In Proc. of ICCV, Kyoto, Japan, September 2009

A direct approach for efficiently tracking with 3D Morphable Models

Enrique Muñoz
Dep. Sistemas Informáticos y Computación
Univ. Complutense Madrid

José M. Buenaposada
Dep. Ciencias de la Computación
Univ. Rey Juan Carlos

Luis Baumela
Dep. Inteligencia Artificial
Univ. Politécnica Madrid

http://www.dia.fi.upm.es/~pcr

Abstract

We present an efficient algorithm for fitting a morphable model to an image sequence. It is built on a projective geometry formulation of perspective projection, which results in a linear mapping from 3D shape to the projective plane, and a factorisation of this mapping into matrices that can be partially computed off-line. This algorithm can cope with full 360 degrees object rotation and linear deformations. We validate our approach using synthetically generated and real sequences. Compared to a plain Lucas-Kanade implementation, we achieve a six fold increase in performance for a rigid object and two fold for a non-rigid face.

1. Introduction

Tracking non-rigid objects and in particular human faces is an active research area for its applications in advanced human computer interaction, performance-driven realistic graphical animation [18] or face recognition [3]. Reliable tracking of faces under large rotations is still a challenging task in computer vision because the face is a low-textured and deformable object whose appearance changes dramatically with pose or facial expression.

Tracking approaches are based on estimating the parameters of a function that describe the pose and deformation of the face in each image of a sequence. This can be achieved by matching a sparse collection of features (feature-based methods) or by directly minimising the difference in image intensity values (direct approaches). The main advantage of feature-based methods is the possibility of working with very large inter-frame motion [20]. This makes them best suited for detection. They require, nevertheless, the existence of highly textured patches uniformly distributed over the target face. Direct approaches, on the other hand, assume that inter-frame motion is small, as is the case in video sequences. They can be used for tracking low or smooth textured surfaces, like human faces [11]. Tracking is usually posed as a Gauss-Newton-like optimisation process, minimising a similarity measure between a reference template (the face model) and the target image. Their main advantage is accuracy, since all visible pixels contribute to the minimisation.

Direct approaches are based on generative linear models of appearance, such as the *Active Appearance Models* (AAMs) [6] or the 3D *Morphable Models* (MMs) [2]. Matthews and Baker used the *Inverse Compositional Image Alignment* (ICIA) algorithm for tracking faces in real-time using AAMs [11]. Their solution is possibly the fastest introduced so far, since the Jacobian and Hessian matrices emerging in the optimisation can be computed off-line. One limitation of their approach is that AAMs are intrinsically 2D models and, although they can be used to track a 3D object, this is achieved at the expense of requiring up to 6 times more shape parameters. In consequence, the minimisation must be properly constrained in order to achieve a robust tracker [12]. Also, 2D models have problems for dealing with self occlusions appearing, for example, with large face rotations. 3D MMs, on the other hand, deal naturally with large rotations and self occlusions. Unfortunately, they have not been used for tracking, since no efficient fitting procedure has been introduced so far. Romdhani and Vetter [14] also used ICIA for efficiently adjusting a 3D MM to the image of a static face. Their solution cannot be used for tracking since their Jacobian and Hessian matrices are locally valid [14]. An important drawback of both approaches is that they work under weak-perspective imaging conditions. This restriction is necessary to decouple the rigid and non-rigid components of target motion. This is a limitation if, for example, we would like to track a face imaged by a camera with short focal length and strong perspective distortion (e.g a low-cost web-cam).

In this paper we introduce a procedure for efficiently fitting a 3D MM to a target image sequence using a full projective camera model. It achieves efficiency by factoring the Jacobian and Hessian matrices appearing in the Gauss-Newton optimisation into the multiplication of a constant

matrix, depending on target structure and texture that can be computed off-line, and a varying matrix that depends on the target motion and deformation parameters. Our approach is directly related to the work of Hager and Belhumeur [8] that introduced an efficient procedure for tracking a planar region with parametric models of geometry and illumination. It can be considered as an extension of their approach to the case of a 3D deforming target. We also introduce necessary and sufficient conditions related to the applicability of this approach. It is also related to Xu and Roy-Chowhury's technique for efficiently estimating 3D motion and illumination [19]. Their approach is based on ICIA and, like Romdhani and Vetter's, their Jacobian and Hessian matrices are only locally valid and have to be recomputed.

In summary, the main contributions of this paper are:

- We revisit Hager and Belhumeur's [8] fitting algorithm and provide necessary and sufficient conditions that must be satisfied to use their approach.
- We introduce a new efficient fitting algorithm based on a projective geometry formulation of perspective projection.This algorithm can cope with full 360 degrees object rotation and linear deformations. It naturally decouples rigid and non-rigid deformations within a projective camera framework.

2. Model-based tracking

The information about the structure, texture and modes of deformation of our target object will follow a convention similar to that used for the 3D MMs [2]. We model shape and colour separately, both on a common bi-dimensional frame space denoted by $\mathbf{v}$. Then, $\mathbf{S}(\mathbf{v})$, mapping the frame space into the $\mathbb{R}^3$ 3D space, defines the 3D shape of an object (see Fig. 1-(b)). Similarly $\mathbf{T}(\mathbf{v})$ (see Fig. 1-(a)), mapping the bi-dimensional frame space into RGB (if coloured) or grey scale (if monochrome) texture space, defines colour or texture.

2.1. Motion model

The 3D motion of a point is the composition of a rigid motion caused by the translation and rotation of the object in space and a non-rigid motion caused by the deformation of the object. Let $\mathbf{x}^{(i)} = (x_i, y_i, z_i)^\top$ denote the co-ordinates of a point in 3D space and let $\mathbf{S} = (\mathbf{x}^{(1)\top}, \mathbf{x}^{(2)\top}, \ldots, \mathbf{x}^{(3)\top})^\top$ be the 3D structure represented by a set of N points in space.

The non-rigid motion of point $\mathbf{x}^{(i)}$ can be described as a linear combination of k basis points. So, the shape of any configuration of the non-rigid object is expressed as a linear combination of a set of k basis shapes stored in matrix $\mathtt{B}_s$ plus a mean vector $\mathbf{S}_0$: $\mathbf{S} = \mathbf{S}_0 + \mathtt{B}_s\mathbf{c}$, $\mathbf{S}, \mathbf{S}_0 \in \mathbb{R}^{3N\times 1}$, $\mathtt{B}_s \in \mathbb{R}^{3N\times k}$, $\mathbf{c} \in \mathbb{R}^{k\times 1}$, where $\mathbf{c} = (c_1, c_2, \ldots, c_k)^\top$ is the vector of shape configuration weights. The mean vector $\mathbf{S}_0$, also called *rigid component*, represents the rigid configuration of the object, and the basis $\mathtt{B}_s$ represents the allowed *modes of deformation*.

The 3D shape can rotate and translate rigidly in space. Let $\mathtt{R}(\alpha, \beta, \gamma) \in \mathbb{R}^{3\times 3}$ and $\mathbf{t} \in \mathbb{R}^{3\times 1}$ be the rotation matrix and translation vector representing such motion. And let $\mathbf{x}_r^{(i)} \in \mathbb{R}^3$ be the 3D co-ordinates of point (i), expressed in the co-ordinate system of the camera associated to a reference image I_r (see Fig. 1-(c)) , whose optical centre is $\mathbf{C}_r$. When the camera moves, the new camera position is $\mathbf{C}_t$ (see Fig. 1-(d)). The co-ordinates of $\mathbf{x}_t^{(i)}$, in this new reference, after a non-rigid deformation $\mathbf{c}_t$ are given by

$$\mathbf{x}_t^{(i)} = \mathtt{R}_t(\mathbf{x}_r^{(i)} + \mathtt{B}_s^{(i)}\mathbf{c}_t) + \mathbf{t}_t, \tag{1}$$

with $\mathtt{R}_t$ and $\mathbf{t}_t$ being the rigid body transformation between reference systems $\mathbf{C}_r$ and $\mathbf{C}_t$. The deformations are local to each point but they are globally parametrised by the deformation coefficients $\mathbf{c}_t$.

Equation (1) can be rewritten as a linear transformation between $\mathbf{x}_r^{(i)}$ and $\mathbf{x}_t^{(i)}$. This can be achieved by assuming that each point $\mathbf{x}_r^{(i)}$ is located on a scene plane $d_r^{(i)}$ units away from $\mathbf{C}_r$ and with normal $\mathbf{n}_r^{(i)}$. So (1) may be rewritten as

$$\mathbf{x}_t = \left(\mathtt{R}_t + (\mathtt{R}_t\mathtt{B}_s\mathbf{c}_t + \mathbf{t}_t)\frac{\mathbf{n}_r^\top}{d_r}\right)\mathbf{x}_r, \tag{2}$$

where we have dropped superscripts for notational convenience. Let $\mathbf{u}_r = \frac{1}{z_r}\mathtt{K}\mathbf{x}_r$ and $\mathbf{u}_t = \frac{1}{z_t}\mathtt{K}\mathbf{x}_t$ denote respectively the projection of $\mathbf{x}$ onto cameras $\mathbf{C}_r$ and $\mathbf{C}_t$, then (2) may be expressed as

$$\mathbf{u}_t \sim \underbrace{\mathtt{K}\left(\mathtt{R}_t + (\mathtt{R}_t\mathtt{B}_s\mathbf{c}_t + \mathbf{t}_t)\frac{\mathbf{n}_r^\top}{d_r}\right)\mathtt{K}^{-1}}_{\mathtt{H}_t}\mathbf{u}_r = \mathbf{f}(\mathbf{u}_r, \boldsymbol{\mu}_t), \tag{3}$$

where $\mathtt{H}_t$ is the homography induced by plane $\mathbf{n}_r$ between cameras $\mathbf{C}_r$ and $\mathbf{C}_t$. Points $\mathbf{u}_r$ and $\mathbf{u}_t$ represent coordinates in $\mathbb{P}^2$, the projective plane. The *brightness constancy assumption* can then be stated as

$$I_r(\mathbf{p}(\mathbf{u}_r)) = I_t(\mathbf{p}(\mathbf{f}(\mathbf{u}_r, \boldsymbol{\mu}_t))), \tag{4}$$

where $\mathbf{p} : \mathbb{P}^2 \mapsto \mathbb{R}^2$ is the projective to Cartesian mapping. Note that, since $\mathbf{f}$ depends on $d_r^{(i)}$ and $\mathbf{n}_r^{(i)}$, each vertex will induce a different $\mathbf{f}^{(i)}(\mathbf{u}, \boldsymbol{\mu})$, which is parametrised with the common object motion parameters $\boldsymbol{\mu}$. For notational convenience we will not represent this dependency in the rest of the paper. See Fig. 1 for further details.

2.2. Tracking as an optimisation problem

Tracking the object at time $t+1$ is equivalent to finding the motion parameters $\boldsymbol{\mu}_{t+1}$ that align the projection of

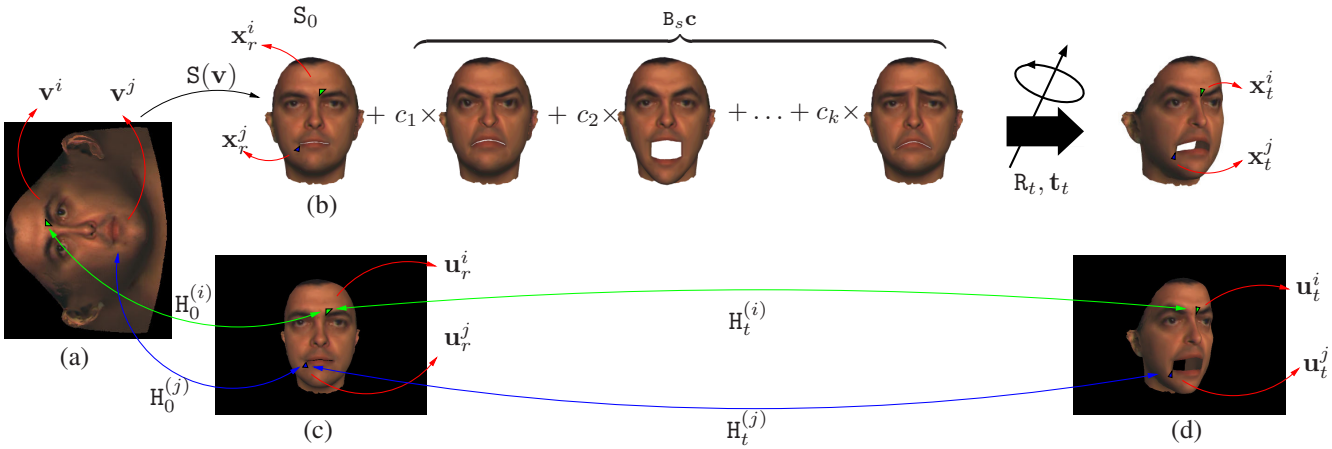

Figure 1. Motion model. (a) Unfolded shape in texture space. We use green and blue colours to display the co-ordinates of triangles i and j respectively. (b) The *reference shape* acts as the model rigid component. (c) We project this shape onto a *reference image* which links texture and image co-ordinates. (d) We compute actual 3D target co-ordinates by modifying the reference shape using a linear combination of deformation basis. This shape is then projected onto the current frame.

the target with the current image. This can be achieved by solving $\boldsymbol{\mu}_{t+1} = \arg\min_{\boldsymbol{\mu}} \mathcal{J}(\boldsymbol{\mu})$, where

$$\mathcal{J}(\boldsymbol{\mu}) = \|\mathbf{I}_r[\mathbf{p}(\mathbf{u}_r)] - \mathbf{I}_{t+1}[\mathbf{p}(\mathbf{f}(\mathbf{u}_r, \boldsymbol{\mu}))]\|^2, \qquad (5)$$

$\mathbf{I}_r[\mathbf{p}(\mathbf{u}_r)]$ and $\mathbf{I}_{t+1}[\mathbf{p}(\mathbf{f}(\mathbf{u}_r, \boldsymbol{\mu}))]$ are $N \times 1$ vectors formed by stacking all image values for $\mathbf{u}_i$, with $i = 1 \ldots N$.

Minimising (5) is a non-linear optimisation task that was originally introduced in the seminal work of Lucas and Kanade [10]. If the vector of motion parameters $\boldsymbol{\mu}$ is known at a previous time instant, $\boldsymbol{\mu}_t$, then (5) may be linearised by making a Taylor series expansion at $\boldsymbol{\mu}_t$

$$\mathcal{J}(\delta\boldsymbol{\mu}) = \left\| \underbrace{\mathbf{I}_r[\mathbf{p}(\mathbf{u}_r)] - \mathbf{I}_{t+1}[\mathbf{p}(\mathbf{f}(\mathbf{u}_r, \boldsymbol{\mu}_t))]}_{\mathbf{e}} - \underbrace{\left.\frac{\partial \mathbf{I}_t[\mathbf{p}(\mathbf{f}(\mathbf{u}_r, \hat{\boldsymbol{\mu}}))]}{\partial \hat{\boldsymbol{\mu}}}\right|_{\hat{\boldsymbol{\mu}}=\boldsymbol{\mu}_t}}_{\mathtt{J}(\boldsymbol{\mu}_t)} \delta\boldsymbol{\mu} \right\|^2, \qquad (6)$$

where $\mathtt{J}(\boldsymbol{\mu}_t)$ is the Jacobian matrix relating changes in template intensity values with motion parameters and $\mathbf{e}$ is the brightness error resulting from projecting the object shape with parameters $\boldsymbol{\mu}_t$ onto image acquired at $t+1$. Note that the output of the minimisation is $\delta\boldsymbol{\mu}$, the motion parameters' increment from instant t to instant $t+1$. The minimum of (6) is given by $\delta\boldsymbol{\mu} = \mathtt{J}(\boldsymbol{\mu}_t)^+\mathbf{e}$. We usually make a few iterations in a Gauss-Newton scheme until convergence.

3. Efficient tracking

The major limitation of the solution introduced in the previous section is the cost associated with the computation, for each incoming image, of $\mathtt{J}(\boldsymbol{\mu}_t)$ in (6), since it is a matrix with as many rows as points in the model and as many columns as parameters in $\boldsymbol{\mu}$. In the case of a typical MM this represents several dozens of thousand rows and up to a few tens of columns. In this section we introduce necessary and sufficient conditions that must be satisfied by $\mathbf{f}(\mathbf{u}, \boldsymbol{\mu})$ to guarantee that $\mathtt{J}(\boldsymbol{\mu}_t)$ can be factored into a large constant matrix, that can be precomputed off-line, and a small matrix that must be recomputed on-line.

3.1. The Gradient Equivalence Equation

Taking derivatives in equation (4) w.r.t. the shape coordinates

$$\left[\left.\frac{\partial \mathbf{I}_r[\mathbf{p}(\hat{\mathbf{x}})]}{\partial \hat{\mathbf{x}}}\right|_{\hat{\mathbf{x}}=\mathbf{u}_r}\right]^\top = \left[\left.\frac{\partial \mathbf{I}_t[\mathbf{p}(\hat{\mathbf{y}})]}{\partial \hat{\mathbf{y}}}\right|_{\hat{\mathbf{y}}=\mathbf{f}(\mathbf{u}_r,\boldsymbol{\mu}_t)}\right]^\top \left[\left.\frac{\partial \mathbf{f}(\hat{\mathbf{x}}, \boldsymbol{\mu}_t)}{\partial \hat{\mathbf{x}}}\right|_{\hat{\mathbf{x}}=\mathbf{u}_r}\right]. \qquad (7)$$

This is the *Gradient Equivalence Equation* (GEE) because it relates the gradients of the target projection in the reference image with the gradients in the image at time t. We are now ready to derive a proposition that introduces a necessary condition on $\mathbf{f}(\mathbf{u}, \boldsymbol{\mu})$ to express the image gradients emerging in $\mathtt{J}(\boldsymbol{\mu}_t)$ in terms of the gradients of the template.

Proposition 3.1. *The image gradient of a point in the projective plane $\mathbb{P}^2$ is a projective line incident to the point.*

Proof. Let $I : \mathbb{R}^2 \to \mathbb{R}$ be an image intensity function on the usual plane and $\mathbf{p} : \mathbb{P}^2 \to \mathbb{R}^2$ be a mapping function from the projective to the Cartesian plane, $\mathbf{p} : (x, y, w) \mapsto (i, j) = (\frac{x}{w}, \frac{y}{w})$. The gradient of the composite function $I \circ \mathbf{p}$ at point $\mathbf{u} = (x, y, w)^\top \in \mathbb{P}^2$ is computed using the chain rule,

$$\begin{aligned}
\left[\left.\frac{\partial I(\mathbf{p}(\hat{\mathbf{x}}))}{\partial \hat{\mathbf{x}}}\right|_{\hat{\mathbf{x}}=\mathbf{u}}\right] &= \left[\left.\frac{\partial I(\hat{\mathbf{p}})}{\partial \hat{\mathbf{p}}}\right|_{\hat{\mathbf{p}}=\mathbf{p}(\mathbf{u})}\right]^\top \left[\left.\frac{\partial \mathbf{p}(\hat{\mathbf{x}})}{\partial \hat{\mathbf{x}}}\right|_{\hat{\mathbf{x}}=\mathbf{u}}\right]^\top, \\
&= \begin{bmatrix}\nabla_i I & \nabla_j I\end{bmatrix}^\top \begin{bmatrix}\frac{1}{x} & 0 & -\frac{x}{w^2} \\ 0 & \frac{1}{y} & -\frac{y}{w^2}\end{bmatrix}, \\
&= \underbrace{\begin{bmatrix}\frac{1}{w}\nabla_i I & \frac{1}{w}\nabla_j I & -\frac{1}{w^2}(x\nabla_i I + y\nabla_j I)\end{bmatrix}}_{\mathbf{l}}.
\end{aligned}$$

Vector $\mathbf{l}$ is a line incident with $\mathbf{u}$, since $\mathbf{l}^\top \mathbf{u} = 0$. □

In particular, as the following corollary states, the motion model used in (3) satisfies Prop. 3.1. Many other transformations such as e.g. motion parallax do not.

Corollary. *The GEE holds in $\mathbb{P}^2$ when the motion model is a homography.*

Proof. It is trivial from (7).

□

3.2. Using model texture

Many previous approaches to 3D tracking, e.g. [7, 15], use a reference image as model for extracting texture information for tracking. This information is valid in the neighbourhood of the reference image. The tracker performance degrades for large rotations [14, 19]. Here we introduce a procedure that uses the morphable model texture for the minimisation in (6).

According to Prop (3.1) and its corollary we can use the GEE if our motion function is a homography. Hence, it can be a combination of several homographies. Let us assume that $\mathtt{H}_0^{(i)}$ is the homography between texture co-ordinate $\mathbf{v}^{(i)}$ and its projection onto the reference image $\mathbf{u}_r^{(i)}$ (see Fig. 1 (a-c)). Then, we can express (4) as $\mathbf{T}[\mathbf{p}(\mathbf{v})] = \mathbf{I}_t[\mathbf{p}(\mathbf{f}'(\mathbf{v}, \boldsymbol{\mu}_t))]$, with $\mathbf{f}'(\mathbf{v}, \boldsymbol{\mu}_t) = \mathtt{H}_t\mathtt{H}_0\mathbf{v}$. We may now rewrite equations (6)-(7) using texture information instead of a reference image. Now, from $\mathtt{J}(\boldsymbol{\mu}_t)$ and and (7) we get

$$\mathtt{J}(\boldsymbol{\mu}_t) = \left[\frac{\partial \mathbf{I}_t[\mathbf{p}(\hat{\mathbf{y}})]}{\partial \hat{\mathbf{y}}}\bigg|_{\hat{\mathbf{y}}=\mathbf{f}'(\mathbf{v},\boldsymbol{\mu}_t)}\right]\left[\frac{\partial \mathbf{f}'(\mathbf{v},\hat{\boldsymbol{\mu}})}{\partial \hat{\boldsymbol{\mu}}}\bigg|_{\hat{\boldsymbol{\mu}}=\boldsymbol{\mu}_t}\right],$$

$$\frac{\partial \mathbf{T}[\mathbf{p}(\hat{\mathbf{x}})]}{\partial \hat{\mathbf{x}}}\bigg|_{\hat{\mathbf{x}}=\mathbf{v}} = \left[\frac{\partial \mathbf{I}_t[\mathbf{p}(\hat{\mathbf{y}})]}{\partial \hat{\mathbf{y}}}\bigg|_{\hat{\mathbf{y}}=\mathbf{f}'(\mathbf{v},\boldsymbol{\mu}_t)}\right]\left[\frac{\partial \mathbf{f}'(\hat{\mathbf{x}},\boldsymbol{\mu}_t)}{\partial \hat{\mathbf{x}}}\bigg|_{\hat{\mathbf{x}}=\mathbf{v}}\right].$$

From where we can express $\mathtt{J}(\boldsymbol{\mu}_t)$ in terms of the model texture

$$\mathtt{J}(\boldsymbol{\mu}_t) = \underbrace{\left[\frac{\partial \mathbf{T}[\mathbf{p}(\hat{\mathbf{x}})]}{\partial \hat{\mathbf{x}}}\bigg|_{\hat{\mathbf{x}}=\mathbf{v}}\right]}_{\mathtt{D}}\underbrace{\left[\frac{\partial \mathbf{f}'(\hat{\mathbf{x}},\boldsymbol{\mu}_t)}{\partial \hat{\mathbf{x}}}\bigg|_{\hat{\mathbf{x}}=\mathbf{v}}\right]^{-1}}_{\mathtt{T}^{-1}}\underbrace{\left[\frac{\partial \mathbf{f}'(\mathbf{x},\hat{\boldsymbol{\mu}})}{\partial \hat{\boldsymbol{\mu}}}\bigg|_{\hat{\boldsymbol{\mu}}=\boldsymbol{\mu}_t}\right]}_{\mathtt{F}}. \tag{8}$$

3.3. Efficient optimisation by factorisation

The result introduced in (8) improves the performance of the tracker, since the texture gradients may be computed off-line (matrix $\mathtt{D}$ is constant). However, we still have to compute the derivatives $\mathtt{T}^{-1}$ and $\mathtt{F}$ for each coordinate in the target region. Hager and Bellhumeur, [8], solved the problem by factorising $\mathtt{DT}^{-1}\mathtt{F}$ into the product of two matrices, $\mathtt{DT}^{-1}\mathtt{F} = \mathtt{SM}$. Matrix $\mathtt{S}$ depends only on the structure of the target (vertexes, plane normal, etc.), and matrix $\mathtt{M}$ depends on the motion parameters. In this section we introduce a proposition that provides sufficient conditions under which the factorisation can always be achieved. Then we compute a factorisation for the motion model used in our tracker.

Lemma 3.2. *We may reorder arbitrarily a sequence of matrix operations (sums, products, reshapings and rearrangements) using reversible operators.*

For a proof of the previous lemma see [4]. In consequence:

Proposition 3.3. $\mathtt{DT}^{-1}\mathtt{F}$ *can be factored into* $\mathtt{DT}^{-1}\mathtt{F} = \mathtt{SM}$ *if it can be expressed in terms of a sequence of matrix operations (sums, products, reshapings and rearrangements).*

Proof. It is immediate from Lemma 3.2. □

We use operators such as Kronecker product, $\otimes$, and column vectorisation $\mathrm{vec}(\mathtt{A})$ [9, 4]. We also introduce the operator $\odot$ which performs a row-wise Kronecker product of two matrices. We give details in appendix A.

3.4. Factorisation of $\mathtt{J}(\boldsymbol{\mu}_t)$

The starting point is to identify the structure and motion terms for matrices $\mathtt{T}^{-1}$ and $\mathtt{F}$. The first term can be inverted using the Sherman-Morrison rule, then

$$\begin{aligned}\mathtt{T}^{-1} =& [\mathtt{H}_t\mathtt{H}_0]^{-1} = \mathtt{H}_0^{-1}\mathtt{K}\left[\mathtt{R}_t + (\mathtt{R}_t\mathtt{B}\mathbf{c}_t + \mathbf{t}_t)\frac{\mathbf{n}_r^\top}{d_r}\right]^{-1}\mathtt{K}^{-1} \\ \propto& \mathtt{H}_0^{-1}\mathtt{K}\left[\mathtt{I}_3 + \mathbf{n}^\top(\mathtt{B}\mathbf{c}_t - \mathbf{t})\mathtt{I}_3 + (\mathbf{t} - \mathtt{B}\mathbf{c}_t)\mathbf{n}^\top\right]\mathtt{R}^\top\mathtt{K}^{-1},\end{aligned} \tag{9}$$

where $\mathbf{n} = \frac{\mathbf{n}_r^\top}{d_r}$, $\mathbf{t} = -\mathtt{R}_t^\top\mathbf{t}_t$ and $\mathtt{R} = \mathtt{R}_t$. The Jacobian of the motion model is given by

$$\mathtt{F} = \mathtt{K}\left[\begin{array}{c|c|c} \dot{\mathtt{R}}\left(\mathtt{K}^{-1}\mathbf{u} + z\mathtt{B}\mathbf{c}_t\right) & z\mathtt{I}_3 & z\mathtt{RB}\end{array}\right], \tag{10}$$

where $z = \mathbf{n}^\top\mathtt{K}^{-1}\mathtt{H}_0\mathbf{v}$ is a constant term for each point and $\dot{\mathtt{R}}$ represents the tensor derivative of rotation matrix $\mathtt{R}$. The product of these two matrices is rearranged using the lemmas given in appendix A. Motion terms are globally common for the whole target, so we can stack all constant terms for each target coordinate into a single matrix $\mathtt{S}$, which multiplies a matrix built from the motion terms, $\mathtt{J} = \mathtt{SM}$, with

$$\mathtt{S} = \left[\begin{array}{c|c|c|c|c} \mathtt{S}_{(1,1)} & \mathtt{S}_{(1,2)} & \mathtt{S}_{(1,3)} & \mathtt{S}_{(1,4)} & \mathtt{S}_{(1,5)} \\ & & \vdots & & \\ \mathtt{S}_{(N,1)} & \mathtt{S}_{(N,2)} & \mathtt{S}_{(N,3)} & \mathtt{S}_{(i,4)} & \mathtt{S}_{(N,5)}\end{array}\right]$$

and

$$\mathtt{M} = \begin{bmatrix}\mathtt{M}_{\alpha_t} & \mathbf{0} & \mathbf{0} & 0 & 0 \\ \mathbf{0} & \mathtt{M}_{\beta_t} & \mathbf{0} & 0 & 0 \\ \mathbf{0} & \mathbf{0} & \mathtt{M}_{\gamma_t} & 0 & 0 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathtt{M}_{\mathbf{t}} & 0 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & 0 & \mathtt{M}_{\mathbf{c}}\end{bmatrix}.$$

In appendix B we give more details on the derivation.

4. Tracking algorithm

In this section we present the complete tracking algorithm (see Alg. 1). The off-line part of the algorithm computes constant matrices $\mathtt{S}$ and Λ, corresponding to each of the 3D model vertexes (see lines 2 and 3). The on-line part (lines 7 to 19) first determines the set of visible 3D points, then performs image rectification and Gauss-Newton iteration, grey level error computation, motion parameters' increment computation and, finally, additive parameters update .

Algorithm 1 Non-rigid 3D tracking algorithm.

Off-line:

1: **for all** $\mathbf{x}$ 3D vertex in the model **do**
2: Compute and store $\mathtt{S}(\mathbf{x})$.
3: Compute and store $\Lambda(\mathbf{x}) = \mathtt{S}(\mathbf{x})^{\top}\mathtt{S}(\mathbf{x})$.
4: **end for**

On-line:

5: Initialise motion parameters $\boldsymbol{\mu}$.
6: **for all** $\mathbf{I}$, image in a sequence **do**
7: Compute $\mathtt{V}$, the set of visible 3D vertexes (Z-buffer algorithm).
8: **while** no convergence **do**
9: Compute $\mathtt{M}(\boldsymbol{\mu})$.
10: **for all** $\mathbf{x} \in \mathtt{V}$ **do**
11: Compute $\mathbf{e} = \mathbf{I}_t[\mathbf{p}(\mathbf{f}'(\mathbf{v}, \boldsymbol{\mu}))] - \mathbf{T}(\mathbf{p}(\mathbf{v}))$.
12: Compute and store $\mathtt{A}(\mathbf{x}) = \mathtt{M}(\boldsymbol{\mu})^{\top}\mathtt{S}(\mathbf{x})^{\top}\mathbf{e}$
13: **end for**
14: Compute $\Lambda = \sum_{\mathbf{x} \in V} \Lambda(\mathbf{x})$.
15: Compute $\mathtt{N} = \mathtt{M}(\boldsymbol{\mu})^{\top}\Lambda\mathtt{M}(\boldsymbol{\mu})$.
16: Compute $\mathtt{A} = \sum_{\mathbf{x} \in V} \mathtt{A}(\mathbf{x})$.
17: Compute $\delta\boldsymbol{\mu} = -\mathtt{N}^{-1}\mathtt{A}$.
18: Update $\boldsymbol{\mu} = \boldsymbol{\mu} + \delta\boldsymbol{\mu}$.
19: **end while**
20: **end for**

5. Experiments

5.1. Rigid tracking

The goal of these experiments is to test the validity of our tracker. We use a synthetic video sequences created from two morphable models: a textured tea-box comprising $30,000$ triangles (Fig. 2, row 1) and a human head (Fig. 2, row 3). We render the sequences using `pov-ray`[1].

The textured tea-box rotates around each reference axis for almost 360 degrees in a 600 frames sequence. We compute the piecewise affine transformation between the texture and reference coordinates, and the structure matrix $\mathtt{S}$. The algorithm can successfully track the target for extreme rotations even when the model texture was not seen in the reference image (see Fig. 2, row 2). This confirms experimentally the theoretical correctness of the algorithm. The human head is a $\approx 150,000$ triangles model. The sequence is 400 frames long and we rotate the head in the Y axis $\pm 70^o$, then in the X axis $\pm 60^o$ and finally a joint rotation in both X and Y for $\pm 30^o$. The tracking is reliable even for extreme rotations (see Fig. 2, rows 4-5), particularly for rotations in the X axis, which are specially difficult given the lack of texture.

[1] Freely available at http://www.povray.org

5.2. Non-rigid tracking

The performance of the tracker estimating non-rigid motion is verified using both synthetic and real-world sequences. The synthetic sequence comprises the first 140 frames of a sequence containing a deformable model. The model performs set of prototypical expressions whilst rotating around Y axis 40^o. The deformable model is built from our head model (see Fig. 3, row 1), which is down-sampled to barely $11,000$ triangles. Then, the model mesh is animated using linear muscles as in [13]. The linear basis of deformation, $\mathtt{B}_s$, are computed using PCA over the generated meshes. We can see that the tracker is able to cope successfully with all the range of expressions, which proves the reliability of the method (see Fig. 3, rows 2-3).

Finally, we process a real example involving translations, rotations and face deformations. We use the previously generated shape basis to model the deformations of the face. These basis were generated synthetically from a graphical model. The model is initialised by fitting the 3D Morphable Model to the first image of the sequence [14]. The algorithm performs quite well during the rigid part. The non rigid section performs also remarkably well, except in those parts of the sequence in which the actor performs expressions which were not included in the training set (see Fig. 3, row 4).

5.3. Timing results

We now compare the computation time of our algorithm with a Lucas-Kanade implementation [10]. Both algorithms were implemented using `MATLAB`. An important issue in our algorithm is that, after Jacobian factorisation, we get heavily sparse matrices (approximately 70% of the elements are zero). We have partially considered this fact in our implementation, which could be further improved using techniques such as e.g. those in [16].

For the rigid case we use the 600 frames long tea-box sequence. We compute both the time to process a single frame and the number of iterations to reach the frame optimum for each algorithm. We restrict the maximum number of iterations per frame to 10 although the algorithm may finish earlier. Timing results yield an average time per frame of 0.6543 seconds for our technique and 3.4529 for the Lucas-Kanade algorithm.

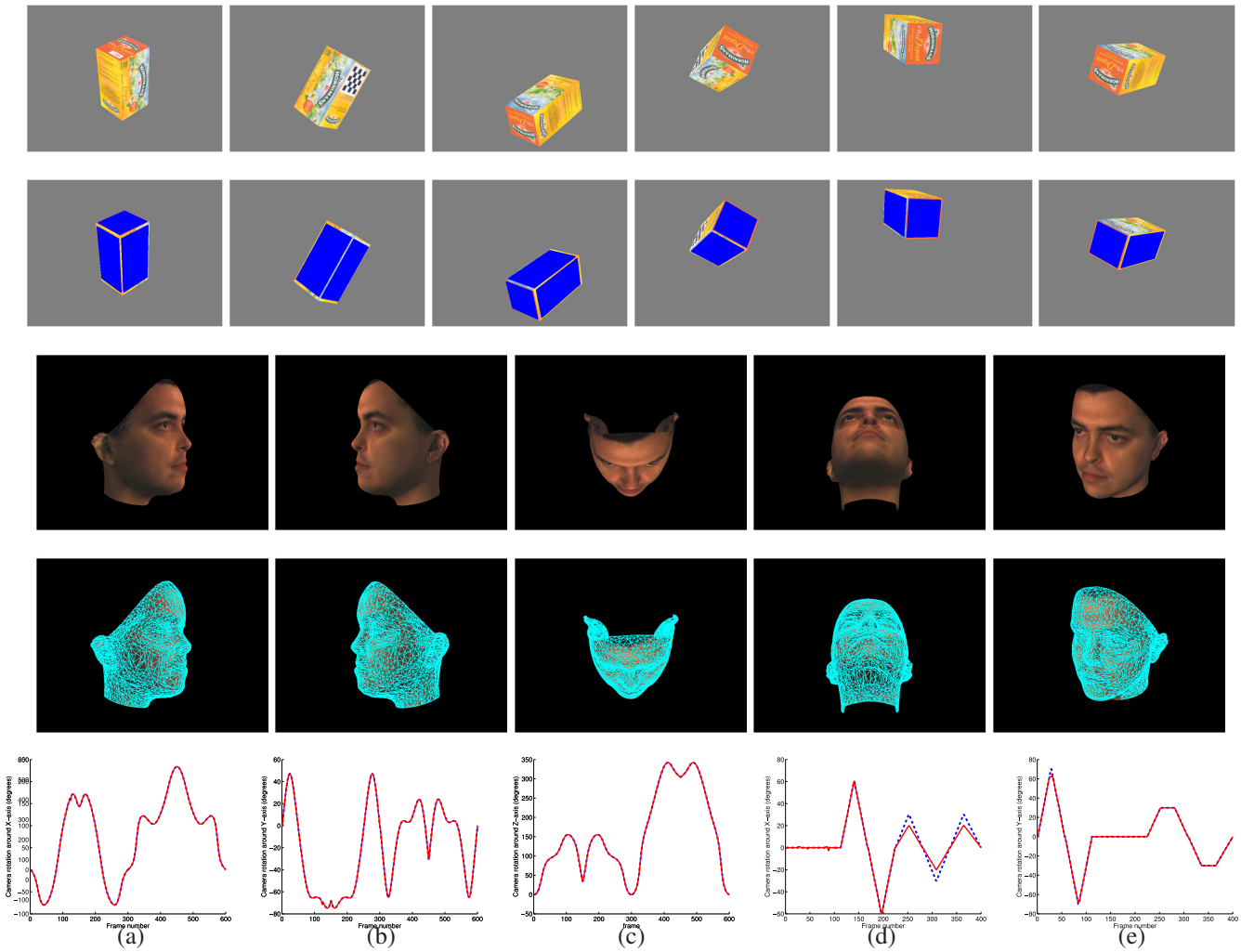

Figure 2. First row: selected frames from the synthetic tea-box sequence. Second row: blue dots represent the projections of the model vertexes using values estimated from the tracker. Third row: selected frames from the synthetic head sequence showing extreme motion. Fourth row: tracking results from the head sequence. A down-sampled mesh is projected onto each image showing the motion estimation from the tracker. Fifth row: Rotation estimation from the tracker vs. ground truth for the tea-box, (a)-(c), and the head sequences, (d)-(e).

We compute timing results for the non-rigid case using the synthetic deforming face image sequence. We restrict the optimisation to 20 iterations per frame. In this case we require 1.892 seconds to process one frame, whereas the Lucas-Kanade algorithm needs 3.563 seconds. For the non-rigid case, our algorithm is only twice as fast as Lucas and Kanade. The decrease in performance compared to the rigid case is caused by the higher complexity and size of the matrices involved in the factorisation.

6. Conclusion and future work

In this paper we have introduced an efficient procedure for fitting a 3D MM to a target image sequence. It generalises Hager and Belhumeur's efficient factorisation algorithm to the case of a 3D deforming target. One of the major limitations of the factorisation-based approach is that it is very difficult to say whether the algorithm can be used for a particular motion model [1]. In this paper we have provided necessary and sufficient conditions that determine the applicability of the algorithm for a given motion model.

We compute the target image gradients off-line, by expressing them in terms of the gradients of the MM texture. Also, by using the MM texture as model template, instead of a reference image, we are able to track the target for 360 degrees rotations. As far as we know, our approach is the only direct method that is able to track all 360 degrees without drift. This result was previously achieved by Vaccheti et al using a feature-based approach with a set of key-frames [17]. In this respect, our model may be seen

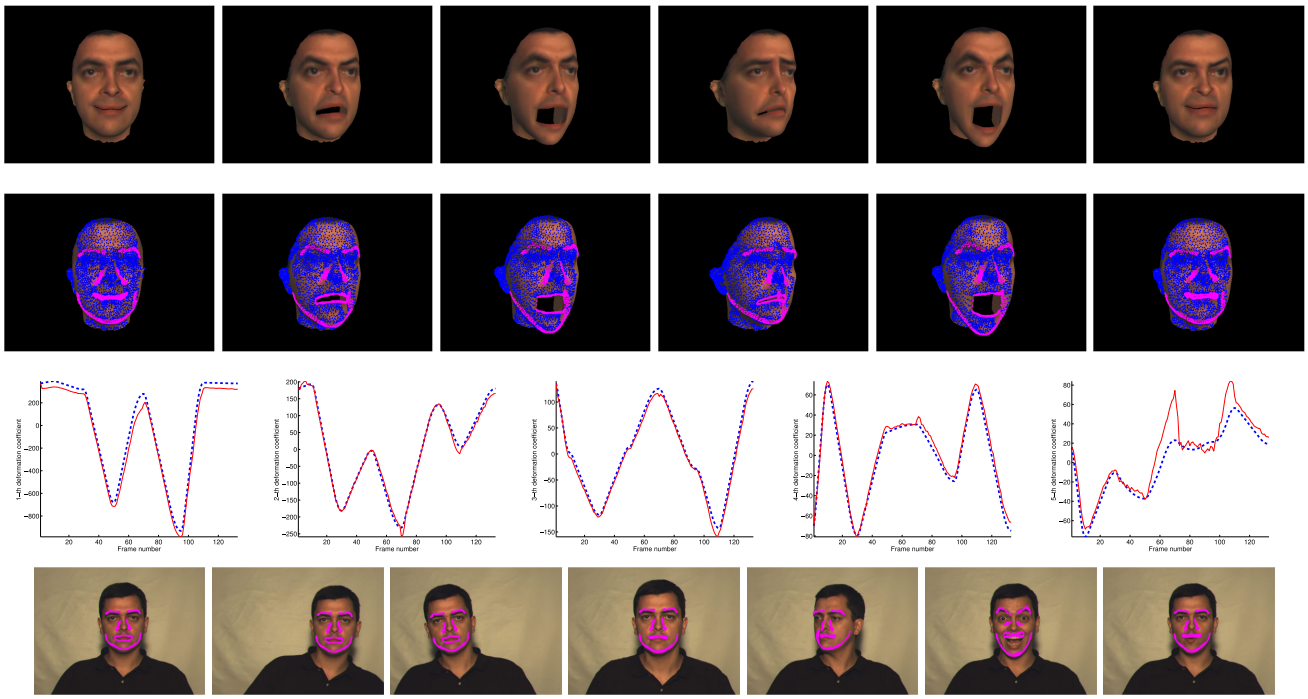

Figure 3. First row: selected frames from the synthetic deformable sequence showing some of the rendered expressions. Second row: Results from the tracker using linear deformation basis: blue dots stand for the projections of the model vertexes. We also coloured in magenta several key vertexes in eyebrows, jaw, lips and nasolabial wrinkles to show the tracked expression. Third row: plots of the first five deformation coefficients. Fourth row: results for a real sequence. We outlined several key vertexes to show the quality of the estimation.

as a generalisation of the key-frames in [17] to the case in which we had one key-frame per surface normal. This was experimentally tested in the tea-box image sequence.

Many non-rigid tracking algorithms, e.g. [11, 14], use weak-perspective camera models to decouple rigid and non-rigid motion components. Our tracking algorithm naturally deals with the more general projective camera model.

We have conducted various synthetic and real experiments validating the robustness of the tracker for large rotations, self occlusions and strong non-rigid deformations.

A future research venue would consider tracking robust to illumination changes. This could be solved using the MM texture basis to model illumination variations. This new model could be efficiently fitted to the image sequence using the factorisation scheme introduced in this paper. A related solution for the 2D planar case has been recently introduced in [5].

Acknowledgements

The authors gratefully acknowledge funding from the Spanish Ministerio de Ciencia e Innovación, project TIN2008-06815-C02-02, and the Consolider Ingenio Program, project CSD2007-00018. We also thank Sami Romdhani and Thomas Vetter for providing some of the models used in our experiments.

References

[1] S. Baker and I. Matthews. Lucas-kanade 20 years on: A unifiying framework. *IJCV*, 56(3):221–255, 2004. 6

[2] V. Blanz and T. Vetter. A morphable model for the synthesis of 3d faces. In *SIGGRAPH*, pages 187–194, 1999. 1, 2

[3] V. Blanz and T. Vetter. Face recognition based on fitting a 3d morphable model. *Trans. on PAMI*, 25(9):1–12, 2003. 1

[4] M. Brand and R.Bhotika. Flexible flow for 3d nonrigid tracking and shape recovery. In *CVPR*, v 1, pp 315–322, 2001. 4

[5] J. Buenaposada, E. Muñoz, and L. Baumela. Efficient illumination independent appearance-based face tracking. *IVC*, 27(5):560–578, 2009. 7

[6] T. Cootes, G. Edwards, and C. Taylor. Active appearance models. *Trans. on PAMI*, 23(6):681–685, 2001. 1

[7] D. Decarlo and D. Metaxas. Optical flow constraints on deformable models with applications to face tracking. *IJCV*, 38(2):99–127, 2000. 4

[8] G. Hager and P. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *Trans. on PAMI*, 20(10):1025–1039, 1998. 2, 4

[9] M. P. K. B. Petersen. The matrix cookbook. 4

[10] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI*, pp 674–679, 1981. 3, 5

[11] I. Matthews and S. Baker. Active appearance models revisited. *IJCV*, 60(2):135–164, 2004. 1, 7

[12] I. Matthews, J. Xiao, and S. Baker. 2d vs. 3d deformable face models: Representational power, construction, and real-time fitting. *IJCV*, 75(1):93–113, 2007. 1

[13] F. I. Parke and K. Waters. *Computer Facial Animation*. AK Peters Ltd, 1996. 5

[14] S. Romdhani and T. Vetter. Efficient, robust and accurate fitting of a 3d morphable model. In *ICCV*, v 1, pp 59–66, 2003. 1, 4, 5, 7

[15] W. Sepp. Efficient tracking in 6-dof based on the image-constancy assumption in 3-D. In *ICPR*, 2006. 4

[16] W. Sepp and G. Hirzinger. Real-time texture-based 3-d tracking. In *DAGM*, pp 330–337, 2003. 5

[17] L. Vacchetti, V. Lepetit, and P. Fua. Stable real-time 3d tracking using online and offline information. *Trans. on PAMI*, 26(10):1385–1391, 2004. 6, 7

[18] D. Vlasic, M. Brand, H. Pfister, and J. Popović. Face transfer with multilinear models. *ACM Trans. Graph.*, 24(3):426–433, 2005. 1

[19] Y. Xu and A. K. Roy-Chowdhury. Inverse compositional estimation of 3d pose and lighting in dynamic scenes. *Trans. on PAMI*, 30(7):1300 – 1307, 2008. 2, 4

[20] W. Zhang, Q. Wang, and X. Tang. Real time feature based 3-d deformable face tracking. In *ECCV*, 2008. 1

A. Matrix lemmas

Lemma A.1. *Let* $\mathtt{A}$ *and* $\mathtt{B}$ *be* $m \times n$ *and* $n \times p$ *matrices respectively. Their product* $\mathtt{AB}$ *can be rewritten as:* $\mathtt{AB} = \left(\mathtt{I}_m \otimes vec(\mathtt{B})^\top\right)\left(\mathtt{I}_m \odot \mathtt{A}\right)$*, with* $\mathtt{I}_m \odot \mathtt{A} = \begin{pmatrix} \mathtt{I}_m \otimes \mathbf{a}_1 \\ \vdots \\ \mathtt{I}_m \otimes \mathbf{a}_m \end{pmatrix}$*, being* $\mathbf{a}_i^\top$ *matrix* $\mathtt{A}$ *rows.*

Next results, which are used in our derivations, follow from the above lemma

- **Corollary 1:**

$$\mathtt{A}_{m\times n}\mathbf{b}_{n\times 1} = \left(\mathtt{I}_m \otimes \mathbf{b}^\top\right)\mathrm{vec}(\mathtt{A}^\top).$$

- **Corollary 2:**

$$(\mathbf{a}_{n\times 1}^\top \mathbf{b}_{n\times 1})\mathtt{I}_m = (\mathtt{I}_m \otimes \mathbf{a}^\top)(\mathtt{I}_m \otimes \mathbf{b}).$$

B. Deriving the factorisation equations

The goal is to rearrange the product of equations 9 and 10 such we separate the structure from the motion. For each target coordinate, i, we have the product two 3×3 by 3×6 matrices. Each column of this product is then factorised using lemmas from appendix A into a matrix containing only constant terms, $\mathtt{S}_i$, and a matrix depending on the current target motion, $\mathtt{M}_i$:

$$\mathtt{J}^{(i)} = \mathtt{D}_i\mathtt{K}\left[\,\mathtt{S}_{(i,1)}\mathtt{M}_{\alpha_t}\,\middle|\,\mathtt{S}_{(i,2)}\mathtt{M}_{\beta_t}\,\middle|\,\mathtt{S}_{(i,3)}\mathtt{M}_{\gamma_t}\,\middle|\,\mathtt{S}_{(i,4)}\mathtt{M}_{\mathbf{t}}\,\middle|\,\mathtt{S}_{(i,5)}\mathtt{M}_{\mathbf{c}}\,\right],$$

where the first three products refer to the rotation parameters, the fourth product depends on the translation and the remaining K columns, $\mathtt{S}_{(i,5)}\mathtt{M}_5$, depend on the shape basis. Structure terms for each target coordinate are:

$$\mathtt{S}_{(i,\{1,2,3\})}^\top = \begin{bmatrix} \mathtt{I}_3 \otimes (\mathtt{H}_0\mathbf{v})^\top \\ \left(\mathtt{I}_3 \otimes \mathbf{n}_i^\top \mathtt{B}_i\right)\left(\mathtt{I}_{3K} \otimes (\mathtt{H}_0\mathbf{v})^\top\right) \\ \left(\mathtt{I}_3 \otimes \mathbf{n}_i^\top\right)\left(\mathtt{I}_9 \otimes (\mathtt{H}_0\mathbf{v})^\top\right) \\ \mathtt{B}_i\left(\mathtt{I}_K \otimes \mathbf{n}_i^\top\right)\left(\mathtt{I}_{3K} \otimes (\mathtt{H}_0\mathbf{v})^\top\right) \\ \left(\mathtt{I}_3 \otimes \mathrm{vec}(\mathbf{n}_i(\mathtt{H}_0\mathbf{v})^\top)^\top\right) \\ \left(\mathtt{I}_3 \otimes \mathrm{vec}(z_i\mathtt{B}_i^\top)^\top\right) \\ \left(\mathtt{I}_3 \otimes \mathbf{n}_i^\top \mathtt{B}_i\right)\left(\mathtt{I}_{3K} \otimes \mathrm{vec}(z_i\mathtt{B}_i^\top)^\top\right) \\ z_i\left(\mathtt{I}_3 \otimes \mathbf{n}_i^\top\right)\left(\mathtt{I}_9 \otimes \mathrm{vec}(\mathtt{B}_i^\top)^\top\right) \\ \mathtt{B}_i\left(\mathtt{I}_k \otimes \mathbf{n}_i^\top\right)\left(\mathtt{I}_{3K} \otimes \mathrm{vec}(z_i\mathtt{B}_i^\top)^\top\right) \\ z_i\left(\mathtt{I}_3 \otimes \mathbf{n}_i^\top\right)\left(\mathtt{I}_9 \otimes \mathrm{vec}(\mathtt{B}_i^\top)^\top\right) \end{bmatrix}$$

$$\mathtt{S}_{(i,4)}^\top = \begin{bmatrix} z_i\mathtt{B}_i \\ z_i\mathtt{I}_3 \otimes \mathbf{n}_i^\top \mathtt{B}_i \\ z_i\mathtt{I}_3 \otimes \mathbf{n}_i^\top \\ z_i\mathtt{B}_i\mathtt{I}_K \otimes \mathbf{n}_i^\top \\ z_i\mathtt{I}_3 \otimes \mathbf{n}_i^\top \end{bmatrix},$$

$$\mathtt{S}_{(i,5)}^\top = \begin{bmatrix} z_i\mathtt{B}_i \\ z_i\left(\mathtt{I}_3 \otimes \mathbf{n}_i^\top \mathtt{B}_i\right)\left(\mathtt{I}_{3K} \otimes \mathrm{vec}(\mathtt{B}_i)^\top\right) \\ z_i\left(\mathtt{I}_3 \otimes \mathbf{n}_i^\top\right)\left(\mathtt{I}_9 \otimes \mathrm{vec}(\mathtt{B}_i)^\top\right) \\ z_i\mathtt{B}_i\left(\mathtt{I}_K \otimes \mathbf{n}_i^\top\right)\left(\mathtt{I}_{3K} \otimes \mathrm{vec}(\mathtt{B}_i)^\top\right) \\ z_i\left(\mathtt{I}_3 \otimes \mathbf{n}_i^\top \mathtt{B}_i\right) \end{bmatrix}.$$

We build motion matrices similarly:

$$\mathtt{M}_{\Delta,\Delta\in\{\alpha_t,\beta_t,\gamma_t\}} = \begin{bmatrix} \mathrm{vec}(\mathtt{K}^{-\top}\dot{\mathtt{R}}_\Delta^\top \mathtt{R}) \\ \mathrm{vec}(\mathtt{K}^{-\top}\dot{\mathtt{R}}_\Delta^\top \mathtt{R}\left(\mathtt{I}_3 \otimes \mathbf{c}_t^\top\right)) \\ -\mathrm{vec}(\mathtt{K}^{-\top}\dot{\mathtt{R}}_\Delta^\top \mathtt{R}\left(\mathtt{I}_3 \otimes \mathbf{t}^\top\right)) \\ -\mathrm{vec}(\mathtt{K}^{-\top}\dot{\mathtt{R}}_\Delta^\top \mathtt{R}\left(\mathbf{c}_t^\top \otimes \mathtt{I}_3\right)) \\ \mathrm{vec}(\mathrm{vec}(\mathtt{R}^\top\dot{\mathtt{R}}_\Delta\mathtt{K}^{-1})\mathbf{t}^\top) \\ \mathrm{vec}(\left(\mathtt{I}_3 \otimes \mathbf{c}_t^\top\right)\dot{\mathtt{R}}_\Delta^\top \mathtt{R}) \\ \mathrm{vec}((\mathbf{c}_t \otimes \mathtt{I}_3)\dot{\mathtt{R}}_\Delta^\top \mathtt{R}\left(\mathtt{I}_3 \otimes \mathbf{c}_t^\top\right)) \\ -\mathrm{vec}((\mathbf{c}_t \otimes \mathtt{I}_3)\dot{\mathtt{R}}_\Delta^\top \mathtt{R}\left(\mathtt{I}_3 \otimes \mathbf{t}^\top\right)) \\ -\mathrm{vec}((\mathtt{I}_3 \otimes \mathbf{c}_t)\dot{\mathtt{R}}_\Delta^\top \mathtt{R}\left(\mathbf{c}_t^\top \otimes \mathtt{I}_3\right)) \\ \mathrm{vec}((\mathtt{I}_3 \otimes \mathbf{c}_t)\dot{\mathtt{R}}_\Delta^\top \mathtt{R}\left(\mathbf{t}^\top \otimes \mathtt{I}_3\right)) \end{bmatrix}$$

$$\mathtt{M}_{\mathbf{t}} = \begin{bmatrix} \mathtt{R}^\top \\ \left((\mathtt{I}_3 \otimes \mathbf{c}_t)\,\mathtt{R}^\top\right) \\ -\left((\mathtt{I}_3 \otimes \mathbf{t})\,\mathtt{R}^\top\right) \\ -\left((\mathbf{c}_t \otimes \mathtt{I}_3)\,\mathtt{R}^\top\right) \\ \left((\mathbf{t} \otimes \mathtt{I}_3)\,\mathtt{R}^\top\right) \end{bmatrix}, \mathtt{M}_{\mathbf{c}} = \begin{bmatrix} \mathtt{I}_K \\ \left((\mathtt{I}_3 \otimes \mathbf{c}_t) \odot \mathtt{I}_K\right) \\ -\left((\mathtt{I}_3 \otimes \mathbf{t}) \odot \mathtt{I}_K\right) \\ -\left((\mathbf{c}_t \otimes \mathtt{I}_3) \odot \mathtt{I}_K\right) \\ \left((\mathbf{t} \otimes \mathtt{I}_K)\right) \end{bmatrix}$$