Camera Pose Estimation via Projective Newton Optimization on the Manifold

Michel Sarkis*, *Member, IEEE,* and Klaus Diepold

Regular Paper

***Abstract*—Determining the pose of a moving camera is an important task in computer vision. In this paper, we derive a projective Newton algorithm on the manifold to refine the pose estimate of a camera. The main idea is to benefit from the fact that the 3D rigid motion is described by the Special Euclidean group which is a Riemannian manifold. The latter is equipped with a tangent space defined by the corresponding Lie algebra. This enables us to compute the optimization direction, i.e. gradient and Hessian, at each iteration of the projective Newton scheme on the tangent space of the manifold. Then, the motion is updated by projecting back the variables on the manifold itself. We also derive another version of the algorithm that employs a homeomorphic parameterization to the Special Euclidean group. We test the algorithm on several simulated and real image data sets. Compared to the standard Newton minimization scheme, we are now able to obtain the full numerical formula of the Hessian with 60% decrease in the computational complexity. Compared to Levenberg-Marquardt, the results obtained are more accurate while having a rather similar complexity.**

***Index Terms*—Newton method, Riemannian manifold, differential geometry, pose estimation**

I. Introduction

THE rigid motion or pose of a camera is composed of two parts: a rotation matrix $\mathbf{R} \in \mathbb{R}^{3\times3}$ that describes the orientation of the camera in the real world and a translation vector $\mathbf{t} \in \mathbb{R}^3$ which reflects the distance between the origin of the camera coordinate system and that of the world coordinate system. Using the camera pose, it is possible to establish the relationship between a point $\mathbf{x} \in \mathbb{R}^2$ in an image and the corresponding point $\mathbf{X} \in \mathbb{R}^3$ in the real world. This relationship is written in the form

$$\mathbf{p} = \mathbf{K}\left[\mathbf{I}_3 \mid \mathbf{0}\right] \cdot \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \cdot \mathbf{P} = \mathbf{K}\left[\mathbf{I}_3 \mid \mathbf{0}\right] \cdot \mathbf{M} \cdot \mathbf{P}, \quad (1)$$

M. Sarkis has conducted this work while being at the Institute for Data Processing, Technische Universität München, Arcisstr. 21, 80290 Munich, Germany. Now, he is affiliated with the Sensing System Laboratory, SONY Deutschland GmbH, Hedelfingerstrasse 61, 70327 Stuttgart, Email:michel.sarkis@sony.de, Phone:+497115858151, Fax:+49711585899151.

K. Diepold is with the Institute for Data Processing, Technische Universität München, Arcisstr. 21, 80290 Munich, Germany, Email: kldi@tum.de, Phone:+49898928923602, Fax:+498928923600.

This work was supported in part by the German Research Foundation (DFG) within the Collaborative Research Center SFB 453 on "High-Fidelity Telepresence and Teleaction".

We would like to thank Chu-Song Chen and Wen-Yay Chang from SINICA for providing us the code of extrinsic calibration.

where $\mathbf{I}_3 \in \mathbb{R}^{3\times3}$ is the identity matrix, $\mathbf{0} \in \mathbb{R}^3$ is the zero vector $\mathbf{K} \in \mathbb{R}^{3\times3}$ is the matrix of intrinsic parameters and $\mathbf{M} \in \mathbb{R}^{4\times4}$ represents the pose of the camera, $\mathbf{p}$ and $\mathbf{P}$ are the points $\mathbf{x}$ and $\mathbf{X}$ respectively in homogeneous coordinates. This relationship describes the rigid transformation between the coordinate system of the world and that of the camera. Determining the pose can be also stated as estimating the rigid transformation $\mathbf{M}$ that maps the world coordinate system to the coordinates of the camera.

In case of a calibrated single moving camera, pose estimation can be performed using structure from motion algorithms followed by bundle adjustment (BA) to refine the results. In this direction, one can find several algorithms that deal with this problem as in [1]–[5] for example. The main idea of such schemes is to find some robust initial estimates of the 3D structure and the motion to guarantee the overall convergence of the problem.

When a stereo rig is used, a partial 3D model can be constructed at each new pose using a stereo-based reconstruction method. Assuming the rig to be fully calibrated and the 3D object to be static, the Euclidean position between the stereo camera and the corresponding partial 3D model will be known. To compute the relative motion between the consecutive views, it is sufficient to estimate the 3D rigid transformation between the matching points of the partial 3D models. This problem is referred to in photogrammetry as the *absolute orientation* and is better understood by looking at the problem setup in Figure 1. Let us denote by $\mathbf{Q} = \{\mathbf{P}_1, \cdots, \mathbf{P}_N\}$ the set of N 3D points in space. Let $\mathbf{Q}^i$ and $\mathbf{Q}^{i-1}$ be the homogeneous coordinates of the points in $\mathbf{Q}$ with respect to the current and previous positions of the stereo camera respectively. The relative motion of the rig can be computed by solving the following least-squares problem

$$\mathbf{Q}^{i-1} = \mathbf{M} \cdot \mathbf{Q}^i, \quad (2)$$

which is nothing but determining the rigid transformation between two sets of corresponding 3D points. Absolute orientation is usually solved by using unit quaternions to represent the rotation matrix or by applying the singular value decomposition [6]–[9]. To determine the set of corresponding points required for working out (2), it is necessary to apply image feature detection and tracking techniques like the KLT tracker which was developed by Tomasi and Kanade in [10] and later enhanced by Bouguet in [11]. These features, however, are susceptible to errors that make the linear computed solution erroneous.

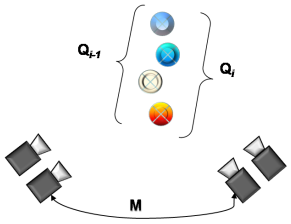


Fig. 1. Problem Setup: using a stereo technique, the 3D homogeneous coordinates of a static set of points $\mathbf{Q}$ are constructed with respect to each position of the camera, i.e. $\mathbf{Q}_i$ and $\mathbf{Q}_{i-1}$. The rigid motion of the camera is the transformation between the coordinates of the points $\mathbf{Q}_i$ and $\mathbf{Q}_{i-1}$ in the two positions.

To rectify the result, the obtained values can be refined with the Levenberg-Marquardt (LM) scheme by minimizing the sum of reprojection errors as usually performed in BA. Like any Newton iterative method, LM requires at each iteration the computation of the gradient and the Hessian. LM uses in the minimization an approximate form of the Hessian to reduce the computational requirements [12], [13]. On one hand, using an approximate form of the Hessian increases the chances of divergence especially if the initial estimate of the motion is not very close to global optimum, see [14]. On the other hand, employing the full numerical form of the Hessian increases the computational complexity of a Newton scheme tremendously, see Table I, which is why this is usually avoided in practice.

Motivated by this dilemma, we propose in this work a projective Newton minimization scheme on the manifold to refine the pose estimate of a moving camera. The key issue is to benefit from the properties of the Riemannian manifold. We derive two versions of the algorithm. The first one operates on the special Euclidean group SE_3 which describes the rigid motion in 3D space. The second one works on $\mathbb{R}^3 \times SO_3$ due to its homeomorphism with SE_3 [15]. At each iteration, the gradient and the Hessian are first computed on the tangent space of the manifold. Then, the motion is updated by projecting back the variables on the manifold itself. This leads to a compact numerical forms of both of the gradient and the Hessian. Compared to a standard Newton scheme, the proposed technique requires 60% less computational complexity to evaluate the full numerical form of the Hessian while preserving the accuracy of the results. Compared to LM, the proposed method leads to a noticeable increase in the accuracy while having a slight increase in the complexity.

Deriving optimization algorithms on the Lie manifold has been widely used in the literature of computer vision and robotics. Example algorithms related to pose estimation can be found in [16]–[18]. In this direction, the contribution of this paper is four folds. Firstly, we derive a Newton algorithm on the manifold to refine the motion estimate on SE_3 and $\mathbb{R}^3 \times SO_3$. The scheme can be applied to the relative point pose problem as well as the perspective n-point problem. Secondly, we provide detailed numerical derivations of the proposed technique. Thirdly, we perform complexity analysis comparisons between the derived scheme and the common algebraic ones used to solve this problem. Fourthly, we show using experimentation that our scheme is less likely to be stuck in local minima, less affected by how far the initial estimate of the pose is from the global optimum, and more robust against the noise.

The rest of this work is divided as follows. Section II presents an overview of the problem setup along with a recall on Newton minimization schemes. Section III derives the proposed projective Newton algorithm in its two versions. Section IV evaluates the performance of the technique and compares it to other schemes. We finally conclude this paper with some discussions in Section V.

II. Related Work

Given are some N point correspondences which are pre-computed using a feature detector and tracked over a sequence, the estimate of the 3D structure computed from a stereo reconstruction technique and the initial pose of the camera obtained from an absolute orientation scheme. Using these values, a point from the 3D structure is projected on the image of the camera at a different position from the pre-computed feature point. The geometrical distance $\mathrm{d}(\cdot)$ between the two points is called the reprojection error. The objective of the minimization is to correct the pose of the camera so that the reprojection error is minimized over all the points. The cost function f to be optimized is written as

$$f = \sum_{j=1}^{N} \mathrm{d}\left(\mathbf{p}_j, \hat{\mathbf{p}}_j\right)^2, \tag{3}$$

where $\mathbf{p}_j$ are the homogeneous coordinates of the j^{th} measured point in the image (camera) and $\hat{\mathbf{p}}_j$ are the homogeneous coordinates of the reprojection of the j^{th} point in the 3D structure $\mathbf{P}_j$ according to (1). The measure $\mathrm{d}(\cdot)$ can be any scale invariant distance as for example the Mahalanobis, the L2 norm or the cosine distances. In the rest of this paper, we will use the cosine distance to measure the reprojection errors. It is given by

$$\mathrm{d}\left(\mathbf{p}_j, \hat{\mathbf{p}}_j\right) = 1 - \frac{\mathbf{p}_j^T \cdot \hat{\mathbf{p}}_j}{\|\mathbf{p}_j\| \cdot \|\hat{\mathbf{p}}_j\|}, \tag{4}$$

where $\|\cdot\|$ denotes the L2 norm. It is also possible to simply use the commonly applied L2 norm difference if preferred. We apply the cosine distance instead since it leads to a simpler computation of the Euclidean derivatives and performs as well as the L2 norm, see [19].

Keeping these points in mind, the objective now is to minimize the cost function f over $\mathbf{R}$ and $\mathbf{t}$. This task is equivalent to refining the relative pose of the camera or the 2D-3D correspondences. In this direction of research, it is possible to find several algorithms that fulfill this task like the ones presented in [20]–[24] for example. However, the goal of this paper is to enhance the accuracy of a Newton minimization scheme without significantly increasing the accompanying computational complexity. As a consequence, we will state in the sequel a general reminder on Newton minimization techniques and show how the motion is usually parameterized in such scenarios.

A. Motion Parameterization

To parameterize the translation vector of the camera, it is sufficient to take its 3 entries as the variables. To parameterize the rotation, it is necessary to find a minimal parameterization which reflects its 3 DOF. A rotational transformation in $\mathbb{R}^3$ is represented by the elements of the special orthogonal group SO_3. The associated Lie algebra $\mathbf{so}_3$ to SO_3 is the set of 3×3 skew-symmetric matrices. The Lie algebra can be 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

$$\mathbf{\Omega}(\omega):\mathbb{R}^3\longrightarrow\mathbf{so}_3,\ \begin{bmatrix}\omega_1\\ \omega_2\\ \omega_3\end{bmatrix}_\times=\begin{bmatrix}0 & -\omega_3 & \omega_2\\ \omega_3 & 0 & -\omega_1\\ -\omega_2 & \omega_1 & 0\end{bmatrix}. \tag{5}$$

To project the Lie algebra to the Lie group, one can use the exponential mapping. Therefore, the rotation matrix $\mathbf{R}$ can be expressed as $\mathbf{R}=\operatorname{expm}(\mathbf{\Omega}(\omega))$. It can be also written using the Rodrigues formula as

$$\begin{aligned}\mathbf{R} &= \operatorname{expm}(\mathbf{\Omega}(\omega)) \\ &= \mathbf{I}_3+\mathbf{\Omega}(\omega)\cdot\frac{\sin(\|\omega\|)}{\|\omega\|}+\mathbf{\Omega}^2(\omega)\cdot\frac{1-\cos(\|\omega\|)}{\|\omega\|^2}.\end{aligned} \tag{6}$$

It is possible to approximate the rotation matrix with $\mathbf{I}_3+\mathbf{\Omega}(\omega)$ if the magnitude ω is small. This leads actually to the local perturbation around $\mathbf{R}$ which is widely used in BA [25]. Using the local perturbations around the rotation is not always accurate especially if the initial estimate of the motion is noisy. Therefore, we will be using in our implementations of the LM algorithm and the vector space Newton the whole term to have an accurate parameterization of the 3D rotation. Note that throughout this paper, we will denote by $\mathbf{a}=\begin{bmatrix}\omega^{\mathrm{T}} & \mathbf{t}^{\mathrm{T}}\end{bmatrix}^{\mathrm{T}}\in\mathbb{R}^6$ as the vector that holds the motion parameters of the camera.

B. Newton Algorithm in the Vector Space

A Newton algorithm finds the minimum of a given cost function iteratively. The update step used to minimize the cost function f in (3) is given by

$$\mathbf{a}^{(k+1)}=\mathbf{a}^{(k)}-\mathbf{H}_f^{-1}\cdot\nabla f. \tag{7}$$

The variable k is the iteration number, $\nabla f\in\mathbb{R}^6$ is the gradient of f and $\mathbf{H}_f\in\mathbb{R}^{6\times6}$ is the corresponding Hessian matrix which is expressed by

$$\mathbf{H}_f=2\mathbf{J}_{\mathbf{g}}^{\mathrm{T}}\cdot\mathbf{J}_{\mathbf{g}}+2\sum_{j=1}^{N}\left(g_j\cdot\mathbf{H}_{g_j}\right). \tag{8}$$

The matrix $\mathbf{H}_{g_j}=\frac{\partial}{\partial\mathbf{a}^{\mathrm{T}}}\frac{\partial g_j}{\partial\mathbf{a}}\in\mathbb{R}^{6\times6}$ denotes the second order derivatives of each term of $\mathbf{g}$ with respect to the camera parameters where $\mathbf{g}$ is the reformulation of the cost function f, i.e. $f=\mathbf{g}^T\mathbf{g}$. The matrix $\mathbf{J}_{\mathbf{g}}$ is the Jacobian and is computed by taking the first order derivatives of the cost function in the form

$$\mathbf{J}_{\mathbf{g}}=\begin{bmatrix}\frac{\partial g_1}{\partial\mathbf{a}^{\mathrm{T}}}\\ \vdots\\ \frac{\partial g_N}{\partial\mathbf{a}^{\mathrm{T}}}\end{bmatrix}\in\mathbb{R}^{N\times6}. \tag{9}$$

By setting the second term of Equation (8) 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. It is also possible to add a damping factor μ to $\mathbf{H}_f$ to regularize the Hessian if it is close to singularity and to vary the speed of convergence of the algorithm. A good description on how to perform such adjustments to the Hessian are found in [26]. The resulting update step will look like

$$\mathbf{a}^{(k+1)}=\mathbf{a}^{(k)}-\left(\mu\cdot\mathbf{I}_6+\mathbf{H}_f\right)^{-1}\cdot\nabla f, \tag{10}$$

where $\mathbf{I}_6\in\mathbb{R}^{6\times6}$ is the identity matrix. Due to μ, the Newton scheme will have the properties of both the steepest descent and the Gauss-Newton techniques. Similarly to steepest descent, the Newton algorithm 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 the approximate Hessian $\hat{\mathbf{H}}_f$ is employed in Equation (10) instead of $\mathbf{H}_f$, the resulting update step is what is used in LM. LM ignores the right addend in (8) since the complexity induced by the computation of $\mathbf{H}_{g_j}$ is very high. To get a feeling how much the computational effort rises, the number of floating point operations (flops) required per iteration are illustrated in Table I. As can be noticed, the inclusion of these terms makes the complexity required for the motion refinement rise by a factor of 5 when compared to LM. This fact shows the reason that makes LM more favorable in practice.

Failing to take the second order derivatives into account, however, increases the risk of divergence especially if the initial estimate of the motion is noisy or far from the global solution [14]. This can be justified since the minimization without these terms makes LM more susceptible to be stuck in local minima which explains why structure from motion algorithms try always to ensure robust initial estimates. For a thorough analysis on Newton minimization schemes, the interested reader may refer to the following excellent references in [1], [12]–[14], [25], [26].

III. The Proposed Projective Newton-Type Approach

Computation of the second order terms in a Newton algorithm, i.e. right addend in Equation (8), is not favorable due to the complexity arising from the computation of the Hessian. The aim of this section is, therefore, to propose a Newton-type optimization method that enhances the accuracy of the motion estimate without a significant increase in the computational requirement. To do that, we will derive a projective Newton minimization scheme on the *manifold*. 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 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. An instance of such manifolds are the *Riemannian* manifolds. A Riemannian manifold 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 example Riemannian manifold is the special orthogonal group SO_3 that describes the 3×3 rotations matrices, see Equation (5). The tangent space to this group is its associated Lie algebra $\mathbf{so}_3$ which is related to the latter via the exponential map.

To derive a Newton-type method, it is necessary to compute first and second order derivatives. 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 often requires an overhead of differential geometric machinery. Instead, a projected Newton-type method is formulated in this paper 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 2. 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 [15], [27]–[30].

A. Parameterization with SE_3

The 3D rigid motion in is represented by the elements of the special Euclidean group SE_3. This group consists of all the 4×4 matrices $\mathbf{M} \in SE_3$ of the form

$$\mathbf{M} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}, \tag{11}$$

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 $\mathbf{m} \in se_3$

$$\mathbf{m} = \begin{bmatrix} \boldsymbol{\Omega}(\omega) & \mathbf{v} \\ \mathbf{0} & 0 \end{bmatrix}, \tag{12}$$

where $\boldsymbol{\Omega}(\omega) \in so_3$ is a skew symmetric-matrix as shown in Equation (5) 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. $\mathbf{M} = \operatorname{expm}(\mathbf{m})$. To design a Newton-type optimization scheme on SE_3, it is necessary to employ ω and $\mathbf{v}$ to parameterize the rotation and the translation, i.e. $\mathbf{a} = [\omega^T \quad \mathbf{v}^T]^T$, respectively.

Definition 1: A smooth and local parameterization ϕ of SE_3 at the camera motion $\mathbf{M}$ can be defined using the corresponding Lie algebra se_3 to be:

$$\begin{aligned} \phi_{(\mathbf{M})} &: \mathbb{R}^6 \rightarrow SE_3, \\ \phi_{(\mathbf{M})}(\mathbf{a}) &= \operatorname{expm}\left(\begin{bmatrix} \boldsymbol{\Omega}(\mathbf{S}_1\mathbf{a}) & \mathbf{S}_2\mathbf{a} \\ \mathbf{0} & 0 \end{bmatrix}\right) \cdot \mathbf{M} \\ &= \boldsymbol{\Lambda}_\phi \cdot \mathbf{M}. \end{aligned} \tag{13}$$

The matrices $\mathbf{S}_1 = [\mathbf{I}\ \mathbf{0}] \in \mathbb{R}^{3\times 6}$ and $\mathbf{S}_2 = [\mathbf{0}\ \mathbf{I}] \in \mathbb{R}^{3\times 6}$ select the appropriate variables. In other words, $\mathbf{S}_1\mathbf{a} = \omega$ whereas $\mathbf{S}_2\mathbf{a} = \mathbf{v}$.

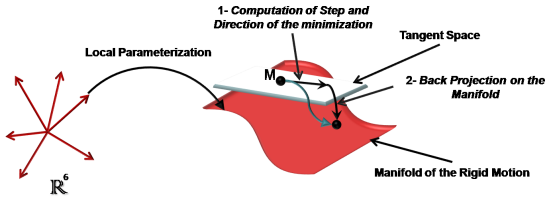


Fig. 2. Projective Newton-type algorithm on the manifold. A local parameterization maps the variables from $\mathbb{R}^6$ to the manifold. The minimization is then performed in two steps: The optimization direction and step size are computed on the tangent space to the manifold. The calculated values are then projected back on the manifold to update the motion.

B. Parameterization Using the Homeomorphism to SE_3

It is possible to think of the rigid motion $\mathbf{M}$ as two consecutive independent motions, i.e. a pure translation $\mathbf{t}$ followed by a pure rotation $\mathbf{R}$. This assumption is performed a lot in the literature because the special Euclidean group SE_3 is a homeomorphic topological space to $\mathbb{R}^3 \times SO_3$ [15]. Therefore, we can use this homeomorphism to derive another parameterization of the motion for our algorithm. To do that, we have to recall that the relationship between $\mathbf{m}$ and $\mathbf{M}$ is defined with the exponential map since SE_3 is a Riemannian manifold. Using the Taylor series of the exponential, one can easily show that this relation can be written as

$$\begin{aligned} \mathbf{R} &= \operatorname{expm}(\boldsymbol{\Omega}(\omega)) \\ \mathbf{t} &= \left(\mathbf{I} + \tfrac{1-\cos(\theta)}{\theta^2}\cdot\boldsymbol{\Omega}(\omega) + \tfrac{\theta-\sin(\theta)}{\theta^3}\cdot\boldsymbol{\Omega}(\omega)^2\right)\cdot\mathbf{v} = \boldsymbol{\Theta}\cdot\mathbf{v}, \end{aligned} \tag{14}$$

where $\theta = \sqrt{\frac{1}{2}\operatorname{trace}(\boldsymbol{\Omega}^{\mathrm{T}}(\omega)\cdot\boldsymbol{\Omega}(\omega))}$. By assuming the two consecutive motions to be independent, the second part of Equation (14) simply reduces to $\mathbf{t} \approx \mathbf{v}$ while the first one remains the same. As a result, using the Homeomorphism to SE_3, the 3D rigid motion can be additionally parameterized with $\mathbf{a} = [\omega^T \quad \mathbf{t}^T]^T \in \mathbb{R}^6$.

Definition 2: This homeomorphism between SE_3 and $\mathbb{R}^3 \times SO_3$ leads to an alternative smooth and local parameterization ϕ for the rigid motion. It is defined as:

$$\begin{aligned} \phi_{(\mathbf{M})} &: \mathbb{R}^6 \rightarrow \mathbb{R}^3 \times SO_3, \\ \phi_{(\mathbf{M})}(\mathbf{a}) &= \begin{bmatrix} \operatorname{expm}(\boldsymbol{\Omega}(\mathbf{S}_1\mathbf{a})) & \mathbf{S}_2\mathbf{a} \\ \mathbf{0} & 1 \end{bmatrix} \cdot \mathbf{M} \\ &= \boldsymbol{\Lambda}_\phi \cdot \mathbf{M}. \end{aligned} \tag{15}$$

C. The Cost Function on the Manifold

Having introduced the two possible parameterizations of the rigid motion, we have to reformulate now the cost function f so that it can accommodate anyone of them. The proposed projective Newton method should compute the update step of the minimization, i.e. gradient and Hessian, at each iteration for the composition $f \circ \phi_{(\mathbf{M})}$ of the cost function f with the local parameterization $\phi_{(\mathbf{M})}$. Then, it should map the update step back to the manifold using the parameterization again, see Figure 2. We denote by $f \circ \phi_{(\mathbf{M})}$ as the mapping

$$f \circ \phi_{(\mathbf{M})} : \mathbb{R}^6 \rightarrow \mathbb{R}, \tag{16}$$

which corresponds to the cost function to be minimized on the manifold.

Definition 3: The cost function that evaluates the composition $f \circ \phi_{(\mathbf{M})}$ of the cost function f with the local parameterization $\phi_{(\mathbf{M})}$ on the manifold is defined by:

$$\begin{aligned} f \circ \phi_{(\mathbf{M})}(\mathbf{a}) &:= \sum_{j=1}^{N} \mathrm{d}\left(\mathbf{p}_j, \hat{\mathbf{p}}_j\right)^2 \\ &= \sum_{j=1}^{N} \mathrm{d}\left(\mathbf{p}_j, \pi \cdot \mathbf{\Lambda}_\phi \cdot \mathbf{M} \cdot \mathbf{P}_j\right)^2. \end{aligned} \tag{17}$$

The distance $\mathrm{d}(\cdot)$ measures the reprojection error while $\pi \in \mathbb{R}^{3\times 4}$ scales the multiplication of the right term of the distance measure from $\mathbb{R}^4$ to $\mathbb{R}^3$ to preserve the dimensions. The matrix $\mathbf{\Lambda}_\phi$ can be any of the two local parameterizations illustrated in Definitions 1 and 2.

D. Computing the Derivatives

As in any Newton-type minimization scheme, the first and second order derivatives of the cost function need to be computed to obtain the direction of the minimization. The main reason behind formulating a projective Newton scheme on the manifold is that the computation of the derivatives is much simpler. To get a glimpse on how to calculate such derivatives, let us compute the first order derivative of the term $(\pi \cdot \mathbf{\Lambda}_\phi \cdot \mathbf{M} \cdot \mathbf{P}_j)$ that represents the coordinate of the reprojected point $\hat{\mathbf{p}}_j$ in the cost function shown above. In a different way from a standard Newton scheme, a projective Newton algorithm computes the derivative with respect to *the manifold step*, which we will denote here by ε, and not the motion parameters, see Section II-B. Moreover, the derivative that we need should be computed at zero or $\varepsilon = 0$ since it is necessary that we find the direction of the minimization at the tangent plane to the manifold [28]. By formulating these points mathematically, the first order derivative is obtained as

$$\begin{aligned} &\frac{\mathrm{d}}{\mathrm{d}\varepsilon}(\hat{\mathbf{p}}_j)(\varepsilon\mathbf{a})\Big|_{\varepsilon=0} = \\ &\pi \cdot \begin{bmatrix} \mathbf{\Omega}(\mathbf{S}_1\mathbf{a}) & \mathbf{S}_2\mathbf{a} \\ \mathbf{0} & 0 \end{bmatrix} \cdot \exp\!\left(\begin{bmatrix} \mathbf{\Omega}(\mathbf{S}_1\varepsilon\mathbf{a}) & \mathbf{S}_2\varepsilon\mathbf{a} \\ \mathbf{0} & 0 \end{bmatrix}\right) \cdot \mathbf{M} \cdot \mathbf{P}_j \Big|_{\varepsilon=0} \\ = \;& \pi \cdot \begin{bmatrix} \mathbf{\Omega}(\mathbf{S}_1\mathbf{a}) & \mathbf{S}_2\mathbf{a} \\ \mathbf{0} & 0 \end{bmatrix} \cdot \mathbf{M} \cdot \mathbf{P}_j, \end{aligned} \tag{18}$$

using the parameterization $\phi_{(\mathbf{M})}$ shown in Definition 1. The corresponding second order derivative of the reprojected point $\hat{\mathbf{p}}_j$ with respect to the manifold step ε is obtained following the same principle. It is given by

$$\frac{\mathrm{d}^2}{\mathrm{d}\varepsilon^2}(\hat{\mathbf{p}}_j)(\varepsilon\mathbf{a})\Big|_{\varepsilon=0} = \pi \cdot \begin{bmatrix} \mathbf{\Omega}(\mathbf{S}_1\mathbf{a}) & \mathbf{S}_2\mathbf{a} \\ \mathbf{0} & 0 \end{bmatrix}^2 \cdot \mathbf{M} \cdot \mathbf{P}_j \tag{19}$$

Since the derivatives are computed with respect to ε at zero and not the motion parameters themselves, the obtained result is more compact as can be noticed from the above equations. This will reduce the computational complexity required to compute the Hessian which is the main motivation behind this work. In addition, there is no need anymore to approximate the parameterization with local perturbations (first terms of the Taylor series expansion of the exponential) as usually done in such algorithms, see [1], [25]. Instead, we can use the full form of the parameterization which will give more robustness against the noise. These two advantages motivate the application of a projective Newton scheme on the manifold for camera pose refinement.

Following this concept, we have to compute the gradient and Hessian of our cost function $f \circ \phi_{(\mathbf{M})}$ at zero. The gradient $\nabla(f \circ \phi_{(\mathbf{M})})(0) \in \mathbb{R}^6$ can be calculated from

$$\frac{\mathrm{d}}{\mathrm{d}\varepsilon}(f \circ \phi_{(\mathbf{M})})(\varepsilon\mathbf{a})\Big|_{\varepsilon=0} = \nabla(f \circ \phi_{(\mathbf{M})})(0)^{\mathrm{T}} \cdot \mathbf{a}. \tag{20}$$

In other words, we have to compute the first order derivative with respect to ε at zero and then rearrange the obtained term into a product of two vectors. The gradient of the cost function at zero is nothing but the transposed of the multiplicand of $\mathbf{a}$.

In a similar manner, the Hessian matrix $\mathbf{H}_{f\circ\phi_{(\mathbf{M})}}(0) \in \mathbb{R}^{6\times 6}$ of $f \circ \phi_{(\mathbf{M})}$ evaluated at zero can be calculated from the quadratic form

$$\frac{\mathrm{d}^2}{\mathrm{d}\varepsilon^2}(f \circ \phi_{(\mathbf{M})})(\varepsilon\mathbf{a})\Big|_{\varepsilon=0} = \mathbf{a}^{\mathrm{T}} \cdot \mathbf{H}_{f\circ\phi_{(\mathbf{M})}}(0) \cdot \mathbf{a} \tag{21}$$

via polarization. The numerical forms of the gradient and Hessian of the cost function are provided in Appendix A for the two parameterizations in Definitions 1 and 2.

E. The Projective Newton Algorithm

Given some initial estimates of the motion $\mathbf{M}$ and the 3D structure, the proposed scheme refines the motion with the following iterative update scheme:

Step 1 (Newton Direction): At each iteration k, compute the vector $\mathbf{a}^{(k)}$ by solving

$$-\left(\mu \cdot \mathbf{I}_6 + \mathbf{H}_{f\circ\phi_{(\mathbf{M})}}(0)\right) \cdot \mathbf{a}^{(k)} = \nabla(f \circ \phi_{(\mathbf{M})})(0). \tag{22}$$

Step 2 (Back Projection on the Manifold): Update the motion parameters

$$\mathbf{M}^{(k)} \leftarrow \mathbf{\Lambda}_\phi^{(k)} \cdot \mathbf{M}^{(k-1)} \tag{23}$$

Step 3: Update the reprojections $\hat{\mathbf{p}}_j$ and compute the mean reprojection error.

Step 4: Repeat until convergence.

F. Analysis of the Derived Scheme

The projective Newton-based minimization algorithm is initialized by some initial values of the motion. Then, it proceeds with the iterative scheme illustrated in the previous section. Each iteration, the algorithm performs an optimization step in Equation (22) 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 (23), see Figure 2. 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. To overcome this problem, it is necessary to determine the optimal μ. For that, there are many derived schemes in the literature. We refer the reader to Chapter 3 in [26] for some examples. Our method can be integrated with any of such schemes. To determine μ in this paper, we first apply the Cholesky factorization with added multiple of the identity to modify the Hessian since it is simple and produces often good results [26]. This leads to the modified Hessian $\hat{\mathbf{H}}_{f\circ\phi_{(\mathbf{M})}}(0)$. Then, we solve for $\mathbf{a}^{(k)}$ as follows

$$\hat{\mathbf{H}}_{f\circ\phi_{(\mathbf{M})}}(0)\cdot\mathbf{a}^{(k)} = -\nabla(f\circ\phi_{(\mathbf{M})})(0) \tag{24}$$

After that, we compute a parameter γ to modify the Hessian. The parameter γ is computed in such a way to satisfy the Wolfe conditions. This is done in our work by performing a backtracing line search using $\mathbf{a}^{(k)}$, see [26]. Then, the value of $\mathbf{a}^{(k)}$ is modified as such

$$\mathbf{a}^{(k)} \leftarrow \gamma\cdot\mathbf{a}^{(k)}. \tag{25}$$

The latter is used in Equation (23) to update the motion parameters. The step we just made is necessary 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.

What remains to be done is to prove that the proposed algorithm will converge to a minimum for a set of observed 3D points and an initial estimate of the motion $\mathbf{M}$. We will demonstrate this point using the *global convergence theorem* derived in Chapter 6 of [31].

Theorem 1: Let $\boldsymbol{\Upsilon}$ denotes a mapping from $SE_3 \to SE_3$ that calculates from the previous motion estimate $\mathbf{M}^{(k-1)}$, the new estimate $\mathbf{M}^{(k)}$ at the k^{th} iteration. The projective Newton algorithm is said to be convergent according to the global convergence theorem if it satisfies the following three conditions:

1) The mapping $\boldsymbol{\Upsilon}$ is closed.
2) All the computed $\mathbf{M}^{(k)}$ by the mapping are contained in a compact set.
3) The mapping $\boldsymbol{\Upsilon}$ is a strictly decreasing function except at the solution.

Proof:

1) To prove the first condition of the theorem, we need to recall that each iteration of the proposed scheme is composed as said before of two mappings. The first one is from $SE_3 \to se_3$ and which is given by the solution of the Newton step in Equation (22). The second is the mapping that projects back on the manifold, i.e. $se_3 \to SE_3$, and which is represented by Equation (23). These two mappings are continuous and smooth according to the properties of a Riemannian manifold; therefore, both mappings are closed. Since $\boldsymbol{\Upsilon}$ is composed of two closed mappings then $\boldsymbol{\Upsilon}$ is also closed.
2) The mapping specified by $\boldsymbol{\Upsilon}$ generates matrices of rigid motion from the set SE_3 which is a compact set by the properties of a Riemannian manifold.
3) The regularization we performed on the Hessian often ensures that it remains positive semi-definite. In addition, the computation of μ in the way previously defined often ensures that the direction of the minimization in this equation is negative or decreasing. Consequently, the step is zero only when the gradient is zero, the point at which the algorithm meets a fixed point (converges). Hence, the third condition of the theorem is also satisfied.

The proposed projective Newton scheme is globally convergent since all the conditions of the global convergence theorem are fulfilled. ■

Corollary 1: The algorithm was proven to be convergent for the parameterization $\phi_{(\mathbf{M})}$ in Definition 1. One can infer from the proof that since $\mathbb{R}^3 \times SO_3$ is homeomorphic to SE_3, the projective Newton scheme is also convergent when the motion is parameterized according to Definition 2.

G. Complexity Analysis of the Algorithm

Now that the algorithm was proven to be globally convergent, the next step is to evaluate its complexity and compare it to the state of the art methods depicted in Section II. The main difference in the proposed scheme is that the optimization step, i.e. evaluation of gradient and Hessian, is conducted on the tangent plane of the Riemannian manifold. By looking at the concept with which these terms are computed in Section III-D, we can anticipate that their form will be compact. This is actually confirmed by depicting their numerical format in Appendix A. There are no lengthy computations in the evaluation of the derivatives of the cost function as in Equations (9) and (8) in the vector space. The only Euclidean derivatives that need to be evaluated are the ones belonging to the distance function $\mathrm{d}(\cdot)$ with respect to $\hat{\mathbf{p}}$, i.e. $\partial \mathrm{d}(\mathbf{p}_j, \hat{\mathbf{p}}_j)/\partial\hat{\mathbf{p}}_j$ and $\partial^2 \mathrm{d}(\mathbf{p}_j, \hat{\mathbf{p}}_j)/\partial\hat{\mathbf{p}}_j^2$. These are also needed in a standard Newton minimization scheme and do not present a bottleneck in the computations.

In order to emphasize on the computational complexity, we present in Table I a comparison of the costs required for the calculation of the gradient and the Hessian for each of the minimization algorithms in a single iteration. We denote in the presented results the standard Newton algorithm described in Section II-B by Newton Vectorspace, the proposed scheme with the parameterization in Definition 1 by Newton SE_3 and the proposed scheme with the parameterization in Definition 2 by Newton $\mathbb{R}^3 \times SO_3$. The factors multiplied by N are the operations that have to be evaluated for each point in the camera while the others are only required once, e.g. Rodrigues formula in Equation (7) 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, i.e. 5 times the computational resources. This shows why

the evaluation of the full Hessian shown in Equation (8) is usually avoided in such problems. However, the amount of computations required by the two versions of the projective Newton algorithm on the manifold is 60% less on average than the latter. Compared to LM, the complexity induced by the proposed method is only twice higher. In addition, 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 can be seen from the fact that taking the full numerical form of the Hessian, i.e. the second order derivatives in Equation (8), makes the optimization more robust against being stuck in local minima [14]. This point will be also visualized in our results in the next section.

IV. Experimental Results

We will describe in this section a comparison regarding the proposed projective Newton technique using both parameterizations, the standard Newton algorithm described in Section II-B and LM. For LM, we use the Matlab implementation which can be called with the command *lsqnonlin*. 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 convergence and accuracy of the estimated motion. The cameras are assumed to be intrinsically calibrated and the 3D structure is computed along with the corresponding image projections. All the used methods are implemented in Matlab. 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 I. Consequently, the variation of the outcome of the algorithms depends only on this difference[1].

Figure 3 shows the output of the algorithms when applied to the simulated data set. We created 100 3D points lying on geometrical shapes consisting of cubes, pyramids and spheres. We also created four cameras at different positions. We projected the 3D points on 256×256 images using these cameras in the noise free case. We fed the coordinates of the feature points and the ground truth 3D structure to each of the Newton algorithms. As for the initial estimates, we made two different experiments. In the first one, we randomly chose the estimates. In the second one, we distorted the ground truth motion values with noise. Doing these two tests will let us know how far is each of the algorithms affected by the value of the initial estimate. In Figure 3a and 3b, we see the mean cosine distance versus the number of iterations to converge. The result illustrated in the plots show the average over 100 trials. One can see that LM failed to properly converge in the first test while it was able to converge like the others in the second one. In Figures 3c and d, we measured the error in the rotation of one camera position from the first test by transforming the rotation matrix to the Euler angles and then calculating the absolute difference between the estimated and the ground truth values. For the translation in Figure 3e, the error is computed by evaluating the distance between the estimated vectors and the corresponding true values. All of these results demonstrate that taking the total Hessian and not just the approximation make the algorithm less affected by the value of the initial motion estimate. To stress on this point, we show in Figure 3f the percentage error between the approximate Hessian used by LM and the accurate one of the test in Figure 3a in both linear and logarithmic scales. One can notice that the difference between the two is significant and this is most likely why there is a difference between the obtained results.

The third test we will perform uses the Hall stereo sequence which we acquired. Example images from the sequence are illustrated in Figure 4. 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 [32]. 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 [33] which is proposed in [34]. We used the epipolar constraint after that to eliminate the outliers and increase the number of feature points by guiding the correspondences. We then applied the absolute orientation technique of [6] to compute a linear estimate of the motion by calculating a rigid transformation between each sequential pair of the partial 3D structures. This results in the eight initial estimates of the camera motion. The structures were then inputted along with the initial estimates of the motion to refine its alignment with the minimization schemes. In these comparisons, we also use the implementation of the iterative closest point (ICP) algorithm of [24] since the problem in hand now corresponds to perspective n-point problem which was also dealt with there. In Figure 5a, we show the average mean cosine distance of the whole sequence plotted versus the number of iterations required to converge. In Figure 5b, we show the corresponding mean reprojection error in pixels. What we can see is that LM is not able to refine the average reprojection error obtained from the linear method while the other Newton schemes and ICP are able to do so. This is also confirmed by Figure 6 where we show 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. As a reference, we also provide in this image the output of the linear algorithm of [6]. We can also realize here that the average error in the motion variables is much lower than the output of LM and is better than that of ICP. This result motivates the application of the manifold based Newton algorithm for motion refinement since it is more robust against the noise. To see why LM was not able to provide a significant enhancement, we show in Figure 7 the norm of the gradient and the damping factor of LM versus

[1]The proposed algorithms can be downloaded from http://wwwt3.ldv.ei.tum.de/index.php?id=248

TABLE I

THE TOTAL NUMBER OF FLOPS NEEDED FOR THE COMPUTATION OF THE GRADIENT AND THE HESSIAN PER ITERATION. N IS THE NUMBER OF POINTS, sqrt IS THE SCALAR SQUARE ROOT, cos IS THE SCALAR COSINE AND sin IS THE SCALAR SINE. THE TERMS expm AND expm_4 ARE THE 3×3 AND 4×4 MATRIX EXPONENTIAL RESPECTIVELY.

Algorithm	Number of flops
LM	$N(114 + 2 \cdot \text{sqrt}) + (227 + \text{expm} + \text{sqrt} + \cos + \sin)$
Newton Vectorspace	$N(564 + 2 \cdot \text{sqrt}) + (1024 + \text{expm} + \text{sqrt} + \cos + \sin)$
Newton SE_3	$N(241 + 2 \cdot \text{sqrt}) + (15 + \text{expm}_4)$
Newton $\mathbb{R}^3 \times SO_3$	$N(235 + 2 \cdot \text{sqrt}) + (6 + \text{expm})$

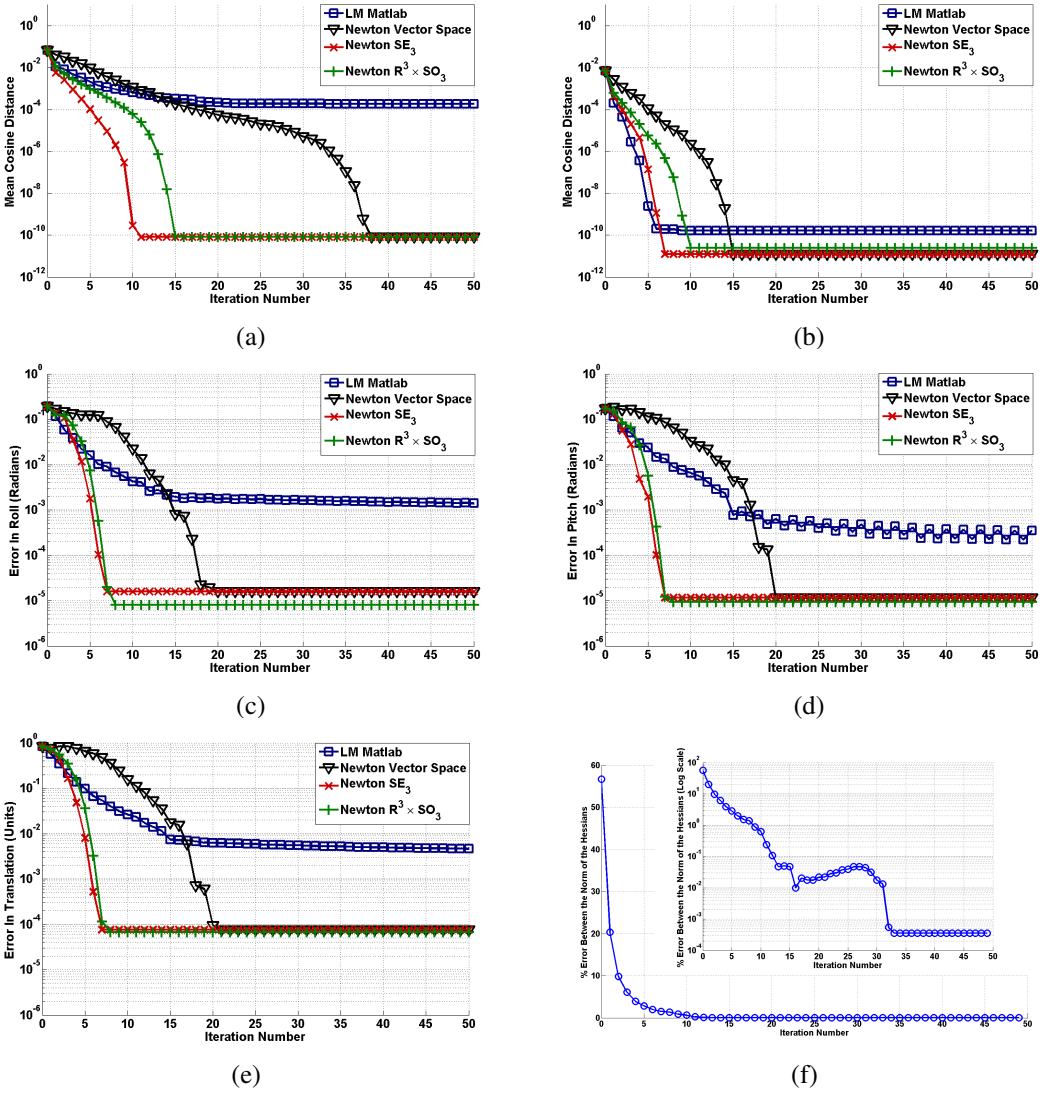


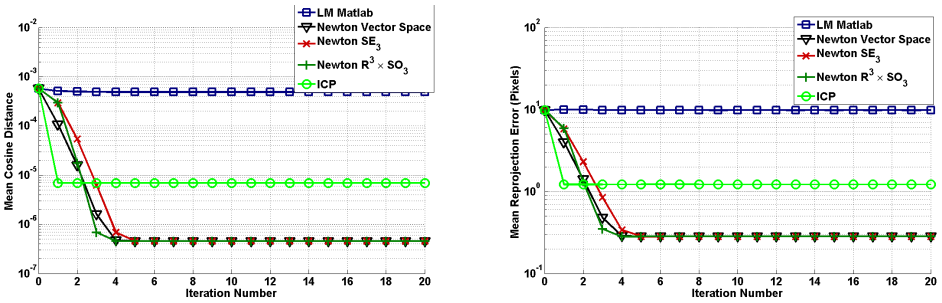
Fig. 3. Result using the simulated data sets. From (a) to (e) in the respective order: Mean cosine distance for the random pose initialization, mean cosine distance for the close pose initialization, error in roll for the random initialization, error in pitch for the random initialization and error in translation for the random initialization versus the number of iterations to converge in the noise free set. In (f), we show the percentage error between the accurate and approximate Hessian (used by LM). We use a linear scale in the outer image and a logarithmic scale in the inner one.

the number of iterations. These show that LM was indeed trapped at a local minimum.

Since the problem in hand is the camera pose estimation, it is also possible to test the algorithms on video sequences originating from a single moving camera. Therefore, 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. To obtain initial estimate of the motion, we first matched and tracked some feature points over each image sequence. We then estimated the intrinsic parameters of the images by applying the technique of [35]. Using the algorithm of [34], we computed the essential matrices between the sequential images, rejected the feature matches that do not satisfy the epipolar constraint and then computed more correspondences by guiding the matches. We perform two tests with these data sets. In one setup of this test we set a lower threshold to reject the matches and in the second one a higher threshold. Therefore, in the

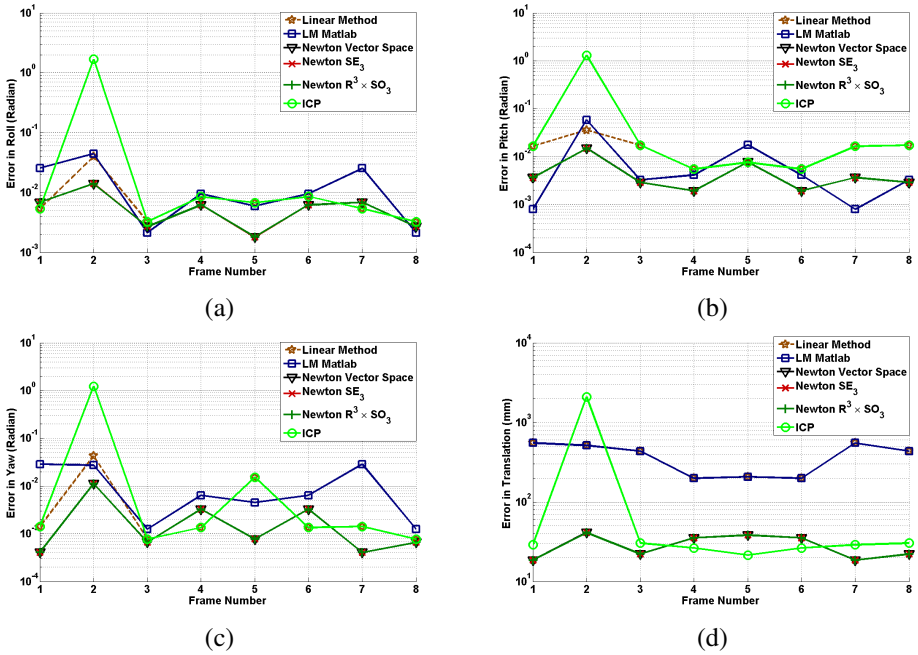
(a) (b) (c)

Fig. 4. Example images from the Hall stereo sequence.



(a) (b)

Fig. 5. (a) The average mean cosine distance versus the number of iterations using the Hall stereo sequence. (b) The average mean reprojection error in pixels versus the number of iterations using the Hall stereo sequence.

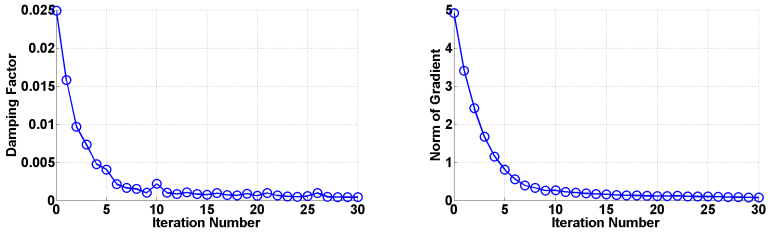


(a) (b) (c) (d)

Fig. 6. Result of the algorithms on the Hall stereo sequence using as initial estimates of the motion the outcome of [6]. From (a) to (d): Error in the estimates of the roll, pitch, yaw and translation respectively.

first case we will have a lower number of false matches and hence lower noise than in the second one. The initial estimates of the motion for each setup are finally 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 of each of the two setups. We then refined the motion estimates using each of the Newton schemes. Note that in the first setup, the initialization of the setup can be considered well conditioned in the sense the data is less noisy and the initial estimates of the motion are more close to the global optimum. In other words, the system is good initialized as usually done in practice. In the second setup, the initialization is noisier and it will be therefore challenging for the Newton algorithms to converge.

The average value of the mean cosine distance and the corresponding mean reprojection error in pixels versus the



(a) (b)

Fig. 7. (a) The damping factor of the LM algorithm versus the number of iterations using the Hall stereo sequence. (b) The corresponding norm of the gradient versus the number of iterations.

iteration number is shown in Figure 8 for the Corridor sequence and in Figure 9 for the Dinosaur sequence. What can we see in both cases is that the LM is pretty sensitive to the initialization of the system. It was not able to converge in the two setups where there was a higher number of mismatches and higher error in the initial value of the motion estimate. Consequently, what we can conclude again is that using the full numerical form of the Hessian makes it more probable for the optimization algorithm to converge to the global optimum.

The refined motion trajectory of the cameras for the two setups with higher noise are shown in Figure 10. In both cases, LM failed in recovering many poses of the camera trajectory while the others have lead to a good output. The correctness of the motion estimates of the Corridor sequence cannot be numerically calculated because this set does not possess a ground truth value of the motion. To the exception of LM, however, the obtained trajectories of the motion look pretty much the same as the ones obtained in [36]. With the Dinosaur sequence, the Newton method with the SE_3 parameterization was able to capture the whole circular motion of the camera. This is also the case with the Newton algorithm in the vector space. The Newton technique with the $\mathbb{R}^3 \times SO_3$ parameterization was able to capture most of the camera poses to the exception of two. The ground truth set of this camera was taken in steps of 10° rotation with a standard deviation of 0.05 [37]. The values obtained with each algorithm are shown in Table II along with the number of outliers, i.e. positions not lying on the circle.

By comparing the two parameterization schemes of the projective Newton algorithm, we notice that their performance was almost the same except with the Dinosaur sequence. There, the technique with the parameterization on $\mathbb{R}^3 \times SO_3$ diverged twice. The reason for that might be because the parameterization $\phi_{(\mathbf{M})}$ in Definition 2 treats each entity, i.e. rotation matrix and translation vector, independently which can also be seen in the form of the obtained gradient and Hessian in Appendix A. In contrast, the parameterization $\phi_{(\mathbf{M})}$ on SE_3 treats all the variables jointly which might has lead to this improvement..

V. Conclusion

We presented in this paper a projective Newton minimization scheme that refines the motion of the camera by optimizing on the Riemannian manifold of the rigid motion. A Riemannian manifold is equipped with a tangent plane that makes the transition between the points smooth. This allows us at each iteration to compute the Newton optimization step on the plane and project back the parameters on the manifold via the exponential map to update the motion. We modified the cost function to accommodate for two parameterization schemes on the manifold. The advantage of the proposed method is that it allows an easier computation of the derivatives which enables us to use the full numerical form of the Hessian in the optimization. Compared to a similar Newton-based technique in the vector space, the proposed algorithm requires 60% less effort to evaluate the Hessian. Compared to LM, the method has a comparable complexity in terms of the Hessian computation although the latter uses an approximate form. Therefore, the application of our scheme leads to more accurate results than LM and is more robust against the noise since the usage of the full numerical form of the Hessian makes it less probable to be stuck in local minima. In addition, it is not required anymore to approximate the motion parameterization with local perturbations as usually done in such schemes. This happens because the derivatives are not computed anymore with respect to the motion variables but to the manifold step, i.e. ε. For these reasons, the proposed projective Newton algorithm on the manifold seems to be a more promising implementation for the non-linear minimization problems.

Appendix

In this section, the term $\mathrm{d}\left(\mathbf{p}_j, \hat{\mathbf{p}}_j\right)$ will be replaced by d for simplicity. In Definitions 1 and 2, we presented two parameterizations of the 3D rigid motion for the proposed projective Newton algorithm. Using the parameterization $\phi_{(\mathbf{M})}$ in Definition 1 along with the concept shown in Equation (20), the explicit formula for the gradient of the local cost function $f \circ \phi_{(\mathbf{M})}$ at zero is given by

$$\begin{aligned} \nabla(f \circ \phi_{(\mathbf{M})})(0) &= 2\mathbf{J}^T \cdot \mathbf{g} \qquad (26) \\ &= 2\sum_{j=1}^{N} \left(\underbrace{\left[-\boldsymbol{\Omega}\left(\pi \mathbf{M} \mathbf{P}_j\right) \quad \mathbf{I}_3\right]^T \cdot \frac{\partial \mathrm{d}}{\partial \hat{\mathbf{p}}_j}}_{\rightarrow \mathbf{J}} \cdot \underbrace{\mathrm{d}}_{\rightarrow \mathbf{g}} \right), \end{aligned}$$

where $\frac{\partial \mathrm{d}}{\partial \hat{\mathbf{p}}_j} \in \mathbb{R}^3$ is the standard Euclidean derivative of the distance function with respect to $\hat{\mathbf{p}}$. The corresponding

TABLE II

THE RESULTS OF THE MINIMIZATION ALGORITHMS ON THE DINOSAUR SEQUENCE ALONG WITH THE GROUND TRUTH VALUES OF THE MOTION. THESE CORRESPOND TO THE SETTING CORRESPONDING TO FIGURE. 9A. 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.

Algorithm	Mean	Std	Number of Outliers
Ground Truth	10	0.05	0
LM	0.82	76.74	25
Newton Vectorspace	10	0.360	0
Newton SE_3	10	0.378	0
Newton $\mathbb{R}^3 \times SO_3$	9.89	8.64	2

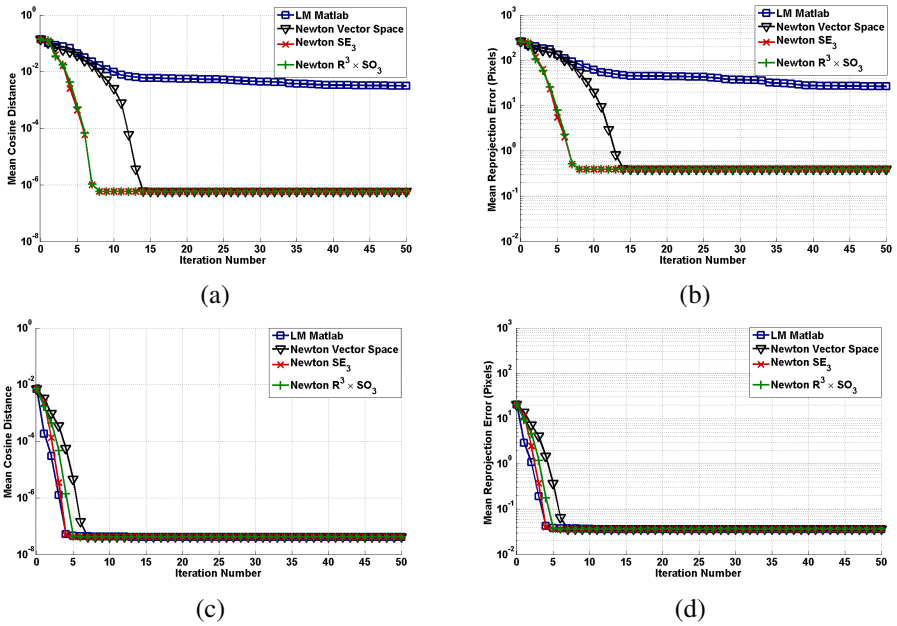


Fig. 8. In the left column, we have the mean cosine distance versus the number of iterations averaged over all the frames of the Corridor sequence while in the right column we have the corresponding mean reprojection error in pixels versus the number of iterations. In (a) & (b), the initialization of the motion is not well conditioned, i.e. the initial estimates are far from the global optimum. In (c) & (d), the initial estimates are well conditioned.

numerical form of the Hessian according to (21) is given by

$$\begin{aligned}\mathbf{H}_{f\circ\phi_{(\mathbf{M})}}(0) &= 2\mathbf{J}^{\mathrm{T}}\cdot\mathbf{J} \\ &+ 2\textstyle\sum_{j=1}^{N}\Big(\mathbf{S}_1^{\mathrm{T}}\cdot\boldsymbol{\Omega}\Big(\frac{\partial \mathrm{d}}{\partial\hat{\mathbf{p}}_j}\Big)\cdot[\boldsymbol{\Omega}(\pi\mathbf{M}\mathbf{P}_j)\ \ \mathbf{I}_3]\cdot\mathrm{d})\Big) \\ &+ 2\textstyle\sum_{j=1}^{N}\Big([-\boldsymbol{\Omega}(\pi\mathbf{M}\mathbf{P}_j)\ \ \mathbf{I}_3]^{T}\cdot\frac{\partial^2\mathrm{d}}{\partial\hat{\mathbf{p}}_j^2}\cdot[-\boldsymbol{\Omega}(\pi\mathbf{M}\mathbf{P}_j)\ \ \mathbf{I}_3]\cdot\mathrm{d}\Big),\end{aligned} \tag{27}$$

where $\frac{\partial \mathrm{d}^2}{\partial\hat{\mathbf{p}}_j^2} \in \mathbb{R}^{3\times 3}$ is the second order derivative of the distance function with respect to $\hat{\mathbf{p}}$.

In a similar fashion, the numerical form of the gradient at zero of the cost function $f\circ\phi_{(\mathbf{M})}$ using the parameterization $\phi_{(\mathbf{M})}$ in Definition 2 is

$$\nabla(f\circ\phi_{(\mathbf{M})})(0) = \tag{28}$$
$$2\textstyle\sum_{j=1}^{N}\Bigg(\underbrace{(\mathbf{S}_2^{\mathrm{T}}+\mathbf{S}_1^{\mathrm{T}}\boldsymbol{\Omega}^{\mathrm{T}}(\mathbf{R}\mathbf{P}_j))\cdot\frac{\partial \mathrm{d}}{\partial\hat{\mathbf{p}}_j}}_{\rightarrow\mathbf{J}}\cdot\underbrace{\mathrm{d}}_{\rightarrow\mathbf{g}}\Bigg),$$

while the corresponding Hessian at zero is given by

$$\begin{aligned}\mathbf{H}_{f\circ\phi_{(\mathbf{M})}}(0) &= 2\mathbf{J}^{\mathrm{T}}\cdot\mathbf{J} \\ &+ 2\textstyle\sum_{j=1}^{N}\Big(\mathbf{S}_1^{\mathrm{T}}\boldsymbol{\Omega}\Big(\frac{\partial \mathrm{d}}{\partial\hat{\mathbf{p}}_j}\Big)\cdot\boldsymbol{\Omega}(\mathbf{R}\mathbf{P}_j)\mathbf{S}_1\cdot\mathrm{d}\Big) \\ &+ 2\textstyle\sum_{j=1}^{N}\Big((\mathbf{S}_2^{\mathrm{T}}-\mathbf{S}_1^{\mathrm{T}}\boldsymbol{\Omega}^{\mathrm{T}}(\mathbf{R}\mathbf{P}_j))\cdot\frac{\partial^2\mathrm{d}}{\partial\hat{\mathbf{p}}_j^2}\cdot(\mathbf{S}_2-\boldsymbol{\Omega}(\mathbf{R}\mathbf{P}_j)\mathbf{S}_1)\cdot\mathrm{d}\Big).\end{aligned} \tag{29}$$

As one can notice, the motion matrix $\mathbf{M}$ is split in this case between the rotation matrix $\mathbf{R}$ and the translation vector $\mathbf{t}$ when calculating the derivatives. This is because the parameterization $\phi_{(\mathbf{M})}$ given in Definition 2 assumes that the motion is the composition of two consecutive motions, i.e. a pure translation and a pure rotation, while the other one in Definition 1 treats the parameters jointly.

REFERENCES

[1] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge University Press, Mar. 2004. 1, 3, 5, 9

[2] M. Pollefeys, L. V. Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch, "Visual modeling with a hand-held camera," *Int. J. Computer Vision*, vol. 59, no. 3, pp. 207–232, Sep. 2004. 1

[3] A. Bartoli and P. Sturm, "Structure-from-motion using lines: Representation, triangulation, and bundle adjustment," *Computer Vision and Image Understanding*, vol. 100, no. 3, pp. 516–441, Dec. 2005. 1

[4] H. Stewénius, C. Engels, and D. Nistér, "Recent developments on direct relative orientation," *ISPRS J. Photogrammetry and Remote Sensing*, vol. 60, no. 4, pp. 284–294, Jun. 2006. 1

[5] K. Ni, D. Steedly, and F. Dellaert, "Out-of-core bundle adjustment for large-scale 3D reconstruction," in *Int. Conf. Computer Vision*, Oct. 2007, pp. 1–8. 1

[6] B. K. P. Horn, "Closed-form solution of absolute orientation using unit quaternions," *J. Optical Society of America A*, vol. 4, no. 4, pp. 629–642, Apr. 1987. 1, 7, 9

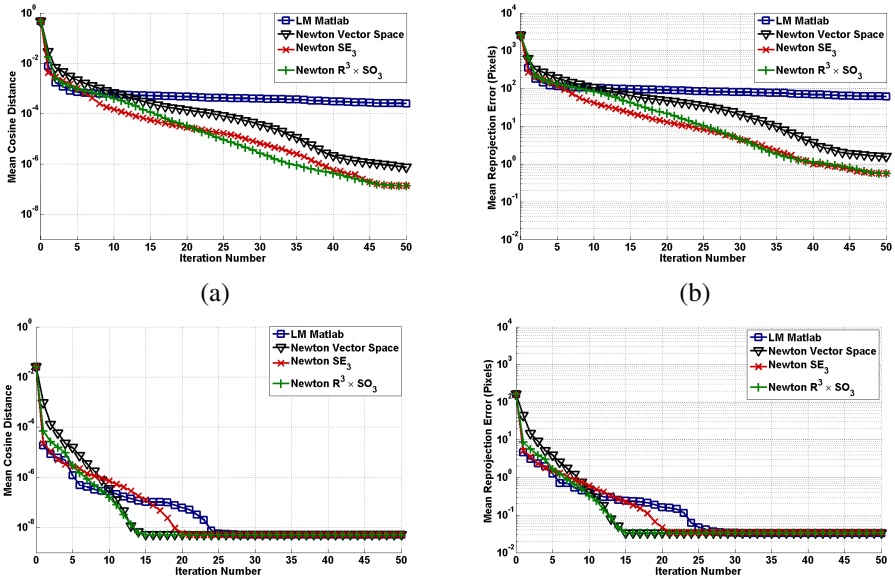


Fig. 9. In the left column, we have the mean cosine distance versus the number of iterations averaged over all the frames of the Dinosaur sequence while in the right column we have the corresponding mean reprojection error in pixels versus the number of iterations. In (a) & (b), the initialization of the motion is not well conditioned, i.e. the initial estimates are far from the global optimum. In (c) & (d), the initial estimates are well conditioned.

[7] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least squares fitting of a two 3D point sets," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 9, no. 4, pp. 698–700, Apr. 1987. 1

[8] B. K. P. Horn, H. M. Hilden, and S. Negahdaripour, "Closed-form solution of absolute orientation using orthonormal matrices," *J. Optical Society of America A*, vol. 5, no. 4, pp. 1127–1135, Apr. 1988. 1

[9] E. Bandari, N. Goldstein, I. Nesnas, and M. Bajracharya, "Efficient calculation of absolute orientation with outlier rejection," in *BMVA Symp. Spatiotemporal Image Processing*, Mar. 2004. 1

[10] C. Tomasi and T. Kanade, "Detection and tracking of point features," Carnegie Mellon University, Tech. Rep. CMU-CS-91-132, Apr. 1991. 1

[11] J. Y. Bouguet, "Pyramidal implementation of the lucas kanade feature tracker: Description of the algorithm," 2000. 1

[12] K. Levenberg, "A method for the solution of certain non-linear problems in least-squares," *Quarterly of Applied Mathematics*, vol. 2, no. 2, pp. 164–168, Jul. 1944. 2, 3

[13] D. Marquardt, "An algorithm for the least-squares estimation for non-linear parameters," *SIAM J. Applied Mathematics*, vol. 11, no. 2, pp. 431–441, Jun. 1963. 2, 3

[14] J. E. Dennis, D. M. Gay, and R. E. Walsh, "An adaptive nonlinear least-squares algorithm," *ACM Transactions on Mathematical Software*, vol. 7, no. 3, Sep. 1981. 2, 3, 7

[15] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, May 2006. 2, 4

[16] T. Drummond and R. Cipolla, "Real-time visual tracking of complex structures," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 932–946, Jul. 2002. 2

[17] S. Krishnan, P.-Y.-Lee, J. B. Moore, and S. Venkatasubramanian, "Global registration of multiple 3D point sets via optimization-on-a-manifold," in *Conf. Geometry Proc.*, Dec. 2005, pp. 1–11. 2

[18] C. Gebken, A. Tolvanen, C. Perwass, and G. Sommer, "Perspective pose estimation from uncertain omnidirectional image data," in *Int. Conf. Pattern Recognition*, Sep. 2006, pp. 793–796. 2

[19] J. R. Smith, "Integrated spatial and image feature system: Retrieval analysis and compression," Ph.D. dissertation, Columbia University, New York, NY, USA, 1997. 2

[20] R. Kumar and A. R. Hanson, "Robust methods for estimating pose and a sensitivity analysis," *Computer Vision, Graphics, and Image Processing: Image Understanding*, vol. 60, no. 3, pp. 313–342, Nov. 1994. 2

[21] L. Quan and Z. Lan, "Linear n-point camera pose determination," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 21, no. 8, pp. 774–780, 1999. 2

[22] C.-P. Lu, "Fast and globally convergent pose estimation from video images," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, no. 6, pp. 610–622, 2000. 2

[23] A. Ansar and K. Daniilidis, "Linear pose estimation from points or lines," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 25, no. 4, pp. 1–22, 2003. 2

[24] C.-S. Chen and W.-Y. Chang, "On pose recovery for generalized visual sensors," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 26, no. 7, pp. 848–861, Jul. 2004. 2, 7

[25] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon, "Bundle adjustment - a modern synthesis," in *Proc. Int. Workshop on Vision Algorithms*, Sep. 2000, pp. 298–375. 3, 5

[26] J. Nocedal and S. Wright, *Numerical optimization*, 2nd ed. Springer, Apr. 2006. 3, 6

[27] C. J. Taylor and D. J. Kriegman, "Minimization on the lie group SO(3) and related manifolds," Yale University, Tech. Rep. 9405, Apr. 1994. 4

[28] C. Udriste, *Convex functions and optimization methods on Riemannian manifolds*. Kluwer Academic Publishers, Jan. 1994. 4, 5

[29] A. Baker, *Matrix groups: An introduction to Lie group theory*. Springer, Oct. 2003. 4

[30] Y. Ma, S. Soatto, J. Kosecka, and S. S. Sastry, *An invitation to 3-D vision*. Springer, Jun. 2005. 4

[31] D. G. Luenberger, *Linear and nonlinear programming*, 2nd ed. Springer, Sep. 2003. 6

[32] J. Y. Bouguet, *Camera calibration toolbox for MATLAB*, Computational Vision Group, California Institute of Technology, Pasadena, CA, USA, 2001. 7

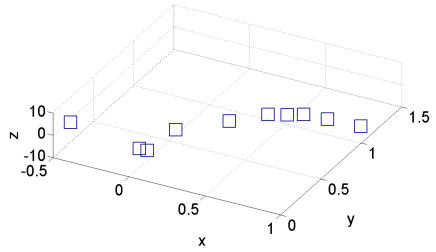
[33] U. Helmke, K. Hüper, P. Lee, and J. Moore, "Essential matrix estimation using Gauss-Newton iterations on a manifold," *Int. J. Computer Vision*, vol. 74, no. 2, pp. 117–136, Aug. 2007. 7

[34] M. Sarkis, K. Diepold, and K. Hüper, "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*, Apr. 2007, pp. I–681–I–684. 7, 8

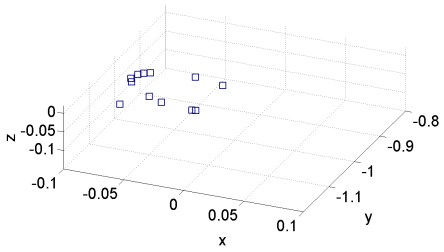
[35] M. Pollefeys, R. Koch, and L. V. Gool, "Self-calibration and metric reconstruction in spite of varying and unknown internal camera parameters," in *Int. Conf. Computer Vision*, Jan. 1998, pp. 90–95. 8

[36] A. W. Fitzgibbon, G. Cross, and A. Zisserman, "Automatic 3D model construction for turn-table sequences," in *Proc. SMILE Workshop on Structure from Multiple Images in Large Scale Environments*, Jun. 1998, pp. 154–170. 10

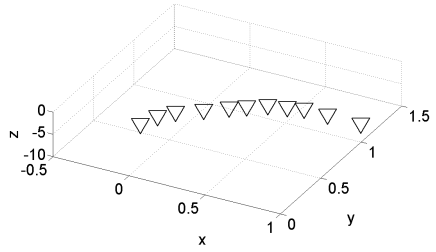
[37] W. Niem, "Robust and fast modelling of 3D natural objects from multiple views," in *SPIE Electronic Imaging: Image and Video Processing II*, Feb. 1994, pp. 338–397. 10



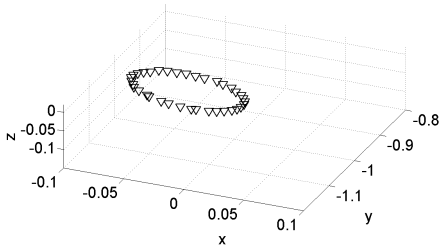
(a) Corridor with LM



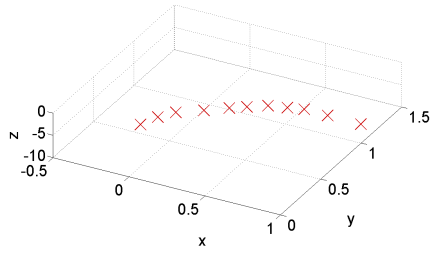
(b) Dinosaur with LM



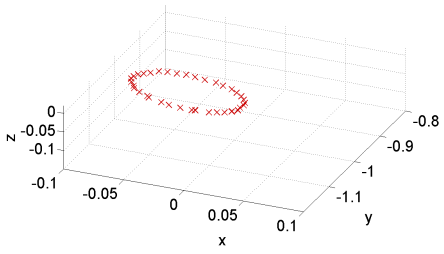
(c) Corridor with Newton Vectorspace



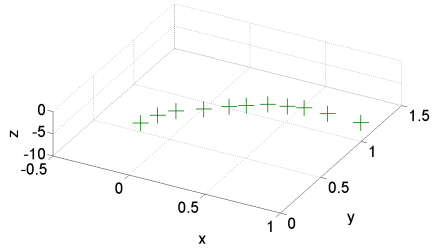
(d) Dinsosaur with Newton Vectorspace



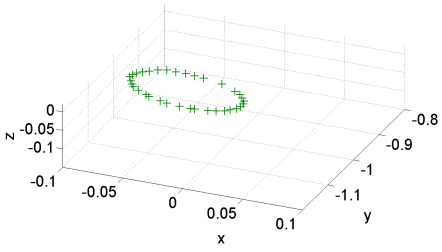
(e) Corridor with Newton SE_3



(f) Dinosaur with Newton SE_3



(g) Corridor with Newton $\mathbb{R}^3 \times SO_3$



(h) Dinosaur with Newton $\mathbb{R}^3 \times SO_3$

Fig. 10. The recovered motion trajectories of the cameras using the different minimization algorithms. These correspond to the settings corresponding to Figures 8a and 9a. The Corridor sequence is in the left column while the Dinosaur sequence is in the right column.

Michel Sarkis received a Bachelor in electrical engineering at the American University of Beirut (AUB), Lebanon, in 2002 and a M.Sc. in electrical engineering at Munich University of Technology (TUM), Germany in 2004 with a Master Thesis emphasizing on Genomic Signal Processing. He pursued his PhD at the institute for Data Processing (LDV) at TUM where he emphasized on stereo reconstruction for Telepresence and Teleaction systems. During that time, he worked on topics related to camera calibration, adaptive meshing, stereo matching, motion estimation and depth compression. Currently, he is working at the Sensing System Laboratory (SSL) in SONY Deutschland GmbH in the reconstruction of microwave and millimeter-wave images. His research interests include image reconstruction, image and video processing, sparse representations, system modeling and optimization algorithms. He is a member at IEEE.

Klaus Diepold was born, raised and educated in Munich, Germany. He received a Dipl.-Ing. degree and a Dr.-Ing. degree both in Electrical Engineering from Technische Universität München (TUM) in 1987 and 1992, respectively. In the years 1993-2002 he worked in the video signal processing, television and video compression industry, and leading the research lab of DynaPel Systems, inc. in Munich. In 2002, Klaus Diepold joined TUM as a full professor at the Department of Electrical Engineering and Information Technology. His main interest lies in the computational aspects of motion picture technology, video processing and compression, computer graphics, computer vision, 3D-audio, machine learning and cognition for technical systems. He is a member of COST IC1003 "Qualitnet", a European initiative dedicated to the topic "Quality of Experience", He was a Principal Investigator in a cooperative research centre on "High-Fidelity Telepresence and Teleaction"; he is a member of the executive board of the Excellence Cluster "Cognition for Technical Systems" and a Scientific Director of the "Centre for Digital Technology and Management".