Proceedings of the 2006 IEEE/RSJ
International Conference on Intelligent Robots and Systems
October 9 - 15, 2006, Beijing, China

Integration of Euclidean constraints in template based visual tracking of piecewise-planar scenes

Selim Benhimane, Ezio Malis
INRIA
2004 route des Lucioles, BP93
06902, Sophia Antipolis cedex, France
Firstname.Secondname@sophia.inria.fr

Abstract— **This papers deals with the problem of tracking a piecewise-planar scene in a video sequence and, at the same time, with the problem of estimating accurately the 3D displacement of the camera for robotic applications. A new approach to the problem is proposed and two noticeable contributions are given. Firstly, the explicit dependency between the 2D image transformation parameters (a homography for each plane) and the 3D camera displacement parameters is computed. Secondly, a second-order optimization algorithm is proposed. The second-oder optimization considerably increases the convergence domain and the convergence rate of standard first-order optimization algorithms while having an almost equivalent computational complexity.**

I. Introduction

Visual tracking of objects in a sequence of images is a fundamental step for various computer vision applications. In many applications, for example in vision-based robot control [1], the final objective of visual tracking is not only the localization of the objects in the image but also the estimation of the camera displacement with respect to a reference frame. Most of the visual tracking algorithms focus on the estimation of the 2D projective transformation of the objects between the images. In this paper, we consider visual tracking methods that minimize a dissimilarity measure between a reference template and the current image using parametric models of the 2D projective transformation (see for example [2], [3], [4], [5], [6]). Then, the camera displacement between the reference frame (attached to the reference template) and the current frame (attached to the current image) can be extracted from the image transformation using the camera intrinsic parameters [7]. Instead, our objective here is to track a piecewise-planar scene (which model is a priori known) in the 2D image and, at the same time, to estimate accurately the 3D displacement of the camera. It is well known that an accurate measure of the displacement can be obtained by adding some known constraints on the camera displacement and/or on the shape of the observed objects (see for example [8], [9], [10], [11]). These constraints are generally expressed in the Euclidean space. For example, the displacement of a camera mounted on a mobile robot can be constrained to be in a plane. The shape of the observed objects may be partially known and different regions of the same rigid object have the same 3D displacement despite they do not undergo the same image transformation. For example, when the scene is piecewise-planar, several planar patches can be independently extracted and tracked. This corresponds to estimating a projective homography for each plane. In the ideal case, the computation of the homographies provides a coherent camera displacement. In practice, this hardly happens if we do not explicitly impose the constraint that all the planes are rigidly attached to each other. Indeed, it can happen that two planar patches with poor textures give two contradictory displacements. In addition, it is generally difficult to transform this additional Euclidean information into simple constraints on the 2D image transformation. To overcome these difficulties, the primary objective of this paper is to compute the explicit dependency between the 2D image transformation parameters and the 3D camera displacement parameters. This makes it easier to integrate Euclidean constraints in the visual tracking algorithm and to give a direct and more accurate camera motion with respect to a reference frame attached to the observed scene. Several papers have addressed the problem of templated-based tracking imposing some Euclidean constraints. In [11], the authors avoid the explicit computation of the Jacobian that relates the variation of the 3D pose parameters to the variation of the appearance in the image by using implicit function derivatives. In that case, the minimization of the image error is done with a coarse approximation of the inverse compositional algorithm [2], [12]. In [13], the authors extend the method proposed in [3] to a homographic warping. The method makes the assumption that the true camera pose can be approximated by the current estimated pose (i.e. the camera displacement is sufficiently small). In addition, the Euclidean constraints are not directly imposed during the tracking, but once the homography has been estimated, the rotation and the translation of the camera are extracted [7]. In [10], the authors go one step further and extend the method proposed in [3] by including the constraint that a set of control points on a three-dimensional surface undergo the same camera displacement. The aim of our work is to propose a visual tracking algorithm to be integrated into vision-based robot control systems. This implies a real-time implementation and the possibility to track objects with fast camera motions. In order to meet these constraints, we suppose that the structure of the scene is known. This limits our system to 6 unknowns (the translation and rotation of the camera). Other methods consider both the motion and the structure as unknowns (see for example [14]).

In this case, the number of unknowns is considerably higher which reduces the real-time performance of the algorithm. Furthermore, even when fixing some unknowns, these methods generally use first-order approximations (e.g. the linearization involved in the Extended Kalman Filter proposed in [14]).

In this paper, we propose a different approach to the problem and we give two noticeable contributions. Firstly, we compute the explicit dependency between the 2D image transformation parameters (a homography for each plane) and the 3D parameters of the camera displacement. Secondly, we propose a second-order optimization algorithm that considerably increase the convergence domain and the convergence rate of standard first-order optimization algorithms while having an almost equivalent computational complexity.

II. Theoretical background

An image region is a $n \times m$ matrix containing the pixel intensities. Let $\mathcal{I}(i,j)$ be the entry of the matrix corresponding to the value of the pixel at the line i and at the column j. Let $\mathbf{p} = (u,v) \in \mathbb{R}^2$ be the vector containing the coordinates (in pixels) of a point in the image region. We suppose that there exists a C^∞ map $I : \mathbb{R}^2 \to \mathbb{R}$ such that, when $\mathbf{p} = (i,j) \in \{1,2,...,n\} \times \{1,2,...,m\}$ then $I(\mathbf{p}) = \mathcal{I}(i,j)$. For non-integer values of (u,v), the value of the function I at $\mathbf{p}$ is obtained by interpolating the measures $\mathcal{I}(i,j)$.

A. The Lie Groups $\mathbb{SL}(3)$ and $\mathbb{SE}(3)$

We suppose that the image contains the projection of a piecewise-planar environment. For each plane in the scene, there exists a (3×3) homography matrix $\mathbf{G}$ that links the coordinates $\mathbf{p} = (u,v)$ of a certain point in a reference image $\mathcal{I}^*$ (for example $\mathcal{I}^*$ can be the first image in the video sequence) to its corresponding point $\mathbf{q}$ in the current image $\mathcal{I}$:

$$\mathbf{q} = \left[\begin{array}{cc} \frac{g_{11}\,u + g_{12}\,v + g_{13}}{g_{31}\,u + g_{32}\,v + g_{33}}, & \frac{g_{21}\,u + g_{22}\,v + g_{23}}{g_{31}\,u + g_{32}\,v + g_{33}} \end{array} \right]$$

such that $\mathcal{I}^*(\mathbf{p}) = \mathcal{I}(\mathbf{q})$ and $\{g_{ij}\}$ are the entries of the matrix $\mathbf{G}$. The homography matrix is defined up to a scale factor. In order to fix the scale, we choose $\mathbf{G} \in \mathbb{SL}(3)$ (the Special Linear group of dimension 3). This choice is well justified since $\det(\mathbf{G}) = 0$ happens only if the observed plane passes through the optical center of the camera (in this singular case the plane is not visible any more). Let $\mathbf{w} : \mathbb{SL}(3) \times \mathbb{R}^2 \to \mathbb{R}^2$ be an action of $\mathbb{SL}(3)$ on $\mathbb{R}^2$ on the left (i.e. $\mathbf{w}(\mathbf{G})(\mathbf{p}) \in \mathbb{R}^2$). The map $\mathbf{w}(\mathbf{G}) : \mathbb{R}^2 \to \mathbb{R}^2$ defines a coordinate transformation (a warping) such that $\mathbf{q} = \mathbf{w}(\mathbf{G})(\mathbf{p})$. Let $\mathbf{I}$ be the identity element of the transformation group. We have the following properties:

- $\mathbf{w}(\mathbf{I})(\mathbf{p})$ is the identity map, i.e. $\forall \mathbf{p} \in \mathbb{R}^2$:

$$\mathbf{w}(\mathbf{I})(\mathbf{p}) = \mathbf{p} \tag{1}$$

- the composition of two actions corresponds to the action of the composition, i.e $\forall \mathbf{p} \in \mathbb{R}^2$, $\forall \mathbf{G}_1,\ \mathbf{G}_2 \in \mathbb{SL}(3)$:

$$\mathbf{w}(\mathbf{G}_1)(\mathbf{w}(\mathbf{G}_2)(\mathbf{p})) = \mathbf{w}(\mathbf{G}_1\mathbf{G}_2)(\mathbf{p}) \tag{2}$$

The camera displacement between two views can be represented by a (4×4) matrix $\mathbf{T} \in \mathbb{SE}(3)$ (the Special Euclidean group):

$$\mathbf{T} = \left[\begin{array}{cc} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{array} \right]$$

where $\mathbf{R} \in \mathbb{SO}(3)$ is the rotation and $\mathbf{t} \in \mathbb{R}^3$ is the translation of the camera. Here and in the following, $\mathbf{0}$ denotes a zero submatrix of appropriate size. The 2D projective transformation $\mathbf{G}$ is similar to a matrix $\mathbf{H} \in \mathbb{SL}(3)$ that depends on $\mathbf{T} \in \mathbb{SE}(3)$:

$$\mathbf{G}(\mathbf{T}) = \mathbf{K}\mathbf{H}(\mathbf{T})\mathbf{K}^{-1}$$

where $\mathbf{K}$ is the upper triangular matrix containing the camera intrinsic parameters (i.e. the focal length f, the skew s, the aspect ratio r and the principal point (u_0, v_0)). The matrix $\mathbf{H}(\mathbf{T})$ can be written as follows:

$$\mathbf{H}(\mathbf{T}) = \frac{\mathbf{R} + \mathbf{t}\mathbf{n}_d^{*\top}}{\sqrt[3]{1 + \mathbf{t}^\top \mathbf{R}\mathbf{n}_d^*}}$$

where $\mathbf{n}_d^* = \mathbf{n}^*/d^*$ is the ratio between the normal vector to the plane $\mathbf{n}^*$ (a unit vector) and the distance d^* of the plane to the origin of the reference frame. The map $\mathbf{H}$ between $\mathbb{SE}(3)$ and $\mathbb{SL}(3)$ is not a group homomorphism (indeed for $\mathbf{T}_1, \mathbf{T}_2 \in \mathbb{SE}(3)$, in general we have: $\mathbf{H}(\mathbf{T}_1)\mathbf{H}(\mathbf{T}_2) \neq \mathbf{H}(\mathbf{T}_1\mathbf{T}_2)$). Thus, the map $\mathbf{w}$ is not an action of $\mathbb{SE}(3)$ on $\mathbb{R}^2$. In general, we have : $\mathbf{w}(\mathbf{G}(\mathbf{T}_1))(\mathbf{w}(\mathbf{G}(\mathbf{T}_2)(\mathbf{p})) \neq \mathbf{w}(\mathbf{G}(\mathbf{T}_1\mathbf{T}_2))(\mathbf{p})$ In some particular cases, when $\mathbf{T}$ belongs to some subgroup of $\mathbb{SE}(3)$, the map $\mathbf{w}$ is an action. For example, if $\mathbf{t} = \mathbf{0}$ then $\mathbf{H}$ depends only on the rotation $\mathbf{R} \in \mathbb{SO}(3)$ and the map $\mathbf{w}$ is an action of $\mathbb{SO}(3)$ on $\mathbb{R}^2$.

B. The Lie Algebra of $\mathbb{SE}(3)$ and the exponential map

Let $\mathbf{A}_i$, with $i \in \{1,2,...,6\}$, be a basis of the Lie Algebra $\mathfrak{se}(3)$ (i.e. the dimension of the Lie Algebra $\mathfrak{se}(3)$ is 6). Any matrix $\mathbf{A} \in \mathfrak{se}(3)$ can be written as a linear combination of the matrices $\mathbf{A}_i$:

$$\mathbf{A}(\mathbf{x}) = \sum_{i=1}^{6} x_i \mathbf{A}_i \tag{3}$$

where $\mathbf{x} = (x_1,\ x_2,\ ...,\ x_6)$ is a (6×1) vector and x_i is the i-th element of the base field. Let the (3×1) vectors $\mathbf{b}_x = (1,0,0)$, $\mathbf{b}_y = (0,1,0)$ and $\mathbf{b}_z = (0,0,1)$ be the natural orthonormal basis of $\mathbb{R}^3$. Knowing that the dimension of the matrices $\mathbf{A}_i$ is (4×4), the generators for the translation are:

$$\mathbf{A}_1 = \left[\begin{array}{cc} \mathbf{0} & \mathbf{b}_x \\ \mathbf{0} & 0 \end{array} \right], \mathbf{A}_2 = \left[\begin{array}{cc} \mathbf{0} & \mathbf{b}_y \\ \mathbf{0} & 0 \end{array} \right], \mathbf{A}_3 = \left[\begin{array}{cc} \mathbf{0} & \mathbf{b}_z \\ \mathbf{0} & 0 \end{array} \right]$$

The generators for the rotation are:

$$\mathbf{A}_4 = \left[\begin{array}{cc} [\mathbf{b}_x]_\times & \mathbf{0} \\ \mathbf{0} & 0 \end{array} \right], \mathbf{A}_5 = \left[\begin{array}{cc} [\mathbf{b}_y]_\times & \mathbf{0} \\ \mathbf{0} & 0 \end{array} \right], \mathbf{A}_6 = \left[\begin{array}{cc} [\mathbf{b}_z]_\times & \mathbf{0} \\ \mathbf{0} & 0 \end{array} \right]$$

where $[\mathbf{b}_i]_\times$ is the antisymmetric matrix associated to the vector $\mathbf{b}_i$ (i.e. $[\mathbf{b}_i]_\times \in \mathfrak{so}(3)$). The exponential map links the Lie Algebra to the Lie Group: $\exp : \mathfrak{se}(3) \to \mathbb{SE}(3)$. There exist an open cube v about $\mathbf{0}$ in $\mathfrak{se}(3)$ and an open neighborhood U of the identity matrix $\mathbf{I}$ in $\mathbb{SE}(3)$ such that $\exp : v \to U$ is smooth and one-to-one onto, with a smooth inverse. The neighborhood U of $\mathbf{I}$ is very large (i.e. the rotation angle must be less than π). Consequently, a homography matrix $\mathbf{H}$ is a function of $\mathbf{T}$ that can be locally parameterized as:

$$\mathbf{H}(\mathbf{T}(\mathbf{x})) = \mathbf{H}\left(\exp\left(\sum_{i=1}^{6} x_i \mathbf{A}_i\right)\right)$$

III. Visual tracking

Let $q = nm$ be the total number of pixels in some image region corresponding to the projection of a planar patch of the scene in the reference image $\mathcal{I}^*$. This image region is called the reference template. To track the template in the current image $\mathcal{I}$ is to find the transformation $\overline{\mathbf{T}} \in \mathbb{SE}(3)$ that warps a pixel of that region in the reference image $\mathcal{I}^*$ into its correspondent in the current image $\mathcal{I}$, i.e find $\overline{\mathbf{T}}$ such that $\forall i$:

$$\mathcal{I}\left(\mathbf{w}\left(\mathbf{G}(\overline{\mathbf{T}})\right)(\mathbf{p}_i)\right) = \mathcal{I}^*(\mathbf{p}_i)$$

For each plane in the scene, we consider its corresponding template and its homography. For the sake of simplicity, we describe here the computations for a single plane. Let us make the following change of coordinates:

$$\mathbf{q}_i = \mathbf{w}(\mathbf{G}(\overline{\mathbf{T}}))(\mathbf{p}_i)$$

Then, we have:

$$\mathbf{p}_i = \mathbf{w}^{-1}\left(\mathbf{G}(\overline{\mathbf{T}})\right)(\mathbf{q}_i) = \mathbf{w}(\mathbf{G}(\overline{\mathbf{T}})^{-1})(\mathbf{q}_i)$$

and:

$$\mathcal{I}(\mathbf{q}_i) = \mathcal{I}^*\left(\mathbf{w}\left(\mathbf{G}(\overline{\mathbf{T}})^{-1}\right)(\mathbf{q}_i)\right) \tag{4}$$

Knowing an approximation $\widehat{\mathbf{T}}$ of the transformation $\overline{\mathbf{T}}$, the problem is to find the incremental transformation $\mathbf{T}(\mathbf{x})$ that minimizes the sum of squared differences (SSD) between the current image region $\mathcal{I}$ warped with $\widehat{\mathbf{T}}\mathbf{T}(\mathbf{x})$ and the reference image $\mathcal{I}^*$, i.e $\mathbf{T}(\mathbf{x})$ that minimizes:

$$\phi(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{q}\left[\mathcal{I}\left(\mathbf{w}(\mathbf{G}(\widehat{\mathbf{T}}\mathbf{T}(\mathbf{x})))(\mathbf{p}_i)\right) - \mathcal{I}^*(\mathbf{p}_i)\right]^2$$

Writing this equation in the same frame by using the equations (2) and (4), the problem becomes find $\mathbf{x}$ that minimizes the function:

$$\phi(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{q}\left[\mathcal{I}^*\left(\mathbf{w}(\mathbf{G}(\overline{\mathbf{T}})^{-1}\mathbf{G}(\widehat{\mathbf{T}}\mathbf{T}(\mathbf{x})))(\mathbf{p}_i)\right) - \mathcal{I}^*(\mathbf{p}_i)\right]^2$$

The global minimum is obviously obtained when $\mathbf{x} = \mathbf{x}_0$ and where $\mathbf{x}_0$ verifies: $\widehat{\mathbf{T}}^{-1}\overline{\mathbf{T}} = \mathbf{T}(\mathbf{x}_0)$. The function ϕ is generally non-linear. As a consequence, the global minimum cannot be found in one iteration. An iterative minimization algorithm is then used. Having an estimated update $\mathbf{x}$, we can update the transformation matrix:

$$\widehat{\mathbf{T}} \longleftarrow \widehat{\mathbf{T}}\mathbf{T}(\mathbf{x}) \tag{5}$$

The nonlinear least-squares minimization can be solved with a standard optimization method using a first-order approximation (section III-A). Instead, we propose to use an optimization method that performs a second-order approximation (section III-B).

A. *First-order approximation*

Consider the $(q \times 1)$ vector $\mathbf{y}(\mathbf{x})$ obtained by rearranging the entries of the image differences:

$$\mathbf{y}(\mathbf{x}) = (y_1(\mathbf{x}), y_2(\mathbf{x}), ..., y_q(\mathbf{x}))$$

where $\forall i \in \{1, 2, ..., q\}$, we have:

$$y_i(\mathbf{x}) = \mathcal{I}^*\left(\mathbf{w}\left(\mathbf{G}(\overline{\mathbf{T}})^{-1}\mathbf{G}(\widehat{\mathbf{T}}\mathbf{T}(\mathbf{x}))\right)(\mathbf{p}_i)\right) - \mathcal{I}^*(\mathbf{p}_i)$$

The cost function ϕ can thus be written as:

$$\phi(\mathbf{x}) = \frac{1}{2}\|\mathbf{y}(\mathbf{x})\|^2 \tag{6}$$

The necessary condition for finding a local or the global minimum of the cost function is that there exists a stationary point $\mathbf{x}_0$ such that:

$$[\nabla_{\mathbf{x}}\phi(\mathbf{x})]_{\mathbf{x}=\mathbf{x}_0} = \mathbf{0} \tag{7}$$

where $\nabla_{\mathbf{x}}$ is the gradient operator with respect to the parameter $\mathbf{x}$. Equation (7) is also non-linear, therefore closed-form solution is generally difficult to obtain. However, we can approximate the cost function by performing first-order Taylor series of $\mathbf{y}(\mathbf{x})$ about $\mathbf{x} = \mathbf{0}$:

$$\mathbf{y}(\mathbf{x}) = \mathbf{y}(\mathbf{0}) + \mathbf{J}(\mathbf{0})\,\mathbf{x} + \mathcal{R}(\|\mathbf{x}\|^2) \tag{8}$$

where $\mathbf{J}(\mathbf{z}) = \nabla_{\mathbf{z}}(\mathbf{y}(\mathbf{z}))$ and $\mathcal{R}(\|\mathbf{x}\|^2)$ is the second-order reminder. Plugging equation (8) in (6) gives:

$$\phi(\mathbf{x}) = \frac{1}{2}\|\mathbf{y}(\mathbf{0}) + \mathbf{J}(\mathbf{0})\mathbf{x}\|^2 + \mathcal{R}(\|\mathbf{x}\|^2) \tag{9}$$

When using the first-order approximation of $\mathbf{y}(\mathbf{x})$, $\widehat{\mathbf{y}}_1(\mathbf{x}) = \mathbf{y}(\mathbf{0}) + \mathbf{J}(\mathbf{0})\mathbf{x}$, the cost function is approximated by:

$$\phi(\mathbf{x}) \approx \frac{1}{2}\|\widehat{\mathbf{y}}_1(\mathbf{x})\|^2$$

Then, the derivative of the cost function can be approximated as follows:

$$\nabla_{\mathbf{x}}(\phi(\mathbf{x})) \approx \nabla_{\mathbf{x}}(\widehat{\mathbf{y}}_1(\mathbf{x}))^\top \widehat{\mathbf{y}}_1(\mathbf{x})$$

and using (7), we find the approximated local minimizer by solving the equation:

$$[\nabla_{\mathbf{x}}(\widehat{\mathbf{y}}_1(\mathbf{x}))]^\top_{\mathbf{x}=\mathbf{x}_0}(\mathbf{y}(\mathbf{0}) + \mathbf{J}(\mathbf{0})\mathbf{x}_0) = 0$$

When $\mathbf{J}(\mathbf{0})$ is a full rank square matrix, the solution is obviously $\mathbf{x}_0 = -\mathbf{J}(\mathbf{0})^{-1}\mathbf{y}(\mathbf{0})$. When $\mathbf{J}(\mathbf{0})$ is neither square nor full rank, any generalized inverse can be used. For example, we can use the pseudo-inverse $\mathbf{J}(\mathbf{0})^+$ of the matrix $\mathbf{J}(\mathbf{0})$. If $\mathbf{J}(\mathbf{0})$ is full rank, the pseudo-inverse can then be written as: $\mathbf{J}(\mathbf{0})^+ = \left(\mathbf{J}(\mathbf{0})^\top\mathbf{J}(\mathbf{0})\right)^{-1}\mathbf{J}(\mathbf{0})^\top$. This corresponds to the well-known Gauss-Newton method. Note that with few modifications, one can use the Levenberg-Marquardt method [15][16]. The Jacobian $\mathbf{J}(\mathbf{0})$ can be written as the product of five Jacobians:

$$\mathbf{J}(\mathbf{0}) = \mathbf{J}_{\mathcal{I}}\,\mathbf{J}_P\mathbf{J}_K\,\mathbf{J}_{\widehat{T}}\,\mathbf{J}_X(\mathbf{0}) \tag{10}$$

As a consequence, the approximated solution for the minimum is:

$$\mathbf{x}_0 = -(\mathbf{J}_{\mathcal{I}}\,\mathbf{J}_P\mathbf{J}_K\,\mathbf{J}_{\widehat{T}}\,\mathbf{J}_X(\mathbf{0}))^+\mathbf{y}(\mathbf{0})$$

Note that, using the current Jacobian given in equation (10), one can design minimization algorithms similar to [5].

B. Second-order approximation

Although the optimization problem can be solved using first-order methods, we propose to use an efficient second-order minimization algorithm. Indeed, with little extra computation, we can make the most of the local quadratic convergence rate of the second-order optimization [17], [6]. A second-order Taylor series of $\mathbf{y}(\mathbf{x})$ about $\mathbf{x} = \mathbf{0}$ gives:

$$\mathbf{y}(\mathbf{x}) = \mathbf{y}(\mathbf{0}) + \mathbf{J}(\mathbf{0})\, \mathbf{x} + \frac{1}{2}\mathbf{M}(\mathbf{0}, \mathbf{x})\mathbf{x} + \mathcal{R}(\|\mathbf{x}\|^3) \tag{11}$$

where $\mathbf{M}(\mathbf{z}, \mathbf{x}) = \nabla_{\mathbf{z}}(\mathbf{J}(\mathbf{z})\mathbf{x})$ and $\mathcal{R}(\|\mathbf{x}\|^3)$ is the third-order reminder. Similarly, we can write the Taylor series of the Jacobian $\mathbf{J}(\mathbf{x})$ about $\mathbf{x} = \mathbf{0}$:

$$\mathbf{J}(\mathbf{x}) = \mathbf{J}(\mathbf{0}) + \mathbf{M}(\mathbf{0}, \mathbf{x}) + \mathcal{R}(\|\mathbf{x}\|^2) \tag{12}$$

Plugging (12) in (11) leads to:

$$\mathbf{y}(\mathbf{x}) = \mathbf{y}(\mathbf{0}) + \frac{1}{2}(\mathbf{J}(\mathbf{0}) + \mathbf{J}(\mathbf{x}))\, \mathbf{x} + \mathcal{R}(\|\mathbf{x}\|^3) \tag{13}$$

Using equation (13), we obtain a second-order approximation of the cost function:

$$\phi(\mathbf{x}) = \frac{1}{2}\left\|\mathbf{y}(\mathbf{0}) + \frac{\mathbf{J}(\mathbf{0}) + \mathbf{J}(\mathbf{x})}{2}\, \mathbf{x}\right\|^2 + \mathcal{R}(\|\mathbf{x}\|^3) \tag{14}$$

Setting $\widehat{\mathbf{y}}_2(\mathbf{x}) = \mathbf{y}(\mathbf{0}) + \frac{1}{2}(\mathbf{J}(\mathbf{0}) + \mathbf{J}(\mathbf{x}))\, \mathbf{x}$, the cost function is approximated by:

$$\phi(\mathbf{x}) \approx \frac{1}{2}\|\widehat{\mathbf{y}}_2(\mathbf{x})\|^2$$

The derivative of the cost function is approximated by:

$$\nabla_{\mathbf{x}}\phi(\mathbf{x}) \approx \nabla_{\mathbf{x}}(\widehat{\mathbf{y}}_2(\mathbf{x}))^\top \widehat{\mathbf{y}}_2(\mathbf{x})$$

and using equation (7), we find the approximated local minimizer by solving the equation:

$$[\nabla_{\mathbf{x}}(\widehat{\mathbf{y}}_2(\mathbf{x}))]^\top_{\mathbf{x}=\mathbf{x}_0} \left(\mathbf{y}(\mathbf{0}) + \frac{\mathbf{J}(\mathbf{0}) + \mathbf{J}(\mathbf{x}_0)}{2}\, \mathbf{x}_0\right) = 0 \tag{15}$$

The Jacobian $\mathbf{J}(\mathbf{x}_0)$ is written again as the product of five Jacobians:

$$\mathbf{J}(\mathbf{x}_0) = \mathbf{J}_{\mathcal{I}^*}\mathbf{J}_P\mathbf{J}_K\mathbf{J}_{\overline{T}}\mathbf{J}_X(\mathbf{x}_0) \tag{16}$$

Despite the Jacobian $\mathbf{J}(\mathbf{x}_0)$ generally depends on the unknown $\mathbf{x}_0$, thanks to the left-invariance property of the vector fields on $\mathbb{SE}(3)$, we have the following identity:

$$\mathbf{J}_X(\mathbf{x}_0)\mathbf{x}_0 = \mathbf{J}_X(\mathbf{0})\mathbf{x}_0$$

Thus, in the equation (15), we can use $\mathbf{J}_X(\mathbf{0})\mathbf{x}_0$ instead of $\mathbf{J}_X(\mathbf{x}_0)\mathbf{x}_0$. On the other hand, the Jacobian $\mathbf{J}_{\overline{T}}$ depends on the unknown transformation $\overline{T}$. However, the Jacobian can be approximated by $\mathbf{J}_{\overline{T}} \approx \mathbf{J}_{\widehat{T}}$. The update $\mathbf{x}_0$ of the second-order minimization algorithm can be computed as follows:

$$\mathbf{x}_0 = -\left(\left(\frac{\mathbf{J}_{\mathcal{I}} + \mathbf{J}_{\mathcal{I}^*}}{2}\right)\mathbf{J}_P\mathbf{J}_K\mathbf{J}_{\widehat{T}}\mathbf{J}_X(\mathbf{0})\right)^+ \mathbf{y}(\mathbf{0})$$

In the next section, we show through experiments that our second-order optimization algorithm converges faster than the standard first-order algorithms. At the same time, the computational complexity is almost the same as for first-order algorithms. Note that, using the reference Jacobian given in equation (16), one can design minimization algorithms similar to [2] but with some additional limitations [12].

IV. Simulation results

A. Tracking a piecewise-planar scene

In order to have a ground truth, we have generated a video sequence with a synthetic piecewise-planar scene. The different planar textures of the scene are extracted from real images to make the simulation as realistic as possible. This simulation proposes to track three different planar regions over a sequence of 100 images. The trajectory of the camera is a closed-loop with the optical axis always pointing towards the planes. We compare two different strategies. In the first one, we estimate the image transformation of the three planar regions separately (see Figure 1). When tracking the planes separately, we use the second-order method proposed in section III-B with the $\mathbb{SL}(3)$ group [6]. Then, knowing the camera intrinsic parameters $\mathbf{K}$ and the ratio $\mathbf{n}_d^*$, we extract the rotation and the translation of the camera [7]. When tracked separately, the constraint that the camera displacement is the same for all the planes cannot be imposed directly. Thus, a second step is needed in order to obtain a coherent estimate of the camera displacement. In the second strategy, we track the three planes jointly (see Figure 1). Thanks to the parameterization of the incremental displacements in the $\mathbb{SE}(3)$ group with its corresponding Lie Algebra, given $\mathbf{n}_d^*$ for each plane and $\mathbf{K}$, it is possible to impose the Euclidean constraint that the rotation and the translation of the camera are the same for the three homographies. For each image of the sequence, we directly estimate the rotation and the translation of the camera with respect to a reference frame. Visually, the results show that tracking the three planes jointly gives better results than tracking them separately. For example, in image 63 we can clearly see that one of the templates is practically lost by the visual tracking algorithm.

B. Advantages of second-order optimization

We compare now the second-order minimization algorithm (called "SO") and two standard first-order Gauss-Newton minimizations algorithms: one, similar to [4], [5]), using the current Jacobian (called "CJ"), and one, similar to [3], [2]), using reference Jacobian (called "RJ"). All of the three algorithms use the Lie Algebra parameterization described in section II-B. The performance of the algorithms have been compared with the same simulation setup. The template has been selected in the center of the image taken at a reference position shown in Figure 2. The camera observing the scene has been moved 1000 times using different rotations and translations. The movements have been computed randomly but such that their effect on the image adds a Gaussian noise to the coordinates of the 4 corners of the template. The

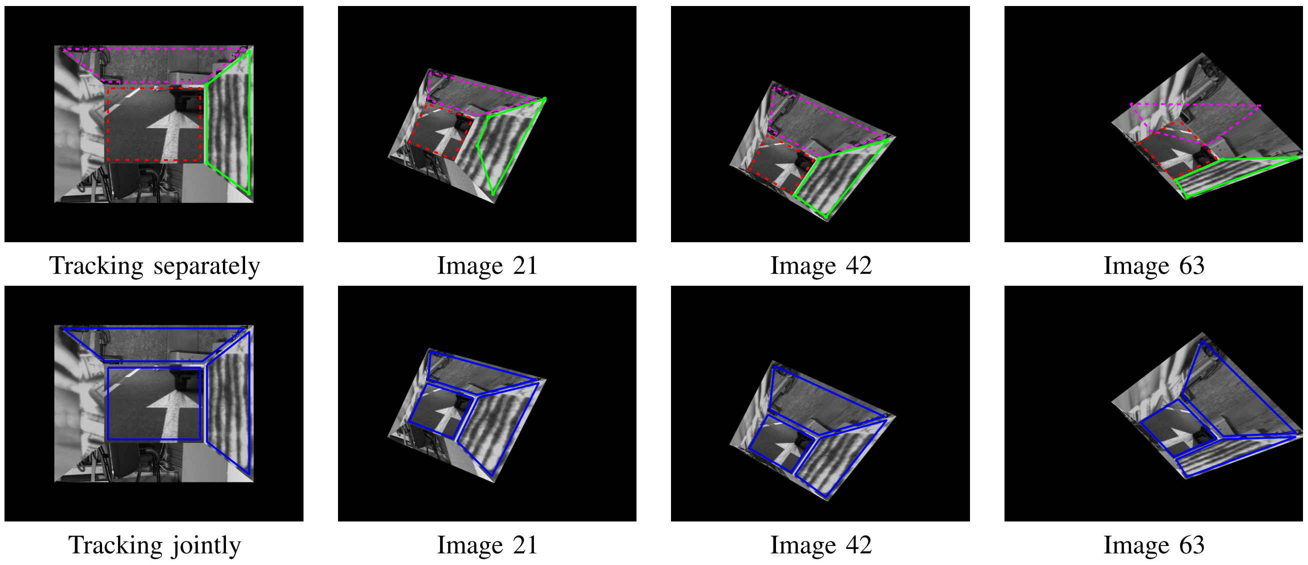

Fig. 1. The red, magenta and green lines represent the tracking error computed respectively the first, second and third plane separately. The blue line represents the tracking error computed considering the three plane at the same time.

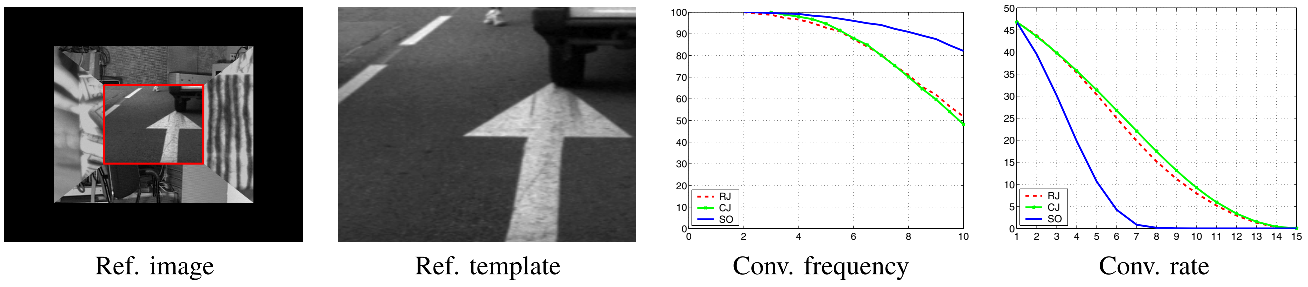

Fig. 2. Comparison between the proposed second-order optimization algorithm (SO) and standard first-order optimization algorithms (RJ and CJ).

standard deviation σ of the Gaussian noise is increased from 0.5 to 10. Figure 2 plots the convergence frequencies (% over 1000 tests). As σ grows, the convergence frequencies of the RJ and the CJ methods decay quicker than the convergence frequency of the second-order method. At the final $\sigma = 10$, the frequency of convergence of the RJ and the CJ methods are only 50% while the frequency of convergence of the second-order method is more than 80%. Figure 2 shows the average convergence rate over the converged tests of the algorithms for $\sigma = 10$. We consider that a test has converged if, after 15 iterations, the position error of each corner of the template is less than 1 pixel. The initial SSD is the same for the three algorithms but the speed of convergence of the second-order method is much higher. Indeed, in the converged tests, the second-order approximation algorithm needs only 7 iterations to converge while the first-order algorithms need much more (14 iterations) to converge. This means that we can perform real-time tracking at higher rates (the algorithm has been tested up to 60 frames per second using a pyramidal implementation for a total of 80000 pixels tracked). Finally, despite the approximation made on the reference Jacobian, the performances of the RJ and CJ algorithm are almost equivalent for small displacements corresponding to small σ (similarly to what it has been observed in [18]).

V. Experimental results

We have tested our algorithm on a sequence of 1000 images acquired by a camera mounted on a mobile robot. Three different planar patches have been selected from the scene observed by the camera in the first image. Each patch belongs to a different plane. The camera is calibrated and the normal vectors to the 3 planes have been roughly estimated. In Figure 3, excerpts from the sequence of tracked images can be seen. We can see that the different patches selected were well tracked along the whole sequence (see the submitted mpeg video). In addition, the 3D displacement of the camera has been accurately estimated (as illustrated by the curves in Figure 4) along a trajectory of more than 6 meters length. Indeed, when compared to a very precise odometry measure obtained by the robot sensors, the visual tracking makes it possible to obtain an accurate estimation of the robot translation (the error is less than 12 cm for a 7 meter long trajectory) and an accurate estimation of the robot rotation (the error is less than 1.6 degrees knowing that the robot did rotations up to 15 degrees).

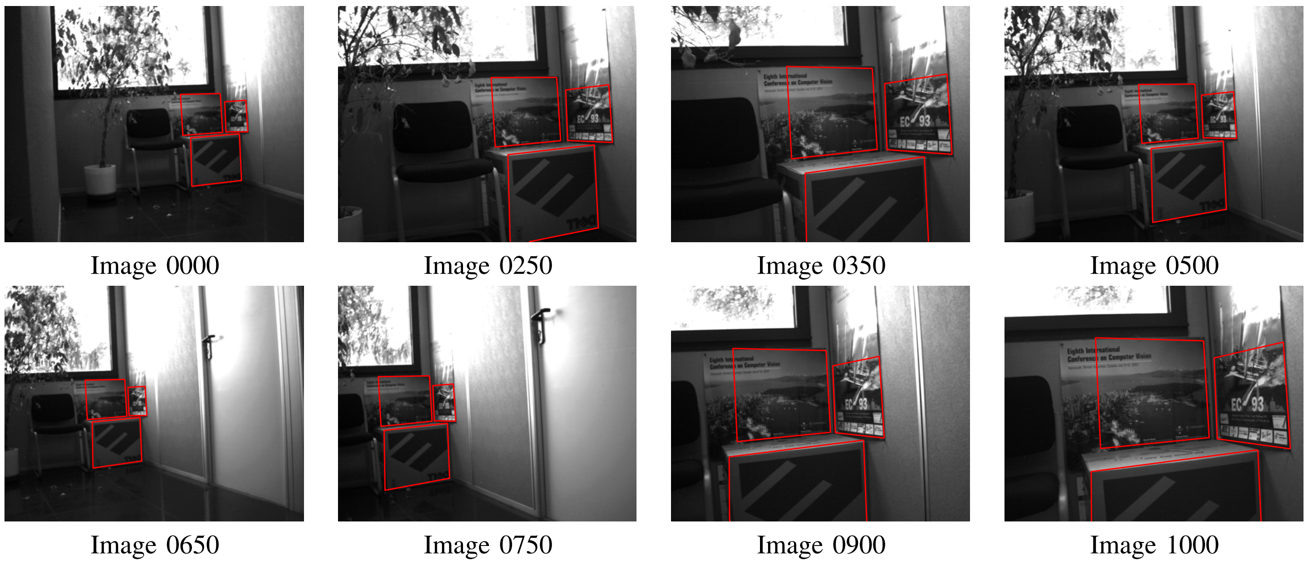

Fig. 3. Excerpts from a sequence of tracked images of 3 different planes.

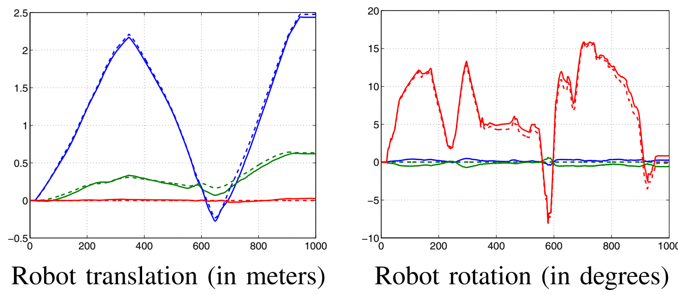

Fig. 4. Comparison between the motion estimation obtained by the tracking algorithm (with solid lines) and obtained by accurate odometry sensors (with dashed lines): the green curves are about the axis, the red are about the y axis and the blue are about the z axis.

VI. Conclusion

In many applications, a camera is used as a sensor for controlling a robot. Tracking the pose of the camera (up to a scale factor for the translation) with respect to a reference frame is as important as tracking in the image space. Extracting the camera pose from the 2D projective transformations (the homographies) of a piecewise-planar scene can be very inaccurate. In order to improve the accuracy of the pose estimation, some known constraints on the environment or on the movement of the camera can be added to the relative motion model. Adding in the projective space an “a priori” knowledge on the observed scene can be possible only in some special cases. In this paper, we have shown that the Euclidean constraints can be added directly in the template-based visual tracking framework. Then, we have proposed a second-order optimization algorithm that enlarges the convergence domain and has faster convergence rate than standard first-order algorithms. Future work will be devoted to the generalization of our algorithm to the tracking of a scene with any structure (not necessarily piecewise-planar) and to the on-line estimation of the structure.

References

[1] S. Hutchinson, G. D. Hager, and P. I. Corke, “A tutorial on visual servo control,” *ITRA*, vol. 12, no. 5, pp. 651–670, 1996.

[2] S. Baker and I. Matthews, “Lucas-kanade 20 years on: a unifying framework,” *IJCV*, vol. 56, no. 3, pp. 221–255, February 2004.

[3] G. Hager and P. Belhumeur, “Efficient region tracking with parametric models of geometry and illumination,” *IEEE Trans. on PAMI*, vol. 20, no. 10, pp. 1025–1039, 1998.

[4] B. Lucas and T. Kanade, “An iterative image registration technique with application to stereo vision,” in *Int. Joint Conf. on Artificial Intelligence*, 1981, pp. 674–679.

[5] H. Shum and R. Szeliski, “Construction of panoramic image mosaics with global and local alignment,” *IJCV*, vol. 16, no. 1, pp. 63–84, 2000.

[6] S. Benhimane and E. Malis, “Real-time image-based tracking of planes using efficient second-order minimization,” in *IROS*, 2004, pp. 943–948.

[7] O. Faugeras and F. Lustman, “Motion and structure from motion in a piecewise planar environment,” *Int. Journal of Pattern Recognition and Artificial Intelligence*, vol. 2, no. 3, pp. 485–508, 1988.

[8] E. Marchand, P. Bouthemy, and F. Chaumette, “A 2d-3d model-based approach to real-time visual tracking,” *Image and Vision Computing*, vol. 19, no. 13, pp. 941–955, November 2001.

[9] T. Drummond and R. Cipolla, “Real-time visual tracking of complex structures,” *IEEE Trans. on PAMI*, vol. 24, no. 7, pp. 932–946, 2002.

[10] W. Sepp and G. Hirzinger, “Real-time texture-based 3-d tracking.” in *DAGM-Symposium*, ser. Lecture Notes in Computer Science, vol. 2781. Springer-Verlag, 2003, pp. 330–337.

[11] D. Cobzas and M. Jagersand, “3D SSD tracking from uncalibrated video,” in *Proc. of Spatial Coherence for Visual Motion Analysis (SCVMA 2004), in conjunction with ECCV*, 2004.

[12] S. Baker, R. Patil, K. Cheung, and I. Matthews, “Lucas-kanade 20 years on: Part 5,” Robotics Institute, Carnegie Mellon University, Tech. Rep. CMU-RI-TR-04-64, November 2004.

[13] J. Buenaposada and L. Baumela, “Real-time tracking and estimation of planar pose,” in *CVPR*, 2002, pp. 697–700.

[14] H. Jin, P. Favaro, and S. S., “A semi-direct approach to structure from motion,” *The Visual Computer*, vol. 6, no. 19, pp. 377–394, 2003.

[15] K. Levenberg, “A method for the solution of certain problems in least squares.” *Quart. Appl. Math.*, vol. 2, pp. 164–168, 1944.

[16] D. W. Marquardt, “An algorithm for least-squares estimation of nonlinear parameters,” *SIAM JAM*, vol. 11, pp. 431–441, 1963.

[17] E. Malis, “Improving vision-based control using efficient second-order minimization techniques,” in *ICRA*, 2004, pp. 1843–1848.

[18] S. Baker and I. Matthews, “Equivalence and efficiency of image alignment algorithms,” in *CVPR*, 2001, pp. 1090–1097.