Tensor Decomposition and Non-linear Manifold Modeling for 3D Head Pose Estimation

Dmytro Derkach, Adrià Ruiz, Federico M Sukno

International Journal of Computer Vision manuscript No.
(will be inserted by the editor)

Tensor Decomposition and Non-linear Manifold Modeling for 3D Head Pose Estimation

Dmytro Derkach · **Adria Ruiz** · **Federico M. Sukno**

Received: date / Accepted: date

Abstract Head pose estimation is a challenging computer vision problem with important applications in different scenarios such as human-computer interaction or face recognition. In this paper, we present a 3D head pose estimation algorithm based on non-linear manifold learning. A key feature of the proposed approach is that it allows modeling the underlying 3D manifold that results from the combination of rotation angles. To do so, we use tensor decomposition to generate separate subspaces for each variation factor and show that each of them has a clear structure that can be modeled with cosine functions from a unique shared parameter per angle. Such representation provides a deep understanding of data behavior. We show that the proposed framework can be applied to a wide variety of input features and can be used for different purposes. Firstly, we test our system on a publicly available database, which consists of 2D images and we show that the cosine functions can be used to synthesize rotated versions from an object from which we see only a 2D image at a specific angle. Further, we perform 3D head pose estimation experiments using other two types of features: automatic landmarks and histogram-based 3D descriptors. We evaluate our approach on two publicly available databases, and demonstrate that angle estimations can be performed by optimizing the combination of these cosine functions to achieve state-of-the-art performance.

Keywords 3D head pose · Manifold learning · Tensor decomposition

D. Derkach and F.M. Sukno
Department of Information and Communication Technologies
Pompeu Fabra University, Barcelona, Spain
E-mail: dmytro.derkach@upf.edu; federico.sukno@upf.edu

Adria Ruiz
INRIA, Univ. Grenoble Alpes, Institute of Enginnering,
CNRS, 38000 Grenoble, France
E-mail: adria.ruiz-ovejero@inria.fr

1 Introduction

Head pose estimation is a relevant problem for several computer vision applications, including human-computer interaction, video conferencing, face recognition and facial motion analysis (Wang et al., 2018). Head pose estimation has traditionally been performed on 2D images, but advances in 3D acquisition systems have led to a growing interest in methods that operate on 3D data (Seemann et al., 2004; Fanelli et al., 2013). These methods are less sensitive to changes in illumination and viewpoint than 2D image-based approaches, which makes them more accurate and robust. Therefore, in this work we focus on head pose estimation from 3D data.

The goal of head pose estimation is to predict the relative orientation between the camera and a 3D mesh of the target head. This orientation is usually represented by three angles: rotation around vertical axis (yaw angle), around the side-to-side axis (pitch angle), and around the front-to-back axis (roll angle). Despite the fact that standard features used to represent 3D meshes lie in high-dimensional spaces, a key observation to solve this problem is that the aforementioned angles define a lower-dimensional manifold with only 3 degrees of freedom. This fact makes tensor decomposition and manifold learning appealing frameworks for the estimation of the orientation parameters. In particular, factorization methods such as multi-linear decomposition (De Lathauwer et al., 2000; Wang et al., 2017b), are able to separate the variations produced by the different factors (i.e. angles) into separate subspaces, thus obtaining specific parametrizations for each of them. On the other hand, manifold learning (Wang et al., 2017a) can be used to find the low-dimensional manifold structure defined by the orientation angles.

In this context, some previous works have attempted to use the frameworks described above for head pose

estimation. Concretely, methods such as Isomap (Raytchev et al., 2004) or Local Linear Embedding (Fu and Huang, 2006) have been explored in order to learn the underlying manifold structure defined by the orientation parameters. Even though the cited methods are able to learn generic low-dimensional data representations, the resulting manifold is only defined implicitly and, therefore, it is difficult to introduce specific constraints to model the inherent structure defined by rotation variations.

In order to address this limitation, we propose a novel approach to learn the manifold defined by 3D rotations. In particular, our method is able to explicitly model its underlying structure with an analytic form which takes into account the specific constraints imposed by orientation variations. For this purpose, we use multi-linear decomposition over data descriptors in order to split the variation factors (i.e. yaw, pitch and roll) and obtain a set of subspaces whose coefficients are governed by an unique parameter. These coefficients define a continuous curve in each of the sub-spaces that corresponds to the pose variation along one of the rotation angles. We demonstrate that the proposed approach can be applied to a wide variety of input features and, further, the obtained coefficients can be modeled in terms of trigonometric functions, which are indeed the bases to explain rotation effects. Thus, we introduce a minimization framework for pose estimation based on tensor decomposition constrained by trigonometric functions so that the solutions obtained are always compatible with the underlying rotation manifold. Preliminary results of our approach were presented in (Derkach et al., 2018).

In our experiments, we start by investigating the structure of the subspace obtained by multi-linear decomposition of 2D images when such subspace corresponds to one rotation angle using 2D images. We show that the obtained coefficients correctly describe the rotation effect, and they can indeed be modeled by a trigonometric function. We also demonstrate that these coefficients can be used to synthesize rotated versions of an object from which we only see one image at a particular rotation angle. Then, we generalize this result to rotations along any of the 3 spatial axes and demonstrate its usefulness by applying it to head pose estimation. We perform 3D head pose estimation experiments using other two types of features: automatic landmarks and histogram-based 3D descriptors. We evaluate our approach over two large and publicly available 3D face corpora: the SASE (Lüsi et al., 2016b) and BIWI databases (Fanelli et al., 2013) and demonstrate that angle estimations can be performed by optimizing the combination of cosine functions to achieve state-of-the-art performance.

The rest of the paper is organized as follows. Section 2 introduces a brief review of the existing approaches for head pose estimation. In Section 3 we give an overview of tensor decomposition methods, especially focusing on the higher order SVD (HOSVD). Section 4 explains how the tensor decomposition framework can be used to model data variations caused by rotations and shows how it can be applied for pose estimation. Then in Section 5 we perform qualitative experiments on a dataset of images with rotation about only one axis. In this experiment we show how the obtained coefficients can be modeled using cosine functions. Once the coefficients are obtained, we show that they can be used to synthesize rotated versions from an object from which we have only seen a 2D image at a specific angle. In Section 6 we demonstrate the application of the proposed framework to the problem of 3D head pose estimation from depth data using two different type of features. Finally, Section 7 concludes the paper.

2 Related work

There exist a considerable number of works that have addressed head pose estimation (Murphy-Chutorian and Trivedi, 2009) and an extensive review of all of them is out of the scope of this paper. Instead, we will focus on those approaches that are more related our work, namely methods that either use manifold learning algorithms or those that perform head pose estimation based on 3D data. It should also be mentioned that there is a recent trend of promising methods based on powerful machine learning algorithms, such us random projection forests (Lee et al., 2015), convolutional random forests (Lee et al., 2017) and convolutional neural networks (CNNs) (Chen et al., 2016; Lathuilière et al., 2017; Lathuilire et al., 2019; Liu et al., 2016; Ahn et al., 2014; Patacchiola and Cangelosi, 2017; Ruiz et al., 2018), although they have been applied mainly to head pose estimation from 2D images. A few notable exceptions are the POSEidon network (Borghi et al., 2017; Borghi et al., 2019), with an architecture based on triple regressive CNNs to combine depth, motion images and appearance cues (all of which are estimated exclusively from the depth stream), and the work from Wang et al. (2019), who employ two sub-networks based on GoogleNet (Szegedy et al., 2015) to perform coarse and fine pose estimation, reporting results both on 2D images and depth-only information.

2.1 Manifold-based methods

Many methods have considered the model of the underlying manifold structure of head pose variations (Sundararajan and Woodard, 2015; Wang et al., 2017a). The main idea behind these methods is that, regardless of the dimensionality of the input features representing the mesh,

there should be at most three degrees of freedom for head pose variation, thus defining a 3D manifold (Raytchev et al., 2004). However, in general, this manifold is embedded non-linearly in the ambient space defined by the features, which has led researchers to explore non-linear manifold learning methods such as Locally Linear Embedding (Fu and Huang, 2006), Isomap (Raytchev et al., 2004), Synchronized Submanifold Embedding (Zhu et al., 2014), Homeomorphic Manifold Analysis (Peng et al., 2014), Neighborhood Preserving Embedding or Locality Preserving Projection (BenAbdelkader, 2010) for head pose estimation from 2D images.

An interesting possibility to enhance the embedding results is to adopt a supervised strategy and use head pose labels in order to learn the manifold structure. For example, Balasubramanian et al. (2007) presented a Biased Manifold Embedding (BME) framework in which the distance metric between features is modified so that heads under similar poses are brought closer to each other than they would be under the unbiased (unsupervised) case. Similarly, Wang and Song (2014) consider head-pose information to constrain the distances between data points and present a regression variant of Fisher Discriminant Analysis (FDA), which they call supervised neighborhood-based FDA. An alternative approach is followed by BenAbdelkader (2010), who firstly apply unsupervised manifold learning methods and then employ the head pose information to train regressors in the resulting low-dimensional manifolds.

Liu et al. (2010) argue that a single manifold is not enough for head pose estimation and that appearance variations such as changes in identity, scale and illumination make it necessary the use of multiple different manifolds to model pose parameters. Thus, authors presented a clustering method to construct multiple manifolds, each of which characterizes the underlying subspace of some subjects. Peng et al. (2014) also learn multiple manifolds; they use Homeomorphic Manifold Analysis to build a separate manifold for each subject and learn non-linear mappings to relate each subject-manifold with a common pose-manifold whose topology is predefined as a unit circle or sphere (for addressing rotations about one or two axes, respectively).

The most similar work to ours is probably the one from Takallou and Kasaei (2014), who learn a non-linear tensor model based on multi-linear decomposition for head pose estimation from 2D images. They build a three-way tensor to account for identity, pose and pixels information, targeting only yaw rotations. During training, they find individual-dependent mappings between each training pose and a unified pose manifold based on tensor decomposition. At test time, each query image is projected into pose and identity subspaces, which results in as many pose coefficients as identities in the training set. The final pose estimate is obtained by validating the available pose coefficients in terms of compliance with the unified pose manifold (e.g. inversely to the distance to training samples).

In contrast to our work, all of the above methods use 2D images and most of them do not target rotations about the three spacial axes, they consider rotations about only one or two axes. Moreover, none of them provides an analytic formulation for the pose manifold.

2.2 3D methods review

Head pose estimation has traditionally been performed on 2D images. But recent advances in 3D acquisition systems have led to a growing interest in methods that operate on 3D data. These methods are less sensitive to changes in illumination and viewpoint than 2D image-based approaches, which makes them more accurate and robust.

An important distinction between different approaches is the type of input data that is used. Firstly, very few methods use only depth information, typically relying on curvatures, symmetry planes or most salient facial landmarks, such as the nose tip (Breitenstein et al., 2008; Sun and Yin, 2008; Li and Pedrycz, 2014).

In contrast, a majority of head pose estimation algorithms working in 3D, use also RGB data as additional source of information, facilitating aspects such as face detection and estimation of fiducial points. In this category we find approaches based on the fusion of 2D and 3D features (e.g. SIFT, HOG) to train regressors (Wang et al., 2013), template fitting (Martin et al., 2014), such as 3D Morphable Models (Ghiass et al., 2015; Yu et al., 2017), or depth features initialized by 2D face detection (Papazov et al., 2015).

Finally, it is also common to take advantage of temporal information for tracking the head pose across sequences of frames (Tulyakov et al., 2014; Gu et al., 2017; Barros et al., 2018; Tan et al., 2018), which considerably improves performance. However, tracking-based algorithms often benefit from the fact that test sequences usually start with nearly frontal head poses and their accuracy to detect initial head poses other than frontal is not clear. Thus, when comparing our results, we will focus on methods that provide estimation results on a per-frame bases, without tracking.

Interestingly, we see that previous methods targeting head pose estimation from 3D data have not taken advantage of the underlying manifold structure of 3D head rotations. In contrast, we present a method that is able to explicitly model its underlying manifold structure with an analytic form that takes into account the specific constraints imposed by orientation variations.

3 Technical background: Tensor decomposition

In this section, we give a review of tensor decomposition methods, especially focusing on the higher order SVD (HOSVD) (Bergqvist and Larsson, 2010; Comon, 2014; Kolda and Bader, 2009).

In many scenarios, data can be naturally represented as multidimensional arrays and, therefore, it is beneficial to take into account its inherent structure in order to analyze it. For this purpose, the use of tensors is a natural solution. In particular, a tensor is also known as a n-way array or a n-mode matrix. Vectors and matrices can be considered as first and second order tensors, respectively. First of all, we will start by reviewing the standard SVD decomposition for second order tensors.

For matrix $A \in \mathbb{R}^{m\times n}$ we recall the SVD as being:

$$A = U\Sigma V^T = \sum_{k=1}^{r} \sigma_k \mathbf{u}_k \mathbf{v}_k^T = \sum_{k=1}^{r} \sigma_k \mathbf{u}_k \otimes \mathbf{v}_k \tag{1}$$

and for the elements a_{ij} of A we have

$$a_{ij} = \sum_{k=1}^{r} u_{ik}\sigma_{kk}v_{ik} \tag{2}$$

Here $\otimes$ denotes the tensor (or outer) product $\mathbf{x}\otimes\mathbf{y} \triangleq \mathbf{x}\mathbf{y}^T$; Σ is a diagonal $(r\times r)$ matrix with nonzero singular values of A (the square roots of the eigenvalues of A^TA) on its diagonal; $\mathbf{u}_k$ and $\mathbf{v}_k$ are the orthonormal columns of the matrix U $(m\times r)$ and V $(n\times r)$, respectively, with $\mathbf{v}_k$ being the eigenvectors of A^TA and $\mathbf{u}_k = A\mathbf{v}_k/\sigma_k$ (Bergqvist and Larsson, 2010).

The SVD is useful whenever we have a two-dimensional data set $\{a_{ij}\}$, which is naturally expressed in term of a matrix A (second order tensor). In the application of this paper we will deal with cases where the dimension is bigger than two. Specifically, we will be interested in modeling data that depends on 5 factors (fifth order tensor): features, identity and 3 rotations about orthogonal spatial axes.

The SVD may be generalized to higher order tensors (or multiway arrays). Given $\mathscr{T} \in \mathbb{R}^{I_1\times I_2\times\ldots\times I_5}$, the decomposition of the fifth order tensor can be expressed as

$$\mathscr{T} = \sum_{J_1}\cdots\sum_{J_5} \mathbf{g}_{J_1J_2\ldots J_5}\mathbf{u}^{(1)}_{J_1}\otimes\mathbf{u}^{(2)}_{J_2}\otimes\cdots\otimes\mathbf{u}^{(5)}_{J_5} \tag{3}$$

or as a mode product (De Lathauwer et al., 2000)

$$\mathscr{T} = \mathscr{G}\times_1 U^{(1)}\times_2 U^{(2)}\cdots\times_5 U^{(5)} \tag{4}$$

where $\mathscr{G}\in\mathbb{R}^{J_1\times J_2\times\ldots\times J_5}$ is the core tensor and $U^{(n)}\in\mathbb{R}^{I_n\times J_n}$ – are the factor matrices (which are orthogonal) and can be thought of as the principal components in each mode.

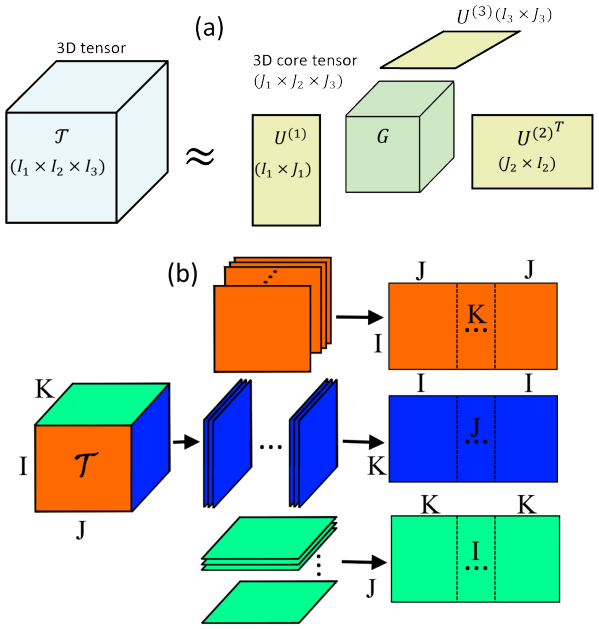

Fig. 1 (a) Illustration of a 3D tensor decomposition. (b) Unfolding of the $(I\times J\times K)$-tensor $\mathscr{T}$ to the $(I\times JK)$-matrix, the $(J\times KI)$-matrix and the $(K\times IJ)$-matrix

The graphic representation of the Higher Order SVD (3D) is shown on the Figu. 1(a);

The n-mode product of a tensor $\mathscr{G}\in\mathbb{R}^{J_1\times J_2\times\ldots\times J_N}$ *by a matrix* $U\in\mathbb{R}^{I_n\times J_n}$ *denoted by* $\mathscr{G}\times_n U$ *is an* $(J_1\times J_2\times\cdots\times J_{n-1}\times I_n\times J_{n+1}\times\cdots\times J_N)$*-tensor of which the entries are given by*

$$(\mathscr{G}\times_n U)_{j_1j_2\cdots j_{n-1}i_nj_{n+1}\cdots j_N} = \sum_{j_n} g_{j_1j_2\cdots j_{n-1}j_nj_{n+1}\cdots j_N}u_{i_nj_n} \tag{5}$$

In HOSVD, all matrices $U^{(n)}$ can be calculated by performing a matrix SVD on the $I_n\times(I_1I_2\cdots I_{n-1}I_{n+1}\cdots I_N)$ matrix obtained by a flattering or unfolding of $\mathscr{T}$ (Bergqvist and Larsson, 2010; De Lathauwer et al., 2000).

The n-mode matricization (or unfolding) of a tensor $\mathscr{T}\in\mathbb{R}^{I_1\times I_2\times\ldots\times I_N}$ *is denoted by* $\mathscr{T}_{(n)}$ *and arranges the n-mode fibers to be columns of the resulting matrix. Tensor element* $(i_1,i_2,\ldots,i_N)$ *maps to matrix element* (i_n,j)*, where*

$$j = 1+\sum_{\substack{k=1\\k\neq n}}^{N}(i_k-1)J_k; \qquad J_k = \begin{cases}\prod_{\substack{m=1\\m\neq n}}^{k-1} I_m, & k>1\\ 0 & otherwise\end{cases} \tag{6}$$

An example of unfolding of the third order tensor $\mathscr{T}$ is shown in Fig. 1(b)

Since $U^{(n)}$ matrices are orthogonal, $\mathscr{G}$ in Eq. 4 is easily calculated using Eq. 7 and it is called the core tensor which

shows the interactions of $U^{(n)}$ matrices – factor matrices (De Lathauwer et al., 2000).

$$\mathscr{G} = \mathscr{T} \times_1 {U^{(1)}}^T \times_2 {U^{(2)}}^T \cdots \times_5 {U^{(5)}}^T \tag{7}$$

4 Proposed method

4.1 Multilinear decomposition and estimation of 3D rotations

In the following, we explain how the tensor decomposition framework described in Sec. 3 can be used to model data variations caused by rotations. Consider a training set composed by N samples $\mathbf{x}_n \in \mathbb{R}^{D_f}$. For instance, in head pose estimation, $\mathbf{x}_n$ refers to a D_f-dimensional vector representing a 3D-descriptor extracted from a mesh. Moreover, we assume that each $\mathbf{x}_n$ is labelled according to its identity plus 3 values defining its corresponding rotation angles (i.e yaw, pitch and roll). By discretizing these values into D_y, D_p and D_r bins respectively, we can represent the whole dataset as a 5-way tensor $\mathscr{T} \in \mathbb{R}^{N_{id} \times D_y \times D_p \times D_r \times D_f}$, where N_{id} is the number of subjects (identities) in the training set.

By using Eq. 4, we can decompose $\mathscr{T}$ as:

$$\mathscr{T} = \mathscr{G} \times U^{(id)} \times U^{(y)} \times U^{(p)} \times U^{(r)} \times U^{(f)} \tag{8}$$

where $\mathscr{G}$ is the core tensor that governs the interaction between the five different factors defining the dataset: identity, rotation in the three angles and appearance (or features) of a sample. Specifically, note that each $U^{(*)}$ is a matrix spanning a subspace for a given factor. Therefore, its rows $\mathbf{u}^{(*)}$ can be seen as vectors representing the data behavior for each parameter of the factor subspace.

For example, the rows of matrix $U^{(id)}$ encode the distinctive characteristics that define the shape of the object $\mathbf{x}_n \in \mathbb{R}^{D_f}$. At the same time, each row in the matrices $U^{(y)}, U^{(p)}$ and $U^{(r)}$ provides coefficients that define the rotation of the object by a particular angle about each axis. And finally, the product by $\mathscr{G}$ and $U^{(f)}$ can be interpreted as a mapping of the shaped and rotated object into feature space.

Thus, from our 5-way tensor, we have 4 modes that correspond to factor subspaces plus another one that corresponds to our input features and is usually combined with the core in the auxiliary variable $\mathscr{W} = \mathscr{G} \times U^{(f)}$. This variable can be understood as a basis that represents the principal axes of variation in feature space across the various factors (the other tensor modes) and how they interact with each other to reconstruct the input features (Vasilescu and Terzopoulos, 2002).

After obtaining the decomposition of the tensor $\mathscr{T}$, a sample $\mathbf{x}$ can be reconstructed as:

$$\mathbf{x} = \mathscr{W} \times \mathbf{u}^{(id)} \times \mathbf{u}^{(y)} \times \mathbf{u}^{(p)} \times \mathbf{u}^{(r)}, \tag{9}$$

where $\mathscr{W} = \mathscr{G} \times U^{(f)}$ and $\{\mathbf{u}^{(y)}, \mathbf{u}^{(p)}, \mathbf{u}^{(r)}\}$ are row vectors from matrices $\{U^{(y)}, U^{(p)}, U^{(r)}\}$. Therefore, it is also theoretically possible to estimate the rotation angles for a given test sample $\mathbf{x} \in \mathbb{R}^{D_f}$ by minimizing its reconstruction error (Tenenbaum and Freeman, 1997; Zhang et al., 2015):

$$\underset{\mathbf{u}^{(y)}, \mathbf{u}^{(p)}, \mathbf{u}^{(r)}, \mathbf{u}^{(id)}}{\operatorname{argmin}} \left\| \mathbf{x} - \mathscr{W} \times \mathbf{u}^{(y)} \times \mathbf{u}^{(p)} \times \mathbf{u}^{(r)} \times \mathbf{u}^{(id)} \right\| \tag{10}$$

Unfortunately, this becomes a minimization problem in which we need to simultaneously solve for 3 viewpoint parameterizations vectors (yaw $\mathbf{u}^{(y)}$, pitch $\mathbf{u}^{(p)}$ and roll $\mathbf{u}^{(r)}$), and the identity vector $\mathbf{u}^{(id)}$. There exist approaches to solve the above minimization, e.g. iterative estimates of one factor at a time or gradient-based optimization (Bakry and Elgammal, 2014). However, they cannot guarantee accurate estimates and the resulting solutions are often not compliant with the manifold structure of the different subspaces. Thus, the final estimates are typically obtained by applying some correction to the results from the minimization of Eq. 10 (e.g. nearest neighbor search (Takallou and Kasaei, 2014)) so that they become compatible with the manifold structure implicitly defined by the training examples.

4.2 Introducing rotation manifold constraints

In Sec. 4.1, we have described the use of tensor decomposition to model the effect of 3D rotations much like any other factor; e.g. notice how rotation factors and identity are treated in an equivalent manner. However, such general treatment does not take advantage of the special structure that can be expected for rotation subspaces leading to the generic formulation in Eq. 10, whose practical use is limited.

Indeed, note that regardless of the resulting dimensionality of the yaw, pitch and roll vectors ($\mathbf{u}^{(y)}$, $\mathbf{u}^{(p)}$ and $\mathbf{u}^{(r)}$), each of them shall theoretically encode only the rotation about only one axis. Hence, the coefficients of each of these vectors are expected to be governed by a single parameter (their corresponding rotation angle), describing a one-dimensional manifold embedded in the higher dimensional ambient space obtained by the tensor decomposition.

To illustrate the above, consider a simple example with a dataset composed of 2D images of different objects rotating with respect to a single axis (e.g, yaw angle). In this scenario, the data depends on three factors: yaw angle, object identity, and features (for which we can directly

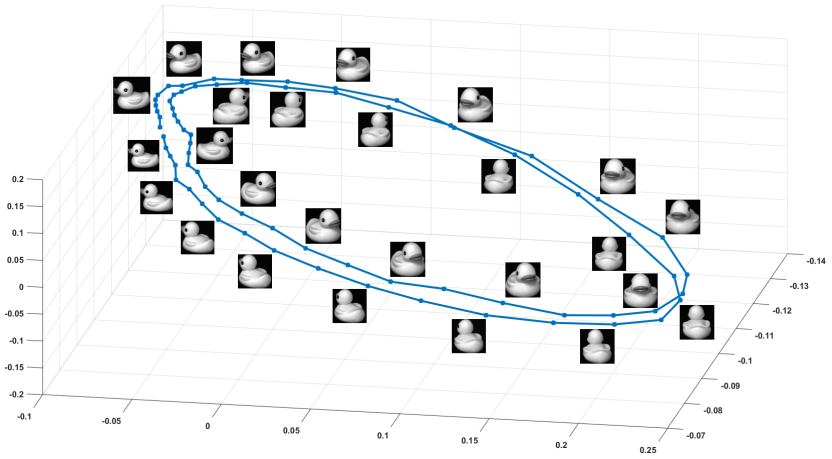

Fig. 2 Visualization of the first three coefficients of the pose variation subspace for a dataset of single object rotated about the vertical axis.

use the pixels). Using this dataset, we can build a tensor and decompose it analogously to Eq. 8. By following this procedure, we obtain three different matrices: $U^{(f)}$, $U^{(y)}$ and $U^{(id)}$, spanning the sub-spaces corresponding to features, rotation and identity, respectively. In Figure 2 we show the values of the first three columns of matrix $U^{(y)}$, corresponding to different images of a "duck" object. We can see that the values displayed in Figure 2 approximately describe a spiral curve, making apparent that the coefficients of the rotation subspace follow a uni-dimensional manifold structure. This is consistent with the fact that the variations captured by this subspace correspond to a single parameter: the rotation angle about the vertical axis. As we will show later in the experiments, this behaviour is not specific for rotations about a single axis but holds for general 3D-rotations and also for different types of features.

Based on this observation, we propose to introduce explicit constraints over the rotation coefficients $\{\mathbf{u}^{(y)}, \mathbf{u}^{(p)}, \mathbf{u}^{(r)}\}$ so that we can carry out their joint estimation directly over the underlying rotation manifold. For this purpose, we re-write Eq. 9 as follows:

$$\begin{aligned}\mathbf{x} &= \mathscr{W} \times \mathbf{u}^{(id)} \times \mathbf{u}^{(y)} \times \mathbf{u}^{(p)} \times \mathbf{u}^{(r)} \\ &= \mathscr{W} \times \mathbf{u}^{(id)} \times \mathbf{f}^{(y)}(\boldsymbol{\omega}^{(y)}) \times \mathbf{f}^{(p)}(\boldsymbol{\omega}^{(p)}) \times \mathbf{f}^{(r)}(\boldsymbol{\omega}^{(r)}) \end{aligned} \tag{11}$$

where $\mathbf{f}^{(*)} : \mathbb{R} \rightarrow \mathbb{R}^{D_*}$ are parametric functions taking as input an angle $\boldsymbol{\omega}^{(*)}$ and giving as output a vector of coefficients $\mathbf{u}^{(*)}$. In this way, we explicitly force the rotation subspaces to be one-dimensional manifolds governed by $\boldsymbol{\omega}^{(*)}$.

By considering the reparametrization defined in Eq. 11, the estimation of the rotation angles given a test sample $\mathbf{x}$ can be obtained by minimizing the following reconstruction error:

$$\begin{aligned}&\underset{\boldsymbol{\omega}^{(y)},\boldsymbol{\omega}^{(p)},\boldsymbol{\omega}^{(r)},\mathbf{u}^{(id)}}{\operatorname{argmin}} \quad \|\mathbf{x} - \hat{\mathbf{x}}\| \\ &\hat{\mathbf{x}} = \mathscr{W} \times \mathbf{u}^{(id)} \times \mathbf{f}^{(y)}(\boldsymbol{\omega}^{(y)}) \times \mathbf{f}^{(p)}(\boldsymbol{\omega}^{(p)}) \times \mathbf{f}^{(r)}(\boldsymbol{\omega}^{(r)})\end{aligned} \tag{12}$$

where $\mathscr{W} = \mathscr{G} \times U^{(f)}$ and the optimized variables are the angles $\boldsymbol{\omega}^*$ and the vector $\mathbf{u}^{(id)}$.

This re-formulation of Eq. 10 allows to minimize the reconstruction error directly fulfilling the manifold structure of the rotation subspaces and reducing the number of coefficients to optimize from $(N_{id} + D_y + D_p + D_r)$ to $(N_{id} + 3)$. As we will show in the experimental results, this offers a crucial advantage with respect to the framework described in Sec. 4.1.

4.3 Constraints definition using trigonometric functions

In the previous section, we have discussed the advantages of incorporating specific constraints into the rotation coefficients $\mathbf{u}^{(*)}$ (see Eq. 9) in order to explicitly model the pose changes of samples $\mathbf{x}$. For this purpose, we have parametrized each $\mathbf{u}^{(*)}$ by using a function $\mathbf{f}^{(*)}$ with a single angle $\boldsymbol{\omega}^{(*)}$ as input. However, we have not discussed yet the specific definition of $\mathbf{f}^{(*)}$ used in this work.

To do so it is important to remember that we are looking for functions that transform a rotation angle into a set of coefficients that produce the desired rotation effect (once appropriately combined by means of $\mathscr{W}$). In this sense we may draw an analogy between the vectors $\mathbf{u}^{(*)} = \mathbf{f}^{(*)}(\boldsymbol{\omega}^{(*)})$ and 3D rotation matrices, which also have multiple coefficients that in reality depend on a single parameter.[1]

[1] Of course this applies to rotations about a single axis; the general 3D-rotation case, depending on 3 parameters, could be anyway decomposed in the product of 3 such rotation matrices, analogous to the product of $\mathbf{f}^{(y)}(\boldsymbol{\omega}^{(y)}) \times \mathbf{f}^{(p)}(\boldsymbol{\omega}^{(p)}) \times \mathbf{f}^{(r)}(\boldsymbol{\omega}^{(r)})$ in Eq. 11.

Therefore, it is reasonable to hypothesize that the relation between the coefficients of the rotation subspaces and their corresponding rotation angles can be modeled by means of trigonometric functions.

Going back to the example from Fig. 2, we can see that the displayed rotation parameters describe an elliptical trajectory, compatible with the above hypothesis. This becomes more evident in Fig. 3, where we show the values of these coefficients separately against the rotation angle. It can be seen that the resulting wave-forms strongly resemble those from the cosine functions. It should be mentioned that, while we only display the first 3 dimensions of the rotation subspace (corresponding to the first 3 columns of matrix $U^{(y)}$), the remaining columns follow a similar pattern. Therefore, we will model $\mathbf{f}^{(*)}$ as vectors of real functions based on cosines parameterized as follows:

$$\mathbf{f}^{(*)}(\omega^{(*)}) = \left(f_1(\omega^{(*)}), f_2(\omega^{(*)}), \dots f_{D_*}(\omega^{(*)})\right) \tag{13}$$

$$f_j(\omega^{(*)}) = \alpha_j^{(*)} cos(\beta_j^{(*)}\omega^{(*)} + \gamma_j^{(*)}) + \varphi_j^{(*)} \tag{14}$$

$$1 \leq j < D_*$$

where $\alpha_j^{(*)}, \beta_j^{(*)}, \gamma_j^{(*)}$ and $\varphi_j^{(*)}$ are parameters defining each specific cosine function. Note that, given a rotation subspace, there will be a different set of parameters for each dimension of the subspace, thus defining a spiral-like structure that analytically represents the underlying manifold.

To obtain the values of the parameters $\alpha_j^{(*)}, \beta_j^{(*)}, \gamma_j^{(*)}$ and $\varphi_j^{(*)}$ that define the analytic curves for each dimension of the rotation subspaces we solve the following minimization:

$$\underset{\alpha_j^{(*)}, \beta_j^{(*)}, \gamma_j^{(*)}, \varphi_j^{(*)}}{\text{argmin}} \quad \|u_{ij}^{(*)} - f_j(\omega_i^{(*)})\| \tag{15}$$

where $u_{ij}^{(*)}$ are the elements of matrices $U^{(y)}, U^{(p)}, U^{(r)}$ obtained from the decomposition of the training tensor and $\omega_i^{(*)}$ is the value corresponding to the i-th bin of the discretized rotation angles used to construct the tensor. For example, given a rotation range of $[-\Theta, \Theta]$ and a uniform discretization:

$$\omega_i^{(*)} = \frac{2i - D_* - 1}{D_* - 1}\Theta \qquad 1 \leq i \leq D_* \tag{16}$$

4.4 Implementation details

Optimization method: Given previous definitions, during learning and inference we need to minimize Eq. 15 and 12, respectively. Given that in both cases the objective function is differentiable with respect to the optimized parameters, we use gradient-descent based optimization. Concretely, we employ the L-BFGS method (Byrd et al., 1994) which is a Quasi-Newton approach. L-BFGS employs the first and second order derivatives of the objective function in order to iteratively update the optimized variables. The key difference between L-BFGS and other Quasi-Newton methods is that the Hessian matrix is approximated with a low-rank compact form, which renders the optimization process much more efficient in terms of space and computational cost. The required first-order derivatives of Eq. 12 and 15 used during optimization are provided in the Appendix. Moreover, note that L-BFGS does not require to explicitly compute the second-order derivatives give that the Hessian matrix is approximated from first-order gradients.

Training: The different steps involved to train the proposed 3D pose estimation framework are summarized in Algorithm 1. We start from a set of feature vectors $\{X\}$, each labeled with an identity and its yaw, pitch and roll angles. Depending on the input data, these angles may need to be discretized so that we obtain angular bins that are consistent across all identities. This allows to organize the input data into the 5D tensor $\mathscr{T}$, which is decomposed to obtain the core tensor $\mathscr{G}$ and the subspace matrices $U^{(*)}$ for each of the 5 factors in the tensor.

Testing: The feature-subspace matrix $U^{(f)}$, representing the input space, is combined with the core to form the auxiliary tensor $\mathscr{W}$; the identity-subspace matrix $U^{(id)}$ is kept unchanged and the rotation-subspace matrices $U^{(y)}$, $U^{(p)}$ and $U^{(r)}$ are reparameterized in terms of cosine functions. Note that, for each rotation angle, there are as many cosine functions as dimensions in the yaw, pitch and roll subspaces, but all the functions in a given subspace are governed by the same unique free variable, respectively $\omega^{(y)}, \omega^{(p)}, \omega^{(r)}$.

During testing, the unknown identity and rotation angles of a feature vector $\mathbf{x}$ are estimated (Algorithm 2). This is done by solving the minimization in Eq. 12 using the auxiliary tensor $\mathscr{W}$ and the parameters learned during training for each function $\mathbf{f}^{(*)}$. Note that, because the factor subspaces are obtained by means of HOSVD, their parameters are sorted in terms of their eigenvalue. Therefore, even though each angular subspace results in $D_* - 1$ cosine functions, those with the smallest eigenvalues are typically discarded due to their sensitivity to noise.

5 Experiments with a single rotation axis

In order to show that the proposed framework can be applied to a wide variety of input features, we perform several experiments. The first experiment is performed on the COIL-20 database (Nene et al., 1996), where we simply use the pixels of 2D images as features and use our framework to impose analytic constraints to out-of/-plane rotations of a variety of objects. Further, in Section 6, we perform 3D head pose estimation experiments using

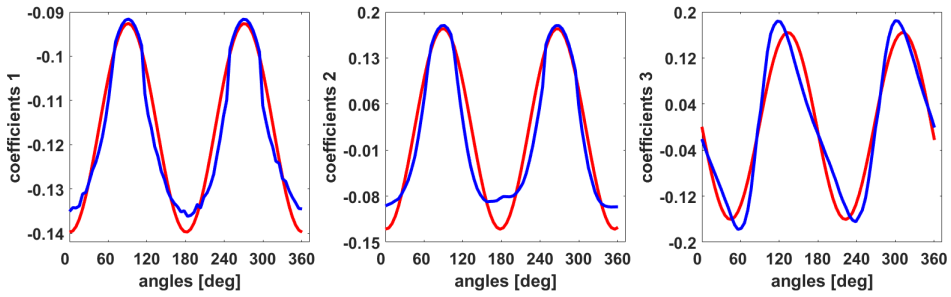

Fig. 3 Values of the first three coefficients of the viewpoint subspace for the example from Figure 2. The blue curves show the actual values of the first three columns of matrix $U^{(y)}$ and the red curves show their least-squares approximation with cosine functions

Algorithm 1: Training Phase

Input : $\{X\}$ – set of feature vectors for each of the subjects and each of the rotation angle labeled with $\omega^{(y)}, \omega^{(p)}, \omega^{(r)}$;
$D^{(*)}$ – number of the bins to discretize rotation;
Output: $\mathscr{W}$ and set of parameters $\alpha^{(*)}, \beta^{(*)}, \gamma^{(*)}, \varphi^{(*)}$ for each of the rotation;

Build 5D tensor $\mathscr{T} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_5}$
Decompose $\mathscr{T}$ using HOSVD (Eq. 8)
  $\mathscr{T} = \mathscr{G} \times_1 U^{(id)} \times_2 U^{(y)} \times_3 U^{(p)} \times_4 U^{(r)} \times_5 U^{(f)}$
Compute $\mathscr{W}$:
  $\mathscr{W} = \mathscr{G} \times_5 U^{(f)}$
foreach $\omega^{(*)} \in \{\omega^{(y)}, \omega^{(p)}, \omega^{(r)}\}$ **do**
  foreach *j-th column in matrix* $U^{(*)}$; $(1 \leq j < D_*)$ **do**
    foreach *i-th bin of the discretized rotation angles* $(1 \leq i \leq D_*)$ **do**
      $\text{argmin}_{\alpha_j^{(*)}, \beta_j^{(*)}, \gamma_j^{(*)}, \varphi_j^{(*)}} \| u_{ij}^{(*)} - f_j(\omega_i^{(*)}) \|$;
      where $f_j(\omega_i^{(*)})$ is from Eq. 14;
    end
  end
end

Algorithm 2: Test Phase

Input : $\mathbf{x}$ – feature vector of unknown subject
Output: estimated angles – $\omega^{(y)}, \omega^{(p)}, \omega^{(r)}$

Initialize :
  $\omega^{(y)} = 0; \omega^{(p)} = 0; \omega^{(r)} = 0;$
  $\mathbf{u}^{(id)}$ as vector with zeros;
Define functions using Eq. 14:
foreach *j-th column in matrix* $U^{(*)}$; $(1 \leq j < D_*)$ **do**
  $f_j^{(y)}(\omega^{(y)}) = \alpha_j^{(y)} cos(\beta_j^{(y)} \omega^{(y)} + \gamma_j^{(y)}) + \varphi_j^{(y)}$
  $f_j^{(p)}(\omega^{(p)}) = \alpha_j^{(p)} cos(\beta_j^{(p)} \omega^{(p)} + \gamma_j^{(p)}) + \varphi_j^{(p)}$
  $f_j^{(r)}(\omega^{(r)}) = \alpha_j^{(r)} cos(\beta_j^{(r)} \omega^{(r)} + \gamma_j^{(r)}) + \varphi_j^{(r)}$
end
Estimate angles $\omega^{(y)}, \omega^{(p)}, \omega^{(r)}$:
$\text{argmin}_{\omega^{(y)}, \omega^{(p)}, \omega^{(r)}, \mathbf{u}^{(id)}} \| \mathbf{x} - \hat{\mathbf{x}} \|$
where $\hat{\mathbf{x}} = \mathscr{W} \times \mathbf{f}^{(y)}(\omega^{(y)}) \times \mathbf{f}^{(p)}(\omega^{(p)}) \times \mathbf{f}^{(r)}(\omega^{(r)}) \times \mathbf{u}^{(id)}$

other two types of features: automatic landmarks and histogram-based 3D descriptors.

Fig. 4 Sample images for the 20 subjects in COIL-20 dataset.

5.1 Image rotation manifold

We start our experiments using 2D images that capture rotations of simple objects along only one axis: the vertical axis (yaw angle). We consider the Columbia University Image Library (COIL-20) data-set (Nene et al., 1996). The COIL-20 is an often-used dataset that contains a total of 1440 grayscale images from 20 different objects. Each image, of size 128 × 128 pixels, shows one of the objects at a particular rotation angle over a black background. There are 72 images for each object, taken at intervals of approximately 5°, thus covering the full range of rotations between 0 and 360 degrees. As the only pre-processing step, each image was downsampled to 32 × 32 pixels and these were concatenated in row vectors of 1024 values which constituted our input features. Figure 4 shows some sample images of this dataset.

Following Algorithm 1, the input data was arranged in a tensor. Due to the fact that images have only rotation about one axis (yaw), we built a 3-D tensor of size 20 × 72 × 1024 ($\mathscr{T} \in \mathbb{R}^{20 \times 72 \times 1024}$). Then we applied tensor decomposition to obtain the core tensor and coefficients for each of the data factors subspaces. We can consider this a special case of Eq. 8 that reduces to:

$$\mathscr{T} = \mathscr{G} \times U^{(id)} \times U^{(y)} \times U^{(f)} \tag{17}$$

where $U^{(id)}$, $U^{(y)}$ and $U^{(f)}$ span the identity, rotation and feature subspaces, respectively. In particular, we are interested in $U^{(y)} \in \mathbb{R}^{72 \times 72}$; each element $u_{ij}^{(y)}$ contains

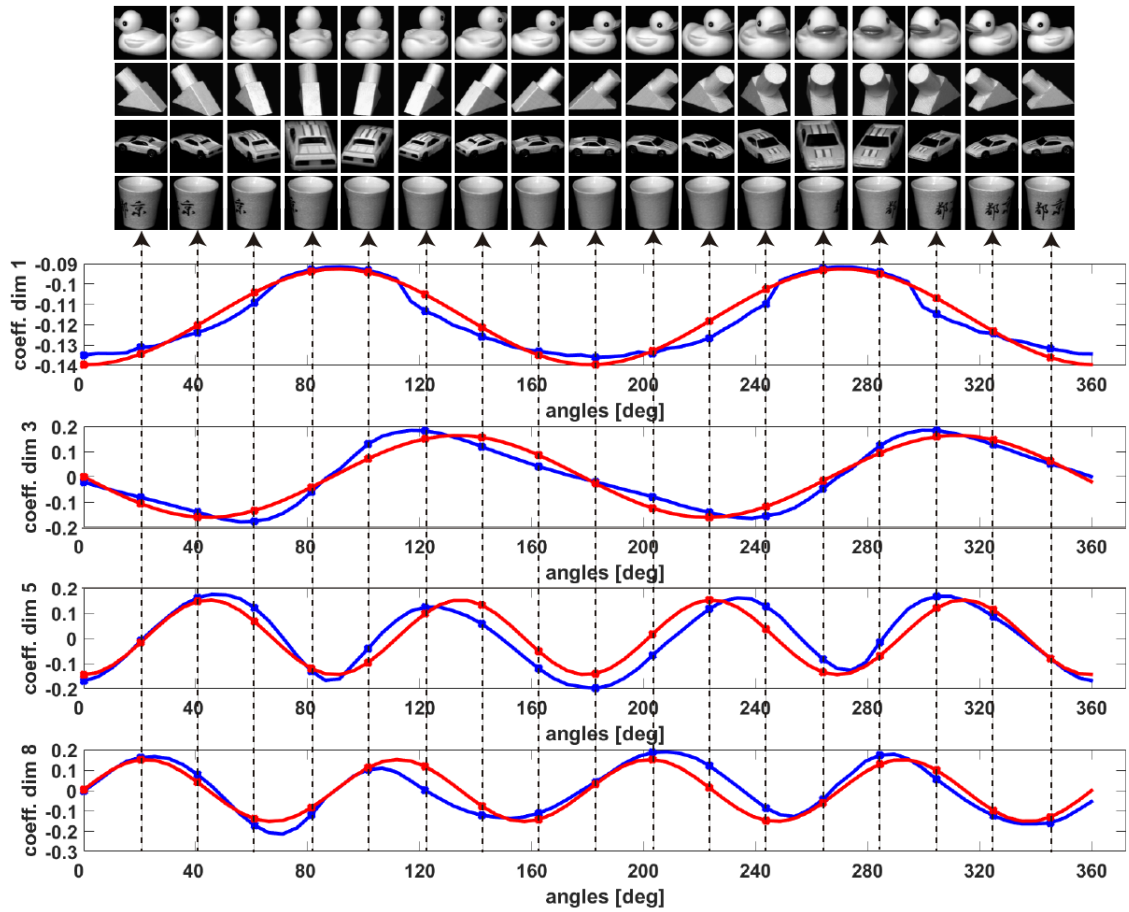

Fig. 5 Coefficients from the first, third, fifth and eighth dimensions of the viewpoint subspace for the entire range of angles. On the top of the figure we also show sample images of some objects with rotation angles in correspondence to those displayed by the coefficient curves (at the bottom). Blue curves show the actual values of matrix $U^{(y)}$ obtained by tensor decomposition while the red curves show their least-squares approximation with cosine functions.

the coefficient of the j-th subspace dimension for the i-th rotation angle. Thus, each column $U^{(y)}$ shows the behaviour of a particular dimension of the rotation subspace when the yaw angle varies and, as hypothesized in Section 4.3, it should approximately generate the waveform of a trigonometric function. Figure 5 shows the coefficients of a few columns of matrix $U^{(y)}$ (in blue color) together with their cosine-based approximations (in red). The latter were computed by minimizing Eq. 14 for each column of matrix $U^{(y)}$, yielding an analytic representation $\mathbf{f}^{(y)}(\omega^{(y)})$ of the underlying manifold structure of the rotation subspace, as defined in Eq. 14.

We can observe in Figure 5 that the curves described by the coefficients of $U^{(y)}$ are indeed quite similar to cosine functions. On the other hand, we also see that the cosine-based approximations do not produce a perfect fit of the curves. This can be reasonably explained by the fact that, apart from rotations, the COIL-20 dataset includes also rather important variations in size, which for our settings can be considered spurious effects. Nevertheless, taking into account that we are trying to model out-of-plane rotations with 2D pictures using directly their pixels as input features, the curves in Figure 5 comply strikingly well with the hypothesized behavior.

5.2 Synthesizing rotations

After rotation coefficients have been modeled, another way to assess the behavior of the proposed framework is by synthesizing multiple views of an object from which only one specific rotation angle has been observed.[2] Specifically, given an input object (represented by feature vector $\mathbf{x}$) whose rotation is $\omega_0^{(y)}$, we can obtain its identity vector

[2] This process is coined *translation* in the seminal work by Tenenbaum and Freeman (2000).

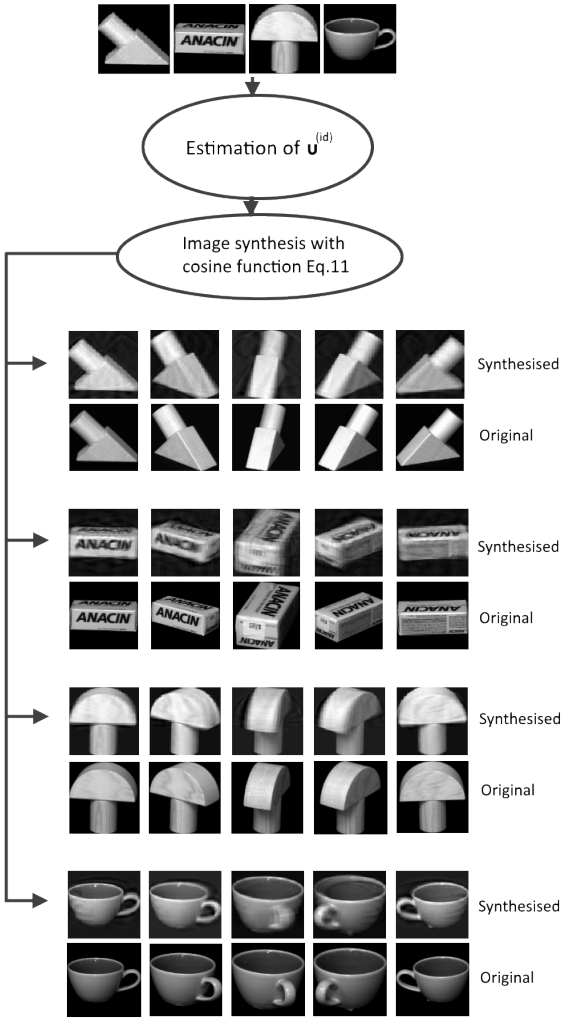

Fig. 6 Image obtained by means of the synthesis procedure (Eq. 11) for a few example objects. The top row shows one image for each of 4 sample objects from the COIL-20 database. These images are used to estimate the identity vectors $\mathbf{u}^{(id)}$ for each object, which are combined with trigonometric function $\mathbf{f}^{(y)}$ in order to synthesize images at different rotation angles. In the lower part of the image we see the synthesized images (top lines) and the corresponding original images in the dataset (bottom lines) for each of the objects.

$\mathbf{u}^{(id)}$ and combine it with $\mathbf{f}^{(y)}$ to synthesize unknown views of the same object according to different rotation angles $\omega^{(y)} \neq \omega_0^{(y)}$ using Eq. 11. Figure 6 illustrates the images obtained by means of this procedure for a few objects, as well as the corresponding actual images from the database (for each experiment, the input tensor was re-build by removing all the images at the rotation angle to be synthesized). We can see, again, that the quality of the synthesized images is not perfect and we can identify some minor artifacts not present in the original images; yet, in all cases we can clearly identify the object as well as the specific rotation angle that was synthesized, indicating that $\mathbf{f}^{(y)}$ has successfully captured the effect of the rotation angle in this dataset.

To quantify the impact of introducing the trigonometric constraints in the synthesis of unseen rotations, we reserved $\sim 10\%$ of the data (the first 7 rotated views[3] of all objects)to be used as test-set and re-built tensor $\mathscr{T}$ with the remaining data (65 views $\times$ 20 objects). The new tensor was decomposed and the resulting rotation subspace was approximated by estimating $\mathbf{f}^{(y)}(\omega^{(y)})$. Then, we set up a *pose transfer experiment*, which consists in using an image from the test set as *source* from which we wish to synthesize images with the same pose for all other objects in the database and compare the result to the actual images captured with that pose.

Let image $\mathbf{x}^{(id_s,\omega_s)}$ from the test set be the *source* with identity id_s and pose ω_s (both of which we assume unknown). We start by estimating its rotation $\hat{\omega}_s^{(y)}$ using the subspaces spanned by the training tensor using a simplified version of Eq. 12 (since there are no pitch and roll effects):

$$(\hat{\omega}_s^{(y)}, \hat{\mathbf{u}}_s^{(id)}) = \underset{\omega^{(y)},\mathbf{u}^{(id)}}{\operatorname{argmin}} \|\mathbf{x}^{(id_s,\omega_s)} - \mathscr{W} \times \mathbf{u}^{(id)} \times \mathbf{f}^{(y)}(\omega^{(y)})\| \tag{18}$$

Then, we can use any other object form the training set as the *target* identity $id_t \neq id_s$, for which we wish to synthesize a new image $\hat{\mathbf{x}}^{(id_t,\omega_s)}$ with the same pose as $\mathbf{x}^{(id_s,\omega_s)}$, as follows:

$$\hat{\mathbf{x}}^{(id_t,\omega_s)} = \mathscr{W} \times \mathbf{u}_t^{(id)} \times \mathbf{f}^{(y)}(\hat{\omega}_s^{(y)}) \tag{19}$$

where $\mathbf{u}_t^{(id)}$ is the identity vector of the target object and is available from the training set. Once the synthesized image $\hat{\mathbf{x}}^{(id_t,\omega_s)}$ is computed, we can evaluate its quality by comparing it with the actual $\mathbf{x}^{(id_t,\omega_s)}$, which is part of the test set. We do so by computing the peak signal to noise ratio (PSNR), taking the difference between the original and synthesized images as *noise*:

$$PSNR = 10 \log_{10} \left(\frac{I_{MAX}^2}{MSE(\hat{\mathbf{x}}^{(id_t,\omega_s)} - \mathbf{x}^{(id_t,\omega_s)})} \right) \tag{20}$$

where $I_{MAX} = 255$ for 8-bit images and MSE is the mean squared error between the actual and synthesized images.

We repeated the above procedure taking each of the 7 $\times$ 20 test images as *source* and transferred their pose to each of the other 19 objects in the dataset, resulting in a total of 7 $\times$ 20 $\times$ 19 = 2,660 comparisons. The average PSNR of the

[3] These samples cover approximately a viewpoint range from 5 to 35 degrees.

synthesized images with respect to the originals was of 18.8 dB ($\pm$ 4.37 dB standard deviation).

To provide a baseline for the above numbers, we repeated the same experiment without the trigonometric constraints. This implies using Eq. 10 instead of Eq. 12, i.e. we estimate directly the coefficients of the rotation subspace $\mathbf{u}^{(y)}$ without constraining them to follow the manifold structure imposed by $\mathbf{f}^{(y)}(\boldsymbol{\omega}^{(y)})$. The resulting PSNR was slightly lower (16.1 $\pm$ 3.52 dB), which suggests that the imperfect fit of the trigonometric approximation (e.g. Fig. 5) is overcome by the benefits of the manifold structure that allows obtaining more accurate estimates of the subspace parameters. In the next section we will see that, when working with 3D data and multiple rotation axes, the difference between the constrained and unconstrained solutions becomes considerably higher.

Finally, note that our framework allows generating synthetic views by directly specifying the desired angle, without the need to estimate the pose subspace parameters from an actual *source* image as we did in the above experiment. However, this would not be possible in the unconstrained case, in which the coefficients $\mathbf{u}^{(y)}$ that correspond to the rotation subspace are not modeled analytically. This led us to design our experiment in terms of *pose transfer*, to allow for a fair comparison of the constrained and unconstrained cases.

6 Experiments on 3D head pose estimation

In this section we demonstrate the application of the proposed framework to the problem of 3D head pose estimation from depth data. In contrast to the tests from Section 5.1, experiments of the present section imply full deployment of our framework, given that we address datasets where head rotations are not constrained to a single axis but contain combinations of yaw, pitch and roll angles at the same time.

Given that our framework focuses on analytically modeling the underlying structure of the rotation manifold in general, it can be applied to a wide variety of input features. Nevertheless, the appropriate selection of input features is important to achieve quantitative results that demonstrate the relevance and applicability of the proposed method. Thus, we have selected the two main features from the system presented in (Derkach et al., 2017), which obtained the first place in the recent Head Pose Estimation Challenge organized on the SASE database (Lüsi et al., 2017). These features are composed by: 1) automatically detected landmarks and 2) histogram-based descriptors extracted around the landmarks. For completeness, we briefly review these features in Section 6.1 and provide pointers to external sources for more detailed information. Then, in Sections 6.2 and 6.4, we report experiments over two large and publicly available 3D face corpora: the SASE (Lüsi et al., 2016b) and BIWI databases (Fanelli et al., 2013).

6.1 Feature extraction

Landmarks: we use Shape Regression with Incomplete Local Features (SRILF) (Sukno et al., 2015) to locate the following 12 facial landmarks: inner and outer eye corners, nose corners, mouth corners, nose root, nose tip and chin tip. The SRILF algorithm combines the response from local feature detectors for each of the targeted landmarks with statistical constraints that ensure the plausibility of landmark positions on a global basis. An important aspect of this algorithm is that it integrates combinatorial search with partial set matching to infer possibly missing landmarks; which inherently provides tolerance to distorted or missing data (occlusions). This is an advantage in applications such as head-pose estimation with sensors like Kinect, which capture depth information from a single view. Under large head rotations, the generated depth maps will have large parts of the face missing due to self-occlusions and it is crucial to be able to exploit partial information (Fig. 7).

Local appearance descriptors: Once the facial landmarks are available, we use their location in order to extract the second type of features, namely local surface descriptors around landmark points. We use 3D Shape Contexts (3DSC) (Frome et al., 2004) as local descriptors, slightly modified to increase their sensitivity to viewpoint and robustness to noise. 3DSC are based on a spherical histogram computed on a neighbourhood of the interest point, *i.e* landmark locations, and have been shown to perform well as descriptors of the facial surface (Sukno et al., 2012). Similarly to other popular descriptors representing 3D geometry (Johnson and Hebert, 1999; Tombari et al., 2010; Rusu et al., 2009), 3DSC uses the surface normal at the interest point to appropriately orient the reference system

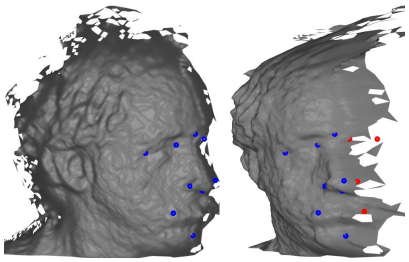

Fig. 7 Positions of the landmarks estimated automatically by SRILF in a head scan showing large yaw rotation. Two views of the same scan are provided: the original view (as seen from the camera) is shown to the left and a rotated view (to simulate a *frontal shot*) is shown to the right. Landmarks lying on the surface are indicated in blue color, while those off-the-surface (estimated by inference) are displayed in red.

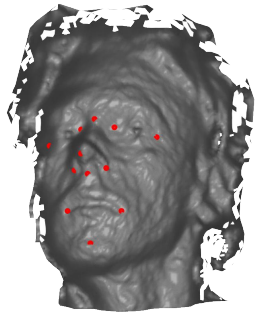

Fig. 8 Example of the 3D mesh of a face with automatically obtained landmarks

of the local neighbourhood, aiming for rotational invariance [4]. Because our objective is to identify viewpoint, such normal-based orientation is not convenient, hence we will orient the reference systems of all local neighbourhoods based on the normal to the camera sensor. This choice avoids also the computation of surface normals, which are known to be especially sensitive to noise (Papazov et al., 2015; Tombari et al., 2010).

6.2 3D head pose estimation using SASE database

The data in SASE has been acquired with Microsoft Kinect 2 camera and contains RGB and depth images in pairs. The entire database includes 50 subjects (32 male and 18 female) in the range of 7-35 years old, with more than 600 frames per subject. For each person, a wide range of yaw, pitch and roll variations are included. Specifically, yaw and pitch angles vary within ± 75 °, while roll angles vary within ± 45 ° (Lüsi et al., 2016a).

The SASE database is divided in two sets: *Training* (comprising 28 subjects with a total of $\sim$ 17K images) and *Validation* (12 subjects, $\sim$ 7K images) (Derkach et al., 2017). Thereby, we have used each of these sets for training and testing, respectively. As mentioned before, we base our tests on the system described in (Derkach et al., 2017), which is used as baseline. Therefore, we use two types of features: automatically detected landmarks and local appearance around landmark points.

Following (Derkach et al., 2017), we start by isolating the head region using clustering and using the obtained result to build a 3D mesh $\mathscr{M}$ that contains the head and part of the shoulders. Then, mesh $\mathscr{M}$ is fed to the SRILF algorithm with the aim to automatically detect 12 prominent facial landmarks. An example of the 3D mesh of the face with the obtained landmarks is illustrated on Figure 8. Once the facial landmarks are available, we use their coordinates as input features to train and test our approach as described in Section 4. It is worth to mention that the use of SRILF to extract the input features provides robustness to both expression changes and missing parts. The latter is especially important in databases such as SASE and BIWI, because large pose variations induce self-occlusions that are likely to affect the visibility of some landmarks. SRILF deals with this problem by statistically inferring missing landmarks, thus providing a complete set of landmark coordinates even under occlusions.

6.2.1 *Pose estimation from landmarks*

During the training phase, we follow Algorithm 1 in order to build a 5D tensor $\mathscr{T} \in \mathbb{R}^{28\times40\times40\times30\times36}$ defined by: 28 subjects, 40 bins discretizing yaw and pitch angles in the range of $[-75°..75°]$, 30 bins for roll in the range of $[-55°..55°]$, and 36-dimensional features (12 landmarks $\times$ 3 coordinates, centered at the nose tip to remove any translation effect). In order to fill all cells in this 5D tensor we need around 1.3 million samples, and it is obvious that the SASE database does not have this amount; it provides only about 5% of them. Thus, $\sim$ 95% of the training data had to be generated synthetically. Specifically, if there is not a sample with i-th identity and target angles yaw $\omega^{(y)}$, pitch $\omega^{(p)}$ and roll $\omega^{(r)}$, we look for the closest sample with the same identity i from the training set and rotate it to the target angles (the amount of rotation is easily computed as the difference between the target angles and the ground-truth angles from the selected sample).

After the tensor is built, we decompose it using Eq. 8, and perform dimensionality reduction, obtaining the core tensor $\mathscr{G} \in \mathbb{R}^{28\times3\times3\times3\times10}$, matrix $U^{(id)} \in \mathbb{R}^{28\times28}$ for the identity subspace, matrices $U^{(y)} \in \mathbb{R}^{40\times3}$, $U^{(p)} \in \mathbb{R}^{40\times3}$ and $U^{(r)} \in \mathbb{R}^{30\times3}$ for yaw, pitch and roll subspaces, and matrix $U^{(F)} \in \mathbb{R}^{36\times10}$ for the features subspace.

Next, we fit cosine functions to the pose coefficients (Eq. 14) and obtain four parameters $(\alpha_j^{(*)}, \beta_j^{(*)}, \gamma_j^{(*)}$ and $\varphi_j^{(*)})$ for each dimension of the three rotation subspaces (yaw, pitch and roll), thus achieving an analytic representation of the structure of the rotation manifolds. The results of the approximated coefficients for 3D pose variations are illustrated in Figure 9. For each of the rotation subspaces (yaw, pitch and roll), the first, second and third coefficients of all angle variations are plotted with two colors. The blue curves are the original values from the first three columns of the matrices $U^{(y)}, U^{(p)}$ and $U^{(r)}$ and the red curves are the approximated values obtained with cosine functions. It can be observed that the trigonometric approximation provides an excellent fit for the three rotation angles, with only minor deviations that could be easily attributed to noise in the data or in the extracted features.

[4] Such invariance, however, is only partially achieved in 3DSC since the orientation of the surface normal still leaves one degree of freedom undefined (the sphere's azimuth) (Sukno et al., 2013)

Based on the obtained coefficients and similarly to the case with 2D images, we can synthetically generate sets of landmarks by sampling our analytic manifold. For example, given a particular subject and target angles $\omega^{(y)}$, $\omega^{(p)}$ and $\omega^{(r)}$, we can synthesize the corresponding set of landmarks using Eq. 11. If this procedure is repeated while varying one of the angles, we can get a graphical illustration of how the effect of this angle has been captured by our framework. Figure 10 shows an example in which identity, pitch and roll angles are fixed, while yaw varies from $-75°$ to $75°$.

After all function parameters are obtained, we use the estimation approach based on the minimization of the reconstruction error (Eq. 12) to test all images in the Validation subset of SASE database ($\sim$ 7K facial images). We compared the obtained results of the proposed framework with respect to the standard approach based on minimizing the reconstruction error without constraints and then *correcting* the solution to lie on the manifold[5]. implicitly defined by the training samples (which we refer to as *Implicit constraints*).

Table 1 summarizes the average pose estimation errors obtained by each approach. It can be seen that the straight-forward application of multi-linear decomposition for head pose estimation yields quite large estimation errors. Nevertheless, by introducing our explicit manifold-compliant constraints, there is a drastic reduction of the estimation errors, which become competitive with state of the art methods (see below).

Table 1 Average pose estimation errors tested on the SASE database using landmark estimates as features

	Yaw	Pitch	Roll	Average
Implicit constraints	12.18	13.51	10.38	12.02
Explicit constraints (proposed)	**6.50**	**7.07**	**6.06**	**6.54**

It is important to note that both methods compared in Table 1 attempt to take into account the fact that rotations define a manifold. The key difference is that in our formulation we incorporate the constraints directly into the minimization (e.g. Eq. 12). In contrast, in the case of using implicit constraints, the minimization is firstly performed without constraints (as in Eq. 10) and compliance to the manifold is enforced only afterwards (the final solution is obtained by iteratively alternating between these two steps). Thus, even though in practice the constraints from both approaches shall define fairly similar manifolds, the unconstrained minimization from Eq. 10 is performed on a space of relatively much higher dimension than the manifold, hence leading to poor performance.

[5] The results obtained from the minimization are forced to comply with the rotation manifold after each iteration using nearest-neighbour search. Results without such *correction* would be worse than those reported and not meaningful for comparison, since this is a widespread practice. Notice that no constraints are applied to the identity subspace in any of the experiments in this paper.

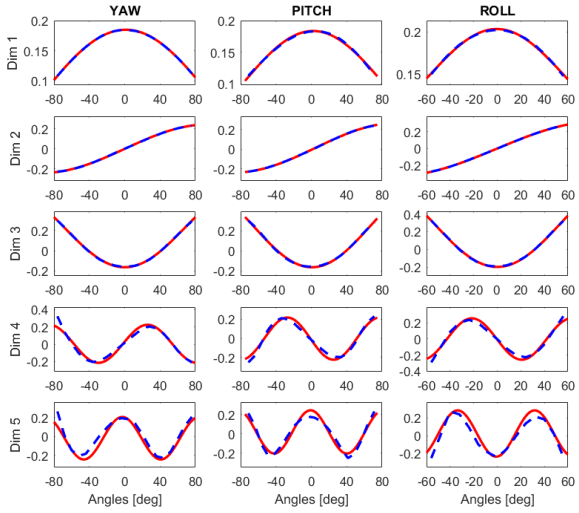

Fig. 9 Curves defined by the coefficients in each of the subspaces corresponding to the head pose variation along one of the rotation axes using landmark extimation as a features. The first column corresponds to yaw rotation and shows the curves built from the coefficients of the first 5 columns of matrix $U^{(y)}$ (blue) and their approximation with a cosine function (red). The second and third columns correspond to pitch and roll angles, respectively

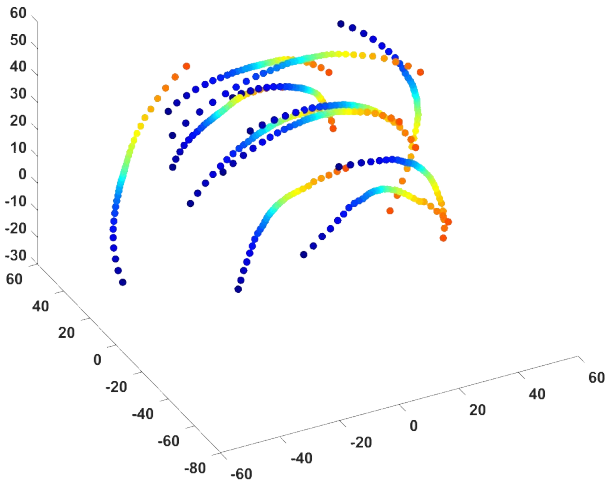

Fig. 10 Example of generated landmarks using trigonometric functions (Eq. 11). Landmark coordinates change with rotation about the vertical axis (yaw angle), i.e. identity, pitch and roll angles are fixed, and the yaw angle varies from $-75°$ to $75°$. Thus, according to the position of the landmarks, we can see how the face progressively rotates from left (orange) to right (blue).

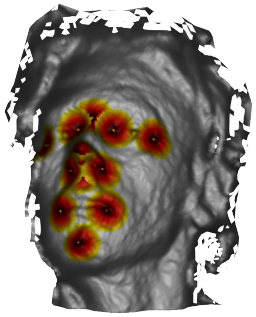

Fig. 11 Illustration of a 3D mesh of the face with the neighbourhoods used to compute local descriptors.

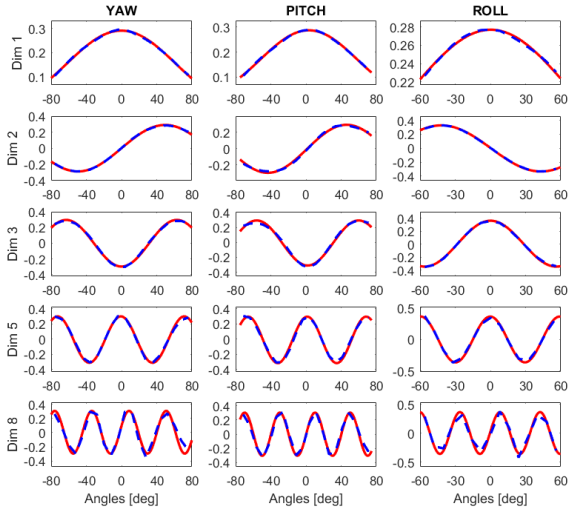

Fig. 12 Curves defined by the coefficients in each of the subspaces corresponding to head pose variation along one of the rotation axes using local descriptors as features. The first column corresponds to yaw rotation and shows the curves built from the coefficients of 5 columns of matrix $U^{(y)}$ (dashed blue lines) and their approximation with a cosine function (red solid lines). The second and third columns correspond to pitch and roll angles, respectively.

6.2.2 Pose estimation from local surface appearance

In this section we perform experiments with the local surface descriptors presented in Section 6.1 as input features. An interesting aspect of using these descriptors is that, because they are based on spatial histograms of local patches from the surface, rotations will have a non-linear effect on the descriptor values. An illustration of a 3D mesh of the face with the local descriptors is provided in Figure 11.

Similarly to the experiment with landmarks from the previous section, for the training phase we firstly build a 5D tensor $\mathcal{T} \in \mathbb{R}^{28\times40\times40\times30\times512}$, i.e. now the features dimension is 512. Secondly, we decompose it using Eq. 8. Thirdly, we estimate the coefficients for each of the factor subspaces to obtain an analytic representation of the structure of the rotation manifolds by fitting cosine functions (Eq. 14). The results of the approximated coefficients for 3D pose variations are illustrated in Figure 12. Finally, we use the modeled coefficients to estimate the 3D head pose based on the minimization of the reconstruction error (Eq. 12).

Note that, following (Derkach et al., 2017), we build 12 different tensors and produce different estimates for each angle (i.e. one per landmark descriptor). However, because of the potential presence of occlusions, it is not guaranteed that all estimated landmarks will actually lie on the mesh surface.[6] Indeed, when parts of the facial surface are missing, it is possible that some landmarks are estimated relatively far from the mesh $\mathscr{M}$, i.e. they are inferred in the position where we would statistically expect them to be, despite no surface has been captured there (Fig.7).

Therefore, we use the indicator function $\mathbb{1}(\|\hat{\mathbf{x}}_\ell - \mathscr{M}\| < \varepsilon)$ to filter out the estimates from landmarks $\hat{\mathbf{x}}_\ell$ that are estimated off the surface and produce our final appearance-based estimate as the average of the remaining ones:

$$\omega^{(*)} = \frac{\sum_\ell \mathbb{1}\left(\|\hat{\mathbf{x}}_\ell - \mathscr{M}\| < \varepsilon\right)\omega_\ell^{(*)}}{\sum_\ell \mathbb{1}\left(\|\hat{\mathbf{x}}_\ell - \mathscr{M}\| < \varepsilon\right)} \tag{21}$$

where $\omega_\ell^{(*)}$ are the estimated angles from the ℓ-th landmark descriptor; the distance from $\hat{\mathbf{x}}_\ell$ to $\mathscr{M}$ is computed as the distance to the nearest mesh vertex:

$$\|\hat{\mathbf{x}}_\ell - \mathscr{M}\| = \min_{v_j \in \mathscr{M}} \|\hat{\mathbf{x}}_\ell - v_j\| \tag{22}$$

Table 2 Average pose estimation errors of the proposed framework and previous works on the SASE database

	Yaw	Pitch	Roll	Average
Lüsi et al. (2016a)	22	19	18	19.67
Derkach et al. (2017)	6.51	7.49	6.52	6.84
Proposed LMK	**6.50**	**7.07**	**6.06**	**6.54**
Proposed DESC	**6.21**	**6.64**	**4.6**	**5.82**

Table 2 summarizes the average pose estimation errors of the proposed framework using both landmarks and appearance features on the SASE database. In this table, we also compare our results to other methods reporting head pose estimation error on the SASE database. Since this database is rather new, only a few papers have reported results on it. We can see that the proposed method outperforms state-of-the-art methods on the same dataset.

In order to give more insights about the effectiveness of the proposed framework, in Table 3 we compare our

[6] We consider that a landmark is *on the surface* when its distance to it is relatively small as compared to the mesh resolution.

Table 3 Pose estimation errors (mean ± standard deviation) of the proposed method against regression and DNNs tested using the same surface descriptors as input.

	Yaw	Pitch	Roll	Average
Regression	6.95 ± 8.17	7.97 ± 10.3	6.22 ± 7.99	7.05 ± 8.88
DNN	6.04 ± 7.62	6.84 ± 10.1	6.44 ± 8.21	6.44 ± 8.71
Proposed	**6.21 ± 7.77**	**6.64 ± 10.0**	**4.60 ± 6.82**	**5.82 ± 8.30**

method based on trigonometric constraints to the results obtained by using linear regression and Deep Neural Networks (DNNs) with the same input features and under the same strategy of producing independent estimates based on each landmark descriptor and combining them by means of eq. 21. We selected the appearance-based features for comparison, as they produced more accurate results than landmarks. Regression experiments were performed using ridge regression with cross-validation to tune the regularization parameter. The DNN experiments were based on the recent architecture from (Borghi et al., 2017), addressed specifically to head pose estimation. Because in our case the features are already available, we used only the last layers of their proposed architecture. Concretely, the used network is composed by three fully-connected layers with 128, 84, and 3 channels, where the two first layers use a *tanh* activation function and the last layer outputs the predictions for the three different angles. As in (Borghi et al., 2017), the network parameters have been optimized by minimizing the L2 loss and using Stochastic Gradient Descent (SGD). The results in Table 3 show that our method compares favorably to the mentioned alternatives. It should be emphasized that the objective of Table 3 is to provide a comparison to other popular techniques under the same input features while removing the impact of the latter. However, DNN approaches are known to benefit from the ability to determine their own features, which is not possible under these experimental settings.

6.3 Discretization, scalability and complexity

The good performance reported in the previous section comes at the price of requiring training data for all angle combinations, e.g., if we discretize yaw, pitch and roll into D_y, D_p and D_r bins, then we would ideally need $D_y \times D_p \times D_r$ training images for each object/person to perform the HOSVD. While there exist alternative ways to decompose the tensor that tolerate missing data (Xu et al., 2015; Zhao et al., 2015), it is worth noting that there are other two aspects that can alleviate this issue in a simpler manner:

1. The fact of working with 3D data allows (under certain conditions[7]) to synthesize *virtual* samples by rotating existing ones to the desired (yaw, pitch, roll) combination. For each bin (i,j,k) of the tensor, with center at $\mathbf{w}_{ijk} = (\omega_i^{(y)}, \omega_j^{(p)}, \omega_k^{(r)})$ (see Eq. 16), we select the closest unused sample $\mathscr{M}$ with ground-truth angles $\mathbf{w}_M = (\omega_M^{(y)}, \omega_M^{(p)}, \omega_M^{(r)})$ and rotate it by $\mathbf{w}_{ijk} - \mathbf{w}_M$ to obtain the synthesized sample $\hat{\mathscr{M}}$ which is now at the desired angles. Input features can then be computed on $\hat{\mathscr{M}}$ and used to fill bin (i,j,k) of tensor $\mathscr{T}$.[8]
2. Because we model each dimension of the rotation manifold by means of cosine functions, we may estimate these accurately even when using smaller numbers of bins.

Based on the above, we repeated the experiments from the previous section while reducing the number of bins used to discretize the rotation angles. We started from the discretization used in the previous section ($40 \times 40 \times 30$ bins), which yields bins of approximately 4 degree for each rotation axis and progressively reduced the number of bins proportionally for the 3 axes.

The results are summarized in Tables 4 and 5, for the landmark and local surface descriptors, respectively, We can see that in both cases, the number of bins can be drastically reduced while maintaining performance relatively unchanged. In the case of landmarks, the errors increased less than 2% while reducing the size of the input tensor by 330 times (to $6 \times 6 \times 4$ bins), after which errors finally rose. Similarly, in the case of local surface descriptors, it was possible to reduce the tensor up to 64 times with no impact on performance.

The reason for this robustness to the discretization size is, again, the use of trigonometric constraints to model the manifold structure of the rotation subspaces. Fig. 13 illustrates this by showing how the input data and the estimated cosines vary as we progressively reduce the number of bins used to discretize the rotation angles. We start by showing the case of $20\times20\times15$ bins, in which the input data clearly defines cosine curves. As we reduce the number of bins, the shapes defined by the input data

[7] Depending on the way in which the data is captured and the extent of the considered rotations, self-occlusions may jeopardize this strategy.

[8] All experiments in this paper have been performed following this strategy,

Table 4 Effect of angular discretization on the proposed method using landmarks as input features.

Num of bins (y,p,r)	Memory used by $\mathscr{T}$	Yaw error	Pitch error	Roll error	Avg. error
$40 \times 40 \times 30$	369.1 MB	6.50 ± 8.74	7.07 ± 10.33	6.06 ± 10.51	6.54 ± 9.89
$20 \times 20 \times 15$	46.14 MB	6.46 ± 8.38	7.07 ± 10.30	5.87 ± 10.21	6.47 ± 9.67
$10 \times 10 \times 8$	6.15 MB	6.44 ± 8.48	7.09 ± 10.05	5.97 ± 10.09	6.50 ± 9.57
$8 \times 8 \times 6$	2.95 MB	6.58 ± 8.61	6.99 ± 9.86	5.90 ± 9.43	6.49 ± 9.31
$6 \times 6 \times 4$	1.11 MB	6.60 ± 8.22	7.16 ± 10.48	5.87 ± 10.18	6.54 ± 9.68
$4 \times 4 \times 3$	378 KB	7.18 ± 8.53	7.15 ± 9.97	6.14 ± 9.20	6.82 ± 9.25
$3 \times 3 \times 2$	142 KB	7.29 ± 9.73	7.37 ± 9.97	7.55 ± 10.97	7.40 ± 10.24

Table 5 Effect of angular discretization on the proposed method using local surface descriptors as input features.

Num of bins (y,p,r)	Memory used by $\mathscr{T}$	Yaw error	Pitch error	Roll error	Avg. error
$40 \times 40 \times 30$	5.13 GB	6.21 ± 7.82	6.64 ± 10.03	4.60 ± 6.82	5.83 ± 8.33
$20 \times 20 \times 15$	656.3 MB	6.33 ± 7.94	6.67 ± 10.03	4.68 ± 6.90	5.89 ± 8.39
$10 \times 10 \times 8$	87.5 MB	6.20 ± 7.83	6.77 ± 9.93	4.68 ± 6.67	5.88 ± 8.25
$8 \times 8 \times 6$	42.0 MB	6.54 ± 7.88	7.31 ± 10.44	5.76 ± 7.55	6.54 ± 8.72
$6 \times 6 \times 4$	15.8 MB	6.59 ± 7.99	7.34 ± 10.06	5.60 ± 7.48	6.51 ± 8.58
$5 \times 5 \times 5$	10.9 MB	6.80 ± 8.18	7.29 ± 10.02	5.61 ± 7.38	6.57 ± 8.60
$4 \times 4 \times 3$	5.3 MB	7.54 ± 8.64	8.11 ± 10.71	6.45 ± 7.77	7.37 ± 9.12

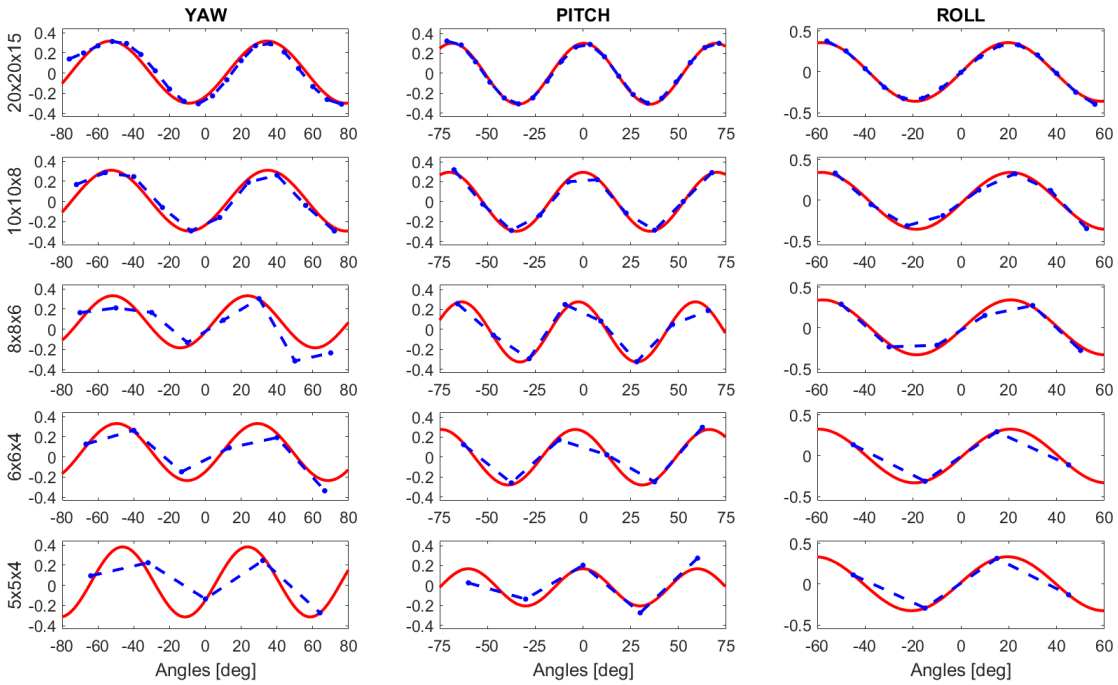

Fig. 13 Effect of the discretization in the estimated cosines used to approximate the rotation subspaces. The blue dots (joined by dashed lines) indicate the actual data samples per subject (bins) used to estimate the cosine functions, which are plotted with red solid lines. Each row shows a different discretization setting, from 20 bins for yaw and pitch (15 for roll) at the top row to just 5 bins (4 for roll) at the bottom row.

seem to deviate from the cosine shapes; yet a least-squares approximation (following Eq. 15) is enough to recover cosine functions with very similar parameters as those obtained with larger numbers of bins. In the last row of Fig. 13, the cosines are estimated from just 5 data points for yaw and pitch (4 points for roll), leading to less accurate estimates. The tensor size, however, is 480 times smaller than the original.

Tables 4 and 5 also indicate the memory required to store the data tensor for the different discretization choices. To compute these values we assumed storage in double-precision using the training set of the SASE

database to build the tensors, which implies storing the data of 28 subjects. The growth of the tensor size is linear in the number of subjects and cubic with respect to the inverse of the bin size. Thus, the scalability of the approach is dominated by the number of bins required to accurately estimate the cosine functions. As discussed above, the approach is robust to small numbers of bins.

In terms of complexity, once the features have been extracted, the run-time during the test stage is dominated by the minimization in Eq. 12. Since our goal was not a real-time system, we solved this minimization using a Matlab package (Schmidt, 2012) without applying any special optimization. In spite of this, the landmark-based estimates took only 37.8 ms per frame (on average) on a desktop PC based on an Intel i7-4770 CPU @3.4 GHz with 24 GB of RAM. On the other hand, the descriptor-based estimates (of considerably higher dimensionality) required an average processing time of 0.83 seconds.

6.4 3D head pose estimation using BIWI database

The BIWI Database (Fanelli et al., 2013), acquired with a Kinect 1 sensor, contains 24 sequences of RGB-D images of subjects moving their heads over a range of roughly $\pm 75^\circ$ for yaw, $\pm 60^\circ$ for pitch and $\pm 50^\circ$ for roll. In total this database consists of around 17K images. Because there is no standard experimental protocol for this database, we perform our experiments under a leave-one-sequence-out strategy, so that no sequence is used for training and test at the same time. All other settings were kept as described in the previous section for the SASE database.

Table 6 summarizes our results, as well as those presented by previous works reporting pose estimation errors on this database. For each method, we show the average absolute error per angle together with the respective standard deviations (when provided by the authors). We also indicate the type of input data that is used (depth, RGB or both) and if pose estimates are produced separately for each frame or using also temporal information (e.g. tracking).

Note that, despite the fact that our approach is the only one using just depth information without tracking, our results are quite competitive. Indeed, we clearly outperform other methods not doing tracking except for (Papazov et al., 2015), who reported smaller averages but considerably higher standard deviations. Additionally, we achieve results that are comparable to some of the tracking-based methods listed in Table 6. Note, however, that we do not use any temporal information and produce our pose estimates independently for each frame, which implies that our results reflect the average performance at fully automatic operation under arbitrary initial head poses.

In the last column of Table 6 we also indicate the methods that are based on DNNs, thus highlighting the growing interest on Deep Learning methods for pose estimation. In the context of such methods, some of which very recent, we still observe that our results are quite competitive. Indeed, we achieve lower overall pose estimation error than all methods reporting per-frame estimates, regardless of whether they use depth or RGB data as input.

On the other hand, even though we have not focused on optimizing operation speed, it is worth noting that several methods in Table 6 are targeted to real time operation. In our case, the limitations to operate in real-time comes firstly from the extracted features (see (Derkach et al., 2017)), and then by the minimization of Eq. 12, which can be time-consuming in the case of high-dimensional features. Nevertheless, our method is not actually tied to any specific feature descriptor, which makes it feasible to replace the features used in this work by real-time alternatives, such as those in (Papazov et al., 2015). As indicated in Section 6.2.2, for low-dimensional features (e.g. $\sim$36 dimensions), the minimization of Eq. 12 is easily achievable in real-time using a Matlab-based implementation (Schmidt, 2012) in a standard desktop-PC; in case of features of considerably higher dimension, there would be the need to investigate a more efficient implementation.

7 Conclusions

In this work we address 3D head pose estimation from depth data by proposing a novel approach to learn the manifold defined by 3D rotations. In particular, our method is able to explicitly model the underlying structure of the rotation manifold with an analytic form that takes into account the specific constraints imposed by orientation variations. For this purpose, we use multi-linear decomposition to split the pose variation factors into separate sub-spaces accounting for yaw, pitch and roll effects. We show that the coefficients within each of these subspaces define a continuous curve that can be modeled in terms of trigonometric functions, which are indeed the bases to explain rotation effects. We exploit this fact to introduce a minimization framework for pose estimation based on tensor decomposition constrained by trigonometric functions so that the obtained solutions are always compliant with variations in rotation parameters. To the best of our knowledge, the proposed method is the first approach combining tensor decomposition with non-linear modelling in order to explicitly impose structure over the learned manifold.

We show that the proposed modeling based on trigonometric functions can accurately model the effect of rotation variations in pose sub-spaces, by means of qualitative examples on 2D and 3D datasets. We also provide quantitative results of head pose estimation in two public databases, which demonstrate the advantages

Table 6 Average pose estimation errors and standard deviations of the proposed frame-work and previous works on the BIWI database

Method	Temporal Information	Errors ± Std			Domain	DNN-based
		Yaw	Pitch	Roll		
Fanelli et al. (2011)	Yes	8.9 ± 13.0	8.5 ± 9.9	7.9 ± 8.3	depth	No
Padeleris et al. (2012)	Yes	2.4 ± 1.8	3.0 ± 2.16	2.8 ± 2.1	depth	No
Baltrušaitis et al. (2012)	Yes	6.3	5.1	11.3	RGB + depth	No
Wang et al. (2013)	No	8.8 ± 14.3	8.5 ± 11.1	7.4 ± 10.8	RGB + depth	No
Ahn et al. (2014)	Yes	2.6 ± 2.5	3.4 ± 2.9	2.8 ± 2.4	RGB	CNN
Meyer et al. (2015)	Yes	2.1	2.1	2.4	depth	No
Papazov et al. (2015)	No Yes	2.5 ± 8.3 3.0 ± 9.6	1.8 ± 4.3 2.5 ± 7.4	2.9 ± 12.8 3.8 ± 16.0	RGB + depth	No
Chen et al. (2016)	No	9.9 ± 12.4	12.8 ± 17.2	6.9 ± 9.8	RGB	No
Li et al. (2016)	Yes	3.0	3.2	5.3	RGB + depth	No
Liu et al. (2016)	Yes	4.5 ± 3.8	4.3 ± 2.7	2.4 ± 1.9	RGB	CNN
Gu et al. (2017)	No Yes	3.91 ± 3.82 3.14 ± 3.12	4.03 ± 3.61 3.48 ± 2.89	3.03 ± 3.05 2.60 ± 2.76	RGB	CNN CNN + RNN
Lathuilière et al. (2017)	No	3.1	4.7	3.1	RGB	No
Yu et al. (2017)	Yes	2.5	1.5	2.2	RGB + depth	VGG-16
Ruiz et al. (2018)	No	4.81	6.61	3.27	RGB	ResNet-50
Borghi et al. (2019)	Yes	2.6 ± 1.5	2.4 ± 1.3	2.9 ± 1.5	depth	POSEidon$^+$
Lathuilire et al. (2019)	No	3.74 2.37	4.02 5.22	3.28 4.04	RGB	VGG-16 ResNet-50
Wang et al. (2019)	No	4.76 ± 4.33 5.16 ± 5.32	5.48 ± 3.23 4.23 ± 5.13	4.29 ± 3.30 5.39 ± 2.61	RGB depth	GoogleNet
Proposed LMK	**No**	**3.6 ± 4.6**	**3.8 ± 4.8**	**5.2 ± 5.8**	**depth**	**No**
Proposed DESC	**No**	**3.3 ± 4.2**	**3.4 ± 4.4**	**3.3 ± 3.7**	**depth**	**No**

introduced by the proposed constraints. Firstly, on the challenging SASE database, we show that directly applying existing multi-linear decomposition approaches yields poor pose estimation errors, which dramatically improve when introducing the proposed trigonometric constraints, reaching the lowest angle estimation errors reported so far on this database. Later, we also report results on the widely used BIWI database, showing that the proposed framework is not only of theoretical interest but it can be translated into a practical system to produce competitive pose estimation results.

ACKNOWLEDGMENTS

This work is partly supported by the Spanish Ministry of Economy and Competitiveness under project grant TIN2017-90124-P, the Ramon y Cajal programme, and the Maria de Maeztu Units of Excellence Programme (MDM-2015-0502). Adria Ruiz work is partially funded by ANR grant ANR-16-CE23-0006.

References

Ahn B, Park J, Kweon IS (2014) Real-time head orientation from a monocular camera using deep neural network. In: Asian Conference on Computer Vision, Springer, pp 82–96

Bakry A, Elgammal A (2014) Untangling object-view manifold for multiview recognition and pose estimation. In: European Conference on Computer Vision, Springer, pp 434–449

Balasubramanian VN, Ye J, Panchanathan S (2007) Biased manifold embedding: A framework for person-independent head pose estimation. In: Computer Vision and Pattern Recognition (CVPR), IEEE, pp 1–7

Baltrušaitis T, Robinson P, Morency LP (2012) 3D constrained local model for rigid and non-rigid facial tracking. In: Computer Vision and Pattern Recognition (CVPR), IEEE, pp 2610–2617

Barros JMD, Mirbach B, Garcia F, Varanasi K, Stricker D (2018) Fusion of keypoint tracking and facial landmark detection for real-time head pose estimation. In: Winter Conference on Applications of Computer Vision (WACV), IEEE, pp 2028–2037

BenAbdelkader C (2010) Robust head pose estimation using supervised manifold learning. In: European Conference on Computer Vision, Springer, pp 518–531

Bergqvist G, Larsson EG (2010) The higher-order singular value decomposition: Theory and an application [lecture notes]. IEEE Signal Processing Magazine 27(3):151–154

Borghi G, Venturelli M, Vezzani R, Cucchiara R (2017) Poseidon: Face-from-depth for driver pose estimation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 4661–4670

Borghi G, Fabbri M, Vezzani R, s calderara, Cucchiara R (2019) Face-from-depth for head pose estimation on depth images. IEEE Transactions on Pattern Analysis and Machine Intelligence *(in press)*.

Breitenstein MD, Kuettel D, Weise T, Van Gool L, Pfister H (2008) Real-time face pose estimation from single range images. In: Computer Vision and Pattern Recognition, IEEE, pp 1–8

Byrd RH, Nocedal J, Schnabel RB (1994) Representations of quasi-newton matrices and their use in limited memory methods. Mathematical Programming 63(1-3):129–156

Chen J, Wu J, Richter K, Konrad J, Ishwar P (2016) Estimating head pose orientation using extremely low resolution images. In: Southwest Symposium on Image Analysis and Interpretation (SSIAI), IEEE, pp 65–68

Comon P (2014) Tensors: a brief introduction. Signal Processing Magazine 31(3):44–53

De Lathauwer L, De Moor B, Vandewalle J (2000) A multilinear singular value decomposition. SIAM journal on Matrix Analysis and Applications 21(4):1253–1278

Derkach D, Ruiz A, Sukno FM (2017) Head pose estimation based on 3-D facial landmarks localization and regression. In: 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017), IEEE, pp 820–827

Derkach D, Ruiz A, Sukno FM (2018) 3D head pose estimation using tensor decomposition and non-linear manifold modeling. In: International Conference on 3D Vision (3DV), IEEE, pp 505–513

Fanelli G, Weise T, Gall J, Van Gool L (2011) Real time head pose estimation from consumer depth cameras. In: Joint Pattern Recognition Symposium, Springer, pp 101–110

Fanelli G, Dantone M, Gall J, Fossati A, Van Gool L (2013) Random forests for real time 3D face analysis. International Journal of Computer Vision 101(3):437–458

Frome A, Huber D, Kolluri R, Bulow T, Malik J (2004) Recognizing objects in range data using regional point descriptors. In: European conference on computer vision, Springer, pp 224–237

Fu Y, Huang TS (2006) Graph embedded analysis for head pose estimation. In: International Conference on Automatic Face and Gesture Recognition, IEEE, pp 6–8

Ghiass RS, Arandjelović O, Laurendeau D (2015) Highly accurate and fully automatic head pose estimation from a low quality consumer-level rgb-d sensor. In: Proceedings of the 2nd Workshop on Computational Models of Social Interactions: Human-Computer-Media Communication, ACM, pp 25–34

Gu J, Yang X, De Mello S, Kautz J (2017) Dynamic facial analysis: From bayesian filtering to recurrent neural network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 1548–1557

Johnson A, Hebert M (1999) Using spin images for efficient object recognition in cluttered 3D scenes. IEEE Transactions on Pattern Analysis and Machine Intelligence 21(5):433–449

Kolda TG, Bader BW (2009) Tensor decompositions and applications. SIAM review 51(3):455–500

Lathuilière S, Juge R, Mesejo P, Muñoz-Salinas R, Horaud R (2017) Deep mixture of linear inverse regressions applied to head-pose estimation. In: Conference on Computer Vision and Pattern Recognition, vol 3, pp 4817–4825

Lathuilire S, Mesejo P, Alameda-Pineda X, Horaud R (2019) A comprehensive analysis of deep regression. IEEE Transactions on Pattern Analysis and Machine Intelligence *(in press)*.:1–1

Lee D, Yang MH, Oh S (2015) Fast and accurate head pose estimation via random projection forests. In: International Conference on Computer Vision, IEEE, pp 1958–1966

Lee D, Yang MH, Oh S (2017) Head and body orientation estimation using convolutional random projection forests. IEEE Transactions on Pattern Analysis and Machine Intelligence pp 1–14

Li D, Pedrycz W (2014) A central profile-based 3D face pose estimation. Pattern Recognition 47(2):525–534

Li S, Ngan KN, Paramesran R, Sheng L (2016) Real-time head pose tracking with online face template reconstruction. IEEE transactions on pattern analysis and machine intelligence 38(9):1922–1928

Liu X, Lu H, Li W (2010) Multi-manifold modeling for head pose estimation. In: International Conference on Image Processing (ICIP), IEEE, pp 3277–3280

Liu X, Liang W, Wang Y, Li S, Pei M (2016) 3D head pose estimation with convolutional neural network trained on synthetic images. In: International Conference on Image Processing (ICIP), IEEE, pp 1289–1293

Lüsi I, Escalera S, Anbarjafari G (2016a) Human head pose estimation on SASE database using random hough regression forests. In: Video Analytics. Face and Facial Expression Recognition and Audience Measurement, Springer, pp 137–150

Lüsi I, Escarela S, Anbarjafari G (2016b) SASE: RGB-depth database for human head pose estimation. In: European Conference on Computer Vision, Springer, pp 325–336

Lüsi I, Jacques Junior JCS, Gorbova J, Baró X, Escalera S, Demirel H, Allik J, Ozcinar C, Anbarjafari G (2017) Joint challenge on dominant and complementary emotion recognition using micro emotion features and head-pose estimation: Databases. In: International Conference on Automatic Face and Gesture Recognition, IEEE, pp 809–813

Martin M, Van De Camp F, Stiefelhagen R (2014) Real time head model creation and head pose estimation on consumer depth cameras. In: International Conference on

3D Vision (3DV), IEEE, vol 1, pp 641–648
Meyer GP, Gupta S, Frosio I, Reddy D, Kautz J (2015) Robust model-based 3D head pose estimation. In: Proceedings of the IEEE International Conference on Computer Vision, IEEE, pp 3649–3657
Murphy-Chutorian E, Trivedi MM (2009) Head pose estimation in computer vision: A survey. IEEE transactions on pattern analysis and machine intelligence 31(4):607–626
Nene SA, Nayar SK, Murase H, et al. (1996) Columbia object image library (coil-20)
Padeleris P, Zabulis X, Argyros AA (2012) Head pose estimation on depth data based on particle swarm optimization. In: Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), IEEE, pp 42–49
Papazov C, Marks TK, Jones M (2015) Real-time 3D head pose and facial landmark estimation from depth images using triangular surface patch features. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4722–4730
Patacchiola M, Cangelosi A (2017) Head pose estimation in the wild using convolutional neural networks and adaptive gradient methods. Pattern Recognition 71:132–143
Peng X, Huang J, Hu Q, Zhang S, Metaxas DN (2014) Head pose estimation by instance parameterization. In: International Conference on Pattern Recognition (ICPR), IEEE, pp 1800–1805
Raytchev B, Yoda I, Sakaue K (2004) Head pose estimation by nonlinear manifold learning. In: International Conference on Pattern Recognition (ICPR), IEEE, vol 4, pp 462–466
Ruiz N, Chong E, Rehg JM (2018) Fine-grained head pose estimation without keypoints. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp 2074–2083
Rusu RB, Blodow N, Beetz M (2009) Fast point feature histograms (fpfh) for 3d registration. In: International Conference on Robotics and Automation, Citeseer, pp 3212–3217
Schmidt M (2012) minfunc: unconstrained differentiable multivariate optimization in matlab. Software available at http://www cs ubc ca/˜ schmidtm/Software/minFunc htm
Seemann E, Nickel K, Stiefelhagen R (2004) Head pose estimation using stereo vision for human-robot interaction. In: International Conference on Automatic Face and Gesture Recognition, IEEE, pp 626–631
Sukno F, Waddington J, Whelan P (2012) Comparing 3D descriptors for local search of craniofacial landmarks. In: International Symposium on Visual Computing, Springer, pp 92–103
Sukno F, Waddington J, Whelan P (2013) Rotationally invariant 3D shape contexts using asymmetry patterns. In: International confernece on computer graphics theory and applications, pp 7–17
Sukno FM, Waddington JL, Whelan PF (2015) 3-D facial landmark localization with asymmetry patterns and shape regression from incomplete local features. IEEE transactions on cybernetics 45(9):1717–1730
Sun Y, Yin L (2008) Automatic pose estimation of 3D facial models. In: International Conference on Pattern Recognition, pp 1–4
Sundararajan K, Woodard DL (2015) Head pose estimation in the wild using approximate view manifolds. In: International Conference on Computer Vision and Pattern Recognition Workshops, IEEE, pp 50–58
Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2015) Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1–9
Takallou HM, Kasaei S (2014) Head pose estimation and face recognition using a non-linear tensor-based model. IET Computer Vision 8(1):54–65
Tan DJ, Tombari F, Navab N (2018) Real-time accurate 3d head tracking and pose estimation with consumer rgb-d cameras. International Journal of Computer Vision 126(2-4):158–183
Tenenbaum JB, Freeman WT (1997) Separating style and content. In: Advances in neural information processing systems, pp 662–668
Tenenbaum JB, Freeman WT (2000) Separating style and content with bilinear models. Neural computation 12(6):1247–1283
Tombari F, Salti S, Di Stefano L (2010) Unique signatures of histograms for local surface description. In: European conference on computer vision, Springer, pp 356–369
Tulyakov S, Vieriu RL, Semeniuta S, Sebe N (2014) Robust real-time extreme head pose estimation. In: International Conference on Pattern Recognition (ICPR), IEEE, pp 2263–2268
Vasilescu MAO, Terzopoulos D (2002) Multilinear analysis of image ensembles: Tensorfaces. In: European Conference on Computer Vision, Springer, pp 447–460
Wang B, Liang W, Wang Y, Liang Y (2013) Head pose estimation with combined 2D SIFT and 3D HOG features. In: International Conference on Image and Graphics (ICIG), IEEE, pp 650–655
Wang C, Song X (2014) Robust head pose estimation via supervised manifold learning. Neural Networks 53:15–25
Wang C, Guo Y, Song X (2017a) Head pose estimation via manifold learning. In: Manifolds-Current Research Areas, InTech
Wang K, Wu Y, Ji Q (2018) Head pose estimation on low-quality images. In: International Conference on Automatic Face & Gesture Recognition (FG 2018), IEEE,

pp 540–547

Wang M, Panagakis Y, Snape P, Zafeiriou S, et al. (2017b) Learning the multilinear structure of visual data. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 4592–4600

Wang Y, Liang W, Shen J, Jia Y, Yu LF (2019) A deep coarse-to-fine network for head pose estimation from synthetic data. Pattern Recognition 94:196–206

Xu Y, Hao R, Yin W, Su Z (2015) Parallel matrix factorization for low-rank tensor completion. Inverse Problems and Imaging 9(2):601–624

Yu Y, Mora KAF, Odobez JM (2017) Robust and accurate 3D head pose estimation through 3dmm and online head model reconstruction. In: International Conference on Automatic Face & Gesture Recognition (FG 2017), IEEE, pp 711–718

Zhang H, El-Gaaly T, Elgammal A, Jiang Z (2015) Factorization of view-object manifolds for joint object recognition and pose estimation. Computer Vision and Image Understanding 139:89–103

Zhao Q, Zhang L, Cichocki A (2015) Bayesian cp factorization of incomplete tensors with automatic rank determination. IEEE transactions on pattern analysis and machine intelligence 37(9):1751–1763

Zhu Y, Xue Z, Li C (2014) Automatic head pose estimation with synchronized sub manifold embedding and random regression forests. International Journal of Signal Processing, Image Processing and Pattern Recognition 7(3):123–134

Appendix

For the objective function of Eq. 15, partial derivatives should be computed with respect to variables $\alpha_j^{(*)}, \beta_j^{(*)}, \gamma_j^{(*)}, \varphi_j^{(*)}$. Let us rewrite this equation as:
$\text{argmin}_{\alpha_j^{(*)},\beta_j^{(*)},\gamma_j^{(*)},\varphi_j^{(*)}} \| u_{ij}^{(*)} - f_j(\omega_i^{(*)}) \| = \text{argmin}_{\alpha_j^{(*)},\beta_j^{(*)},\gamma_j^{(*)},\varphi_j^{(*)}} E(\alpha_j^{(*)}, \beta_j^{(*)}, \gamma_j^{(*)}, \varphi_j^{(*)})$

Where error function E can be written in element form as:
$E(\alpha_j^{(*)}, \beta_j^{(*)}, \gamma_j^{(*)}, \varphi_j^{(*)}) = \frac{1}{2} \sum_i \left(u_{ij}^{(*)} - (\alpha_j^{(*)} \cos(\beta_j^{(*)} \omega_i^{(*)} + \gamma_j^{(*)}) + \varphi_j^{(*)}) \right)^2$

Thus, partial derivatives of the function E become:

$$\frac{\partial E}{\partial \alpha_j^{(*)}} = \sum_i \left(\cos(\beta_j^{(*)} \omega_i^{(*)} + \gamma_j^{(*)}) \cdot (\alpha_j^{(*)} \cos(\beta_j^{(*)} \omega_i^{(*)} + \gamma_j^{(*)}) - u_{ij}^{(*)} + \varphi_j^{(*)}) \right)$$
$$\frac{\partial E}{\partial \beta_j^{(*)}} = \alpha_j^{(*)} \sum_i \left(\omega_i^{(*)} \sin(\beta_j^{(*)} \omega_i^{(*)} + \gamma_j^{(*)}) \cdot (u_{ij}^{(*)} - \alpha_j^{(*)} \cos(\beta_j^{(*)} \omega_i^{(*)} + \gamma_j^{(*)}) - \varphi_j^{(*)}) \right)$$
$$\frac{\partial E}{\partial \gamma_j^{(*)}} = \alpha_j^{(*)} \sum_i \left(\sin(\beta_j^{(*)} \omega_i^{(*)} + \gamma_j^{(*)}) \cdot (u_{ij}^{(*)} - \alpha_j^{(*)} \cos(\beta_j^{(*)} \omega_i^{(*)} + \gamma_j^{(*)}) - \varphi_j^{(*)}) \right)$$
$$\frac{\partial E}{\partial \varphi_j^{(*)}} = \sum_i \left(\alpha_j^{(*)} \cos(\beta_j^{(*)} \omega_i^{(*)} + \gamma_j^{(*)}) + \varphi_j^{(*)} + u_{ij}^{(*)} \right)$$

Similarly to the previous case, the error function in Eq. 12 can be written as:

$$E(\omega^{(y)}, \omega^{(p)}, \omega^{(r)}, \mathbf{u}^{(id)}) = \frac{1}{2} \sum_n \left(\mathbf{x}_n - \hat{\mathbf{x}}_n \right)^2$$

where $\mathbf{x}_n$ using Eq. 14 is:
$\mathbf{x}_n = \sum_i \sum_j \sum_k \sum_l \left(\mathscr{W}_{ijkln} \cdot f_i^{(y)}(\omega^{(y)}) \cdot f_j^{(p)}(\omega^{(p)}) \cdot f_k^{(r)}(\omega^{(r)}) \cdot u_l^{(id)} \right)$

Now we can compute partial derivatives with respect to variables $\omega^{(y)}, \omega^{(p)}, \omega^{(r)}$ and each l-th element in vector $\mathbf{u}^{(id)}$:

$$\frac{\partial E}{\partial \omega^{(y)}} = -\sum_n \left((\mathbf{x}_n - \hat{\mathbf{x}}_n) \cdot \frac{\partial \hat{\mathbf{x}}_n}{\partial \omega^{(y)}} \right);$$
$$\frac{\partial \hat{\mathbf{x}}_n}{\partial \omega^{(y)}} = -\sum_i \sum_j \sum_k \sum_l \left(\mathscr{W}_{ijkln} \cdot \alpha_i^{(y)} \sin(\beta_i^{(y)} \omega^{(y)} + \gamma_i^{(y)}) \beta_i^{(y)} \cdot f_j^{(p)}(\omega^{(p)}) \cdot f_k^{(r)}(\omega^{(r)}) \cdot u_l^{(id)} \right);$$
$$\frac{\partial E}{\partial \omega^{(p)}} = -\sum_n \left((\mathbf{x}_n - \hat{\mathbf{x}}_n) \cdot \frac{\partial \hat{\mathbf{x}}_n}{\partial \omega^{(p)}} \right);$$
$$\frac{\partial \hat{\mathbf{x}}_n}{\partial \omega^{(p)}} = -\sum_i \sum_j \sum_k \sum_l \left(\mathscr{W}_{ijkln} \cdot f_i^{(y)}(\omega^{(y)}) \cdot (\alpha_j^{(p)} \sin(\beta_j^{(p)} \omega^{(p)} + \gamma_j^{(p)}) \beta_j^{(p)}) \cdot f_k^{(r)}(\omega^{(r)}) \cdot u_l^{(id)} \right);$$
$$\frac{\partial E}{\partial \omega^{(r)}} = -\sum_n \left((\mathbf{x}_n - \hat{\mathbf{x}}_n) \cdot \frac{\partial \hat{\mathbf{x}}_n}{\partial \omega^{(r)}} \right);$$
$$\frac{\partial \hat{\mathbf{x}}_n}{\partial \omega^{(r)}} = -\sum_i \sum_j \sum_k \sum_l \left(\mathscr{W}_{ijkln} \cdot f_i^{(y)}(\omega^{(y)}) \cdot f_j^{(p)}(\omega^{(p)}) \cdot (\alpha_k^{(r)} \sin(\beta_k^{(r)} \omega^{(r)} + \gamma_k^{(r)}) \beta_k^{(r)}) \cdot u_l^{(id)} \right);$$
$$\frac{\partial E}{\partial \mathbf{u}_l^{(id)}} = -\sum_n \left((\mathbf{x}_n - \hat{\mathbf{x}}_n) \cdot \frac{\partial \hat{\mathbf{x}}_n}{\partial \mathbf{u}_l^{(id)}} \right);$$
$$\frac{\partial \hat{\mathbf{x}}_n}{\partial \mathbf{u}_l^{(id)}} = \sum_i \sum_j \sum_k \left(\mathscr{W}_{ijkln} \cdot f_i^{(y)}(\omega^{(y)}) \cdot f_j^{(p)}(\omega^{(p)}) \cdot f_k^{(r)}(\omega^{(r)}) \right)$$