Constrained 3D Rotation Smoothing via Global Manifold Regression for Video Stabilization

Chao Jia, *Student Member, IEEE,* and Brian L. Evans, *Fellow, IEEE*

***Abstract*—We present a novel motion smoothing algorithm for hand-held cameras with application to video stabilization. Video stabilization seeks to remove unwanted frame-to-frame jitter due to camera shake. For video stabilization, we use a pure 3D rotation motion model with known camera projection parameters. The 3D camera rotation can be reliably tracked by a gyroscope as commonly found on a smart phone or tablet. In this paper we directly smooth the sequence of camera rotation matrices for the video frames by exploiting the Riemannian geometry on a manifold. Our contributions are (1) formulation of motion smoothing as a geodesic-convex constrained regression problem on a non-linear manifold based on geodesic distance, (2) computation of gradient and Hessian of the objective function using Riemannian geometry for gradient-related manifold optimization, and (3) generalization of the two-metric projection algorithm in Euclidean space to manifolds to solve the proposed manifold optimization problem efficiently. The geodesic-distance-based smoothness metric better exploits the manifold structure of sequences of rotation matrices. The geodesic-convex constraints effectively guarantee that no black borders intrude into the stabilized frames. The proposed manifold optimization algorithm can find the global optimal solution in only a few iterations. Experimental results show that video stabilization based on our motion smoothing algorithm outperforms state-of-the-art methods by generating videos with less jitter and without black borders.**

***Index Terms*—Video stabilization, manifold optimization, special orthogonal group, gradient projection, geodesic convexity.**

I. Introduction

Hand-held video cameras, such as in smart phones and tablets, conveniently capture interesting or memorable moments anywhere and anytime due to their portability. Videos shot with hand-held cameras, however, often suffer from frame-to-frame jitter due to camera shake compared to videos shot with complex lens systems and camera stabilizers such as steadicams and tripods. Video stabilization aims at removing the unwanted jitter to generate visually stable and pleasant videos. Generally video stabilization consists of three major steps [2]: (1) camera motion estimation, (2) camera motion smoothing and (3) frame synthesis. In this paper we focus on the second step.

Our video stabilization algorithm is based on a 3D rotational camera motion model for a calibrated camera with a known intrinsic matrix. Compared to 2D affine or projective motion models, 3D motion models can more accurately reflect the real camera perspective projection, and thus give more realistic motion smoothing and avoid image distortion in frame synthesis. We ignore 3D translation of the camera because (1) the unwanted jitter in videos are primarily caused by camera rotation, and (2) frame synthesis with 3D camera translation would need the depth value at every pixel, which is very difficult to obtain accurately. To estimate the 3D camera rotation we use a gyroscope that is available in many smart phones and tablets. Current gyroscopes in smart phones have very high precision and can return more reliable 3D camera rotation estimates compared to the estimates obtained from visual features in the video sequence, especially when there are many moving objects in the scene or it is difficult to track feature points due to motion blur or illumination changes.

Under a 3D rotational model, camera motion for a video can be considered as a sequence of 3D rotation matrices. We formulate motion smoothing as a regression problem with a regularization term indicating the smoothness of the sequence of rotation matrices. Unlike traditional approaches, we exploit the manifold structure of the sequence of rotation matrices. The formulated problem is based on geodesic distance on the Riemannian manifold.

Due to the change of camera poses introduced by video stabilization, the stabilized frames can be only synthesized for portions of the scene that are visible in the original frames. Therefore, we have to crop the resulting video with a large enough cropping size to keep most of the content of the original video sequence while at the same time guaranteeing that no black borders intrude into the stabilized video frames. In this paper, we introduce a geodesic-convex constraint on the manifold to approximate such requirement so that the entire motion smoothing problem is kept geodesic-convex on the manifold.

Previous methods have only exploited the properties on the manifold of the individual 3D rotation matrix $\mathbf{SO(3)}$ (Special Orthogonal Group), so they can only smooth the camera motion locally through low-pass filtering. Considering the entire set of sequences of rotation matrices as a Riemannian manifold allows us to model the motion smoothing problem globally with proper constraints and solve it optimally.

To solve the formulated constrained smoothing problem on the manifold, we compute the gradient and Hessian of the objective function using Riemannian geometry, and then extend the two-metric projection algorithm in Euclidean space to non-linear manifolds. The proposed manifold optimization algorithm has much better convergence property than normal non-linear optimization algorithms in Euclidean space. Experimental results show that our motion smoothing method out-

The authors are with the Department of Electrical and Computer Engineering, The University of Texas at Austin, Austin, TX, 78712 USA (e-mail: cjia@utexas.edu; bevans@ece.utexas.edu).

This research was supported by gift funding from Texas Instruments, Dallas, TX, USA.

Parts of this paper, including Sections III, V-A, V-B and V-C, are based on our previous work [1].

performs state-of-the-art methods by generating more stable videos with less distortion.

This paper is organized as follows: Section II reviews previous video stabilization algorithms and related optimization background. Section III formulates motion smoothing as a regression problem on the sequence of rotation matrices using geodesic distance. Section IV adds hard geodesic-convex constraints to the optimization problem to guarantee that no black border will be present in the stabilized videos. Section V presents the computation of gradient and Hessian of the objective function using Riemannian geometry and then generalizes two-metric projection algorithm in Euclidean space to non-linear manifolds. Section VI shows the convergence of the proposed algorithms and compares the proposed motion smoothing method against state-of-the-art algorithms. Section VII concludes the paper. We have publicly released the Matlab code for video stabilization using the proposed motion smoothing algorithm [3].

II. Related Work

Camera motion has been commonly modeled using 2D affine or projective approaches [2], [4], [5]. Using full 3D models including both rotation and translation for calibrated cameras was first proposed in [6] and further discussed in [7]. In both papers complicated approximations are used in frame synthesis to handle the problem of missing depth values. In [8], [9] pure 3D rotational models with known intrinsic camera parameters were shown to generate high-quality results while only needing homography-based warping in frame synthesis.

Gyroscopes and other inertial measurement sensors have been widely used in robotic localization problems together with visual measurements [10], [11]. However, they were not used in video stabilization to replace the feature-based motion estimation until they became accurate enough and widely available in cell phones recently [9], [12]. Compared with camera motion estimation using only visual measurements [13], [14], estimation with inertial measurements is faster and more robust, especially for the cell phone cameras that use CMOS image sensors. In these cameras different rows in the same frame are captured sequentially from top to bottom. When there is fast relative motion between the scene and the camera, the frames can be distorted because each row was captured under a different 3D-to-2D projection. This kind of distortion is known as rolling shutter effect [15] and can easily break the multi-view geometry that are used for motion estimation from visual measurements. Using the camera motion estimated with the help of the inertial measurements [9], [12], [16], rolling shutter effect can be effectively rectified so that each frame looks as captured under a single camera pose. For the rest of the paper, we assume that any possible rolling shutter effects are rectified before video stabilization is applied. Our proposed video stabilization methods would therefore work for cameras with or without shutters.

Motion smoothing methods using 2D models are based on Euclidean distance. 2D camera motion can be smoothed using local methods such as Gaussian-kernel low-pass filtering [2], global methods such as ℓ_1-based regularization [4], and real-time methods such as Kalman filtering [17]. 3D rotation smoothing has been implemented locally by low-pass filtering based on either Euclidean distance [9] or geodesic distance on the manifold $\mathbf{SO(3)}$ [7], [12]. Motion smoothing has also been performed directly on the feature trajectories without explicitly estimating the parametric camera motion [18], [19]. These methods are actually also based on 2D motion models.

The manifold structure of 3D rotation has been extensively studied in computer graphics. It has been shown that a linear interpolation on the geodesic between two different poses gives a very smooth and natural animation of rigid body [20]. Such interpolation is equivalent to constructing a curve that minimizes the sum of geodesic distances between every pair of adjacent knots. This fact motivates our formulation of camera motion smoothing on the manifold and the use of geodesic distance as the smoothness metric.

Although $\mathbf{SO(3)}$ has additional applications in computer vision, medical imaging and robotics, the sequence of 3D rotation matrices was hardly investigated as a whole. In [21] discrete regression is first applied on the sequence of rotation matrices with conjugate gradient descent algorithm proposed to solve the formulated problem. In this paper we also directly exploit the manifold structure of sequences of rotation matrices so that we can formulate 3D rotation smoothing as a regression problem. Compared to [21], we further compute the Hessian of the objective function using Riemannian geometry so that the problem can be solved more efficiently using Newton's method on the manifold.

Previous video stabilization methods usually stabilize the video first without considering the cropping size of the result and crop the stabilized video as a post-processing step [7], [12]. Such methods cannot optimally smooth a video sequence with a pre-fixed cropping size and usually have to sacrifice the smoothness. Rendering the unseen part of the frame using mosaicking and inpainting algorithms with the help of neighboring frames allows the original size of the video to be kept [2]. However, the rendered parts usually have much lower image quality, especially for the videos with a lot of moving objects. The cropping size is first considered as a hard constraint in motion smoothing step in [4]. In this paper we approximate this constraint with a geodesic-convex set on the manifold. Constrained optimization on Euclidean space has been extensively studied [22], but not on non-linear manifolds. If the constraint set has some simple structure, such as a Cartesian product of Euclidean balls, an efficient two-metric projection algorithm can be used to solve the optimization problem [23], [24]. The proposed constraint set in this paper is a Cartesian product of geodesic balls on manifold. We extend the two-metric projection algorithm in Euclidean space to general manifolds so the proposed manifold optimization problem can be solved efficiently and optimally. Table I summarizes motion smoothing methods in prior work and in this paper for video stabilization.

III. Smoothness of 3D Rotation Sequence

All of the 3×3 rotation matrices constitute the Special Orthogonal Group $\mathbf{SO(3)}$, in which any element $\mathbf{R}$ satisfies

TABLE I
COMPARISON OF PRIOR WORK AND THIS PAPER ON MOTION SMOOTHING FOR VIDEO STABILIZATION.

Paper	Motion model	Smoothing method	Constrained by cropping size	Global smoothing
[2]	2D	Low-pass filtering	no (full-frame with inpainting)	no
[9]	3D (Euclidean)	Low-pass filtering	no	no
[7] [12]	3D (manifold)	Low-pass filtering	no	no
[4]	2D	Regression	yes	yes
[18]	2D (trajectories)	Regression	no	no
[19]	2D (trajectories)	Subspace low-pass filtering	no	no
Proposed	3D (manifold)	Regression	yes	yes

the constraints $\mathbf{R}\mathbf{R}^{\mathrm{T}} = \mathbf{I}$ and $\det(\mathbf{R}) = 1$. $\mathbf{SO(3)}$ can be also considered as an embedded Riemannian submanifold of Euclidean space $\mathbb{R}^9$ (represented as 3×3 real matrices). A natural extension of Euclidean distance in Euclidean space to the Riemannian manifold $\mathbf{SO(3)}$ is the geodesic distance

$$d_g(\mathbf{R}_i, \mathbf{R}_j) = ||\mathrm{logm}(\mathbf{R}_i^{\mathrm{T}}\mathbf{R}_j)||_F, \tag{1}$$

where $\mathrm{logm}(\cdot)$ is the matrix logarithm operator and $||\cdot||_F$ is the Frobenius norm of a matrix. In fact, $\mathrm{logm}(\mathbf{R}_i^{\mathrm{T}}\mathbf{R}_j)$ is a skew-symmetric matrix representing a tangent vector in the tangent space $T_{\mathbf{R}_i}\mathbf{SO(3)}$ that indicates the non-normalized direction from $\mathbf{R}_i$ to $\mathbf{R}_j$ on $\mathbf{SO(3)}$. Usually we also write $\mathrm{logm}(\mathbf{R}_i'\mathbf{R}_j)$ as $\log_{\mathbf{R}_i}\mathbf{R}_j$ and call it the logarithmic mapping. Inversely, given any tangent vector $\xi \in T_{\mathbf{R}_i}\mathbf{SO(3)}$, we can define $\exp_{\mathbf{R}_i}\xi = \mathbf{R}_i\mathrm{expm}(\xi)$, where $\mathrm{expm}(\cdot)$ is the matrix exponential operator. Here, $\exp_{\mathbf{R}_i}\xi$ is called the exponential mapping and is used to move $\mathbf{R}_i$ along the direction defined by ξ on $\mathbf{SO(3)}$. The logarithmic mapping and exponential mapping together define a curve

$$t \in [0,1] \mapsto \gamma(t) = \exp_{\mathbf{R}_i}\left(t \cdot \log_{\mathbf{R}_i}\mathbf{R}_j\right), \tag{2}$$

which is known as the minimizing geodesic from $\mathbf{R}_i$ to $\mathbf{R}_j$ on $\mathbf{SO(3)}$. The minimizing geodesic is a generalization of the notion of "straight line" in Euclidean space to Riemannian manifolds, representing the shortest path between two points in the manifolds given a Riemannian metric. The length of the minimizing geodesic is defined in (1).

For each video sequence, we can obtain a sequence of 3D rotation matrices corresponding to all of the frames from the gyroscope readings or the estimation using matched feature points. Next we consider the sequence of 3D rotation matrices as a whole and exploit the properties of the Riemannian manifold constituted by these sequences.

Assume $\mathbf{x} = [\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_N]^{\mathrm{T}}$ represents the sequence of 3D camera rotation for any video sequence with N frames. Clearly all of the possible rotation matrix sequences with N elements constitute a manifold $\mathcal{M}_R$ with dimension $3N$. In fact, we have

$$\mathcal{M}_R = \mathbf{SO(3)} \times \mathbf{SO(3)} \times \ldots \times \mathbf{SO(3)}, \tag{3}$$

a Cartesian product of N $\mathbf{SO(3)}$ manifolds. Furthermore, for any $\mathbf{x} \in \mathcal{M}_R$, the tangent space $T_{\mathbf{x}}\mathcal{M}_R$ at $\mathbf{x}$ can be represented as

$$[\eta_1, \eta_2, \ldots, \eta_N]^{\mathrm{T}}, \tag{4}$$

where $\{\eta_i\}$ are real skew-symmetric matrices. In other words, the tangent vectors and corresponding exponential (and logarithmic) mapping are still separable as the elements in the manifold of rotation matrix sequences. This makes the proposed gradient-related optimization algorithms in the next section easy to implement.

The goal of video stabilization is to remove the visible jitter and make the camera motion trajectory change smoothly. Given the manifold structure of $\mathbf{SO(3)}$, it is natural to define the smoothness of a rotation matrix sequence as the sum of geodesic distances between adjacent rotation matrices. At the same time, we need to guarantee that the smoothed camera motion trajectory does not deviate from the original trajectory too much. As a result, we formulate the video stabilization problem as

$$\min_{\{\mathbf{R}_i\}} \sum_{i=1}^{N} \frac{1}{2} d_g^2(\tilde{\mathbf{R}}_i, \mathbf{R}_i) + \alpha \sum_{i=1}^{N-1} \frac{1}{2} d_g^2(\mathbf{R}_i, \mathbf{R}_{i+1}), \tag{5}$$

where $\{\mathbf{R}_i\}$ is the sequence of stabilized rotation matrices, $\{\tilde{\mathbf{R}}_i\}$ is the original sequence of rotation matrices, α is the weighting parameter controlling the smoothness of the stabilized trajectory. (5) is an extension of discrete curve fitting problem in Euclidean space with penalty on the first order difference. Note that although the objective function is derived based on the geodesic distance between elements in $\mathbf{SO(3)}$, it is defined on the rotation matrix sequence manifold $\mathcal{M}_R$.

IV. CONSTRAINED VIDEO STABILIZATION

The proposed objective function in (5) is effective in smoothing the sequence of 3D rotation matrices. However, in the last step of video stabilization, the synthesized frames may contain black borders since not every pixel in the synthesized frame is visible in the original frame due to the change of camera orientation. Therefore, we have to crop the synthesized frames into a smaller size so that there are no black borders in the stabilized video. In other words, given a preferred stabilized size of the video, the video stabilization system must guarantee that every pixel in the cropped stabilized frames is visible in the original frames. This is a hard constraint that has to be considered in the camera motion smoothing algorithm.

Assume the intrinsic projection matrix of the camera is given as $\mathbf{K}$. Under pure rotational camera model, for any pixel $[u_{ij}, v_{ij}]^{\mathrm{T}}$ in the stabilized frame i, its corresponding 2D pixel location in the original frame $[\tilde{u}_{ij}, \tilde{v}_{ij}]^{\mathrm{T}}$ can be computed as

$$\begin{bmatrix} \tilde{u}_{ij} \\ \tilde{v}_{ij} \end{bmatrix} = g\left(\mathbf{K}\tilde{\mathbf{R}}_i{\mathbf{R}_i}^{\mathrm{T}}\mathbf{K}^{-1}\begin{bmatrix} u_{ij} \\ v_{ij} \\ 1 \end{bmatrix}\right), \tag{6}$$

where the function

$$g\left(\begin{bmatrix} x \\ y \\ z \end{bmatrix}\right) = \begin{bmatrix} x/z \\ y/z \end{bmatrix} \tag{7}$$

is used to convert the homogeneous coordinates into inhomogeneous coordinates. Assume that the frame size in the original video is $w \times h$, and the coordinates of the top left corner and bottom right corner of the cropped rectangle in the stabilized video are $[c_1, d_1], [c_2, d_2]$, the hard constraint for video stabilization can be represented as

$$\begin{cases} 0 \le \tilde{u}_{ij} \le w \\ 0 \le \tilde{v}_{ij} \le h \end{cases}, \forall \begin{bmatrix} u_{ij} \\ v_{ij} \end{bmatrix} s.t. \begin{cases} c_1 \le u_{ij} \le c_2 \\ d_1 \le v_{ij} \le d_2 \end{cases} \tag{8}$$

The constraint (8) is very complex with respect to the rotation matrices that we want to compute and no algorithms as far as we know are guaranteed to handle it efficiently (note that for 2D affine or similarity motion models, this constraint is just linear with respect to the variables). To overcome this difficulty we replace the constraint in (8) with a simpler constraint defined on the manifold

$$|| \log_{\tilde{\mathbf{R}}_i} \mathbf{R}_i ||_F \le r_0, \forall i, \tag{9}$$

where r_0 is a fixed threshold depending on the relative size of the cropped rectangle in the stabilized frames. The constraint (9) just means that the geodesic distance between the original and stabilized camera orientations should be less than r_0. r_0 is defined as the largest value to guarantee that for all of the camera orientations satisfying the constraint (9), the constraint (8) is also satisfied. We know that any 3D rotation matrix can be represented by a rotation axis and a rotation angle, so constraint (9) can be also interpreted as the rotation angle of $(\tilde{\mathbf{R}}_i)^{\mathrm{T}} \mathbf{R}_i^{new}$ being no larger than r_0. Constraint (9) is homogeneous on every possible rotation axis and is clearly stricter than constraint (8). Fig. 1 shows that the constraint (9) is a good approximation of the original constraint.

In this example, the original frame size is 720×480 and the cropped rectangle is at the center of the frame with size 540×360. For each possible rotation axis (denoted by the tangent vector $\log_{\tilde{\mathbf{R}}_i} \mathbf{R}_i$ after normalization), we find the maximum geodesic distance that guarantees constraint (8) is satisfied. In Fig. 1 the homogeneous constraint is shown as the sphere and the maximum allowable geodesic distance for each rotation axis (we uniformly sample 1000 rotation axes) is shown as a blue point. We also show three perpendicular views to better illustrate the difference between the two constraints. From Fig. 1 we can observe that for most rotation axes the maximum allowable geodesic distance is close to the homogeneous bound.

The constraint (9) has two significant properties. First, it has a simple form – each rotation matrix in the sequence is constrained in a geodesic ball. As a result, the constraint set is a Cartesian product of geodesic balls. This property guarantees that gradient projection algorithms can be executed efficiently, as shown in the next section. Second, the constraint set is geodesic convex – given any two points in the set, there is a minimizing geodesic contained within the set that joins those two points (The geodesic convexity of the constraint can be easily proved by the triangular inequality of Riemannian metrics). Geodesic convexity is a natural generalization of convexity in Euclidean space to Riemannian manifolds. In the next section we will prove that the objective function (5) is also geodesic convex and thus global optimality can be guaranteed by the proposed optimization algorithms.

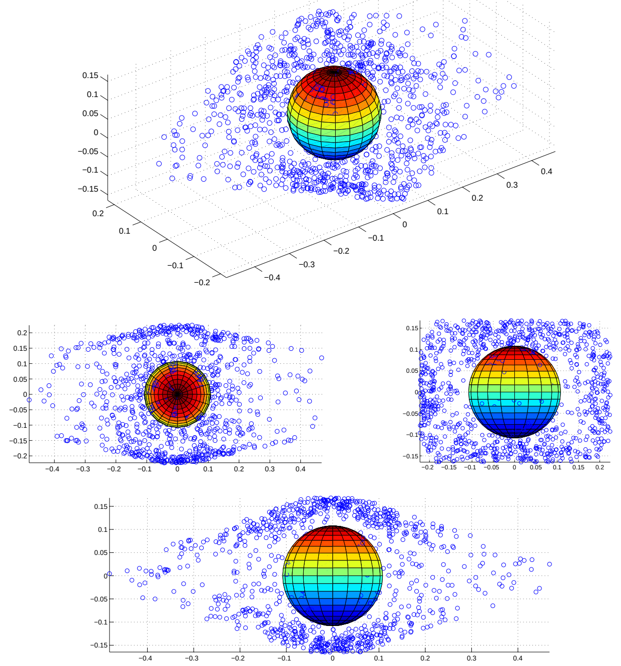
Fig. 1. Approximation of inhomogeneous constraint (8) using homogeneous constraint (9). The maximum allowable geodesic distances for different rotation axes are shown as blue points. The homogeneous geodesic distance constraint is shown as the sphere. The bottom three figures show the same as the top figure from three perpendicular views.

V. Motion Smoothing via Manifold Optimization

For brevity, we use $\mathbf{x} \in \mathcal{M}_R$ to represent the rotation matrix sequence $\{\mathbf{R}_i\}$ and write the objective function to minimize (5) as $f(\mathbf{x})$. In addition, we define $\mathbf{R}_i = A_i \mathbf{x}$, where A_i is a $3 \times 3N$ matrix that is used to extract $\mathbf{R}_i$ from $\mathbf{x}$. Similarly we can map $\mathbf{R}_i$ back to its corresponding location in $\mathbf{x}$ as $A_i^{\mathrm{T}} \mathbf{R}_i$.

The constrained motion smoothing can be finally formulated as

$$\min_{\mathbf{x}} f(\mathbf{x}), \;\; \text{s.t. } \mathbf{x} \in \Omega, \tag{10}$$

where $\Omega = \Omega_1 \times \Omega_2 \times \cdots \times \Omega_N$ is a Cartesian product of geodesic balls. Each geodesic ball is defined as in (9):

$$\Omega_i = \{ \mathbf{R}_i \in \mathbf{SO(3)} : || \log_{\tilde{\mathbf{R}}_i} \mathbf{R}_i ||_F \le r_0 \}. \tag{11}$$

A. Unconstrained Optimization

In this subsection we first ignore the constraint and only minimize the objective function in the entire manifold $\mathcal{M}_R$. We consider the constrained manifold optimization in the next subsection.

As we mentioned, we will directly solve the optimization problem using manifold optimization methods. In other words, the optimization algorithms are based on the geometric structure of the manifold, not its embedding Euclidean space. In fact, the problem in (10) without the constraint is equivalent to an unconstrained quadratic programming problem in Euclidean space. In Euclidean space, such problems have closed-form solution; however, on non-linear manifolds we have to use iterative algorithms.

Similar with Euclidean space, gradient-related iterative algorithms are widely used in optimization on manifolds [25]. The gradient-related algorithms for unconstrained optimization problem on the manifold $\mathcal{M}_R$ can be summarized as follows: For any element $\mathbf{x}$ in the manifold of rotation matrix sequence $\mathcal{M}_R$, given any tangent vector $\xi_{\mathbf{x}} \in T_{\mathbf{x}}\mathcal{M}_R$, we can move $\mathbf{x}$ along the direction defined by $\xi_{\mathbf{x}}$ using the exponential mapping $\exp_{\mathbf{x}}\xi_{\mathbf{x}}$. Note that given the separability property of the tangent vectors the exponential mapping can also be implemented separately for different rotation matrices in the sequence. If $\xi_{\mathbf{x}}$ is a descent direction related to the gradient of the objective function at $\mathbf{x}$, then we have the gradient-related algorithm on the manifold $\mathcal{M}_R$. In fact, similar convergence results of gradient-related algorithms has been extended from Euclidean space to any manifold [25]. The gradient-related algorithms can be classified according to the choice of the descent directions. Popular gradient-related algorithms include steepest gradient descent, conjugate gradient descent, Newton's method, etc.

In this paper we investigate steepest gradient descent and Newton's method, which needs the computation of gradient and Hessian of the objective function.

B. Gradient Computation

In manifold, the gradient of a function is defined as follows:

***Definition** 1:* For any real-valued function $f : \mathcal{M} \to \mathbb{R}$ defined on manifold $\mathcal{M}$, the gradient $\text{grad}f(\mathbf{x})$ is a vector field that satisfies

$$\langle \text{grad}f(\mathbf{x}), \xi_{\mathbf{x}}\rangle_{\mathbf{x}} = \text{D}f(\mathbf{x})[\xi_{\mathbf{x}}], \forall \xi_{\mathbf{x}} \in T_{\mathbf{x}}\mathcal{M}, \tag{12}$$

where $\langle\cdot,\cdot\rangle_{\mathbf{x}}$ on the left-hand side of (12) is any inner product in the tangent space $T_{\mathbf{x}}\mathcal{M}$ that induces a Riemannian metric, $\text{D}f(\mathbf{x})[\cdot]$ on the right-hand side of (12) is the differential map of f at $\mathbf{x}$.

To compute the gradient we first rewrite the objective function as

$$f(\mathbf{x}) = \sum_{i=1}^{N} g_i(\mathbf{x}) + \alpha \sum_{i=1}^{N-1} h_i(\mathbf{x}), \tag{13}$$

where $g_i(\mathbf{x}) = \frac{1}{2}d_g^2(\tilde{\mathbf{R}}_i, \mathbf{R}_i)$ and $h_i(\mathbf{x}) = \frac{1}{2}d_g^2(\mathbf{R}_i, \mathbf{R}_{i+1})$. Note that $\mathbf{R}_i = A_i\mathbf{x}$ in our notation.

If we consider $\frac{1}{2}d_g^2(\tilde{\mathbf{R}}_i, \mathbf{R}_i)$ as a function of $\mathbf{R}_i$, it has been proved [26] that

$$\text{grad}\frac{1}{2}d_g^2(\tilde{\mathbf{R}}_i, \mathbf{R}_i) = -\log_{\mathbf{R}_i}\tilde{\mathbf{R}}_i. \tag{14}$$

Given the separability feature of $\mathbf{x}$, we can further obtain

$$\text{grad}g_i(\mathbf{x}) = -A_i^{\mathrm{T}}\log_{A_i\mathbf{x}}\tilde{\mathbf{R}}_i \tag{15}$$

We propose the following lemma to compute the gradient of $h_i(\mathbf{x})$:

***Lemma** 1:* The gradient of the function $h_i(\mathbf{x})$ defined in (13) on manifold $\mathcal{M}_R$ can be represented as

$$\text{grad}h_i(\mathbf{x}) = -A_i^{\mathrm{T}}\log_{A_i\mathbf{x}} A_{i+1}\mathbf{x} - A_{i+1}^{\mathrm{T}}\log_{A_{i+1}\mathbf{x}} A_i\mathbf{x} \tag{16}$$

Proof: $\forall\xi_{\mathbf{x}} \in T_{\mathbf{x}}\mathcal{M}_R$, define a geodesic curve $\gamma(t) = \exp_{\mathbf{x}}(t\xi_{\mathbf{x}})$, then from the definition of differential map we have

$$\text{D}h_i(\mathbf{x})[\xi_{\mathbf{x}}] = \left.\frac{\text{d}h_i(\gamma(t))}{\text{d}t}\right|_{t=0}. \tag{17}$$

Now consider a family of geodesics

$$c(s,t) = \exp_{A_{i+1}\gamma(t)}(s\log_{A_{i+1}\gamma(t)} A_i\gamma(t)). \tag{18}$$

Denote

$$\begin{cases} c'(s,t) = \frac{\text{d}c(s,t)}{\text{d}s} \\ \dot{c}(s,t) = \frac{\text{d}c(s,t)}{\text{d}t}. \end{cases} \tag{19}$$

According to the definition of exponential mapping we have $c'(s,t) = \log_{A_{i+1}\gamma(t)} A_i\gamma(t)$ and is independent of s. Then we have

$$\begin{aligned}
\frac{\text{d}h_i(\gamma(t))}{\text{d}t} &= \frac{\text{d}}{\text{d}t}\langle \log_{A_{i+1}\gamma(t)} A_i\gamma(t), \log_{A_{i+1}\gamma(t)} A_i\gamma(t)\rangle \\
&= \frac{\text{d}}{\text{d}t}\langle c'(s,t), c'(s,t)\rangle \\
&= \langle \frac{\text{d}}{\text{d}s}\dot{c}(s,t), c'(s,t)\rangle \\
&= \int_0^1 \langle \frac{\text{d}}{\text{d}s}\dot{c}(s,t), c'(s,t)\rangle \text{d}s \\
&= \int_0^1 \frac{\text{d}}{\text{d}s}\langle \dot{c}(s,t), c'(s,t)\rangle \text{d}s \\
&= \langle \dot{c}(1,t), c'(1,t)\rangle - \langle \dot{c}(0,t), c'(0,t)\rangle \\
&= \langle A_i\gamma'(t), \log_{A_{i+1}\gamma(t)} A_i\gamma(t)\rangle - \\
&\quad \langle A_{i+1}\gamma'(t), \log_{A_{i+1}\gamma(t)} A_i\gamma(t)\rangle.
\end{aligned} \tag{20}$$

Note that the tangent vectors in of $\mathcal{M}_R$ also has its Cartesian product structure, so we denote $\xi_i = A_i\xi_{\mathbf{x}}$ and $\xi_{i+1} = A_{i+1}\xi_{\mathbf{x}}$. From the definition of exponential mapping we have $\gamma'(t) = \xi_{\mathbf{x}}$. Therefore, we have

$$\begin{aligned}
\left.\frac{\text{d}h_i(\gamma(t))}{\text{d}t}\right|_{t=0} &= \langle \xi_i - \xi_{i+1}, \log_{A_{i+1}\gamma(0)} A_i\gamma(0)\rangle \\
&= \langle \xi_i - \xi_{i+1}, \log_{A_{i+1}\mathbf{x}} A_i\mathbf{x}\rangle \\
&= \langle \xi_{\mathbf{x}}, -A_i^{\mathrm{T}}\log_{A_i\mathbf{x}} A_{i+1}\mathbf{x} - \\
&\quad A_{i+1}^{\mathrm{T}}\log_{A_{i+1}\mathbf{x}} A_i\mathbf{x}\rangle.
\end{aligned} \tag{21}$$

According to the definition of gradient we have now proved Lemma 1. ■

Now we have derived the gradient of $g_i(\mathbf{x})$ and $h_i(\mathbf{x})$. Using linearity of the gradient, we can obtain

$$\begin{aligned}
\text{grad}f(\mathbf{x}) = -A_1^{\mathrm{T}}(\log_{A_1\mathbf{x}}\tilde{\mathbf{R}}_1 + \log_{A_1\mathbf{x}} A_2\mathbf{x}) - \\
\sum_{i=2}^{N-1} A_i^{\mathrm{T}}(\log_{A_i\mathbf{x}}\tilde{\mathbf{R}}_i + \log_{A_i\mathbf{x}} A_{i+1}\mathbf{x} + \log_{A_i\mathbf{x}} A_{i-1}\mathbf{x}) - \\
A_N^{\mathrm{T}}(\log_{A_N\mathbf{x}}\tilde{\mathbf{R}}_N + \log_{A_N\mathbf{x}} A_{N-1}\mathbf{x}).
\end{aligned} \tag{22}$$

Equation (22) clearly shows the decomposition of $\text{grad}f(\mathbf{x})$ into N skew symmetric matrices corresponding to the N rotation matrices in $\mathbf{x}$. Given the direction, we can use exponential mapping to update $\mathbf{x}$ in each iteration for steepest gradient descent algorithm.

C. Hessian Computation

In Euclidean space the convergence rate of steepest gradient descent is strongly affected by the eigenvalues of the Hessian matrix of the objective function $\text{Hess}f(\mathbf{x})$. This property also holds for non-linear manifolds [25]. In fact we can check that the Hessian matrix of the given objective function is ill-conditioned (the largest eigenvalue is much larger than the smallest eigenvalue). Therefore, the steepest gradient descent method converges only sublinearly.

Newton's method has been proved to converge locally quadratically to the optimal solution for both Euclidean space and non-linear manifolds. Especially for general Riemannian manifolds, the framework of Newton's method was first proposed in [27], [28] with a proof of quadratic convergence.

Newton's method needs calculating the Hessian $\text{Hess}f(\mathbf{x})$. In manifolds the Hessian is defined as following:

***Definition** 2:* For any real-valued function $f : \mathcal{M} \rightarrow \mathbb{R}$ defined on a Riemannian manifold $\mathcal{M}$ with Levi-Civita Connection $\bigtriangledown$, the Hessian $\text{Hess}f(\mathbf{x})$ is mapping from $T_{\mathbf{x}}\mathcal{M}$ to $T_{\mathbf{x}}\mathcal{M}$ satisfying

$$\text{Hess}f(\mathbf{x})[\xi_{\mathbf{x}}] = \bigtriangledown_{\xi_{\mathbf{x}}}\text{grad}f(\mathbf{x}). \tag{23}$$

Note that the Levi-Civita Connection $\bigtriangledown_{\xi_{\mathbf{x}}}\text{grad}f(\mathbf{x})$ is a kind of affine connection that measures in the change in $\text{grad}f(\mathbf{x})$ when $\mathbf{x}$ changes infinitesimally in the direction of $\xi_{\mathbf{x}}$ [29]. The Hessian is also usually defined as an symmetric operator on two tangent vectors as

$$\text{Hess}f(\mathbf{x})(\xi_{\mathbf{x}}, \eta_{\mathbf{x}}) = \langle \text{Hess}f(\mathbf{x})[\xi_{\mathbf{x}}], \eta_{\mathbf{x}} \rangle = \langle \text{Hess}f(\mathbf{x})[\eta_{\mathbf{x}}], \xi_{\mathbf{x}} \rangle \tag{24}$$

To calculate the Hessian on manifolds is a very difficult task. We start to derive the Hessian of the proposed objective function from the following lemma in [30].

***Lemma** 2:* Consider the geodesic distance function $\phi_{\mathbf{Q}}(\mathbf{P}) = \frac{1}{2}d_g^2(\mathbf{P}, \mathbf{Q})$, where $\mathbf{P}, \mathbf{Q} \in \mathbf{SO(3)}$. Let $r = d_g(\mathbf{P}, \mathbf{Q})$ be the geodesic distance. Let $\gamma(t) : [0, r] \rightarrow \mathbf{SO(3)}$ denote the unit speed geodesic connecting $\mathbf{Q}$ to $\mathbf{P}$. $\forall \xi_{\mathbf{P}}, \eta_{\mathbf{P}} \in T_{\mathbf{P}}\mathbf{SO(3)}$, we have the Hessian operator

$$\text{Hess}\phi_{\mathbf{Q}}(\mathbf{P})(\xi_{\mathbf{P}}, \eta_{\mathbf{P}}) = \langle \xi_{\mathbf{P}}^{\parallel}, \eta_{\mathbf{P}}^{\parallel} \rangle + \frac{r}{\tan(r/2)} \langle \xi_{\mathbf{P}}^{\perp}, \eta_{\mathbf{P}}^{\perp} \rangle, \tag{25}$$

where $\parallel$ and $\perp$ signs denote parallel and perpendicular orthogonal components of the tangent vector with respect to $\dot{\gamma}(r)$. Here $\dot{\gamma}(r) \in T_{\mathbf{P}}\mathbf{SO(3)}$ is the parallel translation of $\dot{\gamma}(0) = \log_{\mathbf{Q}}\mathbf{P}$ along the geodesic from $\mathbf{Q}$ to $\mathbf{P}$.

Given Lemma 2 and any orthonormal basis $\{E^n\}_{n=1,2,3}$ of $T_{\mathbf{P}}\mathbf{SO(3)}$ we can compute the matrix representation of the Hessian operator by computing its result on every pair of basis tangent vectors. Lemma 2 gives us a way to compute the Hessian matrix when the objective function is the geodesic distance defined on $\mathbf{SO(3)}$. In our proposed problem we need to find the Hessian for $g_i(\mathbf{x})$ and $h_i(\mathbf{x})$, which are defined on the manifold $\mathcal{M}_R$ of rotation matrix sequences. Note that due to the separability feature (Cartesian product structure) of the tangent vectors of $\mathcal{M}_R$, we can always find an orthonormal basis $\{E_i^n\}_{n=1,2,3;i=1,\ldots,N}$ of $T_{\mathbf{x}}\mathcal{M}_R$, where only $A_iE_i^n$ is non-zero and it is equal to the basis vector E^n defined for $T_{A_i\mathbf{x}}\mathbf{SO(3)}$. In other words, the orthonormal basis of $T_{\mathbf{x}}\mathcal{M}_R$ can be represented by N subgroups and each subgroup corresponds to one particular rotation matrix in the entire sequence. We propose the following proposition:

***Proposition** 1:* Given the decomposed objective functions defined in (13) and an orthonormal basis of $T_{\mathbf{x}}\mathcal{M}_R$ in form of $\{E_i^n\}$, we have

$$\begin{cases} \text{Hess}g_i(\mathbf{x})(E_i^n, E_i^m) = \text{Hess}\phi_{\tilde{\mathbf{R}}_i}(A_i\mathbf{x})(E^n, E^m) \\ \text{Hess}g_i(\mathbf{x})(E_j^n, E_k^m) = 0, \text{if } j \neq i \text{ or } k \neq i \end{cases} \tag{26}$$

$$\begin{cases} \text{Hess}h_i(\mathbf{x})(E_i^n, E_i^m) = \text{Hess}\phi_{A_{i+1}\mathbf{x}}(A_i\mathbf{x})(E^n, E^m) \\ \text{Hess}h_i(\mathbf{x})(E_{i+1}^n, E_{i+1}^m) = \text{Hess}\phi_{A_i\mathbf{x}}(A_{i+1}\mathbf{x})(E^n, E^m) \\ \text{Hess}h_i(\mathbf{x})(E_i^n, E_{i+1}^m) = -\text{Hess}\phi_{A_{i+1}\mathbf{x}}(A_i\mathbf{x})(E^n, E^m) \\ \text{Hess}h_i(\mathbf{x})(E_{i+1}^n, E_i^m) = -\text{Hess}\phi_{A_i\mathbf{x}}(A_{i+1}\mathbf{x})(E^n, E^m) \\ \text{Hess}h_i(\mathbf{x})(E_j^n, E_k^m) = 0, \text{if } j \neq i, i+1 \text{ or } k \neq i, i+1 \end{cases} \tag{27}$$

The computations on the right-hand sides of (26) and (27) have been defined in Lemma 2.

Proposition 1 can be easily proved using the Cartesian product structure of $\mathcal{M}_R$ and the definitions of gradient and Hessian. Using Proposition 1 and linearity of the Hessian we can obtain a $3N \times 3N$ matrix representation H of $\text{Hess}f(\mathbf{x})$ for a given orthonormal basis $\{E_i^n\}$. To compute the direction in Newton's method, we first compute $-\text{grad}f(\mathbf{x})$ and then represent it as a vector v under the orthonormal basis $\{E_i^n\}$. Then we just need to solve the linear system $H \cdot u = v$ and the direction is represented by the vector u under the same basis.

Given any gradient-related update direction we use the Armijo rule [31] to select the step size.

D. Constrained Optimization

In Euclidean space, if the optimization problem is constrained, the update with the descent direction may be outside the constraint set. When the constraint set is convex and the update direction is the gradient, an option is to project the update onto the constraint set in each iteration. This is known as gradient projection algorithm [22] and it only works fast when the constraint set has simple form so the projection step is easy to implement, such as box constraints. The limitation of this algorithm is that the update direction can only be the gradient of the objective function. If the update direction is a scaled version of the gradient, such as in Newton's method, the projection step on the convex set should also be based on the same scaling of the original metric. This will make the projection step very hard to implement even if the constraint set has a very simple form.

In [23], [32] the authors proposed a new version of scaled gradient projection method called "two-metric projection method", which can use the scaled gradient as update direction while keep the projection step based on the original metric. The most important step in two-metric projection method is to decompose the gradient in a pair of dual cones determined by the constraint set and only scale one component. When the gradient is scaled by the Hessian (similar to Newton's method), it has been proved that the two-metric projection method can converge to a stationary point globally and has superlinear convergence rate locally around the stationary point. A clearer summary of the general form of two-metric projection method and its modification can be found in [24], [33]. In this paper we will first apply the two-metric projection algorithm on Euclidean space with the constraint set being a Cartesian product of Euclidean balls. Then we will extend the algorithm into optimization in manifold with the constraint set being a Cartesian product of geodesic balls. To our knowledge this is the first time that the two-metric projection method is extended to solve a constrained manifold optimization problem.

E. Two-metric method in Euclidean space

First we review the algorithm prototype proposed in [23] for generalized constrained optimization problem on a convex set in Euclidean space:

$$\min_{\mathbf{x}\in\Omega} f(\mathbf{x}), \tag{28}$$

where f is any smooth real function on Euclidean space and Ω is any nonempty closed convex set in Euclidean space. The Two-metric Algorithm in Euclidean space (Algorithm 1) is described below.

Algorithm 1 Two-metric Algorithm in Euclidean Space

```
1:  INPUT: $\mathbf{x}^0, k = 0$
2:  repeat
3:      Compute grad$f(\mathbf{x}^k)$
4:      $\mathbf{v}_{\mathcal{N}} = s_{\mathcal{N}}\mathbf{P}_{\mathcal{N}}(-\mathrm{grad}f(\mathbf{x}^k))$
5:      $\mathbf{v}_{\mathcal{T}} = \mathbf{P}_{\mathcal{T}}S_{\mathcal{T}}\mathbf{P}_{\mathcal{N}^*}(-\mathrm{grad}f(\mathbf{x}^k))$
6:      $\mathbf{v} = \mathbf{v}_{\mathcal{N}} + \mathbf{v}_{\mathcal{T}}$
7:      Update $\mathbf{x}^{k+1} = \mathbf{P}_{\Omega}(\mathbf{x}^k + \alpha^k\mathbf{v})$
        ($\alpha^k$ is the step size chosen by the Armijo Rule)
8:      $k = k + 1$
9:      where:
10:     $\mathcal{N} = \{\mathbf{y} : \forall\mathbf{z}\in\Omega, \langle\mathbf{y}, \mathbf{z}-\mathbf{x}^k\rangle \le 0\}$
        (The normal cone at $\mathbf{x}^k$)
11:     $\mathcal{N}^* = \{\mathbf{y}^* : \forall\mathbf{y}\in\mathcal{N}, \langle\mathbf{y}^*, \mathbf{y}\rangle \le 0\}$
        (The dual cone of $\mathcal{N}$)
12:     $\mathcal{T} = \{\mathbf{v}_{\mathcal{N}}\}^{\perp} \cap \mathcal{N}^*$
13:     $[\mathcal{T}]$ = the closed linear hull of $\mathcal{T}$
14:     $S_{\mathcal{T}}$ = a bounded linear map from $[\mathcal{T}]$ into $[\mathcal{T}]$
15:     $s_{\mathcal{N}}$ = a scalar
16: until Convergence
```

$\mathbf{P}_*(\cdot)$ in the algorithm means the projection of a point onto a set. To accelerate convergence the linear map $S_{\mathcal{T}}$ can be chosen based on the inverse of the Hessian of the objective function. Next we show how this algorithm work when the convex constraint set Ω is a Cartesian ball of Euclidean balls:

$$\Omega = \Omega_1 \times \Omega_2 \times \ldots \times \Omega_N, \Omega_i = \{\mathbf{x}_i : ||\mathbf{x}_i - \mathbf{c}_i|| \le r_0\}, \tag{29}$$

where $\{\mathbf{c}_i\}$ are the centers of each Euclidean ball and r_0 is the constant radius of these balls. We also assume that each Euclidean ball is with dimension K.

According to the Cartesian product structure of Ω, we can also decompose the sets $\mathcal{N}, \mathcal{N}^*, \mathcal{T}$ in Algorithm 1 into such kind of Cartesian product form. For any $\mathbf{x}\in\Omega$, define the active index set $I(\mathbf{x}) = \{i : ||\mathbf{x}_i - \mathbf{c}_i|| = r_0\}$, then clearly we have each component of the normal cone $\mathcal{N}$ at $\mathbf{x}$ as

$$\mathcal{N}_i = \begin{cases} \{\mathbf{0}\}, & i \notin I(\mathbf{x}) \\ \{\mathbf{v}_i : \mathbf{v}_i = \lambda(\mathbf{x}_i - \mathbf{c}_i), \lambda \ge 0\}, & i \in I(\mathbf{x}) \end{cases} \tag{30}$$

Then we can get each component of the dual cone $\mathcal{N}^*$ as

$$\mathcal{N}_i^* = \begin{cases} \mathbb{R}^K, & i \notin I(\mathbf{x}) \\ \{\mathbf{v}_i : \langle\mathbf{v}_i, \mathbf{x}_i - \mathbf{c}_i\rangle \le 0\}, & i \in I(\mathbf{x}) \end{cases} \tag{31}$$

If we define a second active index set as

$$\hat{I}(\mathbf{x}) = \{i : i \in I(\mathbf{x}), \mathrm{and}\langle(\mathrm{grad}f(\mathbf{x}))_i, \mathbf{x}_i - \mathbf{c}_i\rangle < 0\}, \tag{32}$$

then we can get each component of the projection of the gradient on to the normal cone $\mathcal{N}$ as

$$(\mathbf{v}_{\mathcal{N}})_i = \begin{cases} \mathbf{0}, & i \notin \hat{I}(\mathbf{x}) \\ -s_{\mathcal{N}}\frac{\langle(\mathrm{grad}f(\mathbf{x}))_i, \mathbf{x}_i - \mathbf{c}_i\rangle}{||\mathbf{x}_i - \mathbf{c}_i||^2}(\mathbf{x}_i - \mathbf{c}_i), & i \in \hat{I}(\mathbf{x}) \end{cases} \tag{33}$$

Finally each component of the set $\mathcal{T}$ and its linear hull $[\mathcal{T}]$ can be represented as

$$\mathcal{T}_i = \begin{cases} \mathbb{R}^K, & i \notin I(\mathbf{x}) \\ \{\mathbf{v}_i : \langle\mathbf{v}_i, \mathbf{x}_i - \mathbf{c}_i\rangle \le 0\}, & i \in I(\mathbf{x}), i \notin \hat{I}(\mathbf{x}) \\ \{\mathbf{v}_i : \mathbf{v}_i \perp (\mathbf{x}_i - \mathbf{c}_i)\}, & i \in \hat{I}(\mathbf{x}) \end{cases} \tag{34}$$

$$[\mathcal{T}]_i = \begin{cases} \mathbb{R}^K, & i \notin \hat{I}(\mathbf{x}) \\ \{\mathbf{v}_i : \mathbf{v}_i \perp (\mathbf{x}_i - \mathbf{c}_i)\}, & i \in \hat{I}(\mathbf{x}) \end{cases} \tag{35}$$

Now we construct the linear map $S_{\mathcal{T}}$. Assume the Hessian matrix of the objective function at $\mathbf{x}$ is H. Without loss of generality, by relabeling the coordinates of $\mathbf{x}$ if necessary, we assume that there exists an index q such that $\hat{I}(\mathbf{x}) = \{q+1, q+2, \ldots, N\}$. We diagonalize the Hessian matrix with respect to $\hat{I}(\mathbf{x})$ as

$$\hat{H} = \begin{bmatrix} \tilde{H} & \mathbf{0} \\ \mathbf{0} & I \end{bmatrix}, \tag{36}$$

where $\tilde{H}$ is the same as the part of H corresponding to $i \notin \hat{I}(\mathbf{x})$, I is the identity matrix corresponding to $\hat{I}(\mathbf{x})$ (the last $(N-q)$ components). We define the matrix representation of the map $S_{\mathcal{T}}$ as $\hat{H}^{-1}$. It is easy to verify that $S_{\mathcal{T}}$ maps any point in $\mathcal{T}$ into $\mathcal{T}$ itself. Thus we can get each component of $\mathbf{v}_{\mathcal{T}}$ as

$$(\mathbf{v}_{\mathcal{T}})_i = \begin{cases} -(\hat{H}^{-1}\mathrm{grad}f(\mathbf{x}))_i, & i \notin I(\mathbf{x}) \\ -\mathbf{P}_{\mathcal{T}_i}(\hat{H}^{-1}\mathrm{grad}f(\mathbf{x}))_i, & i \in I(\mathbf{x}), i \notin \hat{I}(\mathbf{x}) \\ -(\mathrm{grad}f(\mathbf{x}))_i + \frac{\langle(\mathrm{grad}f(\mathbf{x}))_i, \mathbf{x}_i - \mathbf{c}_i\rangle}{||\mathbf{x}_i - \mathbf{c}_i||^2}(\mathbf{x}_i - \mathbf{c}_i), & \\ & i \in \hat{I}(\mathbf{x}) \end{cases} \tag{37}$$

If we choose $s_{\mathcal{N}} = 1$ then finally we can get

$$\mathbf{v}_i = \begin{cases} -(\hat{H}^{-1}\mathrm{grad}f(\mathbf{x}))_i, & i \notin I(\mathbf{x}) \\ -\mathbf{P}_{\mathcal{T}_i}(\hat{H}^{-1}\mathrm{grad}f(\mathbf{x}))_i, & i \in I(\mathbf{x}), i \notin \hat{I}(\mathbf{x}) \\ -(\mathrm{grad}f(\mathbf{x}))_i, & i \in \hat{I}(\mathbf{x}) \end{cases} \tag{38}$$

We can find that only for the indices not in the active set $\hat{I}(\mathbf{x})$ the gradient is scaled by the inverse of the Hessian to get the update direction. In extreme case that the active set is always empty (the boundary of the constraint set is never reached), this algorithms turns into regular Newton's method. Note that due to the Cartesian product structure all the computation except the inverse of the Hessian matrix can be performed independently for each component.

F. Scaled Gradient Projection on Manifold

Now we generalize the two-metric projection algorithm to manifolds. The convex constraint set Ω now becomes the Cartesian of geodesic balls defined in (9). The centers of these balls are the original camera rotation matrices $\{\tilde{\mathbf{R}}_i\}$. Based on the discussion in the previous subsection we propose the Manifold Two-metric Algorithm for Constrained Motion Smoothing (Algorithm 2) below to solve the problem in (10).

Algorithm 2 Manifold Two-metric Algorithm for Constrained Motion Smoothing

```
1: INPUT: A fixed orthonormal basis $\{E_i^n\}_{n=1,2,3;i=1,\dots,N}$ of $T_{\mathbf{x}}\mathcal{M}_R$, $\mathbf{x}^0, k=0$
2: repeat
3:     Compute grad$f(\mathbf{x}^k)$ and its representation vector $\mathbf{u}$ in the fixed basis
4:     Compute the representation matrix $H$ of Hess$f(\mathbf{x}^k)$ in the fixed basis
5:     Find active set $I(\mathbf{x}^k) = \{i : ||\log_{\tilde{\mathbf{R}}_i} A_i\mathbf{x}^k|| = r_0\}$
6:     Find active set
       $\hat{I}(\mathbf{x}^k) = \{i : i \in I(\mathbf{x}), \langle(\text{grad}f(\mathbf{x}^k))_i, \log_{\tilde{\mathbf{R}}_i} A_i\mathbf{x}^k\rangle < 0\}$
7:     Diagonalize $H$ to $\hat{H}$ with respect to $\hat{I}(\mathbf{x}^k)$ as in (36)
8:     Compute the vector representation $\mathbf{v}$ of the update direction as
9:
       $$\mathbf{v}_i = \begin{cases} -(\hat{H}^{-1}\mathbf{u})_i, & i \notin I(\mathbf{x}) \\ -\mathbf{P}_{\mathcal{T}_i}(\hat{H}^{-1}\mathbf{u})_i, & i \in I(\mathbf{x}), i \notin \hat{I}(\mathbf{x}) \\ -\mathbf{u}_i, & i \in \hat{I}(\mathbf{x}) \end{cases} \quad (39)$$
       Compute the update direction $\mathbf{d} \in T_{\mathbf{x}^k}\mathcal{M}_R$ based on its vector representation $\mathbf{v}$
10:    Update $\mathbf{x}^{k+1} = \mathbf{P}_\Omega(\exp_{\mathbf{x}^k} \alpha^k \mathbf{d})$
       ($\alpha^k$ is the step size chosen by the Armijo Rule)
11:    $k = k+1$
12:    where:
13:    $\mathcal{T}_i = \{\xi_i \in T_{A_i\mathbf{x}^k}\mathbf{SO(3)} : \langle \xi_i, \log_{\tilde{\mathbf{R}}_i} A_i\mathbf{x}^k\rangle \le 0\}, i \in I(\mathbf{x}), i \notin \hat{I}(\mathbf{x})$
14: until Convergence
```

G. Geodesic-convexity of Motion Smoothing

In this subsection we will show that the proposed problem in (10) is geodesic convex on the manifold $\mathcal{M}_R$. Geodesic-convexity is a natural generalization of convexity in Euclidean space to manifolds. Similar with Euclidean space, for geodesic-convex optimization problems, local minimum is also global minimum [34]. We mentioned in Section IV that the constraint set is geodesic convex according to triangular inequality of Riemannian metrics. Here we will focus on the proof of the geodesic convexity of the objective function in (13). From the property that a linear combination of geodesic convex functions is still geodesic convex, it is sufficient to prove the functions $g_i(\mathbf{x})$ and $h_i(\mathbf{x})$ are geodesic convex. It has been shown that the geodesic distance function $\phi_{\mathbf{Q}}(\mathbf{P}) = \frac{1}{2}d_g^2(\mathbf{P},\mathbf{Q})$ is geodesic convex with respect to $\mathbf{P}$ inside a geodesic ball centered at $\mathbf{Q}$ with radius less than $\pi/2\sqrt{\Delta}$, where Δ is the upper bound of the sectional curvature of the manifold that $\mathbf{P}$ and $\mathbf{Q}$ lie in [26]. For manifold $\mathbf{SO(3)}$ the sectional curvature is $1/4$ everywhere. So the radius of the geodesic ball centered at $\mathbf{Q}$ should be less than $\pi/2$. Consider $\mathbf{P}$ as a rotation matrix for a certain frame and the $\mathbf{Q}$ as the rotation matrix of its adjacent frame, $\mathbf{P}$ is well guaranteed to be inside the geodesic ball centered at $\mathbf{Q}$ with radius $\pi/2$ since the change of rotation between two consecutive frames is very small. If $\mathbf{Q}$ is the original rotation matrix for a certain frame and $\mathbf{P}$ is the new rotation matrix in the stabilized video for the same frame, then $\mathbf{P}$ is still guaranteed to be inside the geodesic ball centered at $\mathbf{Q}$ with radius $\pi/2$ given the proposed constraint (11).

It has been shown that a function is geodesic convex if and only if the matrix representation of its Hessian is positive semi-definite (PSD) [34]. We have mentioned that the function $\phi_{\tilde{\mathbf{R}}_i}(A_i\mathbf{x})$ is geodesic convex with respect to $A_i\mathbf{x}$. So the matrix representation of its Hessian matrix H_g is PSD. According to Proposition 1 the matrix representation of the Hessian of $g_i(\mathbf{x})$ can be represented as

$$\begin{bmatrix} \mathbf{0} & & & & \\ & \ddots & & & \\ & & H_g & & \\ & & & \ddots & \\ & & & & \mathbf{0} \end{bmatrix} \quad (40)$$

(H_g locates at the coordinates corresponding to frame i) and is clearly also PSD.

Since we have also mentioned that the function $\phi_{A_i\mathbf{x}}(A_{i+1}\mathbf{x})$ is geodesic convex with respect to $A_{i+1}\mathbf{x}$, its Hessian matrix representation H_h is PSD. From Lemma 2 the Hessian matrix representation of the function $\phi_{A_{i+1}\mathbf{x}}(A_i\mathbf{x})$ is also H_h. Thus according to Proposition 1 the Hessian matrix representation of $h_i(\mathbf{x})$ is

$$\begin{bmatrix} \mathbf{0} & & & & & \\ & \ddots & & & & \\ & & H_h & -H_h & & \\ & & -H_h & H_h & & \\ & & & & \ddots & \\ & & & & & \mathbf{0} \end{bmatrix}. \quad (41)$$

It is not hard to show that this matrix is also PSD.

Therefore, we have show that the function $g_i(\mathbf{x})$ and $h_i(\mathbf{x})$ in (13) are geodesic convex, which means the objective function $f(\mathbf{x})$ is geodesic convex. Given the property that the proposed two-metric scaled gradient projection algorithm can always converge to a stationary point, the global optimality of the proposed algorithm is guaranteed.

H. Choice of Regularization Parameter

As we mentioned, the original objective function (5) consists of two different terms: data-fitting term and the regularization term. The parameter α controls the relative weights of the two different terms. The data-fitting term was originally introduced into the unconstrained video stabilization problem to guarantee that the smoothed camera motion trajectory does not deviate from the original trajectory too much. However, we can find that such requirement is redundant if the hard constraint (9) is added. Therefore, the optimal choice of α should be infinity in order to reach the greatest degree of smoothness in the stabilized trajectory. This is equivalent to dropping the data-fitting term $\sum_{i=1}^{N} g_i(\mathbf{x})$.

However, such setting is impractical since the proposed algorithm relies on inverting the matrix representation of Hess$f(\mathbf{x})$. According to the discussion in Section V-G, the Hessian matrix representation of $\sum_{i=1}^{N-1} h_i(\mathbf{x})$ can be written as

$$H = \begin{bmatrix} H_h & -H_h & & & & \\ -H_h & 2H_h & -H_h & & & \\ & -H_h & 2H_h & & & \\ & & & \ddots & & \\ & & & & 2H_h & -H_h \\ & & & & -H_h & H_h \end{bmatrix}. \quad (42)$$

Recall that H is a $3N \times 3N$ matrix given an orthonormal basis of $T_{\mathbf{x}}\mathcal{M}_R$. From the fact that each 3×3 matrix block H_h is PSD, we can decompose it as $H_h = {\Theta_h}^{\mathrm{T}}\Theta_h$. As a result, we can decompose H as $H = \Theta^{\mathrm{T}}\Theta$, where Θ is a $3(N-1) \times 3N$ matrix and

$$\Theta = \begin{bmatrix} \Theta_h & -\Theta_h & & & \\ & \Theta_h & -\Theta_h & & \\ & & & \ddots & & \\ & & & & \Theta_h & -\Theta_h \end{bmatrix}. \quad (43)$$

Clearly the matrix H does not have full rank and thus is not invertible. Therefore, in practice we leave the data-fitting term in the objective function and set α to a very large number to avoid the numerical problem of matrix inversion.

VI. Experimental Results

We first compare the convergence rate of different algorithms in solving the formulated smoothing problem on the sequences of rotation matrices. Fig. 2 is an example showing the convergence rate of steepest gradient descent method and Newton's method in solving the unconstrained formulated problem. In the experiment we try to smooth a sequence of 478 3D rotation matrices (478 frames) with $\alpha = 1000$, which were taken at a frame rate of 30 Hz. Figure 2 shows the values of the objective function in 10 iterations. Newton's method successfully converges in just 2 iterations. Each iteration of the Newton's method takes 2.93s on a 2.3GHz Intel i5 processor machine with MATLAB implementation (without parallel processing). From Fig. 2 we can observe that the scaling of gradient using Hessian matrix can significantly accelerate the convergence, which motivates us to do the same thing in solving the constrained problem.

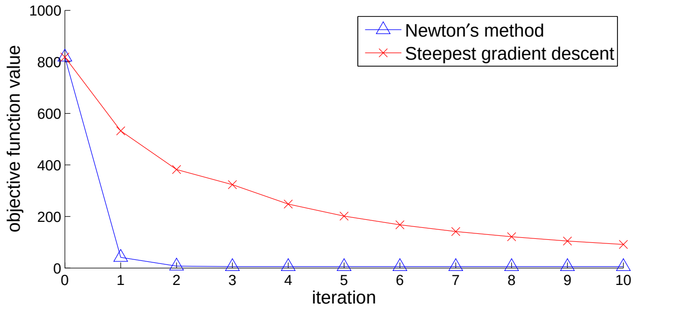

Fig. 2. Convergence of two gradient-related algorithms for unconstrained motion smoothing: Newton's method and steepest gradient descent. For these algorithms, we compute the gradient and Hessian of the objective function using Riemannian geometry.

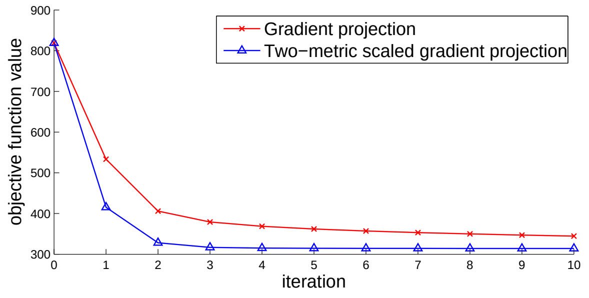

Fig. 3. Convergence of two algorithms for constrained motion smoothing: manifold gradient projection and the manifold two-metric scaled gradient projection (proposed).

In Fig. 3 we compare the convergence rate of gradient projection method and the proposed scaled gradient projection method in solving the constrained motion smoothing problem. The test video sequence is the same as in Fig. 2. The original frame size is 720×480 and the guaranteed cropped size is 540×360. The radius r_0 is found as 0.11. We can observe that proposed two-metric scaled gradient projection method successfully converges in only 4 iterations. Note that the difference in the final convergence values of the objective function between Fig. 2 and Fig. 3 is caused by the constraint in (9). Each iteration of the proposed method takes 3.58s on the same machine as in Section V-C, while the gradient projection method takes 18.66s per iteration. The reason gradient projection method takes such longer time in each iteration is that it needs multiple tries to find the proper (descent) step size using the Armijo step size selection method. For the proposed method usually 1 is the proper step size (no more tries are needed) because the gradient has been scaled by the Hessian of the objective function.

Fig. 4 shows another example comparing the convergence rate of gradient projection method and the proposed scaled gradient projection method. In this example there are 761 frames in the test video sequence. The proposed two-metric scaled gradient projection method successfully converges in only 5 iterations. In Fig. 5 we additionally run the proposed two-metric projection method on a video with 163 frames and show the running time of each iteration for all of the three test videos with respect to their numbers of frames. We can observe that the running time of each iteration increases linearly with

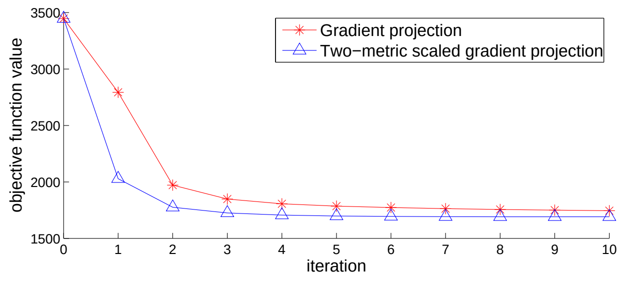

Fig. 4. Convergence of two algorithms for constrained motion smoothin gradient projection and the manifold two-metric scaled gradient projectic (proposed).

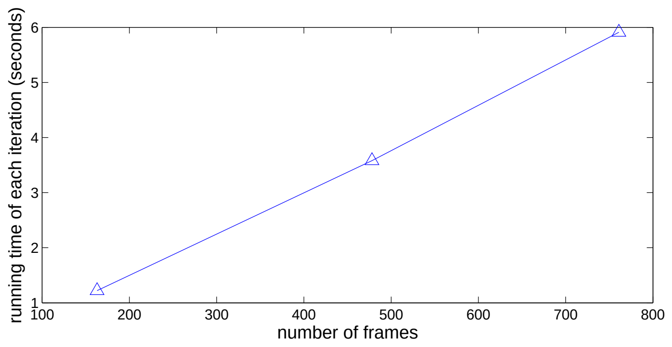

Fig. 5. Running time of each iteration for the proposed manifold two-metric projection algorithm.

the increase in the number of frames, thanks to the sparse structure of the Hessian matrix as shown in Proposition 1.

A comparison between the original camera rotation and the smoothed camera rotation is shown in Fig. 6. In this figure, the camera rotation corresponding to each frame in the video sequence is shown in form of Tait-Bryan angles (a similar representation as Euler angles). Note that, however, our motion smoothing is performed naturally on the manifold $\mathcal{M}_R$ instead of the Euclidean space of rotation angle representation.

Then we use the proposed constrained motion smoothing method in video stabilization. In the experiments we try to stabilize the video sequences captured by Google Nexus S smart phone. While recording videos, we also captured the readings (with timestamps) from the 3-axis gyroscope inside the phone. The camera has been calibrated so the camera intrinsic matrix $\mathbf{K}$ is known. We also assume that the gyroscope and the videos have been synchronized so that we can obtain the camera pose (3D rotation) simply from the gyroscope readings. In practice, the calibration and synchronization can be executed using the method provided in [9].

We use the proposed constrained motion smoothing method in video stabilization and compare our method with the YouTube video editor. The video stabilization in YouTube video editor is based on the approach proposed in [4], which estimates the 2D similarity camera motion (with homography refinement) from frame to frame and uses ℓ_1 regularization to smooth the estimated camera path. This method is one of the state-of-the-art video stabilization algorithms and may be the only one that explicitly considers the constraints for black borders in motion smoothing. We compared the algorithms on videos with original size 720×480. The cropping size used in

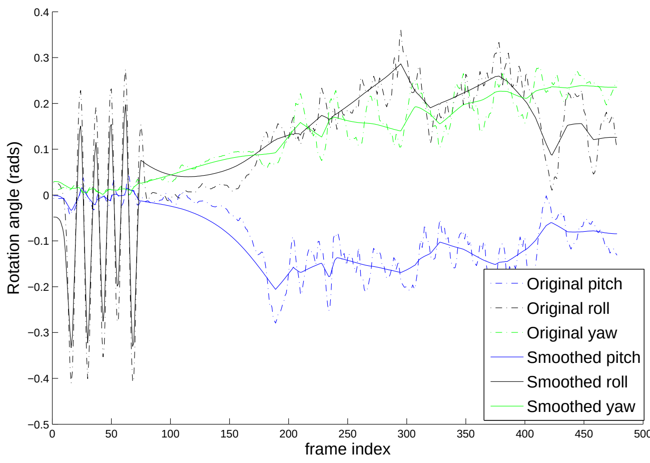

Fig. 6. Comparison of the original and the smoothed camera rotation using the proposed two-metric scaled gradient projection method for constrained smoothing on the manifold of rotation matrices.

YouTube video editor is 540×360 so we use the same size in our method. In all of the experiments we fix the smoothness parameter in our approach as $\alpha = 1000$ and find very little difference in the results with higher value of α.

First, we test the video stabilization algorithms on a video shot by a walking forward person. We use feature trajectories as shown in Fig. 7 as a visualization of the changing frames. We detect Harris corner points in a certain frame (no. 270 in Fig. 7) and track them for ten frames. The feature trajectories are plotted as black curves on top of the starting frame (the frames themselves are plotted using alpha channel 0.5 (more transparent) to make the curves clearer). For a stabilized video the trajectories should be very short since the camera is always facing forward in spite of jitter caused by camera shake. The 2D ℓ_1 regularization method [4] can smooth and shorten the trajectories compared to the original video, but the feature points are still moving up and down. Our algorithm can keep the feature points very steady and the trajectories become almost invisible (just black dots) in the results. Note that we detect and track the feature points independently in the three videos so the location and number of the feature points can be different.

Next we take a test on a video shot while panning the camera. Video stabilization should only remove the unwanted jitter while keeping the panning motion of the camera. In Fig. 8 we do the same test as in Fig. 7, except that the feature points are tracked in twenty consecutive frames instead of ten. Both methods successfully smooth the trajectories of the feature points. However, the trajectories in the stabilization result of [4] is not as straight as those in the result of the proposed method. This comparison, together with the comparison in Fig. 7, show that the proposed method can not only better remove high frequency unwanted jitter but also better smooth the long term motion of the camera.

The stabilization results are best viewed in video form. Please see the online video examples of our paper [3]. The results of YouTube video editor have been compressed so

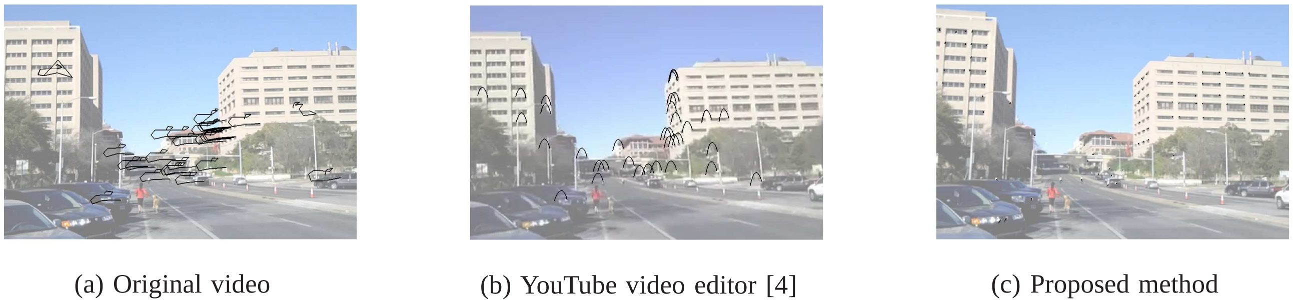

Fig. 7. Stabilization comparison for a video shot by a walking forward person. Features are tracked from frame 270 to frame 280. The feature trajectories are plotted as black curves on frame 270.

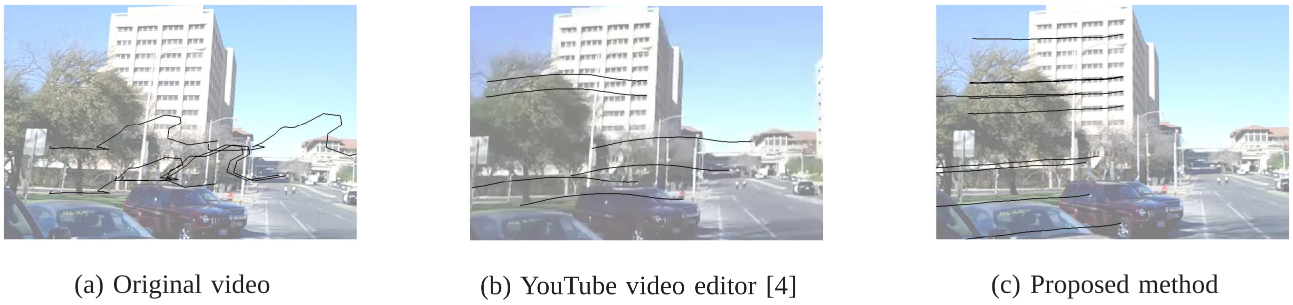

Fig. 8. Stabilization comparison for a video shot while panning the camera. Features are tracked from frame 650 to frame 670. The feature tracks are plotted as black curves on frame 650.

please ignore these compression artifacts in comparison. Besides compression, please note that there is some non-rigid wobble in the results of YouTube video editor. This is not caused by compression but the inaccuracy of the 2D motion model used in [4]. The 3D rotational model in our video stabilization accurately reflect the real camera motion so there is no such non-rigid distortion. In the two video examples features are easy to track since there is very little motion blur in the frames. However, when the videos are shot in low light condition the visual-based motion estimation used in [4] will fail sometimes while 3D rotational video stabilization using gyroscopes is not affected.

As we mentioned in Section II, other 3D rotational video stabilization algorithms [9], [12] are based on local low-pass filtering of the rotation sequence and thus are not able to guarantee that there will be no black borders. Many frames in the stabilized video still have black borders even though adaptive filtering with different window size could be applied to decrease their area. In [12], the authors proposed to use extrapolation to fill the black borders. Extrapolation can be implemented very fast but the image quality is severely sacrificed, as shown in Fig. 9.

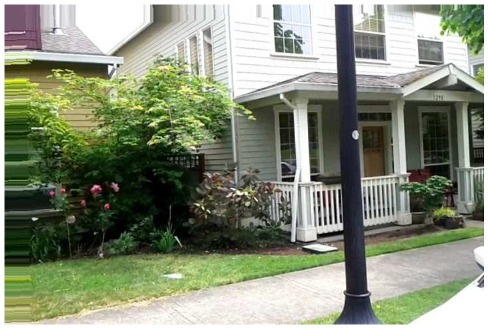

Fig. 9. Extrapolation is used to fill the undefined areas (black borders) on the left and top of the frame [12].

VII. Conclusions

In this paper we propose a novel video stabilization method using a 3D rotational camera motion model. We exploit the manifold structure of not only the 3D rotation matrices, but also the sequences of 3D rotation matrices. This allows us to globally formulate motion smoothing as a regression problem based on geodesic distance. Furthermore, we force the solution to lie on a Cartesian product of geodesic balls so that every pixel in the stabilized frame is visible in the original frame. We directly solve the formulated problem on manifold by generalizing the existing two-metric projection algorithm in Euclidean space. The 3D camera rotation for each frame is obtained reliably using gyroscopes that are equipped in most smart phones and tablets, no matter whether there is motion blur or abrupt illumination change in the videos. We have demonstrated in experiments that our algorithm is very fast and can generate better video stabilization results than state-of-the-art methods.

Acknowledgment

The authors would like to thank Dr. Hamid Sheikh at Texas Instruments for introducing us to open research problems in video rectification and stabilization of cell phone cameras.

References

[1] C. Jia and B. L. Evans, "3D rotational video stabilization using manifold optimization," in *Proc. IEEE Intl. Conf. Acoustics, Speech, and Signal Processing*, May 2013.

[2] Y. Matsushita, E. Ofek, W. Ge, X. Tang, and H.-Y. Shum, "Full-frame video stabilization with motion inpainting," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 28, Jul. 2006.

[3] C. Jia and B. L. Evans, "Video demonstrations for constrained 3D rotation smoothing via global manifold regression for video stabilization," http://users.ece.utexas.edu/∼bevans/papers/2015/stabilization/.

[4] M. Grundmann, V. Kwatra, and I. Essa, "Auto-directed video stabilization with robust L1 optimal camera paths," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Jun. 2011.

[5] Y.-F. Hsu, C.-C. Chou, and M.-Y. Shih, "Moving camera video stabilization using homography consistency," in *Proc. IEEE Intl. Conf. Image Processing*, Sep. 2012.

[6] C. Buehler, M. Bosse, and L. McMillan, "Non-metric image-based rendering for video stabilization," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Dec. 2001, pp. 609–614.

[7] F. Liu, M. Gleicher, H. Jin, and A. Agarwala, "Content-preserving warps for 3D video stabilization," *ACM Trans. Graphics*, vol. 28, no. 3, 2009.

[8] C. Morimoto and R. Chellappa, "Fast 3D stabilization and mosaic construction," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Jun. 1997.

[9] A. Karpenko, D. Jacobs, J. Baek, and M. Levoy, "Digital video stabilization and rolling shutter correction using gyroscopes," Stanford University, Tech. Rep., Mar. 2011.

[10] S.-H. Jung and C. Taylor, "Camera trajectory estimation using inertial sensor measurements and structure from motion results," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 2, Dec. 2001, pp. 732–737.

[11] D. Strelow and S. Singh, "Online motion estimation from image and inerital measurements," in *Proc. Workshop Interation of Vision and Inertial Sensors*, Jun. 2003.

[12] G. Hanning, N. Forslöw, P.-E. Forssén, E. Ringaby, D. Törnqvist, and J. Callmer, "Stabilizing cell phone video using inertial measurement sensors," in *Proc. IEEE Intl. Workshop Mobile Vision*, Nov. 2011.

[13] A. Davison, "Real-time simultaneous localisation and mapping with a single camera," in *Proc. IEEE Intl. Conf. Computer Vision*, vol. 2, Oct. 2003, pp. 1403–1410.

[14] M. Pollefeys, L. Van Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch, "Visual modeling with a hand-held camera," *Intl. Journal Computer Vision*, vol. 59, no. 3, pp. 207–232, 2004.

[15] C. Geyer, M. Meingast, and S. Sastry, "Geometric models of rolling-shutter cameras," in *Proc. Workshop Omnidirectional Vision*, 2005.

[16] C. Jia and B. L. Evans, "Probabilistic 3-D motion estimation for rolling shutter video rectification from visual and inertial measurements," in *Proc. IEEE Intl. Workshop Multimedia Signal Processing*, Sep. 2012.

[17] A. Litvin, J. Konrad, and W. Karl, "Probabilistic video stabilization using Kalman filtering and mosaicking," *Proc. IS&T/SPIE Symp. Electronic Imaging, Image and Video Comm. and Proc.*, pp. 663–674, 2003.

[18] K.-Y. Lee, Y.-Y. Chuang, C. B.-Y., and M. Ouhyoung, "Video stabilization using robust feature trajectories," in *Proc. IEEE Intl. Conf. Computer Vision*, Sep. 2009, pp. 1397–1404.

[19] F. Liu, M. Gleicher, J. Wang, H. Jin, and A. Agarwala, "Subspace video stabilization," *ACM Trans. Graphics*, vol. 30, no. 1, 2011.

[20] K. Shoemake, "Animating rotation with quaternion curves," in *Proc. ACM SIGGRAPH*, 1985, pp. 245–254.

[21] N. Boumal and P.-A. Absil, "A discrete regression method on manifolds and its application to data on so(n)," in *Proc. IFAC World Congress*, 2011.

[22] D. Bertsekas, *Nonlinear Programming: 2nd ed.*, 1999.

[23] E. Gafni and D. Bertsekas, "Two-metric projection methods for constrained optimization," *SIAM Journal Contral and Opt.*, vol. 22, no. 6, pp. 936–964, 1984.

[24] J. Dunn, "A subspace decomposition principle for scaled gradient projection methods: global theory," *SIAM Journal Control and Opt.*, vol. 29, no. 5, pp. 1160–1175, 1991.

[25] P.-A. Absil, R. Mahony, and R. Sepulchrer, *Optimization Algorithm on Matrix Manifolds*, 2008.

[26] H. Karcher, "Riemannian center of mass and mollifier smoothing," *Comm. Pure Appl. Math*, vol. 30, pp. 509–541, 1977.

[27] S. Smith, "Geometric optimization methods for adaptive filtering," Ph.D. dissertation, Harvard University, 1993.

[28] A. Edelman, T. Arias, and S. Smith, "The geometry of algorithms with orthogonality constraints," *SIAM Journal Matrix Anal. Appl.*, vol. 20, no. 2, pp. 303–353, 1998.

[29] J. Gallier, *Notes on Differential Geometry and Lie Groups*. University of Pennsylvannia, 2012.

[30] R. Ferreira, J. Xavier, J. Costeira, and V. Barroso, "Newton method for Riemannian centroid computation in naturally reductive homogeneous spaces," in *Proc. IEEE Intl. Conf. Acoustics, Speech and Signal Processing*, May 2006.

[31] J. Nocedal and S. Wright, *Numerical Optimization*, 1999.

[32] D. Bertsekas, "Projected Newton methods for optimization problems with simple constraints," *SIAM Journal Control and Opt.*, vol. 20, no. 2, pp. 221–246, Mar. 1982.

[33] J. Dunn, "A subspace decomposition principle for scaled gradient projection methods: local theory," *SIAM Journal Control and Opt.*, vol. 31, no. 1, pp. 219–246, 1993.

[34] C. Udriste, *Convex Functions and Optimization Methods on Riemannian Manifolds*, 1994.

PLACE PHOTO HERE

Chao Jia Biography text here.

PLACE PHOTO HERE

Brian L. Evans Biography text here.