Institute for Data Processing
Technische Universität München
Prof. Dr.-Ing. Klaus Diepold

Pose Estimation of a Camera using Newton Optimization on the Manifold

Technical Report

Pose Estimation of a Camera using Newton Optimization on the Manifold

Michel Sarkis*, Alexander Schwing and Klaus Diepold

Institute for Data Processing, Technische Universität München,
Arcisstr. 21, 80290 Munich, Germany
Emails: [michel,kldi]@tum.de,alexander.schwing@mytum.de

Abstract. Bundle adjustment is a minimization method frequently used to refine the structure and motion parameters of a moving camera. In this work, we present a Newton-based approach to enhance the accuracy of the estimated motion parameters in the bundle adjustment framework. The key issue is to parameterize the motion variables of a camera on the manifold of the Euclidean motion by using the underlying Lie group structure of the motion representation. We then reformulate the bundle adjustment cost function and derive the gradient and the Hessian formulation on the manifold to minimize the cost function in such a manner that the minimum is the desired estimate of the motion. This results in a more compact derivation of the Hessian which allows us to use its complete form in the minimization process. Compared to the Levenberg-Marquardt scheme, the proposed algorithm is shown to provide more accurate results while having a comparable complexity although the latter uses an approximate form of the Hessian. The experimental results we performed on simulated and real image sets are evidence that demonstrate our claims.

1 Introduction

Extracting the motion parameters of a moving camera is a significant aim in computer vision and robotics. Based on some features in a sequence of images, it is possible to obtain the initial estimates of the 3D structure of a scene along with the corresponding camera positions. Then, bundle adjustment (BA) is applied to refine those estimates. Examples of algorithms employing BA are reflected in [1–5] while in [6] an excellent survey on the different methodologies to implement this scheme is presented. The core of BA is to minimize the pixel reprojection error of the 3D points so that both the structure and the motion are corrected simultaneously. The error is expressed as the sum of the squared distances between some computed image points obtained via robust feature detectors and the estimated image points from the reprojections of the 3D points. BA is usually implemented using the Levenberg-Marquardt (LM) algorithm, which is an optimization scheme that leads to the least-squares solution of the problem in a non-linear sense [7, 8]. LM has the properties of both the steepest descent and the Gauss-Newton iterative techniques. Similarly to steepest descent, LM converges slowly to the

* This research is sponsored by the German Research Foundation (DFG) as a part of the SFB 453 project, High-Fidelity Telepresence and Teleaction.

solution if the initial estimate is far away from the real solution while it converges fast like Gauss-Newton methods when the initial estimate is close to the real solution.

In order to apply LM, the variables have to be parameterized in such a way that the minimization algorithm can avoid the local minima. The points of the structure are usually described using their 3D coordinates since they reflect the overall degrees of freedom (DOF) of the entity. The translation vector is done in the same way since it consists of 3 entries that represents its 3 DOF. The rotation is described by a 3×3 matrix consisting in a total of 9 entries although it has only 3 DOF. Parameterizing the 3D rotation using the 3 Euler angles is not a good idea in practice since it results in numerical instabilities due to the singularities and the uneven regions that the angles cover. A rotation can be expressed using unit quaternions or using local perturbations of an existing rotation. However, each of these parameterizations has its own drawback. The unit quaternion has to be constrained so that the vector has a unit norm while the update step of the rotation using local perturbations does not result exactly in a rotation matrix [6, 1]. Apart from the parameterization, LM uses in its minimization an approximate formula of the Hessian because the second order derivatives are dropped out in the formulation of the algorithm [7, 8]. It is usually applied in structure from motion problems with the hope that the initial estimates of structure and pose are accurate enough so that the overall system can converge to a global solution. This fact might create instability problems if this is not the case. This can be justified from the fact that BA is a large scale minimization problem. By ignoring the second order terms in the Hessian makes the optimization more susceptible to being stuck in local minima since the minimizer is much small for the terms to be minimized [9], i.e. the reprojection errors.

In this work, we first recall that the motion estimate of the camera in BA can be estimated more accurately by deriving the complete Hessian formulation in a Newton minimization scheme than by just taking its approximation as usually done in LM. We therefore derive a Newton scheme in the vector space with the goal to refine the motion of the camera. Tested on real and simulated data sets, the algorithm is shown to result in a noticeable improvement in the motion estimate when compared to LM. This is due to the complete computation of the Hessian at the cost of an increase in the complexity. We then argue that parameterizing the motion of the camera on the six dimensional manifold of the rigid motion is more favorable than using the current parameterization schemes. To justify that, we formulate a projective Newton-type optimization technique which exploits such parameterization to minimize the BA cost function on the manifold. The proposed scheme is shown also to have a better performance than LM in terms of accuracy of the estimated motion. The advantage in the optimization on a manifold is that the gradient and Hessian computations are easier to derive than in the vector space since they boil down to matrix multiplications [10]. This makes the complexity of the latter considerably less than that of the Newton algorithm in the vector space and comparable to that of LM.

The rest of this work is divided as follows. Section 2 presents a brief overview on BA along with a callback on Newton minimization schemes. Section 3 derives the proposed algorithm. Section 4 evaluates the performance of the technique. In the end, we conclude the paper with some discussions in Section 5.

2 Related Work

Bundle adjustment is to jointly adjust some noisy estimates of the 3D structure along with the motion of the cameras in order to satisfy all the equations of the camera perspective projection with the least possible error. Given are L camera views and N reconstructed points, the cost function f to be minimized is given by

$$f = \sum_{i=1}^{L} \sum_{j=1}^{N} \mathrm{d}\left(\mathbf{x}_j^i, \mathbf{R}_i \mathbf{X}_j + \mathbf{t}_i\right)^2, \tag{1}$$

where $\mathbf{x}_j^i$ are the homogeneous coordinates of the 2D measured points in the i^{th} image (camera), $\mathbf{R}_i$ is a 3×3 rotation matrix representing the orientation of the i^{th} camera in the world coordinate system, $\mathbf{t}_i$ is the translation vector of the corresponding camera between the origin of the world and that of the camera, $\mathbf{X}_j$ are the homogeneous coordinates of the j^{th} point in the 3D structure and d is the geometrical distance measure between a reprojected point $\hat{\mathbf{x}}_j^i = \mathbf{R}_i \mathbf{X}_j + \mathbf{t}_i$ and the corresponding measured point $\mathbf{x}_j^i$. The measure d can be any scale invariant distance such as the Mahalanobis, the L2 norm or the cosine distances.

Assuming that the camera is calibrated, the cost function f has to be minimized over every $\mathbf{R}_i$ and $\mathbf{t}_i$ and $\mathbf{X}_j$. To parameterize the translation vector and the points of the structure, i.e. $\mathbf{X}_j$, it is sufficient to take the 3 entries of each as the variables. To parameterize the rotation, it is necessary to find a minimal parametrization which reflects its 3 DOF. A rotational transformation in $\mathbb{R}^3$ is represented by the elements of the special orthogonal group SO_3. Its associated Lie algebra $\mathbf{so}_3$ is the set of 3×3 skew-symmetric matrices which are considered as the tangent space of SO_3 at the identity. The isomorphism $\mathbf{\Omega}$ that allows to identify $\mathbf{so}_3$ with $\mathbb{R}^3$ is defined as

$$\mathbf{\Omega}\left(\omega^i\right) : \mathbb{R}^3 \longrightarrow \mathbf{so}_3, \begin{bmatrix} \omega_1^i \\ \omega_2^i \\ \omega_3^i \end{bmatrix}_\times = \begin{bmatrix} 0 & -\omega_3^i & \omega_2^i \\ \omega_3^i & 0 & -\omega_1^i \\ -\omega_2^i & \omega_1^i & 0 \end{bmatrix}. \tag{2}$$

To connect the Lie algebra to the Lie group, the exponential map is usually used. The rotation matrix $\mathbf{R}_i$ can thus be expressed in the form $\mathbf{R}_i = \mathrm{expm}\left(\mathbf{\Omega}\left(\omega^i\right)\right)$. It can be written using the Taylor series expansion of the exponential function as

$$\mathbf{R}_i = \mathrm{expm}\left(\mathbf{\Omega}\left(\omega^i\right)\right) = \mathbf{I} + \mathbf{\Omega}\left(\omega^i\right) \cdot \frac{\sin\left(\|\omega^i\|\right)}{\|\omega^i\|} + \mathbf{\Omega}^2\left(\omega^i\right) \cdot \frac{1 - \cos\left(\|\omega^i\|\right)}{\|\omega^i\|^2}. \tag{3}$$

The term to the right is known as the Rodrigues formula of the rotation matrix. If the magnitude of the vector ω^i is small, it can be approximated with $\mathbf{I} + \mathbf{\Omega}\left(\omega^i\right)$ which is nothing but the local perturbation around $\mathbf{R}_i$ [6]. Assuming that the magnitude of ω^i is small is not always accurate especially if the initial estimate of the motion is contains high amount of noise. In our implementations of LM and the vector space Newton, we will be applying the whole term in order to have an accurate parametrization of the 3D rotation. Another reason for this choice is that the Manifold based optimization algorithm results in the accurate rotation matrix. Thus, taking the whole term makes the comparison among the techniques fair.

Note that the main contribution in this work is mainly emphasized by the parameterization the motion variables and developing a projective Newton scheme to fulfill this task. The structure parameterization is left intact as usually done in the literature, see. [6, 1] for example. For simplicity, we will assume throughout this work that the structure is constant; therefore, it will be ignored in the presented derivations unless otherwise specified. With this assumption, the problem in hand turns equivalent to estimating the relative poses of a moving camera. In this direction, one can find several linear and non-linear algorithms that can accomplish this task as the techniques that are presented in [11–14]. The goal of this work, however, is to enhance the accuracy of a Newton minimization scheme without significantly increasing the accompanying computational complexity. For this reason, we will be stating next a brief overview about the Levenberg-Marquardt technique and illustrate its differences from a generalized Newton algorithm.

2.1 Levenberg-Marquardt Algorithm

We will be noting by $\mathbf{a}_i = \begin{bmatrix} \omega^{i\mathrm{T}} & \mathbf{t}_i^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^6$ as the vector that represents the 6 parameters of each camera and $\mathbf{p}$ as the vector that holds the parameters of all the cameras, i.e. $\mathbf{p} = \begin{bmatrix} \mathbf{a}_1^{\mathrm{T}}, \dots, \mathbf{a}_L^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^{6L}$. In order to apply LM, f needs to be formulated in the form $f = \mathbf{g}^{\mathrm{T}}\mathbf{g}$ where $\mathbf{g} = \begin{bmatrix} g_{1,1}, \dots, g_{N,1}, \dots, g_{N,L} \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^{N \cdot L}$ and $g_{j,i} = \mathrm{d}\left(\mathbf{x}_j^i, \mathbf{R}_i\mathbf{X}_j + \mathbf{t}_i\right)$. LM tries to find the solution vector $\widetilde{\mathbf{p}}$ that minimizes Equation (1) iteratively. The update step at the k^{th} iteration can be expressed as

$$\mathbf{p}^{(k+1)} = \mathbf{p}^{(k)} - \left(\mathbf{J}_{\mathbf{g}}^{\mathrm{T}} \cdot \mathbf{J}_{\mathbf{g}} + \mu \cdot \mathbf{I}\right)^{-1} \mathbf{J}_{\mathbf{g}}^{\mathrm{T}} \cdot \mathbf{g}. \tag{4}$$

$\mathbf{I} \in \mathbb{R}^{6L \times 6L}$ is the identity matrix, μ is the damping term that controls the step size of the iteration and $\mathbf{J}_{\mathbf{g}}$ is the Jacobian matrix which is computed by taking the first order derivatives of $\mathbf{g}$ with respect to every $\mathbf{a}_i$

$$\mathbf{J}_{\mathbf{g}} = \begin{bmatrix} \frac{\partial g_{1,1}}{\partial \mathbf{a}_1^{\mathrm{T}}} & \cdots & \frac{\partial g_{1,1}}{\partial \mathbf{a}_L^{\mathrm{T}}} \\ \vdots & & \vdots \\ \frac{\partial g_{N,L}}{\partial \mathbf{a}_1^{\mathrm{T}}} & \cdots & \frac{\partial g_{N,L}}{\partial \mathbf{a}_L^{\mathrm{T}}} \end{bmatrix} \in \mathbb{R}^{N \cdot L \times 6L}. \tag{5}$$

The term $\mathbf{J}_{\mathbf{g}}^T \cdot \mathbf{J}_{\mathbf{g}}$ is the approximation of the Hessian matrix used in the Gauss-Newton algorithm. Hence, $\mu \cdot \mathbf{I}_6$ can be thought of as a regularization term for the Hessian in case if it is close to singularity. Due to this term, LM has the properties of both the steepest descent and the Gauss-Newton iterative techniques. Similarly to steepest descent, LM converges slowly to the solution if the initial estimate is far away from the global solution while it converges fast like Gauss-Newton methods when the initial estimate is close. When solving for Equation (4), LM can be tremendously accelerated when the sparsity of the Hessian is taken into account. A thorough analysis of the scheme is found in [7, 8, 6, 15, 1].

2.2 Newton Algorithm in the Vector Space

The aim of a Newton-type algorithm is to find the extrenum of a given cost function in an iterative scheme. In our case, the update step used to minimize Equation (1) has the

following form

$$\mathbf{p}^{(k+1)} = \mathbf{p}^{(k)} - \mathbf{H}_f^{-1} \cdot \nabla f, \tag{6}$$

where k is the iteration number as before, $\nabla f = 2\mathbf{J}_{\mathbf{g}}^{\mathrm{T}} \cdot \mathbf{g}$ is the gradient of the cost function f and $\mathbf{H}_f \in \mathbb{R}^{6\cdot L \times 6\cdot L}$ is the corresponding Hessian. The Hessian $\mathbf{H}_f$ is expressed by

$$\mathbf{H}_f = 2\mathbf{J}_{\mathbf{g}}^{\mathrm{T}} \cdot \mathbf{J}_{\mathbf{g}} + 2\sum_{i=1}^{L}\sum_{j=1}^{N}\left(g_{j,i} \cdot \mathbf{H}_{g_{j,i}}\right). \tag{7}$$

The matrix $\mathbf{H}_{g_{j,i}}$ denotes the Hessian with respect to each term of $\mathbf{g}$. It is given by

$$\mathbf{H}_{g_{j,i}} = \begin{bmatrix} \frac{\partial}{\partial \mathbf{a}_1^{\mathrm{T}}}\frac{\partial g_{j,i}}{\partial \mathbf{a}_1} & \cdots & \frac{\partial}{\partial \mathbf{a}_L^{\mathrm{T}}}\frac{\partial g_{j,i}}{\partial \mathbf{a}_1} \\ \vdots & & \vdots \\ \frac{\partial}{\partial \mathbf{a}_1^{\mathrm{T}}}\frac{\partial g_{j,i}}{\partial \mathbf{a}_L} & \cdots & \frac{\partial}{\partial \mathbf{a}_L^{\mathrm{T}}}\frac{\partial g_{j,i}}{\partial \mathbf{a}_L} \end{bmatrix} \in \mathbb{R}^{6\cdot L \times 6\cdot L}. \tag{8}$$

By setting the second term of Equation (7) to zero, the approximated Hessian obtained, i.e. $\hat{\mathbf{H}}_f = 2\mathbf{J}_{\mathbf{g}}^{\mathrm{T}} \cdot \mathbf{J}_{\mathbf{g}}$, is what is used in a Gauss-Newton iterative technique. By adding the damping term $\mu \cdot \mathbf{I}$ to $\hat{\mathbf{H}}_f$ leads us to the approximate Hessian used in LM, see Equation (4). The complexity induced by the computation of $\mathbf{H}_{g_{j,i}}$ is the reason that makes LM more favorable in BA. The optimization in hand is, however, a large-scale problem. Failing to take $\mathbf{H}_{g_{j,i}}$ into account in such a case makes the output diverges from the global solution especially if the initial estimate of the motion is noisy or far from the global solution [9]. This can be justified since the minimization without these terms will be more susceptible to be stuck in local minima. It is important to note that the sparsity of $\mathbf{H}_f$ can be explored as in LM to increase the speed of the computations by taking advantage of the zero patterns in the matrix. It is also possible to add a damping factor $\mu \cdot \mathbf{I}$ to $\mathbf{H}_f$ like in Equation (4) to regularize the Hessian if it is close to singularity or to vary the speed of convergence. A good description on how to perform such adjustments to the Hessian are found in [15].

3 The Proposed Projective Newton-Type Approach

Computation of the second order terms in a Newton algorithm, i.e. right addend in Equation (7), is not favorable due to the complexity arising from the computation of the Hessian. To get a glimpse of how the computational complexity rises, the number of floating point operations (FLOPS) required per camera per iteration are illustrated in Table 1. As can be noticed, including these terms makes the complexity required for the motion refinement rises to 5 times more when compared to that needed in LM. The aim of this section is, therefore, to propose an optimization method to enhance the accuracy of the motion estimate in BA without a significant increase in the complexity. To do that, a Newton minimization scheme on the *manifold* will be derived. A manifold is a topological space in which each point has a neighborhood that looks like the Euclidean space. Designing an optimization scheme on a manifold can be visualized as making the minimization iteratively walk on the surface of the curve described by the variables

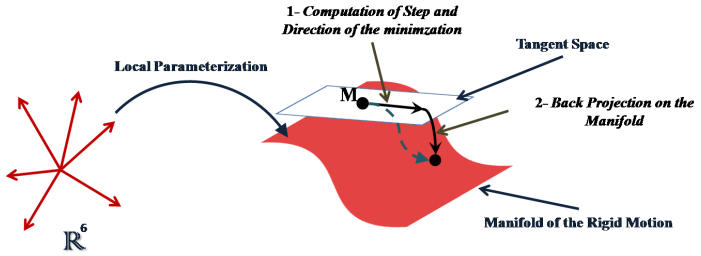

Fig. 1. Projective Newton-type algorithm on the manifold. A local parameterization maps the variables from $\mathbb{R}^6$ to the manifold of the rigid motion. The minimization is then performed in two steps: The optimization direction and step size are computed on the tangent space to the manifold. The calculated values are then projected back on the manifold to update the motion.

until the minimum is achieved. In order to perform such operations, it is necessary that the manifold possesses a special structure. It has to be differentiable and smooth so that notions like distances and angles, necessary to determine the direction of optimization in a Newton scheme, can be defined. Examples are the *Riemannian* manifolds where each has the nice property of being equipped with a tangent space that allows the transitions from a point to another one to be smooth. An instance of a Riemannian manifold is the special orthogonal group SO_3 that describes the $3\times$ 3 rotations matrices, see Equation (2). The tangent space to this group is its associated Lie algebra $\mathbf{so}_3$ which is related to the latter using the exponential map. To derive a Newton-type method, it is necessary to compute first and second order derivatives. An intrinsic Newton method, that is, a Newton method which is intrinsically defined on the Riemannian manifold without referring to any embedding vector space, requires the notion of a Riemannian gradient and a Riemannian Hessian. This idea often requires an overhead of differential geometric machinery. Here, a projected Newton-type method is formulated instead that exploits a local parameterization of the variables on the manifold. The direction and step size of the optimization are computed using the associated tangent space. The result is then projected back on the manifold to update the variables. The mechanism of such schemes is illustrated in Figure 1. To apply this approach, however, a suitable local parameterization of the rigid motion of the camera must be first determined. For good references that provide detailed discussions about Lie theory and differential geometry, see [16, 10, 17–19].

In the following, the derivations will be for simplicity made for one camera since the motion variables of each pose are independent from each other. The index i in Equation (1) that designates the camera number will thus be dropped. The extension of the algorithm to more than one camera is among its lines.

3.1 Parametrization of the Rigid Motion

The rigid motion in $\mathbb{R}^6$ is represented by the elements of the special Euclidean group SE_3 which consists of all the 4×4 matrices $\mathbf{M} \in SE_3$ of the form

$$\mathbf{M} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}, \tag{9}$$

where $\mathbf{R} \in SO_3$ and $\mathbf{t} \in \mathbb{R}^3$ are the rotation and the translation of the camera, respectively. The Lie algebra se_3 associated to SE_3 is the set of 4×4 matrices $\mathbf{m} \in se_3$

$$\mathbf{m} = \begin{bmatrix} \boldsymbol{\Omega}(\omega) & \mathbf{v} \\ \mathbf{0} & 0 \end{bmatrix}, \tag{10}$$

where $\boldsymbol{\Omega}(\omega) \in so_3$ is a skew symmetric-matrix as shown in Equation (2) and $\mathbf{v}$ is a vector in $\mathbb{R}^3$. The matrix $\mathbf{m}$ is related to $\mathbf{M}$ via the exponential map, i.e. $\mathbf{M} = \exp(\mathbf{m})$. Using the Taylor series expansion of the exponential, it can be easily shown that the translation can be written as

$$\mathbf{t} = \left(\mathbf{I} + \frac{1 - \cos(\theta)}{\theta^2} \cdot \boldsymbol{\Omega}(\omega) + \frac{\theta - \sin(\theta)}{\theta^3} \cdot \boldsymbol{\Omega}(\omega)^2 \right) \cdot \mathbf{v} = \boldsymbol{\Theta} \cdot \mathbf{v}, \tag{11}$$

where $\theta = \sqrt{\frac{1}{2}\mathrm{trace}\left(\boldsymbol{\Omega}^{\mathrm{T}}(\omega) \cdot \boldsymbol{\Omega}(\omega)\right)}$. The rotation $\mathbf{R}$ remains the same as was defined in Equation (3), i.e. $\mathbf{R} = \exp(\boldsymbol{\Omega}(\omega))$. To design a Newton-type optimization scheme on a manifold, it is necessary to employ ω and $\mathbf{v}$ to parameterize the rotation and the translation, i.e. $\mathbf{a} = \begin{bmatrix} \omega^{\mathrm{T}} & \mathbf{v}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$, respectively. We therefore propose the following smooth and local parametrization ϕ of SE_3 at the camera motion $\mathbf{M}$ to be

$$\phi_{(\mathbf{M})} : \mathbb{R}^6 \rightarrow SE_3, \phi_{(\mathbf{M})}(\mathbf{a}) = \exp\left(\begin{bmatrix} \boldsymbol{\Omega}(\mathbf{S}_1\mathbf{a}) & \mathbf{S}_2\mathbf{a} \\ \mathbf{0} & 0 \end{bmatrix} \right) \cdot \mathbf{M} = \exp(\boldsymbol{\Lambda}_\phi) \cdot \mathbf{M}, \tag{12}$$

where $\mathbf{S}_1 = [\mathbf{I}\ \mathbf{0}] \in \mathbb{R}^{3\times6}$ and $\mathbf{S}_2 = [\mathbf{0}\ \mathbf{I}] \in \mathbb{R}^{3\times6}$ select the appropriate variables.

3.2 Minimization of the Cost Function

A projected Newton-type method computes a Newton step for the composition $f \circ \phi$ of the cost function f with the local parametrization ϕ and maps (projects) the update step back to the manifold using the parametrization again, see Figure 1. We slightly abuse the notation to write $f \circ \phi_{(\mathbf{M})}$ to denote the mapping

$$f \circ \phi_{(\mathbf{M})} : \mathbb{R}^6 \rightarrow \mathbb{R}, \tag{13}$$

which corresponds here to the cost function for each pose of the camera to be minimized on the manifold. It is defined by

$$\begin{aligned} f \circ \phi_{(\mathbf{m})}(\mathbf{a}) &:= \sum_{j=1}^{N} \mathrm{d}(\mathbf{x}_j, \hat{\mathbf{x}}_j)^2 \\ &= \sum_{j=1}^{N} \mathrm{d}(\mathbf{x}_j, \pi \cdot \exp(\boldsymbol{\Lambda}_\phi) \cdot \mathbf{M} \cdot \mathbf{X}_j)^2, \end{aligned} \tag{14}$$

where $\mathrm{d}(\mathbf{x}_j, \hat{\mathbf{x}}_j)$ is the distance that measures the reprojection error, e.g. Mahalanobis, cosine distance or L2 norm and π is a mapping to scale the multiplication of the right

term of the distance function from $\mathbb{R}^4$ to $\mathbb{R}^3$ to preserve the dimensions. The gradient $\nabla(f \circ \phi_{(\mathbf{M})})(0) \in \mathbb{R}^6$ of the cost function can be calculated from

$$\frac{\mathrm{d}}{\mathrm{d}\varepsilon}(f \circ \phi_{(\mathbf{M})})(\varepsilon \mathbf{a})\bigg|_{\varepsilon=0} = \nabla(f \circ \phi_{(\mathbf{M})})(0)^{\mathrm{T}} \cdot \mathbf{a}. \tag{15}$$

The explicit formula for the gradient of the local cost function of BA at zero is

$$\begin{aligned} \nabla(f \circ \phi_{(\mathbf{M})})(0) &= 2\sum_{j=1}^{N} \left(\underbrace{[-\boldsymbol{\Omega}\,(\pi \mathbf{M}\mathbf{X}_j) \quad \mathbf{I}]^T \cdot \frac{\partial \mathrm{d}\,(\mathbf{x}_j, \hat{\mathbf{x}}_j)}{\partial \hat{\mathbf{x}}_j}}_{\rightarrow \mathbf{J}_i} \cdot \underbrace{\mathrm{d}\,(\mathbf{x}_j, \hat{\mathbf{x}}_j)}_{\rightarrow \mathbf{g}_i} \right) \\ &= 2\mathbf{J}_i^T \cdot \mathbf{g}_i, \end{aligned} \tag{16}$$

where $\frac{\partial \mathrm{d}(\mathbf{x}_j, \hat{\mathbf{x}}_j)}{\partial \hat{\mathbf{x}}_j} \in \mathbb{R}^3$ is the standard Euclidean derivative of the distance function with respect to $\hat{\mathbf{x}}$ and $\mathbf{I} \in \mathbb{R}^{3\times 3}$ is the identity matrix. Equation (16) corresponds to the gradient of the i^{th} camera. It is possible to gather the equations of all the cameras by padding each column vector $\mathbf{J}_i$ into a block diagonal matrix and each $\mathbf{g}_i$ into a single vector. This leads us to

$$\nabla(f \circ \phi_{(\mathbf{M}_1, \cdots, \mathbf{M}_L)})(0) = 2\mathbf{J}^{\mathrm{T}}_{f \circ \phi_{(\mathbf{M}_1, \cdots, \mathbf{M}_{L,})}} \cdot \mathbf{g}, \tag{17}$$

where $\nabla(f \circ \phi_{(\mathbf{M}_1, \cdots, \mathbf{M}_{L,})})(0)$ is the total gradient and $\mathbf{J}_{f \circ \phi_{(\mathbf{M}_1, \cdots, \mathbf{M}_L)}}$ is the corresponding Jacobian matrix for all the system. The Hessian matrix $\mathbf{H}_{f \circ \phi_{(\mathbf{M})}}(0) \in \mathbb{R}^{6\times 6}$ of the local cost function evaluated at zero can be calculated from the quadratic form

$$\frac{\mathrm{d}^2}{\mathrm{d}\varepsilon^2}(f \circ \phi_{(\mathbf{M})})(\varepsilon \mathbf{a})\bigg|_{\varepsilon=0} = \mathbf{a}^{\mathrm{T}} \cdot \mathbf{H}_{f \circ \phi_{(\mathbf{M})}}(0) \cdot \mathbf{a} \tag{18}$$

via polarization. The explicit formula of the Hessian is given by

$$\begin{aligned} \mathbf{H}_{f \circ \phi_{(\mathbf{M})}}(0) = 2\mathbf{J}_i^{\mathrm{T}} \cdot \mathbf{J}_i + 2\textstyle\sum_{j=1}^{N} \left(\mathbf{S}_1^{\mathrm{T}} \boldsymbol{\Omega} \left(\frac{\partial \mathrm{d}(\mathbf{x}_j, \hat{\mathbf{x}}_j)}{\partial \hat{\mathbf{x}}_j} \right) [\boldsymbol{\Omega}\,(\pi \mathbf{M}\mathbf{X}_j) \quad \mathbf{I}] \mathrm{d}\,(\mathbf{x}_j, \hat{\mathbf{x}}_j) \right) \\ + 2\textstyle\sum_{j=1}^{N} \left([-\boldsymbol{\Omega}\,(\pi \mathbf{M}\mathbf{X}_j) \quad \mathbf{I}]^T \cdot \frac{\partial^2 \mathrm{d}(\mathbf{x}_j, \hat{\mathbf{x}}_j)}{\partial \hat{\mathbf{x}}_{\mathbf{j}}^2} [-\boldsymbol{\Omega}\,(\pi \mathbf{M}\mathbf{X}_j) \quad \mathbf{I}] \mathrm{d}\,(\mathbf{x}_j, \hat{\mathbf{x}}_j) \right), \end{aligned} \tag{19}$$

where $\frac{\partial \mathrm{d}^2(\mathbf{x}_j, \hat{\mathbf{x}}_j)}{\partial \hat{\mathbf{x}}_j^2} \in \mathbb{R}^{3\times 3}$ is the second order derivative of the distance function with respect to $\hat{\mathbf{x}}$. As in Equation (17), the Hessian from each camera can be assembled into a sparse block diagonal matrix $\mathbf{H}_{f \circ \phi_{(\mathbf{M}_1, \cdots, \mathbf{M}_L)}}(0) \in \mathbb{R}^{6L\times 6L}$.

By considering the computed gradient and Hessian, we can observe that their form is compact. There is no lengthy computations in the evaluation of the first and second order derivatives of the distance function as in Equations (5) and (7) in the vector space. The only derivatives that need to be evaluated are the first and second order Euclidean derivatives of the cost function with respect to $\hat{\mathbf{x}}$, i.e. $\frac{\partial \mathrm{d}(\mathbf{x}_j, \hat{\mathbf{x}}_j)}{\partial \hat{\mathbf{x}}_j}$ and $\frac{\partial \mathrm{d}^2(\mathbf{x}_j, \hat{\mathbf{x}}_j)}{\partial \hat{\mathbf{x}}_j^2}$, which are simple to obtain. The reason for this is that the computation of the derivatives with

respect to the motion parameters reduces to matrix multiplications which is a property of the projective Newton type optimization algorithms on Riemannian manifolds [10]. This can be illustrated in our derivations by the term $[-\mathbf{\Omega}(\pi\mathbf{M}\mathbf{X}_j) \quad \mathbf{I}]^T$ which represents the derivative with respect to the camera parameters. By comparing Equations (16) and (19), we can see that a lot of terms needed for the computations of the Hessian have to be also computed for the gradient, i.e. $[-\mathbf{\Omega}(\pi\mathbf{M}\mathbf{X}_j) \quad \mathbf{I}]^T$ and $\frac{\partial \mathrm{d}(\mathbf{x}_j, \hat{\mathbf{x}}_j)}{\partial \hat{\mathbf{x}}_j}$.

3.3 Algorithm

Given some initial estimates of the motion $\mathbf{M}_i$ and 3D structure, the proposed optimization algorithm will iterate the following update scheme:

Step 1: (Newton step) Compute the vector $\mathbf{p} = \left[\mathbf{a}_1^{\mathrm{T}}, \ldots, \mathbf{a}_L^{\mathrm{T}}\right]^{\mathrm{T}} \in \mathbb{R}^{6L}$ by solving

$$-\left(\mu \cdot \mathbf{I} + \mathbf{H}_{f \circ \phi_{(\mathbf{M}_1, \cdots, \mathbf{M}_L)}}(0)\right) \cdot \mathbf{p} = \nabla(f \circ \phi_{(\mathbf{M}_1, \cdots, \mathbf{M}_L)})(0). \tag{20}$$

Step 2: Update the motion parameters of each camera

$$\mathbf{M}_i \leftarrow \exp\mathrm{m}(\mathbf{\Lambda}_\phi) \cdot \mathbf{M}_i \tag{21}$$

Step 3: Update the reprojections $\hat{\mathbf{x}}_j^i$ and compute the mean reprojection error.
Step 4: Repeat until convergence.

3.4 Analysis of the Derived Scheme

The projective Newton-based minimization algorithm is initialized by some initial values of structure and motion. Then, it proceeds with the iterative scheme illustrated in the previous section. Each iteration, the algorithm performs an optimization step in Equation (20) on the tangent space of the manifold of the rigid motion followed by a projection step which is an analytic geodesic search described in Equation (21), see Figure 1. The proposed technique implements the Riemannian Newton algorithm on a small neighborhood of the set of local minima of the cost function. Outside of this neighborhood where the Hessian is close to singularity or indefinite, the algorithm might fail. For this reason, Equation (20) is modified using the damping factor μ as in case of the LM technique to regularize the Hessian. This modification is necessary here to enlarge the attraction domain of the local minima and to control the speed of convergence of the algorithm. The direction of the step is determined via the Newton direction and it is zero only when the gradient is zero. Consequently, the algorithm converges to a critical point of the cost function when the gradient is zero.

3.5 Complexity Analysis of the Algorithms

The general core of the proposed Newton-type algorithm in both of its versions is pretty similar to LM. The only difference resides in the way the gradients and the Hessians are evaluated. In order to emphasize on the computational complexity of these terms, we present in Table 1 a comparison of the computational costs required for the gradient

and Hessian for each of the minimization algorithms in a single iteration. The factors multiplied by $L \times N$ are the operations that have to be evaluated for each point and each view while the ones multiplied by L are only required per view, e.g. Rodrigues formula in Equation (3) to compute the rotations. Each of the shown numbers in the table designates the equivalent amount of flops needed while the operations that require more than one flop, e.g. cosine and sine, are mentioned by there names since they depend on the machine used. LM requires the least computational effort to compute these terms while the Newton algorithm in the vector space requires the most. This shows why the evaluation of the full Hessian shown in Equation (7) is usually avoided in BA. However, the amount of computations required by the projective Newton algorithm on the manifold is considerably less (60%) than that of the vector space due to the simpler derivations. Compared to LM, the complexity induced by the proposed method is slightly higher. However, its application will lead to an improvement in the accuracy of the motion estimates and will make the result more robust against the noise. This point will be emphasized in the results.

Algorithm	Number of Flops
Levenberg Marquardt	$L \times N(114 + 2 \cdot \mathrm{sqrt}) + L(227 + \mathrm{sqrt} + \cos + \sin + \mathrm{expm})$
Newton Manifold	$L \times N(241 + 2 \cdot \mathrm{sqrt}) + L(15 + \mathrm{expm})$
Newton Vectorspace	$L \times N(564 + 2 \cdot \mathrm{sqrt}) + L(1024 + \mathrm{sqrt} + \sin + \cos + \mathrm{expm})$

Table 1. Number of flops needed for the computation of the gradient and the Hessian per iteration. N is the number of points, L is the number of cameras, sqrt is the scalar square root, $\cos$ is the scalar cosine, $\sin$ is the scalar sine and expm is the matrix exponential.

3.6 Integrating the Structure Optimization

The method is till the moment only derived for the camera poses. We will briefly sketch now how to integrate the structure in the optimization. Let us recall that each point $\mathbf{X}_j$ of the structure can be parameterized using its coordinates in 3D space. The cost function shown in Equation (14) should be reformulated to designate the mapping

$$f \circ \phi_{(\mathbf{M}_1,\cdots,\mathbf{M}_L,\mathbf{X}_1,\cdots,\mathbf{X}_N)}(\mathbf{z}) \;:\; \mathbb{R}^{6L+3N} \rightarrow \mathbb{R} \qquad (22)$$
$$:= \sum_{i=1}^{L}\sum_{j=1}^{N} \mathrm{d}\left(\mathbf{x}_j^i, \pi \cdot \mathrm{expm}\left(\mathbf{\Lambda}_\phi^i\right) \cdot \mathbf{M}_i \cdot \mathbf{X}_j\right)^2,$$

where the vector $\mathbf{z} \in \mathbb{R}^{6L+3N}$ holds the parameters of all the camera parameters and the 3D points. The derivations of the gradient $\nabla\left(f \circ \phi_{(\mathbf{M}_1,\cdots,\mathbf{M}_L,\mathbf{X}_1,\cdots,\mathbf{X}_N)}\right)(0) \in \mathbb{R}^{6\mathrm{L}+3\mathrm{N}}$ and the Hessian $\mathbf{H}_{f\circ\phi_{(\mathbf{M}_1,\cdots,\mathbf{M}_L,\mathbf{X}_1,\cdots,\mathbf{X}_N)}}(0) \in \mathbb{R}^{6\mathrm{L}+3\mathrm{N}\times 6\mathrm{L}+3\mathrm{N}}$ of the cost function at zero are then computed following a similar concept to Equations (15) and (18). These are used after that to calculate the solution vector $\mathbf{z}$ using the following update scheme

$$-\left(\mu \cdot \mathbf{I} + \mathbf{H}_{f\circ\phi_{(\mathbf{M}_1,\cdots,\mathbf{M}_L,\mathbf{X}_1,\cdots,\mathbf{X}_N)}}(0)\right) \cdot \mathbf{z} = \nabla\left(f \circ \phi_{(\mathbf{M}_1,\cdots,\mathbf{M}_L,\mathbf{X}_1,\cdots,\mathbf{X}_N)}\right)(0). \qquad (23)$$

The Hessian matrix of the overall system is sparse in nature due to the formulation of BA. Therefore, the zero patterns of this entity can be exploited in the inversion operation to obtain the non-linear least-squares solution of $\mathbf{z}$. For more information about this topic, the interested reader may refer to [1, 20] to see how this issue is usually performed.

4 Experimental Results

We will describe in this section a comparison regarding the application of LM, the Newton algorithm in the vector space and the proposed method. We will be testing the techniques using one simulated data set and three real image sequences. The goal of our tests is to demonstrate the performance of the algorithms in terms of the convergence and the accuracy of the estimated motion. We will therefore not perform any structure optimization in order to assess the robustness of each of the optimization schemes. For every algorithm, it is assumed that the cameras are intrinsically calibrated and that the 3D structure is already computed along with the corresponding image projections. It is important to point out that the three methods are implemented in Matlab and they possess the same structure. The only dissimilarity among them resides in the way the gradients and the Hessians are evaluated which was thoroughly discussed in the previous sections and illustrated in Table 1. Consequently, the divergence in the execution is only dependent on this difference.

Figure 2 shows the output of the algorithms when applied to the simulated data set. The set was generated by creating 100 random 3D points and projecting them on four random 256×256 images in the noise free case. The feature points and the ground truth 3D structure present the inputs to the Newton algorithms while the initial estimate for the motion was randomly chosen. The error in the rotation is measured by transforming the rotation matrix to the Euler angles, i.e. roll, pitch and yaw, and then calculating the absolute difference between the estimated and the ground truth values. For the translation, the error is computed by evaluating the distance between the estimated vectors and the corresponding true values. The results illustrated in the plots show the average over 1000 trials. They demonstrate that taking the total Hessian formula and not just the approximation leads to more accuracy in the estimated motion.

The first real image sequence that we will test the algorithms on is the Hall stereo sequence which was acquired by ourselves. Example images from the sequence are illustrated in Figure 3. The sequence consists of nine stereo images of size 640×480. The intrinsic and extrinsic parameters of the stereo rig were computed using the camera calibration toolbox for Matlab [21]. The motion of the rig consists of both rotational and translational movements. The ground truth values of these movements are known. Using some features tracked over all the sequence, we computed nine partial 3D structures using each stereo pair by triangulating the points with the computed camera matrices of the stereo rig. In order to improve the accuracy of the estimate of the scene, we determined the epipolar geometry between each pair using the robust version of the five-point algorithm of [22] proposed in [23], and used the epipolar constraint to eliminate the outliers and to increase the number of feature points by guiding the correspondences. We then applied the absolute orientation technique of [24] to compute a linear estimate of

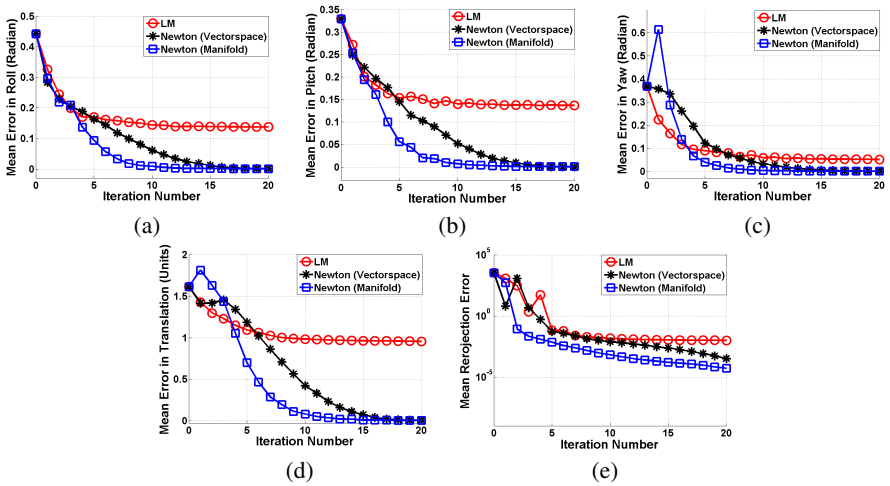

Fig. 2. Result of the algorithms on the simulated data sets. From a to d in the respective order: Error in the roll, pitch, yaw and translation estimates respectively versus the number of iterations to converge in the noise free set. e: Mean reprojection error versus the number of iterations averaged over all images. The result is for 100 points, 4 images of size 256×256 and random initial estimates of the motion for each camera averaged over 1000 trials.

the motion by calculating a homography between the sequential partial structures and hence resulting in eight initial estimates for the motion. We used these estimates to obtain an initial alignment of the nine partial structures into a single one. The structure was then inputted along with the initial estimates of the motion to refine its alignment with the minimization schemes. The result of this experiment is depicted in Figure 4. We show here the average error in the estimated motion for each frame as was done in the simulated data sets but using the ground truth values of the motion along with the average mean reprojection error versus the number of iterations. As a reference, we also provided the output of the linear algorithm of Ref [24]. The first thing we can notice is that the proposed Newton algorithm in its both versions is able to reduce the mean reprojection error to the order of 10^{-5} while LM is not able to effectively minimize the error of the linear method. We can also realize that the average error in the motion variables is almost constant among all the frames of the sequence and is much lower than the output of LM. This result motivates for the employment of the manifold based Newton algorithm in BA since the error does not propagate as in LM.

The last two data sets we will use are the Corridor and the Dinosaur sequences where each one consists of 11 and 36 frames, respectively[1]. As a preprocessing step, we first matched and tracked some feature points over each image sequence. We estimated the intrinsic parameters of the images by applying the technique of [25]. Using the algorithm of [23], we computed the essential matrices between the sequential im-

[1] The two sequences can be downloaded from the website of the VGG group at the University of Oxford: `www.robots.ox.ac.uk/~vgg/data/data-mview.html`.

Fig. 3. Example images from the Hall stereo sequence.

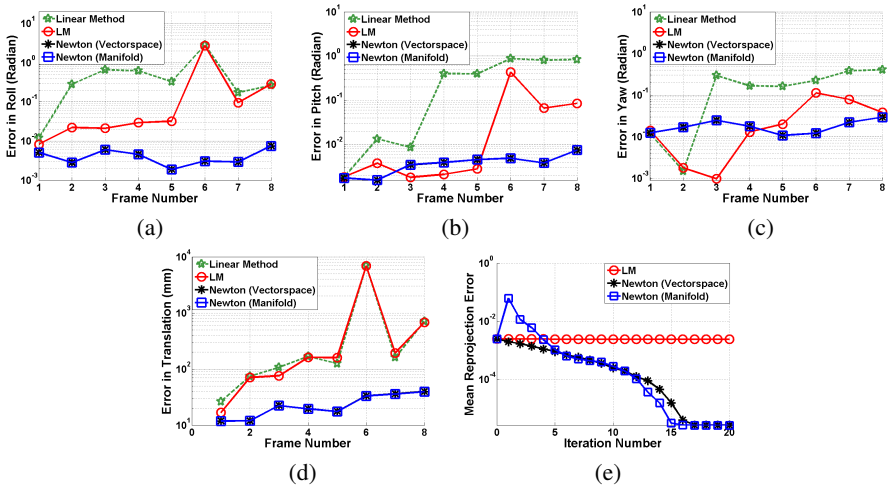

Fig. 4. Result of the minimization algorithms on the Hall stereo sequence using close linear initial estimate of the motion. From a to d: Error in the estimates of the roll, pitch, yaw and translation respectively. e: Mean reprojection error versus the number of iterations averaged over all the sequence.

ages, rejected the feature matches that do not satisfy the epipolar constraint and then computed more correspondences by guiding the matches. The initial estimates of the motion are computed by factorizing the essential matrices, triangulating some of the match points and checking which of the possible camera matrices satisfy the chieralities [1]. The 3D structure was then obtained by triangulating the points with the initial camera matrices. Using the three minimization schemes, we refined the motion estimates and the results are shown in Figure 5. In both sequences, the proposed Newton method and the one in the vector space result in a much more accurate estimate of the motion. By just optimizing the motion, the techniques were able to recover the shapes of the motion. The obtained poses of the camera in the Corridor sequence look pretty similar to the ones shown in [26] while those of the Dinosaur sequence, which is actually a turn-table sequence see [27], have a closed circular shape. The ground truth set of the latter was taken in steps of 10° rotation with a standard deviation of 0.05.

Algorithm	Mean	Std	Number of Outliers
Ground Truth	10	0.05	0
LM	1.11	58.3	20
Newton Vectorspace	10	0.360	0
Newton Manifold	10	0.389	0

Table 2. Results of the minimization schemes on the Dinosaur sequence along with the ground truth values of the motion. The mean and standard deviation of the overall camera poses are in degrees. The outliers are the camera positions that are not lying on the circle.

The values obtained with each of the algorithms are depicted in Table 2 along with the number of outliers which reflect the positions that are not lying on the circle. This table also confirms that the estimated camera motion of the proposed scheme is pretty accurate. The LM has failed in these sets since the obtained initial estimates are not as close as it is usually ensured in such problems. In addition, the proposed projective Newton scheme was able to accurately capture the motion while requiring around 60% less computational requirements than the Newton algorithm in the vector space.

5 Conclusion

We presented in this paper an Newton-type minimization scheme which refines the motion of the camera by optimizing on a Riemannian manifold. We modified the cost function of BA to accommodate for the new parametrization scheme and then derived the full formulas of the gradients and the Hessians to be used in the optimization. Compared to a similar Newton-based technique in the vector space, the proposed algorithm has a simpler evaluation of the gradient and the Hessian since the derivations on a Riemannian manifold are more compact. Compared to LM, the method has a comparable complexity in terms of the evaluation of the gradient and Hessian. In addition, its application leads to more accurate results and it is more robust against the noise. For these reasons, the proposed Newton algorithm on the manifold seems to be a more promising implementation for the non-linear minimization problems arising in BA.

As a future work, we are exploring the possibilities to parameterize the 3D structure by analyzing the underlying manifold it describes. Our aim is to uncover the hidden dimensions of the structure by exploiting the manifold properties in order to optimize it with BA using fewer parameters.

References

1. Hartley, R., Zisserman, A.: Multiple view geometry in computer vision. Cambridge University Press (2004)
2. Pollefeys, M., Gool, L.V., Vergauwen, M., Verbiest, F., Cornelis, K., Tops, J., Koch, R.: Visual modeling with a hand-held camera. Int. J. Computer Vision **59** (2004) 207–232
3. Bartoli, A., Sturm, P.: Structure-from-motion using lines: Representation, triangulation, and bundle adjustment. Computer Vision and Image Understanding **100** (2005) 516–441

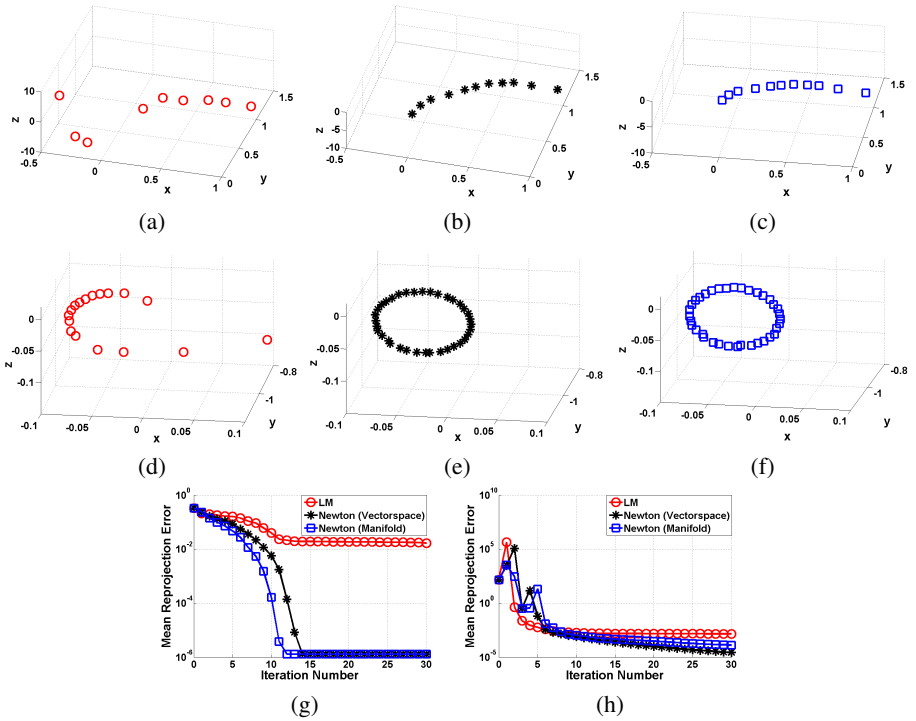

Fig. 5. Result of applying the algorithms to the Corridor and the Dinosaur sequences. a and d: Extracted camera poses of the Corridor and Dinosaur sequences respectively using Levenberg-Marquardt. b and e: same as a and d but using Newton Vectorspace. c and f: same as a and d but using the proposed manifold scheme. g and h: Mean reprojection error versus the number of iterations averaged over all the sequence of the Corridor and Dinosaur sequences respectively.

4. Stewénius, H., Engels, C., Nistér, D.: Recent developments on direct relative orientation. ISPRS J. Photogrammetry and Remote Sensing **60** (2006) 284–294
5. Ni, K., Steedly, D., Dellaert, F.: Out-of-core bundle adjustment for large-scale 3D reconstruction. In: Int. Conf. Computer Vision. (2007) 1–8
6. Triggs, B., McLauchlan, P., Hartley, R., Fitzgibbon, A.: Bundle adjustment - a modern synthesis. In: Proc. Int. Workshop on Vision Algorithms. (2000) 298–375
7. Levenberg, K.: A method for the solution of certain non-linear problems in least-squares. Quarterly of Applied Mathematics **2** (1944) 164–168
8. Marquardt, D.: An algorithm for the least-squares estimation for non-linear parameters. SIAM J. Applied Mathematics **11** (1963) 431–441
9. Dennis, J.E., Gay, D.M., Walsh, R.E.: An adaptive nonlinear least-squares algorithm. ACM Transactions on Mathematical Software **7** (1981)
10. Udriste, C.: Convex functions and optimization methods on Riemannian manifolds. Kluwer Academic Publishers (1994)
11. Kumar, R., Hanson, A.R.: Robust methods for estimating pose and a sensitivity analysis. Computer Vision, Graphics, and Image Processing: Image Understanding **60** (1994) 313–342

12. Quan, L., Lan, Z.: Linear n-point camera pose determination. IEEE Trans. Pattern Analysis and Machine Intelligence **21** (1999) 774–780
13. Lu, C.P.: Fast and globally convergent pose estimation from video images. IEEE Trans. Pattern Analysis and Machine Intelligence **22** (2000) 610–622
14. Ansar, A., Daniilidis, K.: Linear pose estimation from points or lines. IEEE Trans. Pattern Analysis and Machine Intelligence **25** (2003) 1–22
15. Nocedal, J., Wright, S.: Numerical optimization. Springer (2000)
16. Taylor, C.J., Kriegman, D.J.: Minimization on the lie group SO(3) and related manifolds. Technical Report 9405, Yale University (1994)
17. Baker, A.: Matrix groups: An introduction to Lie group theory. Springer (2003)
18. Ma, Y., Soatto, S., Kosecka, J., Sastry, S.S.: An invitation to 3-D vision. Springer (2005)
19. LaValle, S.M.: Planning Algorithms. Cambridge University Press (2006)
20. Lourakis, M., Argyros, A.: The design and implementation of a generic sparse bundle adjustment software package based on the levenberg-marquardt algorithm. Technical Report 340, Institute of Computer Science - FORTH, Heraklion, Greece (2004) http://www.ics.forth.gr/∽lourakis/sba/.
21. Bouguet, J.Y.: Camera calibration toolbox for MATLAB. Computational Vision Group, California Institute of Technology, Pasadena, CA, USA. (2001) http://www.vision.caltech.edu/bouguetj/calib_doc.
22. Helmke, U., Hüper, K., Lee, P., Moore, J.: Essential matrix estimation using Gauss-Newton iterations on a manifold. Int. J. Computer Vision **74** (2007) 117–136
23. Sarkis, M., Diepold, K., Hüper, K.: A fast and robust solution to the five-point relative pose problem using gauss-newton optimization on a manifold. In: IEEE Int. Conf. Acoustics, Speech and Signal Processing. (2007) I–681–I–684
24. Horn, B.K.P.: Closed-form solution of absolute orientation using unit quaternions. J. Optical Society of America A **4** (1987) 629–642
25. Pollefeys, M., Koch, R., Gool, L.V.: Self-calibration and metric reconstruction in spite of varying and unknown internal camera parameters. In: Int. Conf. Computer Vision. (1998) 90–95
26. Fitzgibbon, A.W., Cross, G., Zisserman, A.: Automatic 3D model construction for turn-table sequences. In: Proc. SMILE Workshop on Structure from Multiple Images in Large Scale Environments. (1998) 154–170
27. Niem, W.: Robust and fast modelling of 3D natural objects from multiple views. In: SPIE Image and Video Processing II. (1994) 338–397