Newton Schur* algorithm. The alternated optimization approach was used in Tzimiropoulos et al. (2012) and Antonakos et al. (2014) with complexity $O(n^2F + n^3)$ per iteration. We refer to it as the *SSD Inverse Gauss-Newton Alternated* algorithm.

On the other hand, the forward version of the previous algorithm was first proposed by Martins et al. in (2010).[24] 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 if the algorithm is the same as in the naive inverse approach of Gross et al. In this paper, we refer to this algorithm as the *SSD Forward Gauss-Newton* algorithm. It is important to notice that Tzimiropoulos and Pantic (2013) 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 appearance parameters. Their algorithm is equivalent to the previous *Project-Out Forward Gauss-Newton*.

Finally, Kossaifi et al. derived the *SSD Inverse Newton Schur* algorithm in Kossaifi et al. (2014). 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 Sect. 3, 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*

[23] Notice that, in Amberg et al. (2009), 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.

[24] Note 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.

- *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*
- *SSD Bidirectional Wiberg*

have never been published before and are also a key contribution of the presented work.

Note that the iterative solutions of all CGD algorithms studied in this paper are given in Appendix 3.

5 Experiments

In this section, we analyze the performance of the CGD algorithms derived in Sect. 3 on the specific problems 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 Labelled Faces Parts in-the-Wild (LFPW) (Belhumeur et al. 2011) database. 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. In the third experiment, we study the effect that varying the value of the parameters α and β has on the performance of *Asymmetric* algorithms. The fourth experiment explores the effect of optimizing the cost functions using reduced subsets of the total number of pixels (Fig. 3) and quantifies the impact that this has on the accuracy and computational efficiency of CGD algorithms. Finally, in the fifth experiment, we report the performance of the most accurate CGD algorithms on the test set of the Helen (Le et al. 2012) database and on the entire Annotated Faces in-the-Wild (AFW) (Zhu and Ramanan 2012) database.

Throughout this section, we abbreviate CGD algorithms 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 the previous 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 group[25] (Sagonas et al. 2013a, b) and fitting accuracy is reported using the point-to-point error measure normalized by the *face size*[26] proposed in Zhu and Ramanan (2012) over the 49 interior points of the iBug annotation scheme.

In all face alignment experiments, we use a single AAM, trained using the ~800 and ~2000 training images of the LFPW and Helen databases. Similar to Tzimiropoulos and Pantic (2014), we use a modified version of the *Dense* Scale Invariant Feature Transform (DSIFT) (Lowe 1999; Dalal and Triggs 2005) to define the appearance representation of the previous AAM. In particular, we describe each pixel with a reduced SIFT descriptor of length 8 using the public implementation provided by the authors of Vedaldi and Fulkerson (2010). All algorithms are implemented in a coarse to fine manner using a Gaussian pyramid with 2 levels (face images are normalized to a *face size*[27] of roughly 150 pixels at the top level). In all experiments, we optimize over 7 shape parameters (4 similarity transform and 3 non-rigid shape parameters) at the first pyramid level and over 16 shape parameters (4 similarity transform and 12 non-rigid shape parameters) at the second one. The dimensionality of the appearance models is kept to represent 75 % of the total variance in both levels. This results in 225 and 280 appearance parameters at the first and second pyramid levels respectively. The previous choices were determined by testing on a small hold out set of the training data.

In all experiments, algorithms are initialized by perturbing the similarity transform that perfectly aligns the model's mean shape (a frontal pose and neutral expression looking shape) with the ground truth shape of each image. These transforms are perturbed by adding uniformly distributed random noise to their scale, rotation and translation parameters. Exemplar initializations obtained by this procedure for different amounts of noise are shown in Fig. 4. Notice that we found that initializing using 5 % uniform noise is (statistically) equivalent to initializing with the popular OpenCV (Bradski 2000) implementation of the well-known Viola and Jones face detector (Viola and Jones 2001) on the test images of the LFPW database.

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*

[25] http://ibug.doc.ic.ac.uk/resources/300-W/.

[26] The face size is computed as the mean of the height and width of the bounding box containing a face.

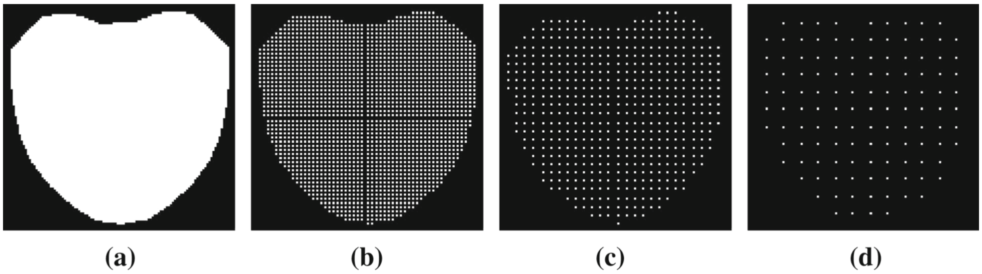


Fig. 3 Subset of pixels on the reference frame used to optimize the SSD and Project-Out cost functions for different sampling rates. **a** 100 %, **b** 50 %, **c** 25 %, **d** 12 %

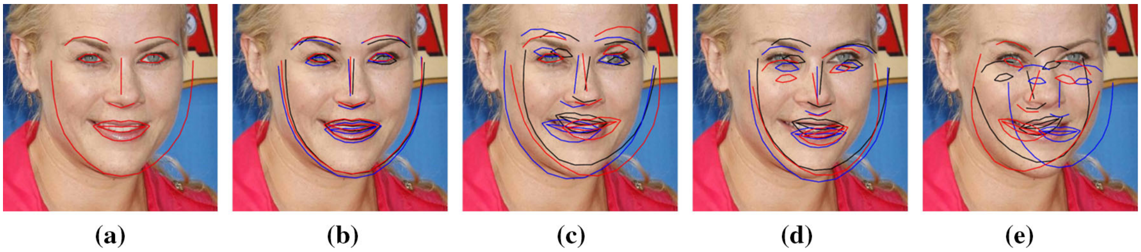


Fig. 4 Exemplar initializations obtained by varying the percentage of uniform noise added to the similarity parameters. Note that, increasing the percentage of noise produces more challenging initialization **a** 0 %, **b** 2.5 %, **c** 5 %, **d** 7.5 %. **e** 10 %

algorithms are obtained using the Bayesian project-out cost function defined by Eq. 22; and (v) results for *Asymmetric* algorithms are reported for the special case of symmetric composition i.e. $\alpha = \beta = 0.5$ in Eq. 34.

Finally, in order to encourage open research and facilitate future comparisons with the results presented in this section, we make the implementation of all algorithms publicly available as part of the Menpo Project[1] (Alabort-i-Medina et al. 2014).

5.1 Comparison on LFPW

In this experiment, we report 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, we group algorithms by cost function (i.e. *SSD* or *Project-Out*), and optimization method (i.e. *Gauss-Newton*, *Newton* or *Wiberg*).

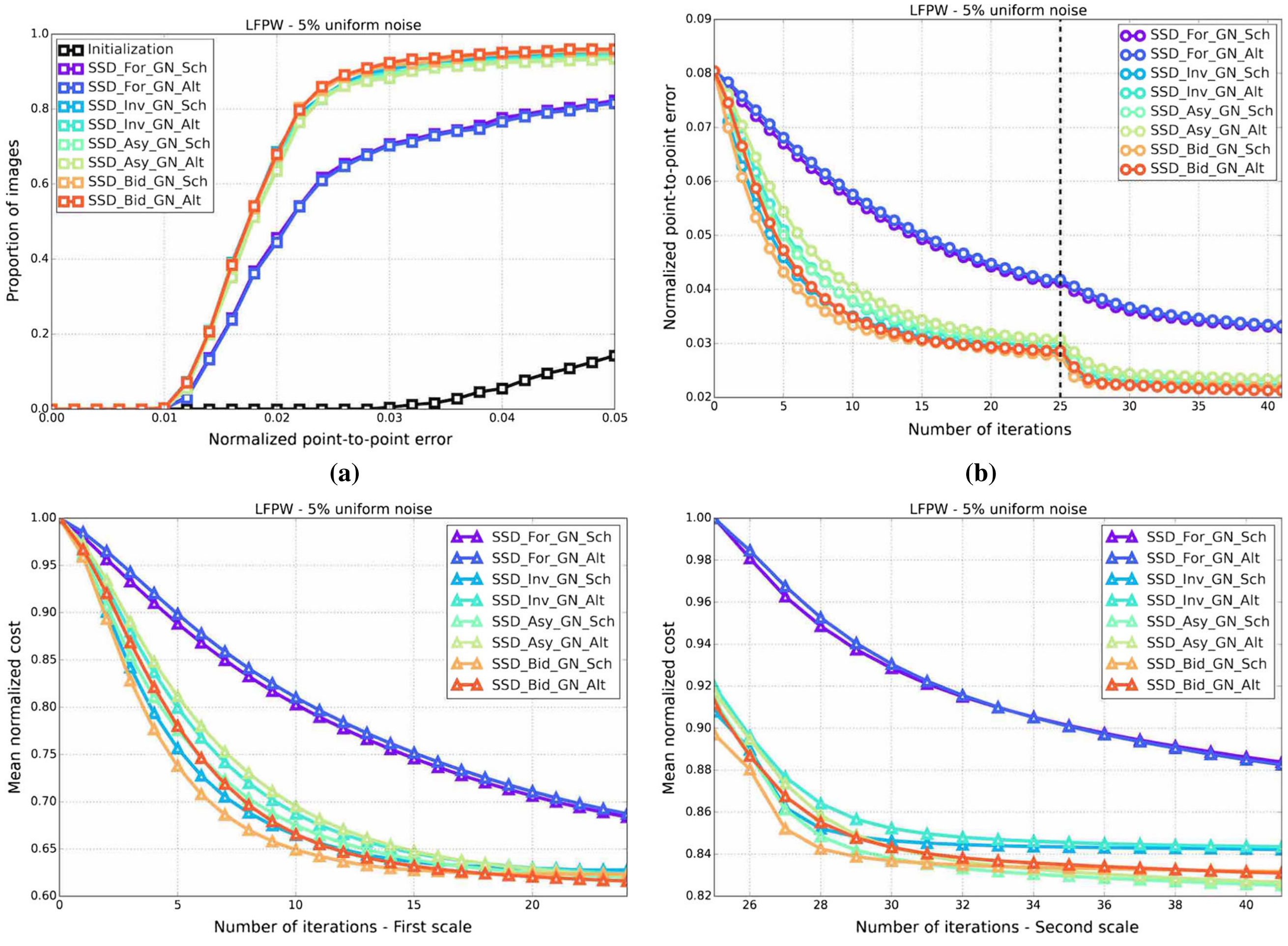
Results for this experiment are reported in Figs. 5, 6, 7, 8, 9 and 10. These figures have all the same structure and are composed of four figures and a table. Figs. 5a, 6a, 7a, 8a, 9a and 10a report the Cumulative Error Distribution (CED), i.e the proportion of images versus normalized point-to-point error for each of the algorithms' groups. Figures 5e, 6e, 7e, 8e, 9e, and 10e summarize and complete the information on the previous CEDs 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. The aim of the previous figures and tables is to help us compare the final fitting accuracy obtained by each algorithm. On the other hand, Figs. 5b, 6b, 7b, 8b, 9b and 10b report the mean normalized point-to-point error at each iteration while Figs. 5c, 5d, 6c, 6d, 7c, 7d, 8c, 8d, 9c, 9d and 10c, 10d report the mean normalized cost at each iteration.[27] The aim of these figures is to help us compare the convergence properties of every algorithm.

5.1.1 SSD Gauss-Newton algorithms

Results for *SSD Gauss-Newton* algorithms are reported in Fig. 5. We can observe that *Inverse*, *Asymmetric* and *Bidirectional* algorithms obtain a similar performance and significantly outperform *Forward* algorithms in terms of fitting accuracy, Fig. 5a, e. 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 5b–d there seems to be a clear (and obviously 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*, *Asym-*

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



(a)

(b)

(c)

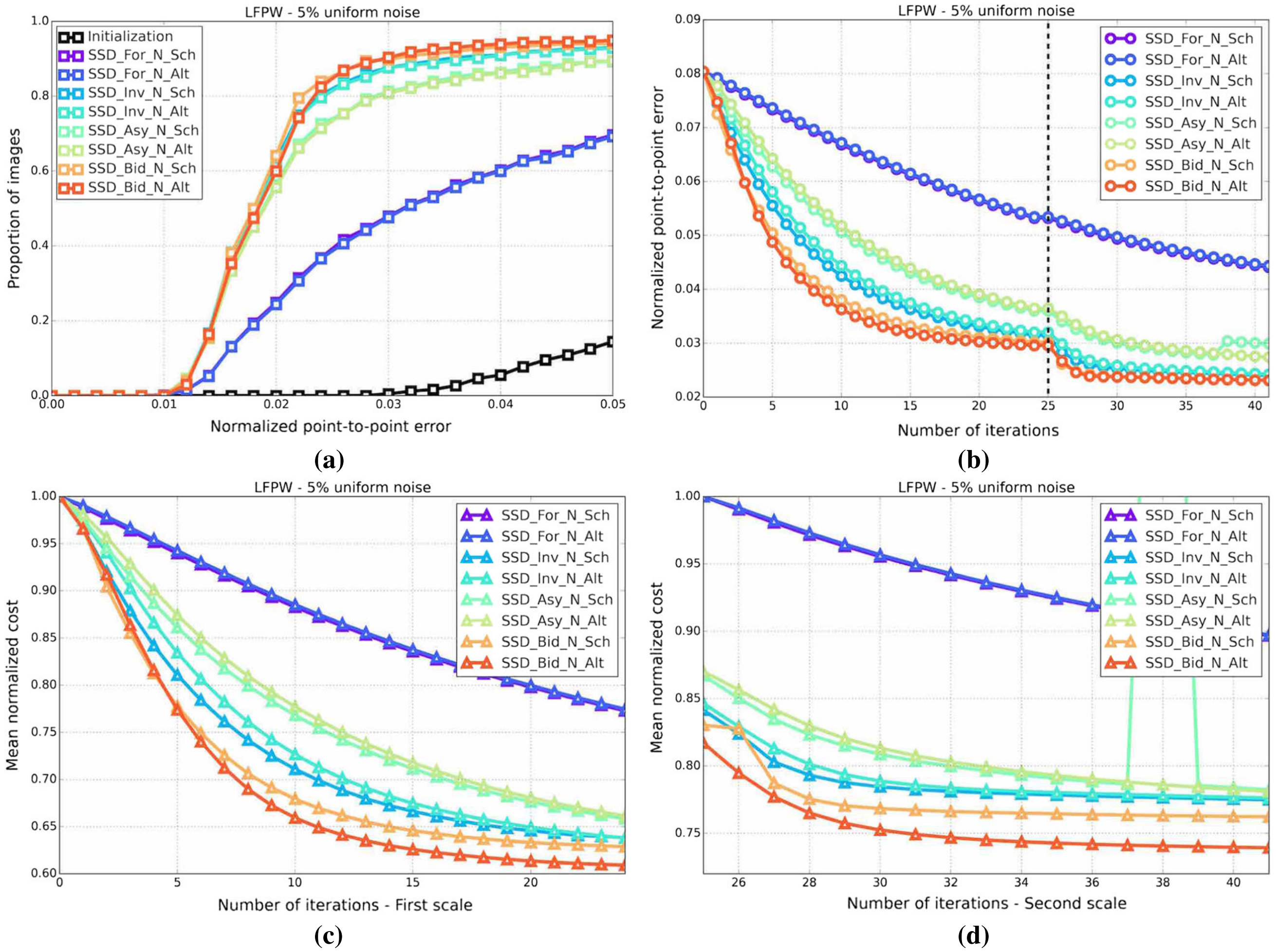
(d)

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)

Fig. 5 Results showing the fitting accuracy and convergence properties of the SSD Gauss-Newton algorithms on the LFPW test dataset initialized with 5 % uniform noise. **a** CED on the LFPW test dataset for all SSD Gauss-Newton algorithms initialized with 5 % uniform noise. **b** Mean normalized point-to-point error versus number of iterations on the LFPW test dataset for all SSD Gauss-Newton algorithms initialized with 5 % uniform noise. **c** Mean normalized cost versus number of first scale iterations on the LFPW test dataset for all SSD Gauss-Newton algorithms initialized with 5 % uniform noise. **d** Mean normalized cost versus number of second scale iterations on the LFPW test dataset for all SSD Gauss-Newton algorithms initialized with 5% uniform noise. **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 initialized with 5 % uniform noise

metric 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.



(a) **(b)** **(c)** **(d)**

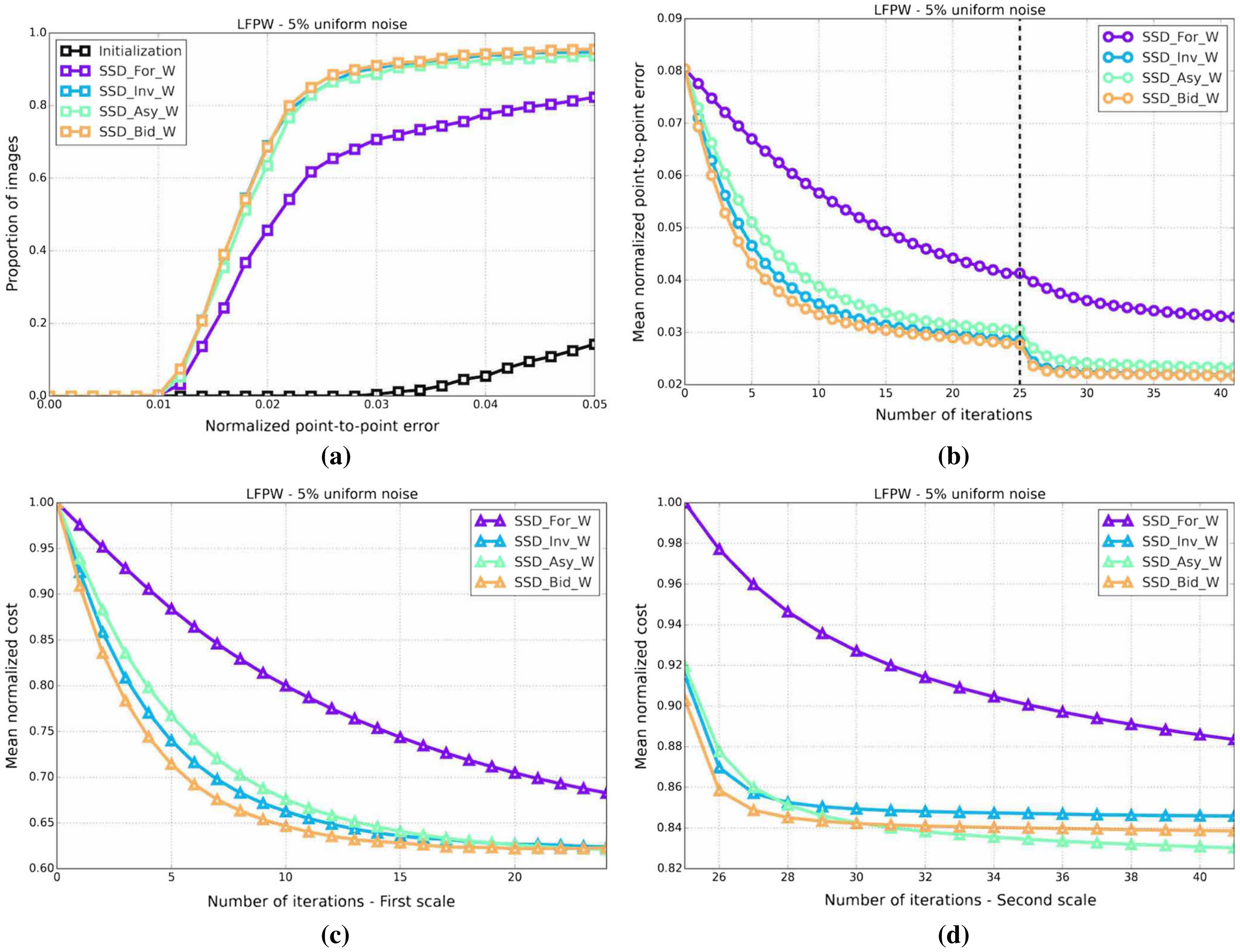
Algorithm	< 0.02	< 0.03	< 0.04	Mean	Std	Median	Run-Time
Initialization	0.000	0.004	0.055	0.080	0.028	0.078	-
SSD_For_N_Sch	0.249	0.479	0.603	0.044	0.033	0.031	∼ 2.3 s
SSD_For_N_Alt	0.244	0.476	0.600	0.044	0.033	0.032	∼ 2.1 s
SSD_Inv_N_Sch	0.626	0.876	0.909	0.024	0.022	**0.018**	∼ 2.3 s
SSD_Inv_N_Alt	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)

Fig. 6 Results showing the fitting accuracy and convergence properties of the SSD Newton algorithms on the LFPW test dataset initialized with 5 % uniform noise. **a** Cumulative error distribution on the LFPW test dataset for all SSD Newton algorithms initialized with 5 % uniform noise. **b** Mean normalized point-to-point error versus number of iterations on the LFPW test dataset for all SSD Newton algorithms initialized with 5 % uniform noise. **c** Mean normalized cost versus number of first scale iterations on the LFPW test dataset for all SSD Newton algorithms initialized with 5 % uniform noise. **d** Mean normalized cost versus number of second scale iterations on the LFPW test dataset for all SSD Newton algorithms initialized with 5 % uniform noise. **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 initialized with 5 % uniform noise

5.1.2 SSD Newton algorithms

Results for *SSD Newton* algorithms are reported on Fig. 6. In this case, we can observe that the fitting performance of all algorithms decreases with respect to their *Gauss-Newton* counterparts Fig. 6a, e. This is most noticeable in the case of *Forward* algorithms for which there is ∼ 20% drop in the proportion of images fitted below 0.02, 0.03



(a)

(b)

(c)

(d)

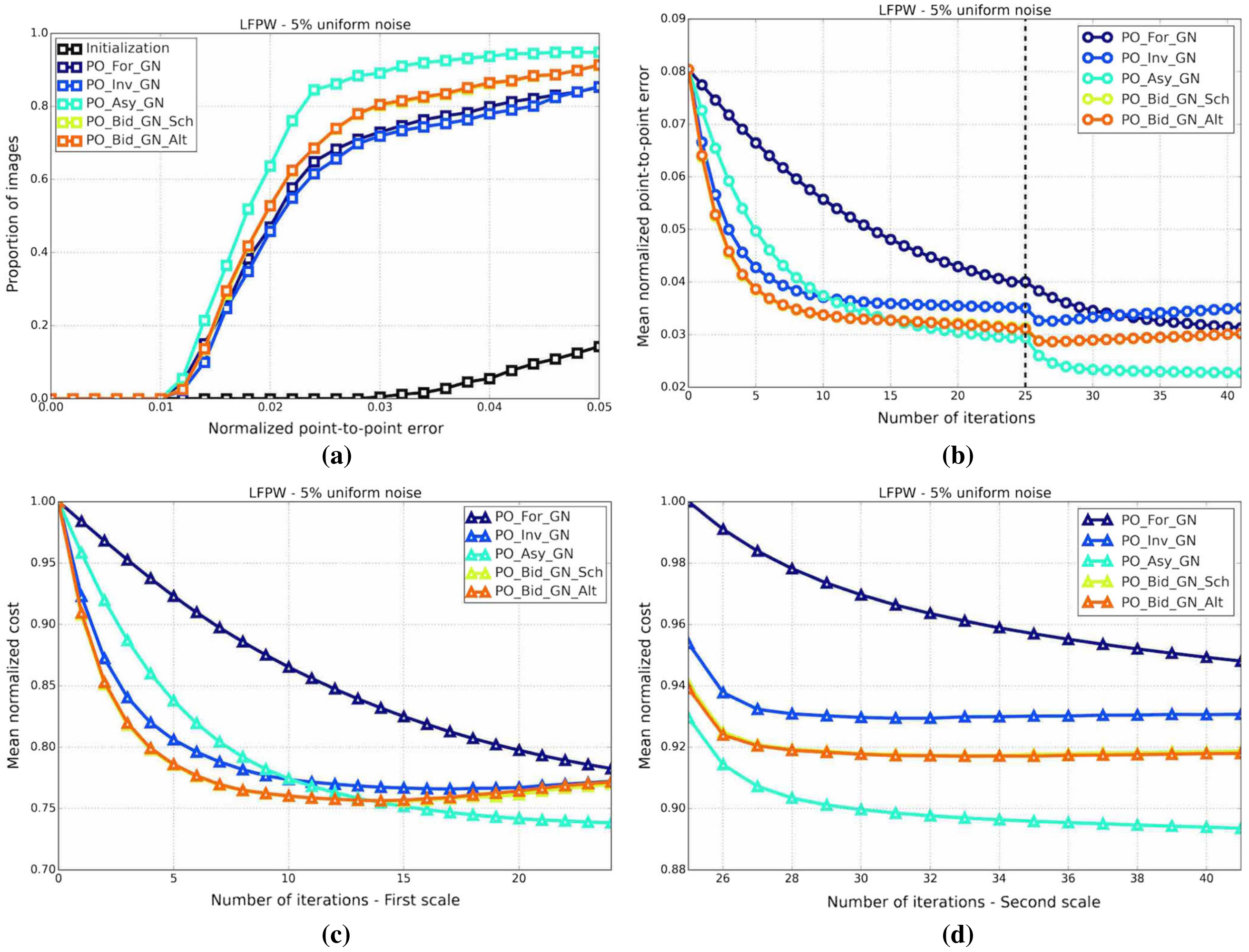
Algorithm	< 0.02	< 0.03	< 0.04	Mean	Std	Median	Run-Time
Initialization	0.000	0.004	0.055	0.080	0.028	0.078	-
SSD_For_W	0.457	0.707	0.777	0.33	0.030	0.021	∼ 1.6 s
SSD_Inv_W	**0.689**	0.903	0.939	**0.22**	**0.019**	**0.017**	∼ 1.6 s
SSD_Asy_W	0.635	0.887	0.926	0.23	0.021	0.018	∼ 1.7 s
SSD_Bid_W	0.686	**0.911**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

(e)

Fig. 7 Results showing the fitting accuracy and convergence properties of the SSD Wiberg algorithms on the LFPW test dataset. **a** CED on the LFPW test dataset for all SSD Wiberg algorithms initialized with 5 % uniform noise. **b** Mean normalized point-to-point error versus number of iterations on the LFPW test dataset for all SSD Wiberg algorithms initialized with 5 % uniform noise. **c** Mean normalized cost versus number of first scale iterations on the LFPW test dataset for all SSD Wiberg algorithms initialized with 5 % uniform noise. **d** Mean normalized cost versus number of second scale iterations on the LFPW test dataset for all SSD Wiberg algorithms initialized with 5 % uniform noise. **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 initialized with 5 % uniform noise

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 Fig. 6b–d. *Bidirectional* algorithms are the fastest to converge followed by



(a) **(b)**

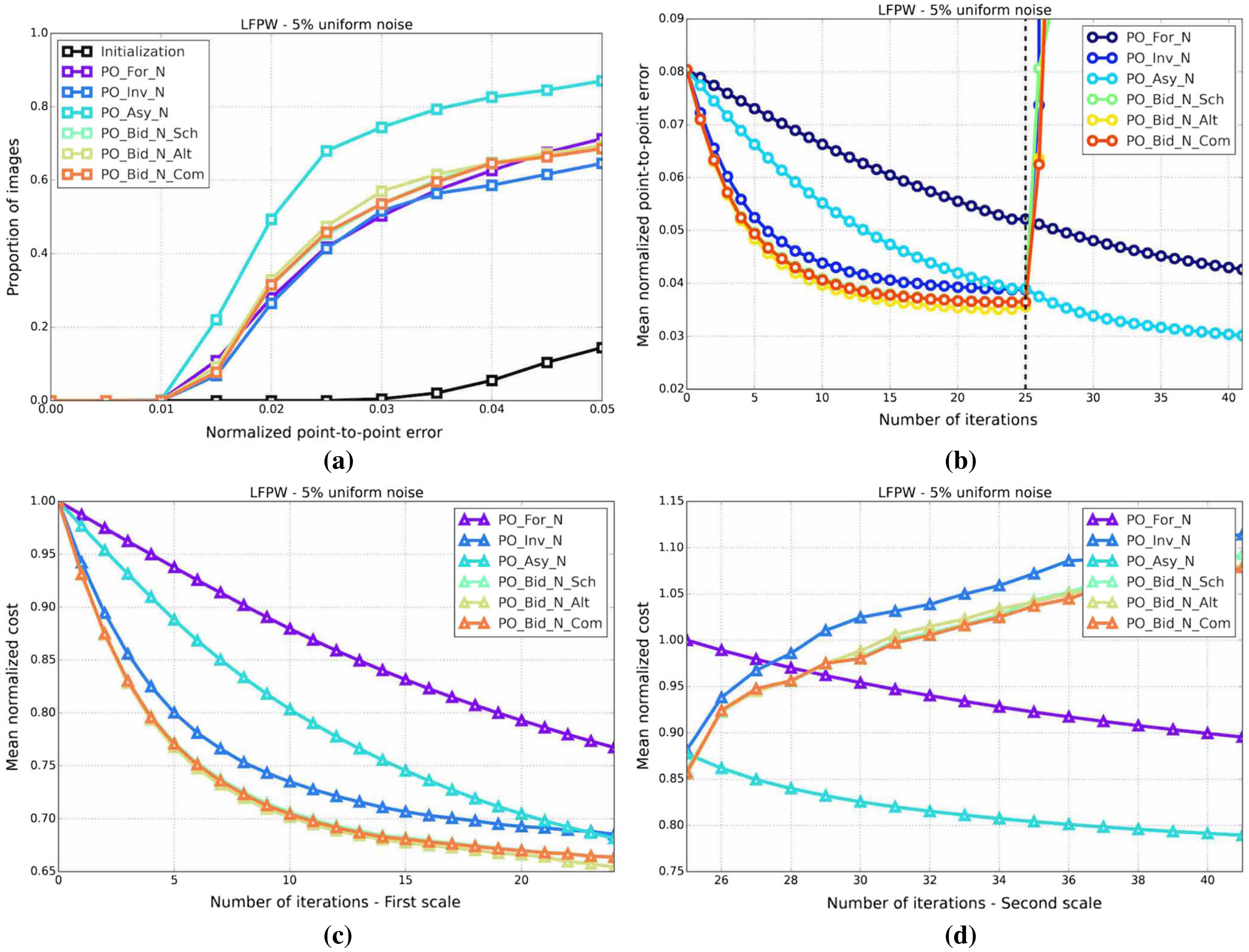
(c) **(d)**

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)

Fig. 8 Results showing the fitting accuracy and convergence properties of the Project-Out Gauss-Newton algorithms on the LFPW test dataset. **a** CED graph on the LFPW test dataset for all Project-Out Gauss-Newton algorithms initialized with 5 % uniform noise. **b** Mean normalized point-to-point error versus number of iterations on the LFPW test dataset for all Project-Out Gauss-Newton algorithms initialized with 5 % uniform noise. **c** Mean normalized cost versus number of first scale iterations on the LFPW test dataset for all Project-Out Gauss-Newton algorithms initialized with 5 % uniform noise. **d** Mean normalized cost versus number of second scale iterations on the LFPW test dataset for all Project-Out Gauss-Newton algorithms initialized with 5 % uniform noise. **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 Project-Out Gauss-Newton algorithms initialized with 5 % uniform noise

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.



(a) **(b)**

(c) **(d)**

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)

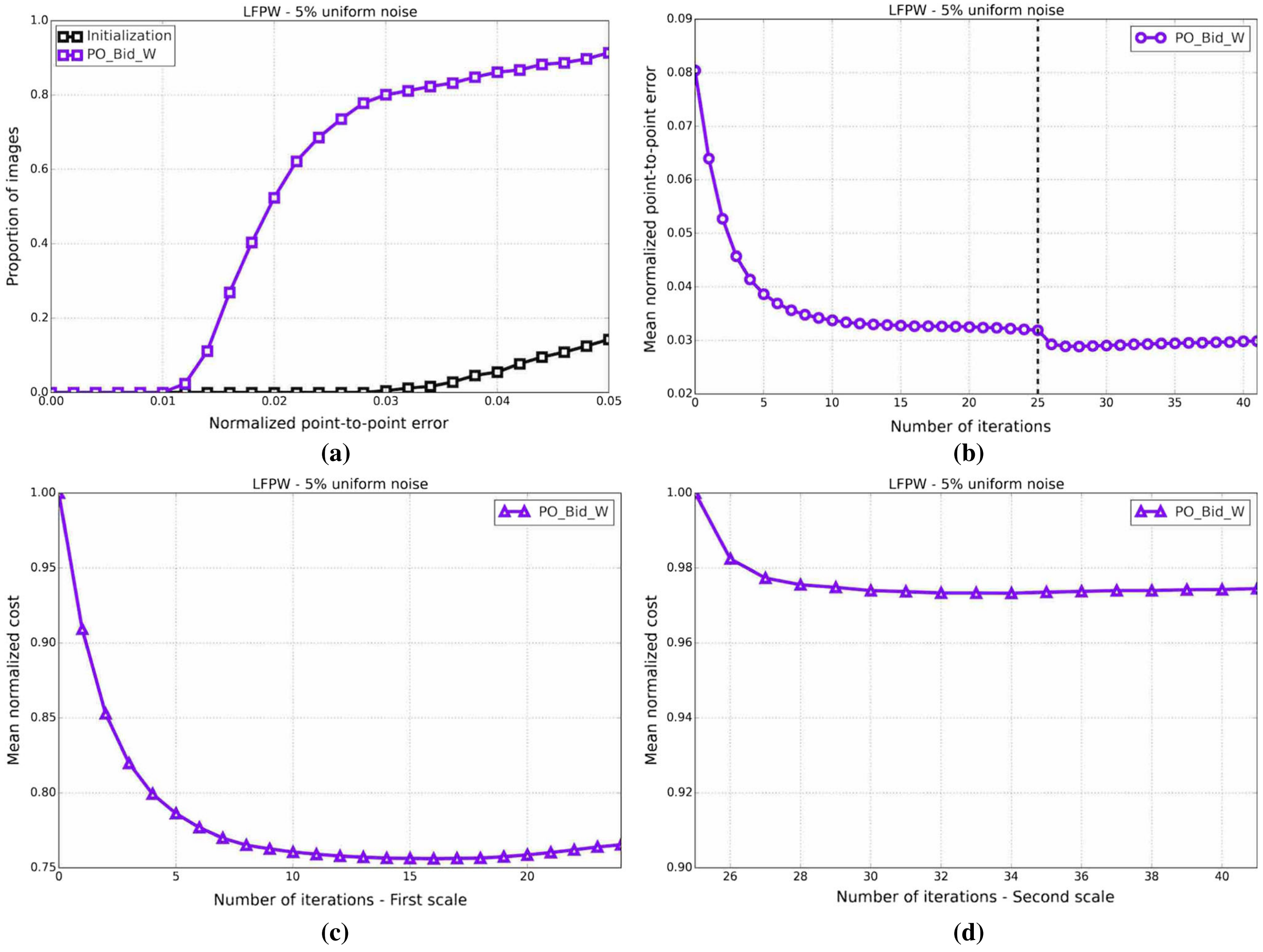
Fig. 9 Results showing the fitting accuracy and convergence properties of the Project-Out Newton algorithms on the LFPW test dataset. **a** CED graph on the LFPW test dataset for all Project-Out Newton algorithms initialized with 5 % uniform noise. **b** Mean normalized point-to-point error versus number of iterations on the LFPW test dataset for all Project-Out Newton algorithms initialized with 5 % uniform noise. **c** Mean normalized cost versus number of first scale iterations on the LFPW test dataset for all Project-Out Newton algorithms initialized with 5 % uniform noise. **d** Mean normalized cost versus number of second scale iterations on the LFPW test dataset for all Project-Out Newton algorithms initialized with 5 % uniform noise. **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 Project-Out Newton algorithms initialized with 5 % uniform noise

5.1.3 SSD Wiberg algorithms

Results for *SSD Wiberg* algorithms are reported on Fig. 7. Figure 7a–e show that these results are (as one would expect) virtually equivalent to those obtained by their *Gauss-Newton* counterparts.

5.1.4 Project-Out Gauss-Newton algorithms

Results for *Project-Out Gauss-Newton* algorithms are reported on Fig. 8. We can observe that, there is significant drop in terms of fitting accuracy for *Inverse* and *Bidirectional* algorithms with respect to their *SSD* versions, Fig. 8a, e. As



(a)

(b)

(c)

(d)

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)

Fig. 10 Results showing the fitting accuracy and convergence properties of the Project-Out Wiberg algorithms on the LFPW test dataset. **a** Cumulative Error Distribution graph on the LFPW test dataset for all Project-Out Wiberg algorithms initialized with 5 % uniform noise. **b** Mean normalized point-to-point error versus number of iterations graph on the LFPW test dataset for all Project-Out Wiberg algorithms initialized with 5 % uniform noise. **c** Mean normalized cost versus number of first scale iterations graph on the LFPW test dataset for all Project-Out Wiberg algorithms initialized with 5 % uniform noise. **d** Mean normalized cost versus number of second scale iterations graph on the LFPW test dataset for all Project-Out Wiberg algorithms initialized with 5 % uniform noise. **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 Project-Out Wiberg algorithms initialized with 5 % uniform noise

expected, the *Forward* algorithm achieves virtually the same results as its *SSD* counterpart. The *Asymmetric* algorithm obtains similar accuracy to that of the best performing *SSD* algorithms.

Looking at Figures 8b–d we 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 convergent algorithm.

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.

5.1.5 *Project-Out Newton algorithms*

Results for *Project-Out Newton* algorithms are reported on Fig. 9. It can be clearly seen that *Project-Out 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 *Project-Out Gauss-Newton* algorithms, Fig. 9a, e. In fact, by looking at Fig. 9b–d 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.

5.1.6 *Project-Out Wiberg algorithms*

Results for the *Project-Out Bidirectional Wiberg* algorithm are reported on Fig. 10. As expected, the results are virtually identical to those of the obtained by *Project-Out Bidirectional Gauss-Newton* algorithms.

5.2 Weighted Bayesian project-out

In this experiment, we quantify the importance of each of the two terms in our Bayesian project-out cost function, Eq. 22. To this end, we introduce the parameters, $\rho \in [0, 1]$ and $\gamma = 1 - \rho$, to weight up the relative contribution of both terms:

$$\rho ||\mathbf{i}[\mathbf{p}] - \bar{\mathbf{a}}||^2_{\mathbf{A}\mathbf{D}^{-1}\mathbf{A}^T} + \frac{\gamma}{\sigma^2} ||\mathbf{i}[\mathbf{p}] - \bar{\mathbf{a}}||^2_{\hat{\mathbf{A}}} \tag{124}$$

Setting $\rho = 0$, $\gamma = 1$ reduces the previous cost function to the original project-out loss proposed in Matthews and Baker (2004); completely disregarding the contribution of the prior distribution over the appearance parameters i.e the Mahalanobis distance *within* the appearance 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 appearance subspace. Finally setting $\rho = \gamma = 0.5$ leads to the standard Bayesian project-out cost function proposed in Sect. 3.1.2.

In order to assess the impact that each term has on the fitting accuracy obtained by the previous *Project-Out* algorithm we repeat the experimental set up of the first experiment and test all *Project-Out Gauss-Newton* algorithms for different values of the parameters $\rho = 1 - \gamma$. Notice that, in this case, we only report the performance of *Gauss-Newton* algorithms 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 Fig. 11. We can see that, regardless of the type of composition, a weighted combination of the two previous 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 appearance 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.

5.3 Optimal asymmetric composition

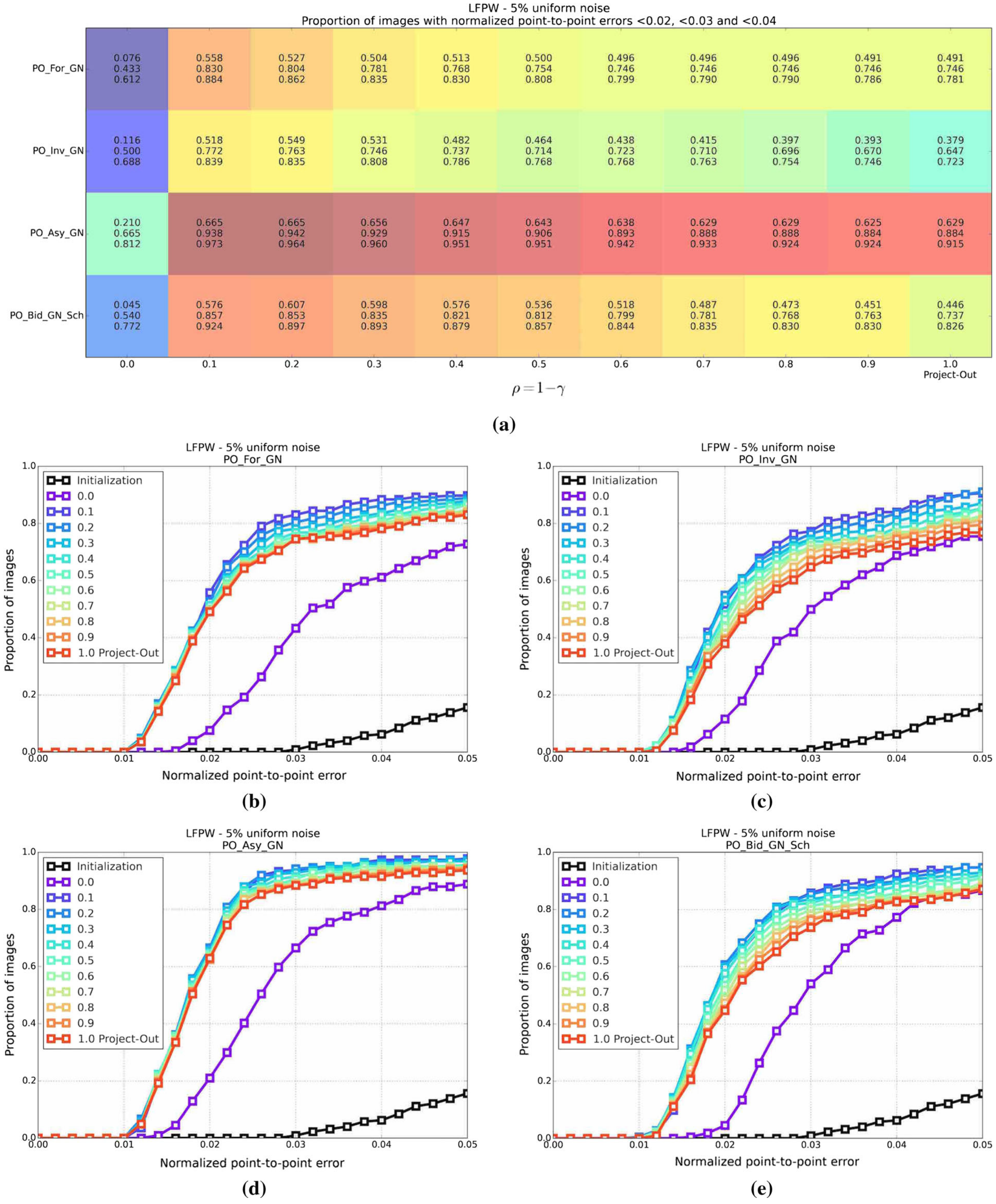
This experiment quantifies the effect that varying the value of the parameters $\alpha \in [0, 1]$ and $\beta = 1 - \alpha$ in Eq. 34 has in the fitting accuracy obtained by the *Asymmetric* algorithms. Note that for $\alpha = 1$, $\beta = 0$ and $\alpha = 0$, $\beta = 1$ these algorithms reduce to their *Forward* and *Inverse* versions respectively. Recall that, in previous experiments, we used the *Symmetric* case $\alpha = \beta = 0.5$ to generate the results reported for *Asymmetric* algorithms. Again, we only report performance for *Gauss-Newton* algorithms.

We again repeat the experimental set up described in the first experiments and report the fitting accuracy obtained by the *Project Out* and *SSD Asymmetric Gauss-Newton* algorithms for different values of the parameters $\alpha = 1 - \beta$. Results are shown in Fig. 12. For the *BPO Asymmetric* algorithm, the best results are obtain by setting $\alpha = 0.4$, $\beta = 0.6$, Figs. 12a (top) and 12b. These results slightly outperform those obtain by the default *Symmetric* algorithm and this particular configuration of the *BPO 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 $\alpha = 0.2$, $\beta = 0.8$, Figs. 12a (bottom) and 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.

5.4 Sampling and Number of Iterations

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

The first one consists of optimizing the SSD and Project-Out cost functions using only a subset of all pixels in the reference frame. In AAMs the total number of pixels on the reference frame, F, is typically several orders of magnitude bigger than the number of shape, n, and appearance,



(a)

(b)

(c)

(d)

(e)

Fig. 11 Results quantifying the effect of varying the value of the parameters $\rho = 1 - \gamma$ in Project-Out Gauss-Newton algorithms. **a** Proportion of images with normalized point-to-point errors smaller than 0.02, 0.03 and 0.04 for the Project-Out and SSD Asymmetric Gauss-Newton algorithms for different values of $\rho = 1 - \gamma$ and initialized with 5 % noise. Colors encode overall fitting accuracy, from highest to lowest: *red*, *orange*, *yellow*, *green*, *blue* and *purple*. **b** CED on the LFPW test dataset for Project-Out Forward Gauss-Newton algorithms for different values of $\rho = 1 - \gamma$ and initialized with 5 % noise. **c** CED on the LFPW test dataset for Project-Out Inverse Gauss-Newton algorithms for different values of $\rho = 1 - \gamma$ and initialized with 5 % noise. **d** CED on the LFPW test dataset for Project-Out Asymmetric Gauss-Newton algorithms for different values of $\rho = 1 - \gamma$ and initialized with 5 % noise. **e** CED on the LFPW test dataset for Project-Out Bidirectional Gauss-Newton algorithms for different values of $\rho = 1 - \gamma$ and initialized with 5 % noise (Color figure online)

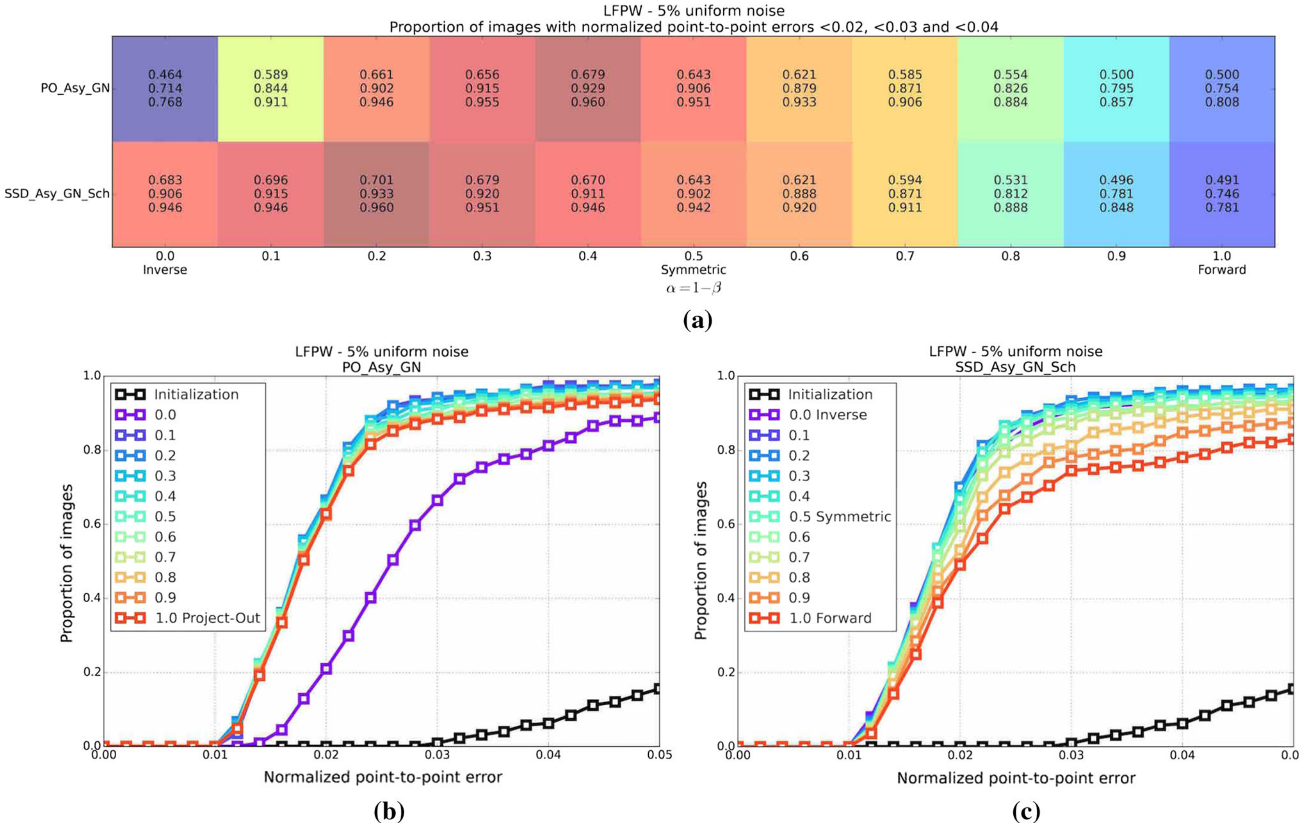


Fig. 12 Results quantifying the effect of varying the value of the parameters $\alpha = 1 - \beta$ in Asymmetric algorithms. **a** Proportion of images with normalized point-to-point errors smaller than 0.02, 0.03 and 0.04 for the Project-Out and SSD Asymmetric Gauss-Newton algorithms for different values of $\alpha = 1 - \beta$ and initialized with 5 % noise. Colors encode overall fitting accuracy, from highest to lowest: *red*, *orange*, *yellow*, *green*, *blue* and *purple*. **b** CED on the LFPW test dataset for Project-Out Asymmetric Gauss-Newton algorithm for different values of $\alpha = 1 - \beta$ and initialized with 5 % noise. **c** CED on the LFPW test dataset for the the SSD Asymmetric Gauss-Newton algorithm for different values of $\alpha = 1 - \beta$ and initialized with 5 % noise (Color figure online)

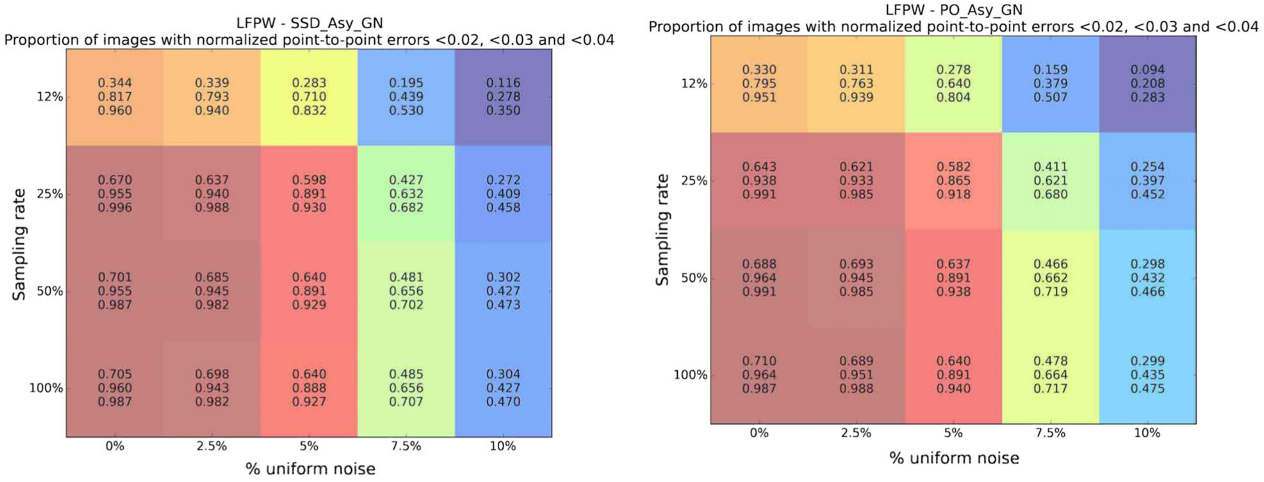
m, components i.e. $F >> m >> n$. Therefore, a significant reduction in the complexity (and running time) of CGD algorithms can be obtained by decreasing the number of pixels that are used to optimize the previous cost functions. To this end, we compare the accuracy obtained by using 100, 50, 25 and 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, Fig. 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, we compare the accuracy obtained by running the algorithms for 40 (24 + 16) and 20 (12 + 8) iterations.

Note that, in order to further highlight the advantages and disadvantages of using the previous strategies, we report the fitting accuracy obtained by initializing the algorithms using different amounts of uniform noise.

Once more we repeat the experimental set up of the first experiment and report the fitting accuracy obtained by the Project Out and SSD Asymmetric Gauss-Newton algorithms. Results for this experiment are shown in Fig. 13. 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, Fig. 13a, b. Also, performance seems to be consistent along the amount of noise. In terms of run time, Fig. 13c, reducing the number of pixels to 50, 25 and 12 % offers speed ups of ∼2.0x, ∼2.9x and ∼3.7x for the *BPO* algorithm and of ∼1.8x, ∼2.6x and ∼2.8x for the *SSD* algorithm respectively.

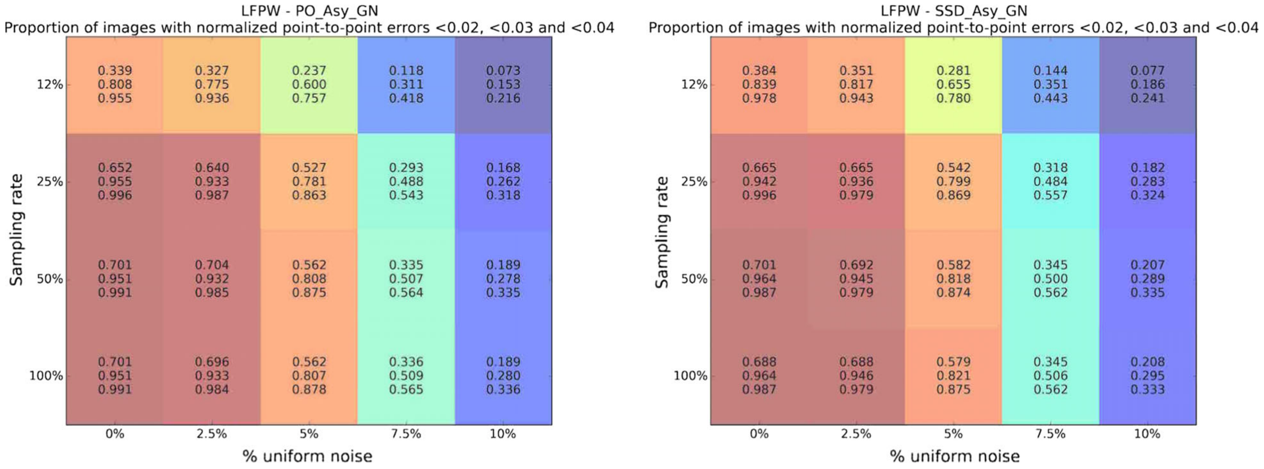
On the other hand, reducing the number of iterations from 40 (24 + 16) to 20 (12 + 8) has no negative impact in performance for levels of noise smaller than 2 % but has a noticeable negative impact for levels of noise bigger than 5 %. Notice that remarkable speed ups, Fig. 13f, can be obtain for both algorithms by combining the previous two strategies at the expenses of small but noticeable decreases in fitting accuracy.



(a) **(b)**

	100%	< 50%	< 25%	< 12%
SSD_Asy_GN_Sch	∼ 1680 ms	∼ 930 ms	∼ 650 ms	∼ 590 ms
PO_Asy_GN	∼ 1400 ms	∼ 680 ms	∼ 480 ms	∼ 380 ms

(c)

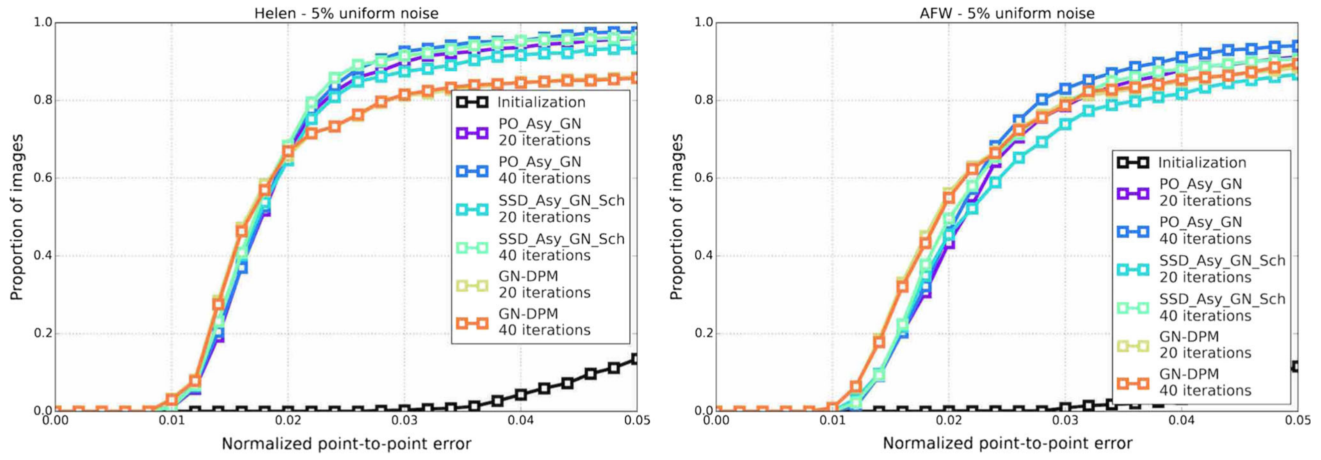


(d) **(e)**

	100%	< 50%	< 25%	< 12%
SSD_Asy_GN_Sch	∼ 892 ms	∼ 519 ms	∼ 369 ms	∼ 331 ms
PO_Asy_GN	∼ 707 ms	∼ 365 ms	∼ 235 ms	∼ 211 ms

(f)

Fig. 13 Results assessing the effectiveness of sampling for the best performing Project-Out and SSD algorithms on the LFPW database. **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. Colors encode overall fitting accuracy, from highest to lowest: *red*, *orange*, *yellow*, *green*, *blue* and *purple*. **b** Proportion of images with normalized point-to-point errors smaller than 0.02, 0.03 and 0.04 for the Project-Out Asymmetric Gauss-Newton algorithm using different sampling rates, 40 $(24 + 16)$ iterations, and initialized with different amounts of noise. Colors encode overall fitting accuracy, from highest to lowest: *red*, *orange*, *yellow*, *green*, *blue* and *purple*. **c** Table showing run time of each algorithm for different amounts of sampling and 40 $(24+16)$ iterations. **d** Proportion of images with normalized point-to-point errors smaller than 0.02, 0.03 and 0.04 for the Project-Out Asymmetric Gauss-Newton algorithm using different sampling rates, 20 $(12 + 8)$ iterations, and initialized with different amounts of noise. Colors encode overall fitting accuracy, from highest to lowest: *red*, *orange*, *yellow*, *green*, *blue* and *purple*. **e** 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. Colors encode overall fitting accuracy, from highest to lowest: *red*, *orange*, *yellow*, *green*, *blue* and *purple*. **f** Table showing run time of each algorithm for different amounts of sampling and 20 $(12 + 8)$ iterations (Color figure online)



(a) **(b)**

Fig. 14 Results showing the fitting accuracy of the SSD and Project-Out Asymmetric Gauss-Newton algorithms on the Helen and AFW databases. **a** CED on the Helen test dataset for the Project-Out and SSD Asymmetric Gauss-Newton algorithms initialized with 5 % noise. **b** CED on the AFW database for the Project-Out and SSD Asymmetric Gauss-Newton algorithm initialized with 5 % noise

5.5 Comparison on Helen and AFW

In order to facilitate comparisons with recent prior work on AAMs (Tzimiropoulos and Pantic 2013; Antonakos et al. 2014; Kossaifi et al. 2014) and with other state-of-the-art approaches in face alignment (Xiong and De la Torre 2013; Asthana et al. 2013), in this experiment, we report the fitting accuracy of the *SSD* and *Project-Out Asymmetric Gauss-Newton* algorithms on the widely used test set of the Helen database and on the entire AFW database. Furthermore we compare the performance of the previous two algorithms with the one obtained by the recently proposed Gauss-Newton Deformable Part Models (GN-DPMs) proposed by Tzimiropoulos and Pantic in Tzimiropoulos and Pantic (2014); which was shown to achieve state-of-the-art results in the problem of face alignment in-the-wild.

For both our algorithms, we report two different types of results: (i) sampling rate of 25 % and 20 (12 + 8) iterations; and (ii) sampling rate of 50 % and 40 (24 + 16) iterations. For GN-DPMs we use the authors public implementation to generate the results. In this case, we report, again, two different types of results by letting the algorithm run for 20 and 40 iterations.

Result for this experiment are shown in Fig. 14. Looking at Fig. 14a we can see that both, *SSD* and *Project-Out 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-DPMs for normalized point-to-point errors <0.2 and significantly better for <0.3, <0.4. As expected, the best results for both our 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 *Project-Out Asymmetric Gauss-Newton*. In general, these results are consistent with the ones obtained on the LFPW test dataset, Experiments 5.1 and 5.3.

On the other hand, the performance of both algorithms drops significantly on the AFW database, Fig. 14b. In this case, GN-DPMs achieves slightly better results than the *SSD* and *Project-Out Asymmetric Gauss-Newton* algorithms for normalized point-to-point errors <0.2 and slightly worst for <0.3, <0.4. Again, both our 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 recent works on AAMs (Tzimiropoulos and Pantic 2014; Alabort-i-Medina and Zafeiriou 2014; Antonakos et al. 2014; Alabort-i-Medina and Zafeiriou 2015) and it is attributed to large difference in terms of shape and appearance statistics between the images of the AFW dataset and the ones of the training sets of the LFPW and Helen datasets where the AAM model was trained on.

Exemplar results for this experiment are shown in Figs. 15 and 16.

5.6 Analysis

Given the results reported by the previous six experiments we conclude that:

1. Overall, *Gauss-Newton* and *Wiberg* algorithms vastly outperform *Newton* algorithms for fitting AAMs. Experiment 5.1 clearly shows that the former algorithms provide significantly higher levels of fitting accuracy at

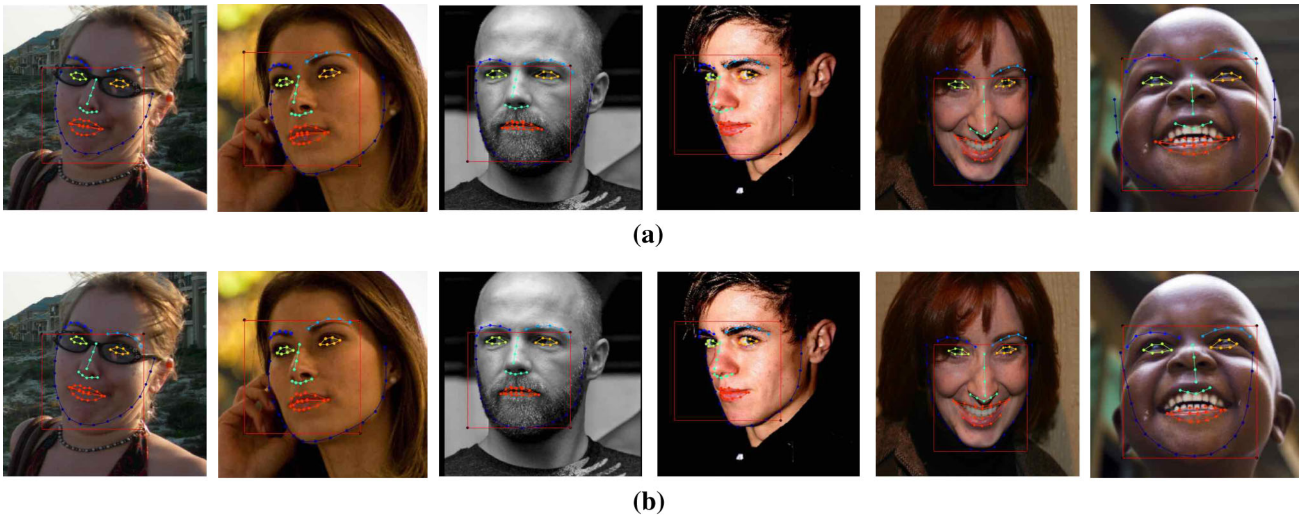


Fig. 15 Exemplar results from the Helen test dataset. **a** Exemplar results from the Helen test dataset obtained by the Project-Out Asymmetric Gauss-Newton Schur algorithm. **b** Exemplar results from the Helen test dataset obtained by the SSD Asymmetric Gauss-Newton Schur algorithm

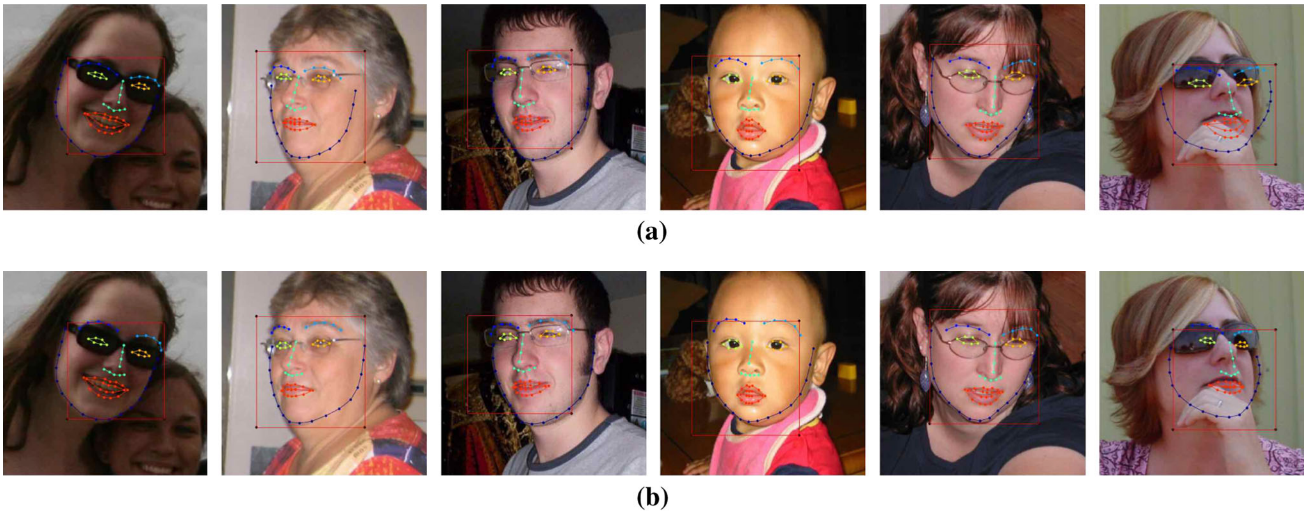


Fig. 16 Exemplar results from the AFW dataset. **a** Exemplar results from the Helen test dataset obtained by the Project-Out Asymmetric Gauss-Newton Schur algorithm. **b** Exemplar results from the AFW dataset obtained by the SSD Asymmetric Gauss-Newton Schur algorithm

considerably lower computational complexities and run times. These findings are consistent with existent literature in the related field of parametric image alignment (Matthews and Baker 2004) and also, to certain extend, with prior work on *Newton* algorithms for AAM fitting (Kossaifi et al. 2014). We attribute the bad performance of *Newton* algorithms 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 performance. The results in Experiment 5.1 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. Our *Bayesian* project-out formulation leads to significant improvements in fitting accuracy without adding extra computational cost. Experiment 5.2 shows that a weighted combination of the two terms forming *Bayesian* project-out loss always outperforms the *classic* project out formulation.
4. The *Asymmetric* composition proposed in this work leads to CGD algorithms that are more accurate and that converge faster. In particular, the *SSD* and *Project-Out Asymmetric Gauss-Newton* algorithms are shown to achieve significantly better performance than their *Forward* and *Inverse* counterparts in Experiments 5.1 and 5.3.
5. Finally, a significant reduction in the computational complexity and runtime of CDG algorithms can be obtained by limiting the number of pixels considered during optimization of the loss function and by adjusting the number

of iterations that the algorithms are run for, Experiment 5.4.

6 Conclusion

In this paper we have thoroughly studied the problem of fitting AAMs using CGD algorithms. We have presented a unified and complete framework for these algorithms and classified them with respect to three of their main characteristics: (i) *cost function*; (ii) type of *composition*; and (iii) *optimization method*.

Furthermore, we have extended the previous framework by:

- Proposing a novel *Bayesian cost function* for fitting AAMs that can be interpreted as a more general formulation of the well-known project-out loss. We have assumed a probabilistic model for appearance generation with both Gaussian noise and a Gaussian prior over a latent appearance space. Marginalizing out the latent appearance space, we have derived a novel cost function that only depends on shape parameters and that can be interpreted as a valid and more general probabilistic formulation of the well-known project-out cost function (Matthews and Baker 2004). In the experiments, we have showed that our Bayesian formulation considerably outperforms the original project-out cost function.
- Proposing *asymmetric* and *bidirectional* compositions for CGD algorithms. We have shown the connection between Gauss-Newton Asymmetric algorithms and ESM algorithms and experimentally proved that these two novel types of composition lead to better convergent and more robust CGD algorithm for fitting AAMs.
- Providing new valuable insights into existent CGD algorithms by reinterpreting them as direct applications of the *Schur complement* and the *Wiberg method*.

Finally, in terms of future work, we plan to:

- Adapt existent Supervised Descent (SD) algorithms for face alignment (Xiong and De la Torre 2013; Tzimiropoulos 2015) to AAMs and investigate their relationship with the CGD algorithms studied in this paper.
- Investigate if our Bayesian cost function and the proposed asymmetric and bidirectional compositions can also be successfully applied to similar generative parametric models, such as the Gauss-Newton Parts-Based Deformable Model (GN-DPM) proposed in Tzimiropoulos and Pantic (2014).

Acknowledgments The work of Joan Alabort-i-Medina is funded by a DTA studentship from Imperial College London and by the Qualcomm Innovation Fellowship. The work of S. Zafeiriou has been partly funded by the EPSRC project Adaptive Facial Deformable Models for Tracking (ADAManT), EP/L026813/1.

Appendix 1: Asymmetric Gauss-Newton Algorithms as Efficient Second-order Minimization (ESM)

In this section, we show that the *Asymmetric Gauss-Newton* algorithms derived in Sect. 3.3.1 are, in fact, also 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 (Malis 2004; Benhimane and Malis 2004; Mégret et al. 2008, 2010). Following a similar line of reasoning, we will show that *Asymmetric Gauss-Newton* algorithms for fitting AAMs can also be also interpreted as ESM algorithms.

In order to show the previous relationship we will make use of the simplified data term[28] introduced by Eq. 25. Using *forward* composition, the optimization problem defined by:

$$\Delta\mathbf{p}^* = \arg\min_{\Delta\mathbf{p}} \frac{1}{2}\mathbf{r}_f^T \bar{\mathbf{A}} \mathbf{r}_f \tag{125}$$

where the forward residual $\mathbf{r}_f$ is defined as:

$$\mathbf{r}_f = \mathbf{i}[\mathbf{p} \circ \Delta\mathbf{p}] - \mathbf{a} \tag{126}$$

As seen before, Gauss-Newton solves the previous optimization problem by performing a *first* order Taylor expansion of the residual around $\Delta\mathbf{p}$:

$$\begin{aligned}\hat{\mathbf{r}}_f(\Delta\mathbf{p}) &= \mathbf{r}_f + \frac{\partial \mathbf{r}_f}{\partial \Delta\mathbf{p}}\Delta\mathbf{p} + \underbrace{O_{\mathbf{r}_f}(\Delta\mathbf{p}^2)}_{\text{remainder}} \\ &= \mathbf{i}[\mathbf{p}] - \bar{\mathbf{a}} + \mathbf{J}_{\mathbf{i}}\Delta\mathbf{p} + O_{\mathbf{r}_f}(\Delta\mathbf{p}^2)\end{aligned} \tag{127}$$

and solving the following approximation of the original problem:

$$\Delta\mathbf{p}^* = \arg\min_{\Delta\mathbf{p}} \frac{1}{2}\hat{\mathbf{r}}_f^T \hat{\mathbf{r}}_f \tag{128}$$

However, note that, instead of performing a first order Taylor expansion, we can also perform a *second* order Taylor expansion of the residual:

[28] Notice that similar derivations can also be obtained using the SSD and Project-Out data terms, but we use the simplified one here for clarity.

$$\begin{aligned}\check{\mathbf{r}}_f(\Delta\mathbf{p}) &= \mathbf{r}_f + \frac{\partial \mathbf{r}_f}{\partial \Delta\mathbf{p}}\Delta\mathbf{p}\\ &\quad + \frac{1}{2}\Delta\mathbf{p}^T\frac{\partial^2\mathbf{r}_f}{\partial^2\Delta\mathbf{p}}\Delta\mathbf{p} + O_{\mathbf{r}_f}(\Delta\mathbf{p}^3)\\ &= \mathbf{i}[\mathbf{p}] - \mathbf{a} + \mathbf{J_i}\Delta\mathbf{p}\\ &\quad + \frac{1}{2}\Delta\mathbf{p}^T\mathbf{H_i}\Delta\mathbf{p} + O_{\mathbf{r}_f}(\Delta\mathbf{p}^3)\end{aligned} \tag{129}$$

Then, given the second main assumption behind AAMs (Eq. 7) the following approximation must hold:

$$\begin{aligned}\nabla\mathbf{i}[\mathbf{p}]\frac{\partial\mathcal{W}}{\partial\Delta\mathbf{p}} &\approx \nabla\mathbf{a}\frac{\partial\mathcal{W}}{\partial\Delta\mathbf{p}}\\ \mathbf{J_i} &\approx \mathbf{J_a}\end{aligned} \tag{130}$$

and, because the previous $\mathbf{J_i}$ and $\mathbf{J_a}$ are functions of $\Delta\mathbf{p}$, we can perform a *first* order Taylor expansion of $\mathbf{J_i}$ to obtain:

$$\begin{aligned}\mathbf{J_i}(\Delta\mathbf{p}) &\approx \mathbf{J_i} + \Delta\mathbf{p}^T\frac{\partial\mathbf{J_i}}{\partial\Delta\mathbf{p}} + \underbrace{O_{\mathbf{J_i}}(\Delta\mathbf{p}^2)}_{\text{remainder}}\\ &\approx \mathbf{J_i} + \Delta\mathbf{p}^T\mathbf{H_i} + O_{\mathbf{J_i}}(\Delta\mathbf{p}^2)\\ \mathbf{J_a} &\approx \mathbf{J_i} + \Delta\mathbf{p}^T\mathbf{H_i} + O_{\mathbf{J_i}}(\Delta\mathbf{p}^2)\\ \Delta\mathbf{p}^T\mathbf{H_i} &\approx \mathbf{J_a} - \mathbf{J_i} - O_{\mathbf{J_i}}(\Delta\mathbf{p}^2)\end{aligned} \tag{131}$$

Finally, substituting the previous approximation for $\Delta\mathbf{p}^T\mathbf{H_i}$ into Eq. 129 we arrive at:

$$\begin{aligned}\check{\mathbf{r}}_f(\Delta\mathbf{p}) &= \mathbf{i}[\mathbf{p}] - \mathbf{a} + \mathbf{J_i}\Delta\mathbf{p}\\ &\quad + \frac{1}{2}\Delta\mathbf{p}^T\mathbf{H_i}\Delta\mathbf{p} + O_{\mathbf{r}_f}(\Delta\mathbf{p}^3)\\ &= \mathbf{i}[\mathbf{p}] - \mathbf{a} + \mathbf{J_i}\Delta\mathbf{p}\\ &\quad + \frac{1}{2}\left(\mathbf{J_a} - \mathbf{J_i} - O_{\mathbf{J_i}}(\Delta\mathbf{p}^2)\right)\Delta\mathbf{p}\\ &\quad + O_{\mathbf{r}_f}(\Delta\mathbf{p}^3)\\ &= \mathbf{i}[\mathbf{p}] - \mathbf{a} + \frac{1}{2}(\mathbf{J_i} + \mathbf{J_a})\Delta\mathbf{p}\\ &\quad + O_{\text{total}}(\Delta\mathbf{p}^3)\end{aligned} \tag{132}$$

where the total remainder is cubic with respect to $\Delta\mathbf{p}$:

$$O_{\text{total}}(\Delta\mathbf{p}^3) = O_{\mathbf{r}_f}(\Delta\mathbf{p}^3) - O_{\mathbf{J_i}}(\Delta\mathbf{p}^2)\Delta\mathbf{p} \tag{133}$$

The expression in Eq. 132 constitutes a true *second* order approximation of the forward residual $\mathbf{r}_f$ where the term $\frac{1}{2}(\mathbf{J_i} + \mathbf{J_a})$ is equivalent to the asymmetric Jacobian in Eq. 47 when $\alpha = \beta = 0.5$:

$$\begin{aligned}\frac{1}{2}(\mathbf{J_i} + \mathbf{J_a}) &= \left(\frac{1}{2}\mathbf{J_i} + \frac{1}{2}\mathbf{J_a}\right)\\ &= \left(\frac{1}{2}\nabla\mathbf{i}[\mathbf{p}]\frac{\partial\mathcal{W}}{\partial\Delta\mathbf{p}} + \frac{1}{2}\nabla\mathbf{a}\frac{\partial\mathcal{W}}{\partial\Delta\mathbf{p}}\right)\\ &= \left(\frac{1}{2}\nabla\mathbf{i}[\mathbf{p}] + \frac{1}{2}\nabla\mathbf{a}\right)\frac{\partial\mathcal{W}}{\partial\Delta\mathbf{p}}\\ &= (\nabla\mathbf{t})\frac{\partial\mathcal{W}}{\partial\Delta\mathbf{p}}\\ &= \mathbf{J_t}\end{aligned} \tag{134}$$

and, consequently, *Asymmetric Gauss-Newton* algorithms for fitting AAMs can be viewed as ESM algorithms that only require *first* order partial derivatives of the residual and that have the same computational complexity as *first* order algorithms.

Appendix 2: Terms in SSD Newton Hessians

In this section we define the individual terms of the Hessian matrices used by the *SSD Asymmetric* and *Bidirectional Newton* optimization algorithms derived in Sect. 3.3.2.

(a) Asymmetric

The individual terms forming the Hessian matrix of the *SSD Asymmetric Newton* algorithm defined by Eq. 83 are defined as follows:

$$\begin{aligned}\frac{\partial^2\mathcal{D}_a}{\partial\Delta\mathbf{c}^2} &= \frac{\partial - \mathbf{A}^T\mathbf{r}_a}{\partial\Delta\mathbf{c}}\\ &= -\mathbf{A}^T\frac{\partial\mathbf{r}_a}{\partial\Delta\mathbf{c}}\\ &= \underbrace{\mathbf{A}^T\mathbf{A}}_{\mathbf{I}}\end{aligned} \tag{135}$$

$$\begin{aligned}\frac{\partial^2\mathcal{D}_a}{\partial\Delta\mathbf{c}\partial\Delta\mathbf{p}} &= \frac{\partial - \mathbf{A}^T\mathbf{r}_a}{\partial\Delta\mathbf{p}}\\ &= \frac{\partial - \mathbf{A}^T}{\partial\Delta\mathbf{p}}\mathbf{r}_a - \mathbf{A}^T\frac{\partial\mathbf{r}_a}{\partial\Delta\mathbf{p}}\\ &= -\beta\mathbf{J}_\mathbf{A}^T\mathbf{r}_a - \mathbf{A}^T\mathbf{J_t}\end{aligned} \tag{136}$$

where we have defined $\mathbf{J_A} = [\nabla\mathbf{a}_1, \ldots, \nabla\mathbf{a}_m]^T\frac{\partial\mathcal{W}}{\partial\Delta\mathbf{p}}$.

$$\begin{aligned}\frac{\partial^2\mathcal{D}_a}{\partial\Delta\mathbf{p}^2} &= \frac{\partial\mathbf{J}_\mathbf{t}^T\mathbf{r}_a}{\partial\Delta\mathbf{p}}\\ &= \frac{\partial\mathbf{J}_\mathbf{t}^T}{\partial\Delta\mathbf{p}}\mathbf{r}_a + \mathbf{J}_\mathbf{t}^T\frac{\partial\mathbf{r}_a}{\partial\Delta\mathbf{p}}\\ &= \left(\frac{\partial\mathcal{W}}{\partial\Delta\mathbf{p}}^T\nabla^2\mathbf{t}\frac{\partial\mathcal{W}}{\partial\Delta\mathbf{p}} + \underbrace{\nabla\mathbf{t}\overbrace{\frac{\partial^2\mathcal{W}}{\partial^2\mathbf{p}}}^{\mathbf{0}}}_{\mathbf{0}}\right)\mathbf{r}_a\\ &\quad + \mathbf{J}_\mathbf{t}^T\mathbf{J_t}\\ &= \left(\frac{\partial\mathcal{W}}{\partial\Delta\mathbf{p}}^T\nabla^2\mathbf{t}\frac{\partial\mathcal{W}}{\partial\Delta\mathbf{p}}\right)\mathbf{r}_a + \mathbf{J}_\mathbf{t}^T\mathbf{J_t}\end{aligned} \tag{137}$$

(b) Bidirectional

The individual terms forming the Hessian matrix of the *SSD Bidirectional Newton* algorithm defined by Eq. 86 are defined as follows:

$$\begin{aligned}\frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{c}^2} &= \frac{\partial - \mathbf{A}^T \mathbf{r}_b}{\partial \Delta \mathbf{c}} \\ &= -\mathbf{A}^T \frac{\partial \mathbf{r}_b}{\partial \Delta \mathbf{c}} \\ &= \underbrace{\mathbf{A}^T \mathbf{A}}_{\mathbf{I}} \qquad (138)\end{aligned}$$

$$\begin{aligned}\frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{c} \partial \Delta \mathbf{p}} &= \frac{\partial - \mathbf{A}^T \mathbf{r}_b}{\partial \Delta \mathbf{p}} \\ &= -\mathbf{A}^T \frac{\partial \mathbf{r}_b}{\partial \Delta \mathbf{p}} \\ &= -\mathbf{A}^T \mathbf{J}_\mathbf{i} \qquad (139)\end{aligned}$$

$$\begin{aligned}\frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{c} \partial \Delta \mathbf{q}} &= \frac{\partial - \mathbf{A}^T \mathbf{r}_b}{\partial \Delta \mathbf{q}} \\ &= \frac{\partial - \mathbf{A}^T}{\partial \Delta \mathbf{q}} \mathbf{r}_b - \mathbf{A}^T \frac{\partial \mathbf{r}_b}{\partial \Delta \mathbf{q}} \\ &= -\mathbf{J}_\mathbf{A}^T \mathbf{r}_b + \mathbf{A}^T \mathbf{J}_\mathbf{a} \qquad (140)\end{aligned}$$

$$\begin{aligned}\frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{p}^2} &= \frac{\partial \mathbf{J}_\mathbf{i}^T \mathbf{r}_b}{\partial \Delta \mathbf{p}} \\ &= \frac{\partial \mathbf{J}_\mathbf{i}^T}{\partial \Delta \mathbf{p}} \mathbf{r}_b + \mathbf{J}_\mathbf{i}^T \frac{\partial \mathbf{r}_b}{\partial \Delta \mathbf{p}} \\ &= \left(\frac{\partial \mathcal{W}}{\partial \Delta \mathbf{p}}^T \nabla^2 \mathbf{i}[\mathbf{p}] \frac{\partial \mathcal{W}}{\partial \Delta \mathbf{p}} \right) \mathbf{r}_b + \mathbf{J}_\mathbf{i}^T \mathbf{J}_\mathbf{i} \qquad (141)\end{aligned}$$

$$\begin{aligned}\frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{p} \partial \Delta \mathbf{q}} &= \frac{\partial \mathbf{J}_\mathbf{i}^T \mathbf{r}_b}{\partial \Delta \mathbf{q}} \\ &= -\mathbf{J}_\mathbf{i}^T \mathbf{J}_\mathbf{a} \qquad (142)\end{aligned}$$

$$\begin{aligned}\frac{\partial^2 \mathcal{D}_b}{\partial \Delta \mathbf{q}^2} &= \frac{\partial - \mathbf{J}_\mathbf{a}^T \mathbf{r}_b}{\partial \Delta \mathbf{q}} = \frac{\partial - \mathbf{J}_\mathbf{a}^T}{\partial \Delta \mathbf{q}} \mathbf{r}_b - \mathbf{J}_\mathbf{a}^T \frac{\partial \mathbf{r}_b}{\partial \Delta \mathbf{q}} \\ &= -\left(\frac{\partial \mathcal{W}}{\partial \Delta \mathbf{q}}^T \nabla^2 (\mathbf{a} + \mathbf{A}\mathbf{c}) \frac{\partial \mathcal{W}}{\partial \Delta \mathbf{q}} \right) \mathbf{r}_b + \mathbf{J}_\mathbf{a}^T \mathbf{J}_\mathbf{a} \\ & \qquad (143)\end{aligned}$$

Appendix 3: Iterative Solutions of All Algorithms

In this section we report the iterative solutions of all CGD algorithms studied in this paper. In order to keep the information structured algorithms are grouped by their cost function. Consequently, iterative solutions for all SSD and Project-Out algorithms are stated in Tables 1 and 2.

Table 1 Iterative solutions of all SSD algorithms studied in this paper

SSD algorithms	Iterative solutions		
	$\Delta \mathbf{p}$	$\Delta \mathbf{q}$	$\Delta \mathbf{c}$
SSD_For_GN_Sch Amberg et al. (2009), Tzimiropoulos and Pantic (2013)	$\Delta \mathbf{p} = -\hat{\mathbf{H}}_\mathbf{i}^{-1} \mathbf{J}_\mathbf{i}^T \bar{\mathbf{A}} \mathbf{r}$ $\hat{\mathbf{H}}_\mathbf{i} = \mathbf{J}_\mathbf{i}^T \bar{\mathbf{A}} \mathbf{J}_\mathbf{i}$		$\Delta \mathbf{c} = \mathbf{A} \left(\mathbf{r} + \mathbf{J}_\mathbf{i} \Delta \mathbf{p} \right)$
SSD_For_GN_Alt	$\Delta \mathbf{p} = -\mathbf{H}_\mathbf{i}^{-1} \mathbf{J}_\mathbf{i}^T \left(\mathbf{r} - \mathbf{A} \Delta \mathbf{c} \right)$ $\mathbf{H}_\mathbf{i} = \mathbf{J}_\mathbf{i}^T \mathbf{J}_\mathbf{i}$		$\Delta \mathbf{c} = \mathbf{A} \left(\mathbf{r} + \mathbf{J}_\mathbf{i} \Delta \mathbf{p} \right)$
SSD_For_N_Sch	$\Delta \mathbf{p} = -\left(\hat{\mathbf{H}}_\mathbf{i}^\mathrm{N} \right)^{-1} \mathbf{J}_\mathbf{i}^T \bar{\mathbf{A}} \mathbf{r}$ $\hat{\mathbf{H}}_\mathbf{i}^\mathrm{N} = \frac{\partial \mathcal{W}}{\Delta \mathbf{p}}^T \nabla^2 \mathbf{i} \frac{\partial \mathcal{W}}{\Delta \mathbf{p}} \mathbf{r} + \hat{\mathbf{H}}_\mathbf{i}$		$\Delta \mathbf{c} = \mathbf{A} \left(\mathbf{r} + \mathbf{J}_\mathbf{i} \Delta \mathbf{p} \right)$
SSD_For_N_Alt	$\Delta \mathbf{p} = -\left(\mathbf{H}_\mathbf{i}^\mathrm{N} \right)^{-1} \mathbf{J}_\mathbf{i}^T \bar{\mathbf{A}} \left(\mathbf{r} - \mathbf{A} \Delta \mathbf{c} \right)$ $\mathbf{H}_\mathbf{i}^\mathrm{N} = \frac{\partial \mathcal{W}}{\Delta \mathbf{p}}^T \nabla^2 \mathbf{i} \frac{\partial \mathcal{W}}{\Delta \mathbf{p}} \mathbf{r} + \mathbf{H}_\mathbf{i}$		$\Delta \mathbf{c} = \mathbf{A} \left(\mathbf{r} + \mathbf{J}_\mathbf{i} \Delta \mathbf{p} \right)$
SSD_For_W	$\Delta \mathbf{p} = -\hat{\mathbf{H}}_\mathbf{i}^{-1} \mathbf{J}_\mathbf{i}^T \bar{\mathbf{A}} \mathbf{r}$		$\Delta \mathbf{c} = \mathbf{A} \mathbf{r}$
SSD_Inv_GN_Sch Papandreou and Maragos (2008), Tzimiropoulos and Pantic (2013)	$\Delta \mathbf{p} = \hat{\mathbf{H}}_\mathbf{a}^{-1} \mathbf{J}_\mathbf{a}^T \bar{\mathbf{A}} \mathbf{r}$ $\hat{\mathbf{H}}_\mathbf{a} = \mathbf{J}_\mathbf{a}^T \bar{\mathbf{A}} \mathbf{J}_\mathbf{a}$		$\Delta \mathbf{c} = \mathbf{A} \left(\mathbf{r} - \mathbf{J}_\mathbf{a} \Delta \mathbf{p} \right)$
SSD_Inv_GN_Alt Tzimiropoulos et al. (2012), Antonakos et al. (2014)	$\Delta \mathbf{p} = \mathbf{H}_\mathbf{a}^{-1} \mathbf{J}_\mathbf{a}^T \left(\mathbf{r} - \mathbf{A} \Delta \mathbf{c} \right)$ $\mathbf{H}_\mathbf{a} = \mathbf{J}_\mathbf{a}^T \mathbf{J}_\mathbf{a}$		$\Delta \mathbf{c} = \mathbf{A} \left(\mathbf{r} - \mathbf{J}_\mathbf{a} \Delta \mathbf{p} \right)$
SSD_Inv_N_Sch	$\Delta \mathbf{p} = \left(\hat{\mathbf{H}}_\mathbf{a}^\mathrm{N} \right)^{-1} \mathbf{J}_\mathbf{a}^T \bar{\mathbf{A}} \mathbf{r}$ $\hat{\mathbf{H}}_\mathbf{a}^\mathrm{N} = \frac{\partial \mathcal{W}}{\Delta \mathbf{p}}^T \nabla^2 \mathbf{a} \frac{\partial \mathcal{W}}{\Delta \mathbf{p}} \mathbf{r} + \hat{\mathbf{H}}_\mathbf{a}$		$\Delta \mathbf{c} = \mathbf{A} \left(\mathbf{r} - \mathbf{J}_\mathbf{a} \Delta \mathbf{p} \right)$

Table 1 continued

SSD algorithms	Iterative solutions		
	$\Delta\mathbf{p}$	$\Delta\mathbf{q}$	$\Delta\mathbf{c}$
SSD_Inv_N_Alt	$\Delta\mathbf{p} = \left(\mathbf{H}_{\mathbf{a}}^{\mathrm{N}}\right)^{-1} \mathbf{J}_{\mathbf{a}}^{T} \bar{\mathbf{A}} \left(\mathbf{r} - \mathbf{A}\Delta\mathbf{c}\right)$		$\Delta\mathbf{c} = \mathbf{A}\left(\mathbf{r} - \mathbf{J}_{\mathbf{a}}\Delta\mathbf{p}\right)$
	$\mathbf{H}_{\mathbf{a}}^{\mathrm{N}} = \frac{\partial\mathcal{W}}{\Delta\mathbf{p}}^{T} \nabla^{2}\mathbf{i} \frac{\partial\mathcal{W}}{\Delta\mathbf{p}} \mathbf{r} + \mathbf{H}_{\mathbf{a}}$		
SSD_Inv_W	$\Delta\mathbf{p} = \hat{\mathbf{H}}_{\mathbf{a}}^{-1} \mathbf{J}_{\mathbf{a}}^{T} \bar{\mathbf{A}}\mathbf{r}$		$\Delta\mathbf{c} = \mathbf{A}\mathbf{r}$
SSD_Asy_GN_Sch	$\Delta\mathbf{p} = -\hat{\mathbf{H}}_{\mathbf{t}}^{-1} \mathbf{J}_{\mathbf{t}}^{T} \bar{\mathbf{A}}\mathbf{r}$		$\Delta\mathbf{c} = \mathbf{A}\left(\mathbf{r} + \mathbf{J}_{\mathbf{t}}\Delta\mathbf{p}\right)$
	$\hat{\mathbf{H}}_{\mathbf{t}} = \mathbf{J}_{\mathbf{t}}^{T} \bar{\mathbf{A}}\mathbf{J}_{\mathbf{t}}$		
SSD_Asy_GN_Alt	$\Delta\mathbf{p} = -\mathbf{H}_{\mathbf{t}}^{-1} \mathbf{J}_{\mathbf{t}}^{T} \left(\mathbf{r} - \mathbf{A}\Delta\mathbf{c}\right)$		$\Delta\mathbf{c} = \mathbf{A}\left(\mathbf{r} + \mathbf{J}_{\mathbf{t}}\Delta\mathbf{p}\right)$
	$\mathbf{H}_{\mathbf{t}} = \mathbf{J}_{\mathbf{t}}^{T} \mathbf{J}_{\mathbf{t}}$		
SSD_Asy_N_Sch	$\Delta\mathbf{p} = -\left(\hat{\mathbf{H}}_{\mathbf{t}}^{\mathrm{N}}\right)^{-1} \mathbf{J}_{\mathbf{t}}^{T} \bar{\mathbf{A}}\mathbf{r}$		$\Delta\mathbf{c} = \mathbf{A}\left(\mathbf{r} + \mathbf{J}_{\mathbf{t}}\Delta\mathbf{p}\right)$
	$\hat{\mathbf{H}}_{\mathbf{t}}^{\mathrm{N}} = \frac{\partial\mathcal{W}}{\Delta\mathbf{p}}^{T} \nabla^{2}\mathbf{t} \frac{\partial\mathcal{W}}{\Delta\mathbf{p}} \mathbf{r} + \hat{\mathbf{H}}_{\mathbf{t}}$		
SSD_Asy_N_Alt	$\Delta\mathbf{p} = -\left(\mathbf{H}_{\mathbf{t}}^{\mathrm{N}}\right)^{-1} \mathbf{J}_{\mathbf{t}}^{T} \bar{\mathbf{A}} \left(\mathbf{r} - \mathbf{A}\Delta\mathbf{c}\right)$		$\Delta\mathbf{c} = \mathbf{A}\left(\mathbf{r} + \mathbf{J}_{\mathbf{t}}\Delta\mathbf{p}\right)$
	$\mathbf{H}_{\mathbf{t}}^{\mathrm{N}} = \frac{\partial\mathcal{W}}{\Delta\mathbf{p}}^{T} \nabla^{2}\mathbf{t} \frac{\partial\mathcal{W}}{\Delta\mathbf{p}} \mathbf{r} + \mathbf{H}_{\mathbf{t}}$		
SSD_Asy_W	$\Delta\mathbf{p} = -\hat{\mathbf{H}}_{\mathbf{t}}^{-1} \mathbf{J}_{\mathbf{t}}^{T} \bar{\mathbf{A}}\mathbf{r}$		$\Delta\mathbf{c} = \mathbf{A}\mathbf{r}$
SSD_Bid_GN_Sch	$\Delta\mathbf{p} = -\hat{\mathbf{H}}_{\mathbf{i}}^{-1} \mathbf{J}_{\mathbf{i}}^{T} \bar{\mathbf{A}}\mathbf{r}_{1}$	$\Delta\mathbf{q} = \check{\mathbf{H}}_{\mathbf{a}}^{-1} \mathbf{J}_{\mathbf{a}}^{T} \mathbf{P}\mathbf{r}$	$\Delta\mathbf{c} = \mathbf{A}\mathbf{r}_{2}$
	$\mathbf{r}_{1} = \left(\mathbf{r} - \mathbf{J}_{\mathbf{a}}\Delta\mathbf{q}\right)$	$\check{\mathbf{H}}_{\mathbf{a}} = \mathbf{J}_{\mathbf{a}}^{T} \mathbf{P}\mathbf{J}_{\mathbf{a}}$	$\mathbf{r}_{2} = \left(\mathbf{r} + \mathbf{J}_{\mathbf{i}}\Delta\mathbf{p} - \mathbf{J}_{\mathbf{a}}\Delta\mathbf{q}\right)$
		$\mathbf{P} = \bar{\mathbf{A}} - \bar{\mathbf{A}}\mathbf{J}_{\mathbf{i}}\hat{\mathbf{H}}_{\mathbf{i}}^{-1} \mathbf{J}_{\mathbf{i}}^{T} \bar{\mathbf{A}}$	
SSD_Bid_GN_Alt	$\Delta\mathbf{p} = -\mathbf{H}_{\mathbf{i}}^{-1} \mathbf{J}_{\mathbf{i}}^{T} \mathbf{r}_{3}$	$\Delta\mathbf{q} = \mathbf{H}_{\mathbf{a}}^{-1} \mathbf{J}_{\mathbf{a}}^{T} \mathbf{r}_{4}$	$\Delta\mathbf{c} = \mathbf{A}\mathbf{r}_{2}$
	$\mathbf{r}_{3} = \left(\mathbf{r} - \mathbf{A}\Delta\mathbf{c} - \mathbf{J}_{\mathbf{a}}\Delta\mathbf{q}\right)$	$\mathbf{r}_{4} = \left(\mathbf{r} - \mathbf{A}\Delta\mathbf{c} + \mathbf{J}_{\mathbf{i}}\Delta\mathbf{p}\right)$	
SSD_Bid_N_Sch	$\Delta\mathbf{p} = -\left(\hat{\mathbf{H}}_{\mathbf{i}}^{\mathrm{N}}\right)^{-1} \mathbf{J}_{\mathbf{i}}^{T} \bar{\mathbf{A}}\mathbf{r}_{1}$	$\Delta\mathbf{q} = \left(\check{\mathbf{H}}_{\mathbf{a}}^{\mathrm{N}}\right)^{-1} \mathbf{J}_{\mathbf{a}}^{T} \mathbf{P}^{\mathrm{N}}\mathbf{r}$	$\Delta\mathbf{c} = \mathbf{A}\mathbf{r}_{2}$
		$\check{\mathbf{H}}_{\mathbf{a}}^{\mathrm{N}} = \frac{\partial\mathcal{W}}{\Delta\mathbf{p}}^{T} \nabla^{2}\mathbf{t} \frac{\partial\mathcal{W}}{\Delta\mathbf{p}} \mathbf{r} + \check{\mathbf{H}}_{\mathbf{a}}$	
		$\mathbf{P}^{\mathrm{N}} = \bar{\mathbf{A}} - \bar{\mathbf{A}}\mathbf{J}_{\mathbf{i}} \left(\hat{\mathbf{H}}_{\mathbf{i}}^{\mathrm{N}}\right)^{-1} \mathbf{J}_{\mathbf{i}}^{T} \bar{\mathbf{A}}$	
SSD_Bid_N_Alt	$\Delta\mathbf{p} = -\left(\mathbf{H}_{\mathbf{i}}^{\mathrm{N}}\right)^{-1} \mathbf{J}_{\mathbf{i}}^{T} \mathbf{r}_{3}$	$\Delta\mathbf{q} = \left(\mathbf{H}_{\mathbf{a}}^{\mathrm{N}}\right)^{-1} \mathbf{J}_{\mathbf{a}}^{T} \mathbf{r}_{4}$	$\Delta\mathbf{c} = \mathbf{A}\mathbf{r}_{2}$
SSD_Bid_W	$\Delta\mathbf{p} = -\hat{\mathbf{H}}_{\mathbf{i}}^{-1} \mathbf{J}_{\mathbf{i}}^{T} \bar{\mathbf{A}}\mathbf{r}$	$\Delta\mathbf{q} = \check{\mathbf{H}}_{\mathbf{a}}^{-1} \mathbf{J}_{\mathbf{a}}^{T} \mathbf{P}\mathbf{r}$	$\Delta\mathbf{c} = \mathbf{A}\mathbf{r}$

Table 2 Iterative solutions of all Project-Out algorithms studied in this paper

Project-Out algorithms	Iterative solutions	
	$\Delta\mathbf{p}$	$\Delta\mathbf{q}$
PO_For_GN Amberg et al. (2009), Tzimiropoulos and Pantic (2013)	$\Delta\mathbf{p} = -\hat{\mathbf{H}}_{\mathbf{i}}^{-1} \mathbf{J}_{\mathbf{i}}^{T} \bar{\mathbf{A}}\mathbf{r}$	
	$\hat{\mathbf{H}}_{\mathbf{i}} = \mathbf{J}_{\mathbf{i}}^{T} \bar{\mathbf{A}}\mathbf{J}_{\mathbf{i}}$	
PO_For_N	$\Delta\mathbf{p} = -\left(\hat{\mathbf{H}}_{\mathbf{i}}^{\mathrm{N}}\right)^{-1} \mathbf{J}_{\mathbf{i}}^{T} \bar{\mathbf{A}}\mathbf{r}$	
	$\hat{\mathbf{H}}_{\mathbf{i}}^{\mathrm{N}} = \frac{\partial\mathcal{W}}{\partial\Delta\mathbf{p}}^{T} \nabla^{2}\mathbf{i} \frac{\partial\mathcal{W}}{\partial\Delta\mathbf{p}} \bar{\mathbf{A}}\mathbf{r} + \hat{\mathbf{H}}_{\mathbf{i}}$	
PO_Inv_GN Matthews and Baker (2004)	$\Delta\mathbf{p} = \hat{\mathbf{H}}_{\mathbf{a}}^{-1} \mathbf{J}_{\bar{\mathbf{a}}}^{T} \bar{\mathbf{A}}\mathbf{r}$	
	$\hat{\mathbf{H}}_{\bar{\mathbf{a}}} = \mathbf{J}_{\bar{\mathbf{a}}}^{T} \bar{\mathbf{A}}\mathbf{J}_{\bar{\mathbf{a}}}$	
PO_Inv_N	$\Delta\mathbf{p} = \left(\hat{\mathbf{H}}_{\bar{\mathbf{a}}}^{\mathrm{N}}\right)^{-1} \mathbf{J}_{\bar{\mathbf{a}}}^{T} \bar{\mathbf{A}}\mathbf{r}$	
	$\hat{\mathbf{H}}_{\bar{\mathbf{a}}}^{\mathrm{N}} = \frac{\partial\mathcal{W}}{\Delta\mathbf{p}}^{T} \nabla^{2}\bar{\mathbf{a}} \frac{\partial\mathcal{W}}{\Delta\mathbf{p}} \bar{\mathbf{A}}\mathbf{r} + \hat{\mathbf{H}}_{\bar{\mathbf{a}}}$	
PO_Asy_GN	$\Delta\mathbf{p} = -\hat{\mathbf{H}}_{\mathbf{t}}^{-1} \mathbf{J}_{\mathbf{t}}^{T} \bar{\mathbf{A}}\mathbf{r}$	
	$\hat{\mathbf{H}}_{\mathbf{t}} = \mathbf{J}_{\mathbf{t}}^{T} \bar{\mathbf{A}}\mathbf{J}_{\mathbf{t}}$	

Table 2 continued

Project-Out algorithms	Iterative solutions	
	$\Delta\mathbf{p}$	$\Delta\mathbf{q}$
PO_Asy_N	$\Delta\mathbf{p} = -\left(\hat{\mathbf{H}}^{\mathrm{N}}_{\mathbf{t}}\right)^{-1}\mathbf{J}^T_{\mathbf{t}}\bar{\mathbf{A}}\mathbf{r}$ $\hat{\mathbf{H}}^{\mathrm{N}}_{\mathbf{t}} = \frac{\partial\mathcal{W}}{\partial\Delta\mathbf{p}}^T\nabla^2\mathbf{t}\frac{\partial\mathcal{W}}{\partial\Delta\mathbf{p}}\bar{\mathbf{A}}\mathbf{r} + \hat{\mathbf{H}}_{\mathbf{t}}$	
PO_Bid_GN_Sch	$\Delta\mathbf{p} = -\hat{\mathbf{H}}^{-1}_{\mathbf{i}}\mathbf{J}^T_{\mathbf{i}}\bar{\mathbf{A}}\left(\mathbf{r} - \mathbf{J}_{\bar{\mathbf{a}}}\Delta\mathbf{q}\right)$	$\Delta\mathbf{q} = \check{\mathbf{H}}^{-1}_{\bar{\mathbf{a}}}\mathbf{J}^T_{\mathbf{i}}\mathbf{P}\mathbf{r}$ $\check{\mathbf{H}}_{\bar{\mathbf{a}}} = \mathbf{J}^T_{\bar{\mathbf{a}}}\mathbf{P}\mathbf{J}_{\bar{\mathbf{a}}}$ $\mathbf{P} = \bar{\mathbf{A}} - \bar{\mathbf{A}}\mathbf{J}_{\mathbf{i}}\hat{\mathbf{H}}^{-1}_{\mathbf{i}}\mathbf{J}^T_{\mathbf{i}}\bar{\mathbf{A}}$
PO_Bid_GN_Alt	$\Delta\mathbf{p} = -\hat{\mathbf{H}}^{-1}_{\mathbf{i}}\mathbf{J}^T_{\mathbf{i}}\bar{\mathbf{A}}\left(\mathbf{r} - \mathbf{J}_{\bar{\mathbf{a}}}\Delta\mathbf{q}\right)$	$\Delta\mathbf{q} = \hat{\mathbf{H}}^{-1}_{\bar{\mathbf{a}}}\mathbf{J}^T_{\bar{\mathbf{a}}}\bar{\mathbf{A}}\left(\mathbf{r} + \mathbf{J}_{\mathbf{i}}\Delta\mathbf{p}\right)$
PO_Bid_N_Sch	$\Delta\mathbf{p} = -\left(\hat{\mathbf{H}}^{\mathrm{N}}_{\mathbf{i}}\right)^{-1}\mathbf{J}^T_{\mathbf{i}}\bar{\mathbf{A}}\left(\mathbf{r} - \mathbf{J}_{\bar{\mathbf{a}}}\Delta\mathbf{q}\right)$	$\Delta\mathbf{q} = \left(\check{\mathbf{H}}^{\mathrm{N}}_{\bar{\mathbf{a}}}\right)^{-1}\mathbf{J}^T_{\bar{\mathbf{a}}}\mathbf{P}^{\mathrm{N}}\mathbf{r}$ $\check{\mathbf{H}}^{\mathrm{N}}_{\bar{\mathbf{a}}} = \frac{\partial\mathcal{W}}{\Delta\mathbf{p}}^T\nabla^2\bar{\mathbf{a}}\frac{\partial\mathcal{W}}{\Delta\mathbf{p}}\bar{\mathbf{A}}\mathbf{r} + \check{\mathbf{H}}_{\bar{\mathbf{a}}}$ $\mathbf{P}^{\mathrm{N}} = \bar{\mathbf{A}} - \bar{\mathbf{A}}\mathbf{J}_{\mathbf{i}}\left(\hat{\mathbf{H}}^{\mathrm{N}}_{\mathbf{i}}\right)^{-1}\mathbf{J}^T_{\mathbf{i}}\bar{\mathbf{A}}$
PO_Bid_N_Alt	$\Delta\mathbf{p} = -\left(\hat{\mathbf{H}}^{\mathrm{N}}_{\mathbf{i}}\right)^{-1}\mathbf{J}^T_{\mathbf{i}}\bar{\mathbf{A}}\left(\mathbf{r} - \mathbf{J}_{\bar{\mathbf{a}}}\Delta\mathbf{q}\right)$	$\Delta\mathbf{q} = \left(\hat{\mathbf{H}}^{\mathrm{N}}_{\bar{\mathbf{a}}}\right)^{-1}\mathbf{J}^T_{\bar{\mathbf{a}}}\bar{\mathbf{A}}\left(\mathbf{r} + \mathbf{J}_{\mathbf{i}}\Delta\mathbf{p}\right)$
PO_Bid_W	$\Delta\mathbf{p} = -\hat{\mathbf{H}}^{-1}_{\mathbf{i}}\mathbf{J}^T_{\mathbf{i}}\bar{\mathbf{A}}\mathbf{r}$	$\Delta\mathbf{q} = \check{\mathbf{H}}^{-1}_{\bar{\mathbf{a}}}\mathbf{J}^T_{\bar{\mathbf{a}}}\mathbf{P}\mathbf{r}$

Appendix 4: Additional Experiment: Comparison on MIT StreetScene Dataset

In order to showcase the broader applicability of AAMs, we complete the main experimental section by performing an additional experiment on the problem of non-rigid car alignment in-the-wild. To this end, we report the fitting accuracy of the best performing CGD algorithms on the MIT StreetScene[29] database.

We use the first view of the MIT StreetScene[30] dataset containing a wide variety of frontal car images obtained in the wild. We use 10-fold cross-validation on the ∼500 images of the previous dataset to train and test our algorithms. We report results for the two versions of the *SSD Asymmetric Gauss-Newton* and the *Project-Out Asymmetric Gauss-Newton* algorithms used in Experiment 5.5.

Result for this experiment are shown in Fig. 17. We can observe that all algorithms obtain similar performance and that they vastly improve upon the original initialization. Exemplar results for this experiment are shown in Fig. 18.

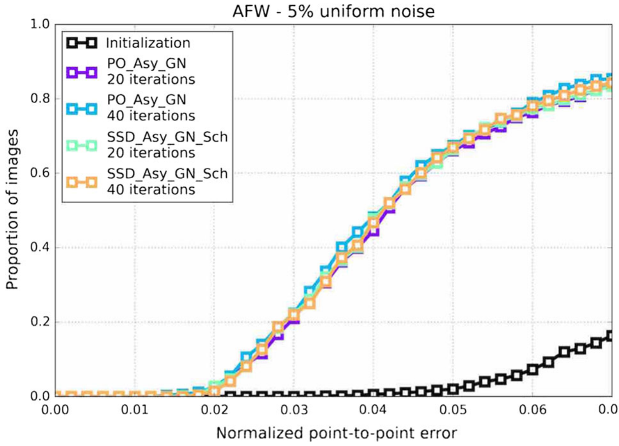


Fig. 17 CED on the first view of the MIT StreetScene test dataset for the Project-Out and SSD Asymmetric Gauss-Newton algorithms initialized with 5 % noise

[29] http://cbcl.mit.edu/software-datasets/streetscenes.

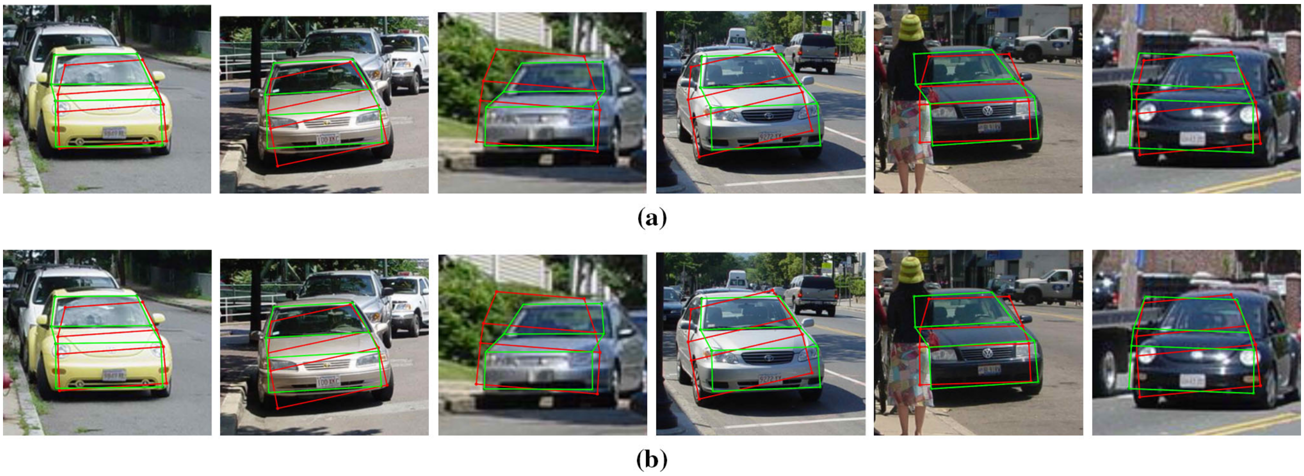


Fig. 18 Exemplar results from the MIT StreetScene test dataset. **a** Exemplar results from the MIT StreetScene test dataset obtained by the Project-Out Asymmetric Gauss-Newton Schur algorithm. **b** Exemplar results from the MIT StreetScene test dataset obtained by the SSD Asymmetric Gauss-Newton Schur algorithm

References

Alabort-i-Medina, J., & Zafeiriou, S. (2014). Bayesian active appearance models. In *IEEE Conference on computer vision and pattern recognition (CVPR)*.

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

Alabort-i-Medina, J., Antonakos, E., Booth, J., Snape, P., & Zafeiriou, S. (2014). Menpo: A comprehensive platform for parametric image alignment and visual deformable models. In *ACM international conference on multimedia (ACMM)*.

Amberg, B., Blake, A., & Vetter, T. (2009). On compositional image alignment, with an application to active appearance models. In *IEEE conference on computer vision and pattern recognition (CVPR)*.

Antonakos, E., Alabort-i-Medina, J., Tzimiropoulos, G., & Zafeiriou, S. (2014). Feature-based lucas-kanade and active appearance models. *IEEE Transactions on Image Processing (TIP)*.

Asthana, A., Zafeiriou, S., Cheng, S., & Pantic, M. (2013). Robust discriminative response map fitting with constrained local models. In *IEEE conference on computer vision and pattern recognition (CVPR)*.

Authesserre, J. B., & Berthoumieu, Y. (2010). Bidirectional composition on lie groups for gradient-based image alignment. *IEEE Transactions on Image Processing (TIP)*, *19*, 2369–2381.

Autheserre, J. B., Mégret, R., & Berthoumieu, Y. (2009). Asymmetric gradient-based image alignment. In *IEEE international conference on acoustics, speech and signal processing (ICASSP)*.

Bach, F., & Jordan, M. (2005). A probabilistic interpretation ofcanonical correlation analysis. *Technical report*, Department of Statistics. Berkeley: University of California

Baker, S., & Matthews, I. (2004). Lucas-kanade 20 years on: A unifying framework. *International Journal of Computer Vision (IJCV)*, *56*, 221–255.

Batur, A., & Hayes, M. (2005). Adaptive active appearance models. *IEEE Transactions on Image Processing (TIP)*, *14*, 1707–1721.

Belhumeur, P. N., Jacobs, D. W., Kriegman, D. J., & Kumar, N. (2011). Localizing parts of faces using a consensus of exemplars. In *Conference on computer vision and pattern recognition (CVPR)*.

Benhimane, S., & Malis, E. (2004). Real-time image-based tracking of planes using efficient second-order minimization. In *IEEE international conference on intelligent robots and systems (IROS)*.

Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge: Cambridge University Press.

Bradski, G. (2000). The opencv library. Dr Dobb's Journal of Software Tools.

Cootes, T. F., & Edwards, G. J. (2001). Active appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, *6*, 681–685.

Cootes, T. F., & Taylor, C. J. (2001). On representing edge structure for model matching. In *IEEE conference on computer vision and pattern recognition (CVPR)*.

Cootes, T. F., & Taylor, C. J. (2004). Statistical models of appearance for computer vision. *Technical report*, Imaging Science and Biomedical Engineering, University of Manchester.

Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. In *IEEE conference on computer vision and pattern recognition (CVPR)*.

De la Torre, F. (2012). A least-squares framework for component analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, *34*, 1041–1055.

Donner, R., Reiter, M., Langs, G., Peloschek, P., & Bischof, H. (2006). Fast active appearance model search using canonical correlation analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, *10*, 1690–1694.

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

Hou, X., Li, S.Z., Zhang, H., & Cheng, Q. (2001). Direct appearance models. In *IEEE conference on computer vision and pattern recognition (CVPR)*.

Kossaifi, J., Tzimiropoulos, G., & Pantic, M. (2014). Fast newton active appearance models. In *IEEE international conference on image processing (ICIP)*.

Le, V., Jonathan, B., Lin, Z., Boudev, L., & Huang, T.S. (2012). Interactive facial feature localization. In *European conference on computer vision (ECCV)*.

Liu, X. (2009). Discriminative face alignment. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, *31*, 1941–1954.

Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *IEEE international conference on computer vision (ICCV)*.

Lucey, S., Navarathna, R., Ashraf, A. B., & Sridharan, S. (2013). Fourier lucas-kanade algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, *35*, 1383–1396.

Malis, E. (2004). Improving vision-based control using efficient second-order minimization techniques. In *International conference on robotics and automation (ICRA)*.

Martins, P., Batista, J., & Caseiro, R. (2010). Face alignment through 2.5d active appearance models. In *British machine vision conference (BMVC)*.

Matthews, I., & Baker, S. (2004). Active appearance models revisited. *International Journal of Computer Vision (IJCV)*, *60*, 135–164.

Mégret, R., Authesserre, J.B., & Berthoumieu, Y. (2008). The bi-directional framework for unifying parametric image alignment approaches. In *European conference on computer vision (ECCV)*.

Moghaddam, B., & Pentland, A. (1997). Probabilistic visual learning for object representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, *19*, 696–710.

Muñoz, E., Márquez-Neila, P., & Baumela, L. (2014). Rationalizing efficient compositional image alignment. *International Journal of Computer Vision (IJCV)*, *112*, 354–372.

Nicolaou, M. A., Zafeiriou, S., & Pantic, P. (2014). A unified framework for probabilistic component analysis. In *Machine learning and knowledge discovery in databases (ECML PKDD)*.

Okatani, T., & Deguchi, K. (2006). On the wiberg algorithm for matrix factorization in the presence of missing components. *International Journal of Computer Vision (IJCV)*, *72*, 329–337.

Papandreou, G., & Maragos, P. (2008). Adaptive and constrained algorithms for inverse compositional active appearance model fitting. In *IEEE conference on computer vision and pattern recognition (CVPR)*.

Prince, S., Li, P., Fu, Y., Mohammed, U., & Elder, J. H. (2012). Probabilistic models for inference about identity. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, *34*, 144–157.

Roweis, S. (1998). Em algorithms for pca and spca. *Advances in Neural Information Processing Systems (NIPS)*, *10*, 626–632.

Sagonas, C., Tzimiropoulos, G., Zafeiriou, S., & Pantic, M. (2013a). 300 faces in-the-wild challenge: The first facial landmark localization challenge. In *IEEE international conference on computer vision workshop (ICCV-W)* (pp. 397–403).

Sagonas, C., Tzimiropoulos, G., Zafeiriou, S., & Pantic, M. (2013b). A semi-automatic methodology for facial landmark annotation. In *IEEE conference on computer vision and pattern recognition workshops (CVPRW)* (pp. 896–903).

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

Sauer, P., Cootes, T., & Taylor, C. (2011). Accurate regression procedures for active appearance models. In *British machine vision conference (BMVC)*.

Strelow, D. (2012). General and nested wiberg minimization: L2 and maximum likelihood. In *European conference on computer vision (ECCV)*.

Tipping, M. E., & Bishop, C. M. (1999). Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *61*, 611–622.

Tresadern, P.A., Sauer, P., & Cootes, T.F. (2010). Additive update predictors in active appearance models. In *British machine vision conference (BMVC)*.

Tzimiropoulos, G. (2015). Project-out cascaded regression with an application to face alignment. In *IEEE conference on computer vision and pattern recognition (CVPR)*.

Tzimiropoulos, G., & Pantic, M. (2013). Optimization problems for fast aam fitting in-the-wild. In *IEEE international conference on computer vision (ICCV)*.

Tzimiropoulos, G., & Pantic, M. (2014). Gauss-newton deformable part models for face alignment in-the-wild. In *IEEE conference on computer vision and pattern recognition (CVPR)*.

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

van der Maaten, L., & Hendriks, E. (2010). Capturing appearance variation in active appearance models. In *IEEE conference on computer vision and pattern recognition workshop (CVPR-W)*.

Vedaldi, A., & Fulkerson, B. (2010). VLFeat: An open and portable library of computer vision algorithms.

Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *IEEE conference on computer vision and pattern recognition (CVPR)*.

Woodbury, M. A. (1950). *Inverting modified matrices*. Princeton: Princeton University.

Xiong, X., & De la Torre, F. (2013). Supervised descent method and its applications to face alignment. In *IEEE conference on computer vision and pattern recognition (CVPR)*.

Zhu, X., & Ramanan, D. (2012). Face detection, pose estimation, and landmark localization in the wild. In *Conference on computer vision and pattern recognition (CVPR)*.