Efficient Homography-Based Tracking and 3-D Reconstruction for Single-Viewpoint Sensors

Christopher Mei, Selim Benhimane, Ezio Malis, *Associate Member, IEEE*, and Patrick Rives, *Member, IEEE*

***Abstract*—This paper addresses the problem of motion estimation and 3-D reconstruction through visual tracking with a single-viewpoint sensor and, in particular, how to generalize tracking to calibrated omnidirectional cameras. We analyze different minimization approaches for the intensity-based cost function (sum of squared differences). In particular, we propose novel variants of the efficient second-order minimization (ESM) with better computational complexities and compare these algorithms with the inverse composition (IC) and the hyperplane approximation (HA). Issues regarding the use of the IC and HA for 3-D tracking are discussed. We show that even though an iteration of ESM is computationally more expensive than an iteration of IC, the faster convergence rate makes it globally faster. The tracking algorithm was validated by using an omnidirectional sensor mounted on a mobile robot.**

***Index Terms*—Omnidirectional vision, structure from motion, visual tracking.**

I. Introduction

WIDE-field-of-view cameras are becoming increasingly popular in mobile robotics as they offer advantages in tasks such as motion estimation, autonomous navigation, and localization [1]. Recent research using omnidirectional cameras has been focused on ego-motion estimation [2], [3] and visual servoing [4], [5]. Visual tracking, which is a fundamental step for various computer vision and robotic applications, has seen very few articles. The advantages of direct visual tracking in terms of precision have been shown for perspective cameras [6] and stereo pairs [7], which motivates the extension of tracking to omnidirectional cameras.

The tracking approach presented in this paper minimizes a dissimilarity measure and more specifically a sum of squared differences (SSD) between a reference template and the current image taken with a large-field-of-view sensor. This leads to a nonlinear optimization problem that can be solved for small displacements (the type of movement that would be expected in a scene at video rate). The advantage of SSD tracking is precision (all the information is being used, leading to subpixel accuracy) and speed ($\sim$100 Hz). This is why these techniques are particularly well adapted to robotic tasks such as visual servoing and as an input to simultaneous localization and mapping (SLAM) algorithms. The downside is the need for a strong overlap between the reprojected and the real object for the algorithm to converge.

The apparent difficulty of tracking with omnidirectional sensors comes from the nonlinear projection model resulting in changes of shape in the image that makes the direct use of methods such as Kanade–Lucas–Tomasi (KLT) [8], [9] nearly impossible. Parametric models [10]–[12], such as the homography-based approach presented in this paper, are well adapted to this problem. Previous related work using homography-based tracking for perspective cameras include [13] and [14], which extend the work proposed by Hager and Belhumeur [10]. Homographies have also been used for visual servoing with central catadioptric cameras [5], [15] and share with our approach the notion of homographies for points belonging to the sphere of the unified projection model. We assume in this paper that the camera has been calibrated. The single-viewpoint property means it would be possible to track in an unwarped perspective view. This is, however, undesirable for the following reasons.

1) It introduces a discontinuity in the Jacobian (at least two planes are needed to represent the 360° field of view).
2) The nonuniform resolution is not taken into account.
3) The approach is inefficient (in terms of speed and memory usage).

To our knowledge, this is the only work on SSD tracking for omnidirectional sensors. The closest work is that of Barreto *et al.* [4]. The authors propose a method for tracking omnidirectional lines using a contour-to-point tracker to avoid the problem of quadric-based catadioptric line fitting. In [16] and [17], the authors estimate the position of a perspective camera assuming known plane positions. In our paper, the plane normals and distances are estimated. Compared to methods such as [18] and [19], we proceed in an iterative minimization to achieve better accuracy and more reliable convergence, as will be discussed in Section IV-B. This paper is also related to [17], where the authors introduce the efficient second-order minimization (ESM) algorithm for 2-D homography-based tracking. In this paper, we propose novel variants to the ESM and show how to apply the algorithm to jointly estimate the pose and plane positions (instead of separate tracking) in a more general omnidirectional framework.

In this paper, we will not discuss the important aspects of a complete visual SLAM solution, such as automatic plane

Manuscript received July 30, 2007; revised September 21, 2008. First published December 9, 2008; current version published December 30, 2008. This paper was recommended for publication by Associate Editor J. D. Tardos and Editor K. Lynch upon evaluation of the reviewers' comments. This paper was presented in part at the IEEE/RSJ International Conference on Intelligent Robot Systems, Beijing, China, October 2006 and at the British Machine Vision Conference, Edinburgh, U.K., September 2006.

C. Mei is with the Robotics Research Group, Department of Engineering Science, University of Oxford, Oxford OX1 3PJ, U.K. (e-mail: christopher.mei@eng.ox.ac.uk).

S. Benhimane is with the Technical University of Munich, 85748 Munich, Germany (e-mail: selim.benhimane@cs.tum.edu).

E. Malis and P. Rives are with the Institut National de Recherche en Informatique et en Automatique, 06902 Sophia Antipolis, France (e-mail: ezio.malis@sophia.inria.fr; patrick.rives@sophia.inria.fr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TRO.2008.2007941

initialization [20], [21] and relocalization [22], [23], that are required in a complete visual SLAM solution. The focus of this paper is on the tracking aspect, how to extend tracking to omnidirectional cameras, its relation to motion estimation, and efficient minimization approaches.

The paper will be organized in the following way. We will start by introducing the concept of spherical perspective projection that is adapted to large-field-of-view sensors. We will then detail the geometric transformation considered in this paper: planar homographies. Homographies have the advantage of encompassing all perspective planar deformations, and thus, enable to track over long sequences and avoid drift. Changes in illumination will not be considered for clarity. Affine photometric models can be introduced without changing the underlying results using, for example, [10]. More advanced models, taking into account, for example, specularities, are still objectives of research and not the topic of this paper. Section III will discuss the problem of minimization and how to obtain second-order convergence and apply it to single-plane tracking (SPT) and multiplane tracking (MPT) with single-viewpoint sensors (thus obtaining the pose of the camera and the positions of the 3-D planes). Simulated and real data experiments validate the proposed algorithms and confirm the advantages compared to standard algorithms such as the inverse compositional (IC) [24] or hyperplane approximation (HA) [25]. The algorithm was evaluated on the motion estimation of a mobile robot, and the results are compared with the precise odometry considered as ground truth.

II. Projection Models and Geometric Transformations

Single-viewpoint omnidirectional sensors are perspective sensors, and therefore, all standard properties such as planar homographies stay valid as we will illustrate in this section. The standard projection model needs to be adapted, however, to allow points to be "behind" the camera; this will be done through the use of the spherical perspective projection.

A. Perspective Projection Models

1) Planar Perspective Projection: The standard projection model used for perspective cameras can be summarized in the following steps (Fig. 1).

1) A 3-D point $(\boldsymbol{\mathcal{P}})_{\mathcal{F}_c} = (X, Y, Z)$ in the camera reference frame is projected to the *normalized* plane π_m

$$(\boldsymbol{\mathcal{P}})_{\mathcal{F}_c} \longrightarrow \mathbf{m} = (x, y, 1) = \left(\frac{X}{Z}, \frac{Y}{Z}, 1\right).$$

2) With $\mathbf{K}$ being the matrix of intrinsic parameters [26], the projection of $\mathbf{m}$ in homogeneous coordinates to the image plane π_p is obtained linearly by

$$\mathbf{p} = (u, v, 1) = \mathbf{K}\mathbf{m}.$$

For a field of view greater than 180°, this model is not adapted. With a unique plane, there is an ambiguity between the front and the back of the camera that makes metric reconstruction impossible.

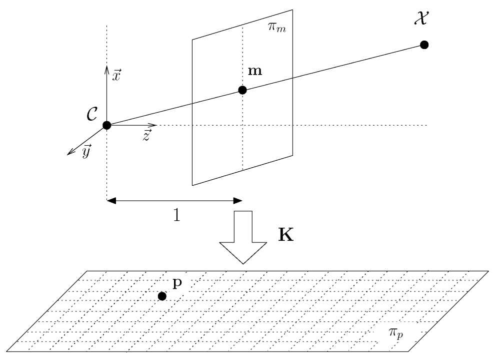


Fig. 1. Planar perspective projection.

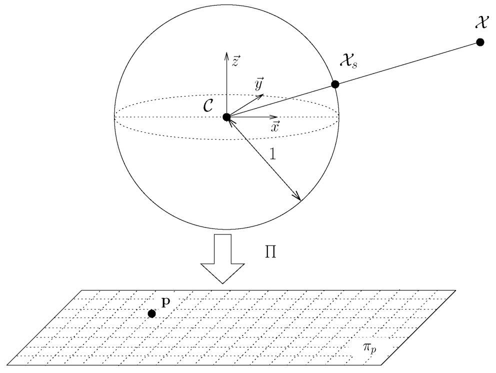


Fig. 2. Spherical perspective projection.

2) Spherical Perspective Projection: Instead of projecting the points to the unit plane π_m, we can project the points to the unit sphere $\mathbb{S}^2 = \{\boldsymbol{\mathcal{S}} \in \mathbb{R}^3 | \|\boldsymbol{\mathcal{S}}\| = 1\}$ (see Fig. 2). We will denote by $\boldsymbol{\mathcal{S}}$ the points on $\mathbb{S}^2$. The cheirality constraint[1] can be expressed in the following way: Then scale factor λ relating points on the sphere to the 3-D points must be positive

$$\exists \boldsymbol{\mathcal{S}} \in \mathbb{S}^2 \implies \exists \lambda > 0 | \boldsymbol{\mathcal{P}} = \lambda \boldsymbol{\mathcal{S}}. \tag{1}$$

From the unit sphere, we can then apply a projection function $\Pi : \Upsilon \subsetneq \mathbb{S}^2 \to \Omega \subset \mathbb{R}^2$ that depends on the intrinsic parameters of the sensor.

An important observation is that for a single-viewpoint sensor, Π is a bijective function defined over a subset of $\mathbb{S}^2$ ($\mathbb{S}^2$ and $\mathbb{R}^2$ do not share the same topology). Π^{-1} will relate points from the image plane to their projective rays (lifting).

In this paper, the projection model used for the omnidirectional sensors is the model proposed in [27] based on the work by Geyer and Daniilidis [28] and Barreto and Araujo [29]. It has the advantage of being valid for a large class of sensors: central catadioptric sensors, fish-eye lenses, and spherical mirrors.

B. Homographies for the Spherical Perspective Projection Model

Let $\mathbf{R} \in \mathbb{SO}(3)$ be the rotation of the camera and $\mathbf{t} \in \mathbb{R}^3$ its translation. The standard planar homography matrix $\mathbf{H}$ is defined up to a scale factor

$$\mathbf{H} \sim \mathbf{R} + \mathbf{t}\mathbf{n}_d^{*\top} \tag{2}$$

[1]Constraint that the scene points are in front of the camera.

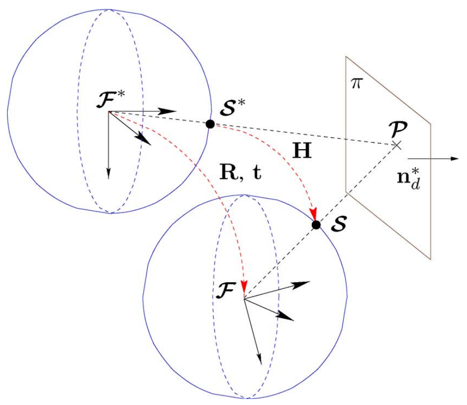


Fig. 3. Planar homography for the spherical perspective projection.

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. In the following sections, we will call $\mathbf{n}_d^*$ the plane normal by "abuse of language." The transformation induced by a homography stays valid for all single-viewpoint sensors. Fig. 3 illustrates the transformation induced by a planar homography using the spherical perspective projection model. The points $\mathcal{S}^*$ and $\mathcal{S}$ are related by

$$\exists(\rho, \rho^*) \in \mathbb{R}^2, \qquad \mathcal{P} = \rho\mathcal{S} = \rho^*\mathbf{H}\mathcal{S}^*. \tag{3}$$

A homography is defined up to a scale factor. In order to fix the scale, we choose $\det(\mathbf{H}) = 1$, i.e., $\mathbf{H} \in \mathbb{SL}(3)$ (the special linear group of dimension 3). Indeed, $\det(\mathbf{H}) = 0$ happens only if the observed plane passes through the optical center of the camera and is then no longer visible.

From a homography, it is possible to extract the transformation and plane normal. However, two solutions are obtained that explain why we have to distinguish the tracking of a single plane (through a unique homography) and tracking multiple planes.

C. Warping

We will call *warping*, denoted by $\mathbf{w} : \mathbb{SL}(3) \times \mathbb{S}^2 \longrightarrow \mathbb{S}^2$, a coordinate transformation induced by a homography $\mathbf{H} \in \mathbb{SL}(3)$ between points on a sphere: this group transformation has the following properties.

1) $\mathbf{w}\langle\mathbf{I}\rangle\langle\mathcal{S}\rangle$ is the identity map: $\forall\mathcal{S} \in \mathbb{S}^2, \mathbf{w}\langle\mathbf{I}\rangle\langle\mathcal{S}\rangle = \mathcal{S}$.
2) The composition of two actions corresponds to the action of the composition

$$\forall\mathcal{S} \in \mathbb{S}^2, \qquad \forall(\mathbf{H}_1, \mathbf{H}_2) \in \mathbb{SL}(3)^2$$
$$\mathbf{w}\langle\mathbf{H}_1\rangle\langle\mathbf{w}\langle\mathbf{H}_2\rangle\langle\mathcal{S}\rangle\rangle = \mathbf{w}\langle\mathbf{H}_1\mathbf{H}_2\rangle\langle\mathcal{S}\rangle.$$

Practically, warping consists of finding the intensity of the transformation of an image point in a new view. Due to discretization, we will have to calculate an *approximate* intensity in the new position. For omnidirectional sensors, the intensity at a given point depends on the solid angle formed by a pixel and the interpolation should take into account the geodesic distance on the unit sphere. However, what is important is the relative distance that will change only slightly due to distortion as the calculation is *local*. In this paper, the image warpings were done with a bilinear transformation taken in the image.

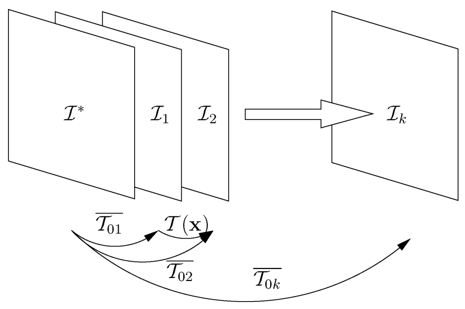

$\mathcal{I}^*$
$\mathcal{I}_1$
$\mathcal{I}_2$
$\mathcal{I}_k$
$\overline{\mathcal{T}}_{01}$
$\mathcal{T}(\mathbf{x})$
$\overline{\mathcal{T}}_{02}$
$\overline{\mathcal{T}}_{0k}$

Fig. 4. Incremental calculation of the transformation.

III. Homography-Based Visual Tracking

A. Incremental Tracking and Motion Estimation

Let $\mathcal{I}^*$ be the reference image. We will call reference template, denoted by $\mathcal{R}$, a region $\mathcal{I}^*$ corresponding to the projection of a planar region of the scene. Let $\mathcal{I}_t, t = 1, \ldots, T$, be a sequence of images. Tracking corresponds to finding the projection of the pixels $\mathbf{p} \in \mathcal{R}$ in the sequence of images.

The estimation of a generic transformation $\mathcal{T}$ in a image sequence is done incrementally, the previous calculated values being used as initial estimates for the following transformations: $\widehat{\mathcal{T}}_{0(i+1)} = \overline{\mathcal{T}}_{0i}$. If convergence is obtained at each step, we obtain the optimal transformation $\overline{\mathcal{T}}$ between the first and the last view without drift, as illustrated in Fig. 4. $\mathcal{T}(\mathbf{x})$ corresponds to the increment that is estimated at each step.

The projection of a point belonging to a planar region follows a planar homography $\mathbf{H}$. Thus, if the only transformation that has occurred to the template is geometric (as is assumed in this paper), tracking at time t will correspond to finding the optimal homography $\overline{\mathbf{H}}$ such that

$$\forall\mathbf{p} \in \mathcal{R}, \qquad \mathcal{I}_t(\mathcal{T}_{\overline{\mathbf{H}}}(\mathbf{p}^*)) = \mathcal{I}^*(\mathbf{p}^*). \tag{4}$$

In the case of multiple planes, we can either track the planes individually or impose the same camera motion as described in Section III-D. We will see that the latter choice leads to more robust motion estimates.

Finding the optimal transformation can be written as an optimization problem over the difference in intensities. If the L_2-norm is chosen, the approach is called SSD tracking, and the minimization is written as

$$\overline{\mathcal{T}}_t = \mathrm{argmin}_{\mathcal{T}_i} \frac{1}{2} \sum_{\mathbf{p}^* \in \mathcal{R}} \|\mathcal{I}_t(\mathcal{T}_i(\mathbf{p}^*)) - \mathcal{I}^*(\mathbf{p}^*)\|^2. \tag{5}$$

B. Efficient Second-Order Minimization

SSD tracking generally relies on iterative methods to minimize the cost function. First-order methods or methods with final superlinear convergence (e.g., Levenberg–Marquardt or dog leg) are generally employed as calculating the Hessian to obtain full quadratic convergence is computationally expensive.

In fact, through the Lie algebra parameterization, we can obtain second-order convergence with a computational cost of the same order as a first-order method; this technique was dubbed ESM [14], [30]. For completeness, we will derive the main steps.

Consider the general least-squares minimization problem

$$F(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{n}(f_i(\mathbf{x}))^2 = \frac{1}{2}\|\mathbf{f}(\mathbf{x})\|^2. \quad (6)$$

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 $[\nabla_\mathbf{x} F]_{\mathbf{x}=\mathbf{x}_0} = \mathbf{0}$, where $\nabla_\mathbf{x}$ is the gradient operator with respect to $\mathbf{x}$. If this equation is nonlinear, a closed-form solution is generally difficult to obtain.

A second-order Taylor series of $\mathbf{f}$ about $\mathbf{x} = \mathbf{0}$ gives

$$\mathbf{f}(\mathbf{x}) = \mathbf{f}(\mathbf{0}) + \mathbf{J}(\mathbf{0})\,\mathbf{x} + \frac{1}{2}\mathbf{M}(\mathbf{0},\mathbf{x})\mathbf{x} + \mathcal{R}(\|\mathbf{x}\|^3) \quad (7)$$

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

$$\mathbf{J}(\mathbf{x}) = \mathbf{J}(\mathbf{0}) + \mathbf{M}(\mathbf{0},\mathbf{x}) + \mathcal{R}(\|\mathbf{x}\|^2). \quad (8)$$

Plugging (8) into (7) leads to

$$\mathbf{f}(\mathbf{x}) = \mathbf{f}(\mathbf{0}) + \frac{1}{2}(\mathbf{J}(\mathbf{0}) + \mathbf{J}(\mathbf{x}))\,\mathbf{x} + \mathcal{R}(\|\mathbf{x}\|^3). \quad (9)$$

As $\mathbf{x}_0 \approx \mathbf{0}$, a second-order approximation of $\mathbf{f}$ in $\mathbf{x}_0$ is

$$\mathbf{f}(\mathbf{x}_0) \approx \mathbf{f}(\mathbf{0}) + \frac{1}{2}(\mathbf{J}(\mathbf{0}) + \mathbf{J}(\mathbf{x}_0))\,\mathbf{x}_0. \quad (10)$$

Under certain conditions, which we will detail in the following sections, $\mathbf{J}(\mathbf{x}_0)\mathbf{x}_0$ can be calculated without knowing the value of $\mathbf{x}_0$. This is the basis of the ESM algorithm.

Let $\mathbf{J}(\mathbf{x}_0)\mathbf{x}_0 = \mathbf{J}'\mathbf{x}_0$, at the solution, $\mathbf{f}(\mathbf{x}_0) = 0$, so our second-order least-square minimizer is the solution to (with $^+$ indicating the pseudoinverse)

$$\mathbf{x}_0 = -\left(\frac{\mathbf{J}(\mathbf{0}) + \mathbf{J}'}{2}\right)^+ \mathbf{f}(\mathbf{0}). \quad (11)$$

For comparison, the first-order minimizers for the *forward compositional* (FC; current Jacobian) [10] and the IC (reference Jacobian) [24] methods, respectively, are

$$\mathbf{x}_0 = -\mathbf{J}(\mathbf{0})^+\mathbf{f}(\mathbf{0}), \qquad \mathbf{x}_0 = -\mathbf{J}'^+\mathbf{f}(\mathbf{0}). \quad (12)$$

An alternative formulation proposed by Jurie and Dhome [25], HA, consists of finding a matrix $\mathbf{J}^\oplus$ that plays the same role as the pseudoinverse but is obtained by learning the linear relationship between the incremental motion $\mathbf{x}_0$ and $\mathbf{f}(\mathbf{0})$.

C. Tracking a Single Plane

Consider the following diagram, which is illustrated by Fig. 3:

$$\mathbf{p}^* \xrightarrow{\Pi^{-1}} \boldsymbol{\mathcal{S}}^* \xrightarrow{\mathbf{w}\langle\mathbf{H}\rangle\langle\cdot\rangle} \boldsymbol{\mathcal{S}} \xrightarrow{\Pi} \mathbf{p}. \quad (13)$$

To track the template in the current image $\mathcal{I}$ is to find the transformation $\overline{\mathbf{H}} \in \mathbb{SL}(3)$ that warps the lifting of that region to the lifting of the reference template of $\mathcal{I}^*$, i.e., find $\overline{\mathbf{H}}$ with

$$\forall i, \qquad \mathcal{I}\left(\Pi\left(\mathbf{w}\langle\overline{\mathbf{H}}\rangle\langle\boldsymbol{\mathcal{S}}_i^*\rangle\right)\right) = \mathcal{I}^*(\mathbf{p}_i^*). \quad (14)$$

Knowing an approximation $\widehat{\mathbf{H}}$ of the transformation $\overline{\mathbf{H}}$, the problem is to find the incremental transformation $\mathbf{H}(\mathbf{x})$ that minimizes the SSD

$$\begin{cases} F(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{q}\|\mathbf{f}_i\|^2 \\ \mathbf{f}_i = \mathcal{I}\big(\Pi\big(\mathbf{w}\langle\widehat{\mathbf{H}}\mathbf{H}(\mathbf{x})\rangle\langle\boldsymbol{\mathcal{S}}_i^*\rangle\big)\big) - \mathcal{I}^*(\mathbf{p}_i^*) \end{cases} \quad (15)$$

with $\mathbf{H}(\mathbf{x})$ parameterized locally by the Lie algebra of $\mathbb{SL}(3)$, with $\mathbf{A}_i, i = 1,\ldots,8$, generators of the algebra (further details are given in Appendix II)

$$\mathbf{H}(\mathbf{x}) = \exp\left(\sum_{i=1}^{8} x_i\mathbf{A}_i\right). \quad (16)$$

Surprisingly, this parameterization is not standard [31] even though it has the advantage of ensuring that at each iteration, the homography belongs to $\mathbb{SL}(3)$ and that we use the minimal amount of parameters.

The current Jacobian $\mathbf{J}(\mathbf{0})$ and the reference Jacobian $\mathbf{J}(\mathbf{x}_0)$ can be written as the product of four Jacobians. The complete expressions can be found in [32] (similar to Appendix IV)

$$\mathbf{J}(\mathbf{0}) = \mathbf{J}_\mathcal{I}\mathbf{J}_\Pi\mathbf{J}_w\mathbf{J}_{\mathbf{Hx}}(\mathbf{0}) \quad (17)$$

$$\mathbf{J}(\mathbf{x}_0) = \mathbf{J}_{\mathcal{I}^*}\mathbf{J}_\Pi\mathbf{J}_w\mathbf{J}_{(\overline{\mathbf{H}}^{-1}\widehat{\mathbf{H}}\mathbf{Hx})}(\mathbf{x}_0). \quad (18)$$

Using Proposition 1 in Appendix III, we obtain the property

$$\mathbf{J}_{(\overline{\mathbf{H}}^{-1}\widehat{\mathbf{H}}\mathbf{Hx})}(\mathbf{x}_0)\mathbf{x}_0 = \mathbf{J}_{\mathbf{Hx}}(\mathbf{0})\mathbf{x}_0. \quad (19)$$

The scalar exponential function can help gain an insight into this property

$$\forall t, \qquad \exp(-t_0)\exp(t) = \exp(-t_0 + t). \quad (20)$$

By a change of variable, we obtain

$$[\nabla_t \exp(-t_0 + t)]_{t=t_0} = [\nabla_t \exp(t)]_{t=0}$$

For the matrix exponential, the property (20) is not valid in general (noncommutativity); however, the unknown value $\mathbf{x}_0$ defines a subgroup in which the property holds.

Thus, in (10), we can use $\mathbf{J}_{\mathbf{Hx}}(\mathbf{0})\mathbf{x}_0$ instead of $\mathbf{J}_{(\overline{\mathbf{H}}^{-1}\widehat{\mathbf{H}}\mathbf{Hx})}(\mathbf{x}_0)\mathbf{x}_0$, where $\mathbf{J}_{\mathbf{Hx}}(\mathbf{0})$ is a constant Jacobian matrix. The update $\mathbf{x}_0$ of the second-order minimization algorithm can be computed as follows:

$$\mathbf{x}_0 = -\left(\left(\frac{\mathbf{J}_\mathcal{I} + \mathbf{J}_{\mathcal{I}^*}}{2}\right)\mathbf{J}_\Pi\mathbf{J}_w\mathbf{J}_{\mathbf{Hx}}(\mathbf{0})\right)^+ \mathbf{f}(\mathbf{0}). \quad (21)$$

The computational complexity is almost the same as for first-order algorithms as the reference Jacobian $\mathbf{J}_{\mathcal{I}^*}$ needs to be calculated only once.

Obtaining $\mathbf{J}_{\mathcal{I}^*}$ and $\mathbf{J}_\mathcal{I}$, which are the Jacobians taken in the images (for example, using a Sobel filter), is a remarkable property. It indicates that we can take into account the nonlinear properties of the sensor simply through the Jacobian of the projection function $\mathbf{J}_\Pi$ (this appears in the derivation of the Jacobians in Appendix IV).

D. Tracking Multiple Planes

When tracking multiple planes, we have the choice either to track the planes independently or to constrain the same motion for each plane, which is the objective of this section.

From (2), we can parameterize a homography by a transformation $\mathbf{T} \in \mathbb{SE}(3)$ and a plane normal $\mathbf{n}_d \in \mathbb{R}^3$. With the Lie algebra parameterization of $\mathbf{T}$

$$\mathbf{H}(\mathbf{T}(\mathbf{x}), \mathbf{n}_d) = \mathbf{H}\left(\exp\left(\sum_{i=1}^{6} x_i \mathbf{A}_i\right), \mathbf{n}_d\right). \tag{22}$$

We can now reformulate the problem by parameterizing each plane with the same transformation $\mathbf{T}$. To track the template j in the current image $\mathcal{I}$ is to find the transformation $\mathbf{H}(\overline{\mathbf{T}}, \overline{\mathbf{n}}_d^j)$ that warps the lifting of that region to the lifting of the reference template of $\mathcal{I}^*$

$$\forall i, j, \qquad \mathcal{I}\big(\Pi\big(\mathbf{w}\langle \mathbf{H}(\overline{\mathbf{T}}, \overline{\mathbf{n}}_d^j)\rangle\langle \boldsymbol{\mathcal{S}}_{ij}^*\rangle\big)\big) = \mathcal{I}^*(\mathbf{p}_{ij}^*). \tag{23}$$

In other words, knowing an approximation $\widehat{\mathbf{T}}$ of $\overline{\mathbf{T}}$ and $\widehat{\mathbf{n}}_d^j$ of $\overline{\mathbf{n}}_d^j$, the problem is to find the incremental transformations $\mathbf{T}(\mathbf{x})$ and $\mathbf{n}_d^j(\mathbf{x})$ that minimize the SSD over all the pixels and over the l planes ($\mathbf{x}$ contains the six transformation parameters and the $3 \times l$ parameters for the normals and depths)

$$\begin{cases} F(\mathbf{x}) = \frac{1}{2}\sum_{j=1}^{l}\sum_{i=1}^{q_j} \|\mathbf{f}_{ij}\|^2 \\ \mathbf{f}_{ij} \quad = \mathcal{I}\big(\Pi\big(\mathbf{w}\langle \mathbf{H}(\mathbf{T}(\mathbf{x})\widehat{\mathbf{T}}, \widehat{\mathbf{n}}_d^j + \mathbf{n}_d^j(\mathbf{x}))\rangle\langle \boldsymbol{\mathcal{S}}_{ij}^*\rangle\big)\big) \\ \qquad\quad -\mathcal{I}^*(\mathbf{p}_{ij}^*). \end{cases} \tag{24}$$

The minimal number of parameters in (24) is, in fact, $6 + 3 \times l - 1$ because the first homography has only eight degrees of freedom. However, the extra degree of freedom empirically gave better results probably due to the better conditioning of the Jacobian (all the values for the normals have the same amplitude).

Similarly to the previous case, the Jacobians $\mathbf{J}(\mathbf{0})$ and $\mathbf{J}(\mathbf{x}_0)$, which, respectively, correspond to the current and the reference Jacobians, can be written as (see Appendix IV)

$$\mathbf{J}(\mathbf{0}) = \mathbf{J}_{\mathcal{I}}\, \mathbf{J}_{\Pi}\, [\mathbf{J}_{H_T} \mathbf{J}_T(\mathbf{0}) \quad \mathbf{J}_{H_n} \mathbf{J}_n(\mathbf{0})] \tag{25}$$

$$\mathbf{J}(\mathbf{x}_0) = \mathbf{J}_{\mathcal{I}^*}\, \mathbf{J}_{\Pi}\, [\mathbf{J}_{H_T^*} \mathbf{J}_{T^*}(\mathbf{x}_0) \quad \mathbf{J}_{H_n^*} \mathbf{J}_{n^*}(\mathbf{x}_0)] \tag{26}$$

with H_T being the homography seen as a function of the transformation $\mathbf{T}$ and H_n the homography seen as a function of the plane normal $\mathbf{n}_d$.

Using Proposition 1, $\mathbf{J}_{T^*}(\mathbf{x}_0)\mathbf{x}_0 = \mathbf{J}_T(\mathbf{0})\mathbf{x}_0$ and $\mathbf{J}_{n^*}(\mathbf{x}_0)\mathbf{x}_0 = \mathbf{J}_n(\mathbf{0})\mathbf{x}_0$. If we assume that $\widehat{\mathbf{T}} \approx \overline{\mathbf{T}}$ and $\widehat{\mathbf{n}} \approx \overline{\mathbf{n}}$, $\mathbf{J}_{H_T^*} \approx \mathbf{J}_{H_T}$, and $\mathbf{J}_{H_n^*} \approx \mathbf{J}_{H_n}$, the update $\mathbf{x}_0$ can be computed as follows:

$$\mathbf{x}_0 = -\left(\left(\frac{\mathbf{J}_{\mathcal{I}} + \mathbf{J}_{\mathcal{I}^*}}{2}\right)\mathbf{J}_{\Pi}\, [\mathbf{J}_{H_T T}(\mathbf{0}) \quad \mathbf{J}_{H_n n}(\mathbf{0})]\right)^{+} \mathbf{f}(\mathbf{0}). \tag{27}$$

The inversion of this matrix can be done efficiently by exploiting its sparse structure, as discussed in detail in Appendix I.

E. Efficient ESM Implementation and Variants

In this section, we will detail how to improve the computational cost of the ESM and stay close to the second-order convergence rate.

Algorithm 1 details a basic iterative descent method for tracking a single plane. The computation is split between evaluation of the cost function and calculation of the descent direction.

Algorithm 1: ESM tracking method: basic algorithm for a single plane

```
Data: Current image $\mathcal{I}$ and reference image $\mathcal{I}^*$.
      $(\varepsilon, k_{max})$ : thresholds, $\widehat{\mathbf{H}}$ : initial estimate
Result: Local minimum $\overline{\mathbf{H}}$
Calculate $\mathbf{J}_{\mathcal{I}^*}$ (eg. Prewitt, Sobel). k := 0; found := false
while (not found) and ($k++ < k_{max}$) do
    Calculate $\mathbf{J}_{\mathcal{I}}$. Calculate $\mathbf{J}_{esm}$ from equation (21).
    $\mathbf{x}_0 = -(\mathbf{J}_{esm}^\top \mathbf{J}_{esm})^{-1}\mathbf{J}_{esm}^\top \mathbf{f}(\mathbf{0})$
    $\widehat{\mathbf{H}} \leftarrow \widehat{\mathbf{H}} e^{A(\mathbf{x}_0)}$
    if $\|\mathbf{x}_0\| < \varepsilon$ then
        found := true
    end
end
```

Let q be the number of pixels of the template (several hundreds or several thousands) and m the number of parameters.The authors in [33] detail the cost in terms of operations of the different steps when tracking. The pseudoinversion and, more specifically, the product $\mathbf{J}_{\mathrm{ESM}}^\top \mathbf{J}_{\mathrm{ESM}}$, is the most expensive step and requires $o(m^2 q)$ operations. Calculating the cost function in comparison has a complexity of $o(mq)$.

The idea behind the IC algorithm is to reduce the complexity to $o(mq + m^3)$ by precalculating the pseudoinverse $(\mathbf{J}_{\mathrm{inv}}^\top \mathbf{J}_{\mathrm{inv}})^{-1}\mathbf{J}_{\mathrm{inv}}^\top$, with $\mathbf{J}_{\mathrm{inv}}$ being the *reference* Jacobian. We may note that changes in the cost function, for example, due to occlusion (including the template partly going out of the image) cannot be handled without needing to recalculate $\mathbf{J}_{\mathrm{inv}}^+$. It should also not be used when estimating the motion directly *even* when the structure is known as the reference Jacobian that depends on a *combination* of the displacement and structure. Equation (41) shows the dependency of every element of the Jacobian on motion *and* structure. We simulate possible effects of using an incorrect constant Jacobian in Section IV-A1.

Similarly to the IC, we can consider alternative increments reducing the complexity to $o(mq + m^3)$

1) *αESM*:

$$\mathbf{x}_0 = -\frac{(\mathbf{g}^\top \mathbf{J}_{\mathrm{inv}}^p \mathbf{g})}{\|\mathbf{J}_{\mathrm{ESM}} \mathbf{J}_{\mathrm{inv}}^p \mathbf{g}\|^2}\mathbf{J}_{\mathrm{inv}}^p \mathbf{g} \tag{28}$$

with $\mathbf{g} = \mathbf{J}_{\mathrm{ESM}}^\top \mathbf{f}(\mathbf{0})$ and $\mathbf{J}_{\mathrm{inv}}^p = (\mathbf{J}_{\mathrm{inv}}^\top \mathbf{J}_{\mathrm{inv}})^{-1}$. The corrective term was found in the following way.

a) The parameter $(-\mathbf{g})$ can be considered as a second-order steepest descent direction.

b) We then look for the corrective term α that minimizes the second-order approximation to the cost function

$$\begin{aligned} F(-\alpha \mathbf{J}_{\mathrm{inv}}^p \mathbf{g}) &= \frac{1}{2}\|\mathbf{f}(\mathbf{0}) - \alpha \mathbf{J}_{\mathrm{ESM}} \mathbf{J}_{\mathrm{inv}}^p \mathbf{g}\|^2 \\ &= F(\mathbf{0}) - \alpha \mathbf{f}(\mathbf{0})^\top \mathbf{J}_{\mathrm{ESM}} \mathbf{J}_{\mathrm{inv}}^p \mathbf{g} \\ &\quad - \frac{1}{2}\alpha^2 \|\mathbf{J}_{\mathrm{ESM}} \mathbf{J}_{\mathrm{inv}}^p \mathbf{g}\|^2. \end{aligned}$$

By differentiation, we obtain the optimal α and the desired expression.

2) *iESM*:

$$\mathbf{x}_0 = -\frac{(\mathbf{g}^\top \mathbf{J}^p_{\mathrm{ESM}}\mathbf{g})}{\|\mathbf{J}_{\mathrm{ESM}}\mathbf{J}^p_{\mathrm{ESM}}\mathbf{g}\|^2}\mathbf{J}^p_{\mathrm{ESM}}\mathbf{g} \tag{29}$$

with $\mathbf{g} = \mathbf{J}^\top_{\mathrm{ESM}}\mathbf{f}(\mathbf{0})$ and $\mathbf{J}^p_{\mathrm{ESM}} = (\mathbf{J}^\top_{\mathrm{ESM}}\mathbf{J}_{\mathrm{ESM}})^{-1}$ calculated at the beginning of the iterations. This approach is valid for explicit structure and motion. It can be justified by saying that at the beginning, we have the best second-order estimate, and then, we correct the Jacobian so that it stays valid at the optimum. It is not as satisfying as the previous methods since the Jacobian will *not* be correct at the optimum. However, if it is sufficiently good, it will lead to the optimum anyway.

Reducing the computational cost of the iterations is only one side of the problem. Ideally, we would also want to diminish the number of iterations. The ESM algorithm, as a second-order method, converges faster and more often than a first-order method.

In Section IV, the different algorithms will be compared with the effect of the approximations on the convergence.

F. Outlier Rejection

Outlier rejection is an important part of visual tracking when dealing with sequences in uncontrolled environments. Specularities or occlusions are common sources of tracking failure.

A standard technique to improve least-square estimates is to use a robust cost function (M-estimators) such as Tukey or Huber (see [10] and [26, App.]). The underlying idea is that the errors should follow a given profile (e.g., Gaussian distribution) and that all the points that do not verify the distribution are considered as outliers. Outliers are given lower weights so as to avoid influencing the results. Applying this technique directly as in [10] means we are considering each pixel as an independent value. However, the error is generally spatially correlated. This observation was used in [34] to devise a robust approach to tracking. The idea is to split the image in blocks and calculate an error that takes into account the standard deviation of the initial block (errors are expected to be higher in texture-rich regions). The authors use the following error for a block B_i:

$$\mathbf{e}_i = \frac{F(\mathbf{x})_{\mathbf{p}\in B_i}}{\sigma[\mathcal{I}^*(\mathbf{p})]_{\mathbf{p}\in B_i}}. \tag{30}$$

They then order the errors and keep a predefined percentage of blocks. Choosing a predefined value is not satisfactory as when there are few outliers, useful information will be discarded, and if there is a lot of noise, outliers will be kept. We propose to simply use robust statistics on the errors in each pixel weighted by the standard deviation of a rectangular region around the pixel. We found that this approach gave stable results over quite a wide range of rectangle sizes (5×5 to 15×15), and results were better than the per-pixel method, as we will see in Section IV.

IV. Experimental Validation

In this section, we will make a distinction between 2-D tracking, using, for example, homography estimation (as described in Section III-C), and 3-D tracking, where the camera motion and the plane normals are explicitly estimated (as described in Section III-D).

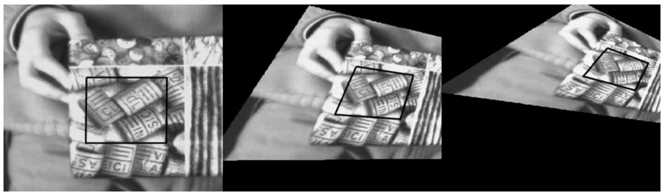


Fig. 5. Images 1, 50, and 100 of the simulated sequence.

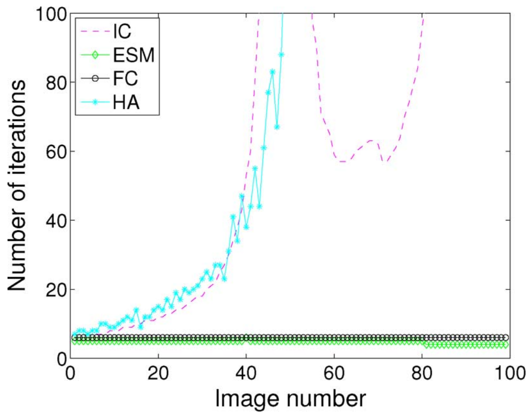


Fig. 6. Comparison of the number of iterations for the simulation sequence.

A. Simulation

1) Applying the IC To Explicit Motion Estimation (3-D Tracking): As explained previously, the reference Jacobian for multiple planes depends not only on the structure but also on the camera pose. Using the IC or HA algorithm with a constant Jacobian, in this case, can lead to poor results as the Jacobian can have arbitrarily large errors. Alas, the IC has often been used to track in this situation. We will now simulate possible effects of using the IC and HA in this situation.

Our simulation setup consists of a sequence of 100 images without any added noise and with small interframe displacements. Fig. 5 shows images 1, 50, and 100, respectively, of the simulated sequence. The correct plane normal and distance (i.e., the structure) were given as input to the different algorithms. In the case of the HA, a linear relationship was learnt between the Lie representations of small random $\mathbb{SE}(3)$ transformations (generating a homography through the given correct normal) and the difference between the reference image and the warped image.

Fig. 6 shows the number of iterations taken to minimize and Fig. 7 shows the reprojection error after minimization for each image of the sequence. As is evident from the graph, if the Jacobians are not recalculated, the IC and HA take more and more iterations. At the end of the sequence, the IC and HA took more than 500 iterations to converge. The FC took systematically six iterations and the ESM took five iterations at the same computational cost.

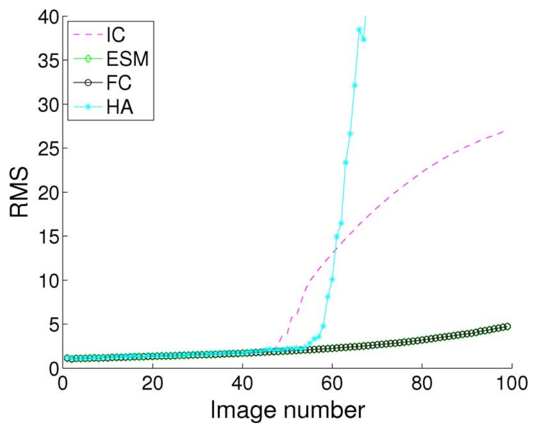


Fig. 7. Comparison of the reprojection error for the simulation sequence.

This experiment confirms that without recalculating the IC Jacobian (by explicit calculation or through learning)—which is the main advantage of using an IC formulation—IC and HA can lead to incorrect results for 3-D tracking *even* when the structure is known and the interframe displacement is small. This result is particularly important for omnidirectional visual tracking as we expect to track templates over larger image regions than for small-field-of-view cameras.

These remarks do not, of course, apply to 2-D tracking (such as [25]) where the Jacobian is exactly constant (this comes from the subgroup property explained in Appendix III). The authors in [19] reestimate the Jacobian for the 3-D tracking with IC, thus removing the advantage of this formulation of the problem. It is unclear if this is the case in [16].

2) Comparison Between Methods: The Matlab program written at the Carnegie Mellon University for the project "Lucas–Kanade 20 Years On" was used to compare the convergence of the different algorithms.[2]

The following methods were compared: FC [10], IC [24], HA [25], ESM, αESM, and iESM, as described previously.

In the following, the *spatial noise* corresponds to a Gaussian reprojection error in the image of the corners of the tracked plane. The *image noise* is an independent and identically distributed Gaussian noise added to the image intensities (reference or warped image as specified by the context). The criterion chosen for the convergence of an algorithm is the rms reprojection error of the image corners. An algorithm is said to have converged if the reprojection rms of the plane corners is less than 1.

HA is different from the other algorithms in the sense that it requires a learning step to evaluate a matrix $\mathbf{J}^{\oplus}$ that plays the same role as the pseudoinverse in (11). In this evaluation, we learnt $\mathbf{J}^{\oplus}$ for every four pixels in the image and applied 20 random motions for each pixels (in other words, $20(100/4)^2 = 12\,500$ random homography transforms) with a spatial variance of four pixels. For $\mathbf{J}^{\oplus}$ to be valid in the presence of noise, we also found it essential to learn in the presence of noise. In this case, we learnt with a Gaussian random intensity noise added

[2]The Matlab source code can be downloaded from http://www.ri.cmu.edu/projects/project_515.html.

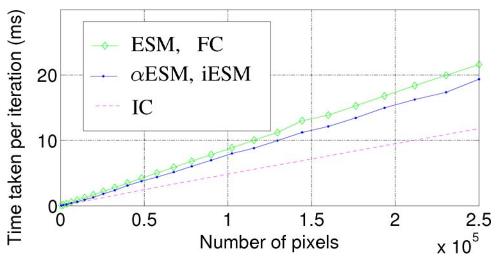


Fig. 8. Iteration time (in milliseconds) versus the number of pixels.

TABLE I
ESTIMATED TIMES FOR AN ITERATION AND NUMBER OF ITERATIONS

	Time taken per iteration	Number of iterations given "at infinity"	Number of iterations given at "fixed time"
IC, HA	1	150	20
αESM, iESM	1.33	112	15
ESM, FC	1.8	83	11

to the warped template of six pixels (the same amount of noise used for the evaluation). For a fair comparison, only every four pixels were used for *each* of the algorithms.

a) Evaluating the computational time per iteration: To evaluate the time taken for an iteration, we programmed the iteration steps in C language and tested the times for image sizes ranging from 20×20 to 500×500. The IC has the lowest computational cost and was compared to the other methods. In Fig. 8, we plot the iteration time versus the number of pixels. The ratios are reported in Table I. These results require comment. Why does the computational time increase *linearly* with the template size? Why is the difference between the IC and the other methods not greater?

The linear increase in computational time comes from the use of continuous blocks of memory that enables the instruction pointer (IP) to predict the next load by a constant stride calculation (here, a "perfect" stride access pattern). The IP prefetcher then generates a prefetch request and loads the memory in the L1 cache. (This property can be checked by making the same calculations but this time accessing the memory at random. The computational time is then no longer linear with template size.)

The explanation concerning the computational time requires an explanation on the implementation to reduce memory access. A naive implementation would start by calculating $\mathbf{J}_{\mathrm{ESM}}$ (a matrix of size mp) and then $\mathbf{J}^{+}_{\mathrm{ESM}}\mathbf{f}(\mathbf{0})$ in two separate iterations. However, it is not necessary to calculate $\mathbf{J}_{\mathrm{ESM}}$ explicitly. Instead, it is possible to build $\mathbf{J}^{\top}_{\mathrm{ESM}}\mathbf{J}_{\mathrm{ESM}}$ (a symmetric matrix with $p(p-1)/2$ distinct values) and $\mathbf{J}_{\mathrm{ESM}}\mathbf{f}(\mathbf{0})$ (a vector of size p) in one loop over the image pixels with one access to the reference and current images, and their gradients. The small amount of memory required for the computation ensures that the values can be kept cached by the processor for read and write (and avoids the risk of random memory access). This aspect is rarely taken into account when calculating the computational cost of an algorithm: it can take longer to access memory than to recompute part of the data. The naive implementation leads to an algorithm that is about 3 times slower than the IC instead of only 1.8 in this one-pass approach.

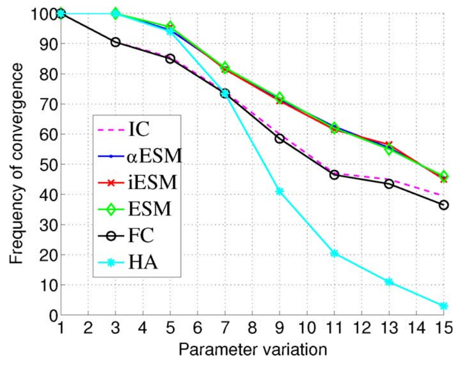


Fig. 9. Frequency of convergence versus homography motion for an "infinite time."

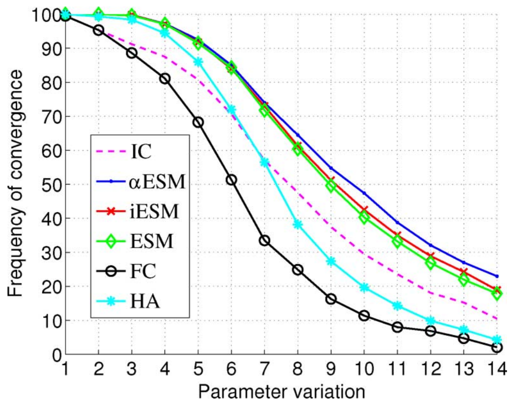


Fig. 10. Frequency of convergence versus homography motion for a "fixed time" with noise.

We believe that this aspect of computation should be taken into account when analyzing the efficiency of computer vision algorithms as precalculating values (and thus making many memory accesses) can have the paradoxical effect of slowing down the algorithm. Only considering the number of operations can lead to incorrect conclusions.

b) Tests: The following tests were undertaken.

1) "At infinity": Each algorithm is given a large amount of iterations (typically 150, see Table I). This is to test the convergence properties.
2) "Fixed time with added noise": An image noise of variance 6 was added to the warped image. This test simulates unmodeled errors that could appear through lighting changes or occlusion. We give a typical amount of iterations (e.g., 20 for the IC; see Table I). The number of iterations are calculated in order to give the same amount of time to each algorithm.

Figs. 9 and 10 show the frequency of convergence of the different algorithms as the homography transformation increases (the x-axis corresponds to the spatial error in pixels). The number of tests for each parameter variation was of 200 "at infinity" and 1000 in the other case.

The test "at infinity" confirms that there are two groups: the algorithms with a second-order convergence (ESM and variants) and the algorithms with first-order convergence properties. This experiment indicates that the ESM variants keep the second-order convergence rate despite the approximations. The HA has a good convergence rate for the range of spatial errors learnt but drops rapidly for stronger motion (and gives worse results than the IC or FC for very strong motion). The values during the learning phase were chosen to obtain a compromise between the rate of convergence and the robustness to noise. Learning stronger motions lead to a drop of the convergence rate for smaller motions.

The test at "fixed time with noise" shows that the two variants αESM and iESM are possible alternatives to the direct ESM approach. The reason iESM has the same convergence rate as αESM is that the calculation of the pseudoinverse at the beginning of the iterations compensates, in part, for the image errors. Again, HA provides a good convergence rate for small motions but this degrades with stronger motion.

3) Conclusion: In the simulation experiments, we compared the convergence properties of the ESM algorithm and different variants to more standard approaches such as the IC and the FC. We showed that even though the iteration step of the ESM (and variants) is computationally more expensive than an iteration of the IC, the higher convergence rate makes it globally faster. If we add the possibility of working directly on occlusion (without having to calculate constant Jacobians by block such as in [34]), the algorithm is altogether a good alternative to the IC for image-based tracking. Variants of the ESM, αESM and iESM, were shown to improve the results further. An experiment also confirmed that the inverse compositional (IC and HA) with a constant Jacobian can lead to poor convergence for 3-D tracking even if the structure is known.

B. Real Data

The algorithm was tested on real data obtained from a mobile robot. The central catadioptric camera comprised the S80 parabolic mirror from RemoteReality with a telecentric lens and a perspective camera of resolution 1024 $\times$ 768, and calibrated using the open-source toolbox described in [27].

The experimental sequence consisted of 120 gray-level images taken with a fixed shutter speed and gain set to 0 over a distance of about 2 m. The sequence was taken so as to minimize slippage and enable to consider the robot odometry as ground truth. These experiments were undertaken to analyze the precision of the algorithm for motion estimation. Another aspect that needed to be confirmed is the advantage of imposing the same camera motion when tracking different planes.

Fig. 11(a) shows the templates chosen by hand and tracked in the experiment. The planes are numbered in counterclockwise order from 1 to 3 starting from the top left. To fix the scale factor, we measured the distance from the camera to plane 3 (0.5 m) (the plane that proved the stablest while tracking).

In the case of SPT, for each homography, a translation $\mathbf{t}$ up to a scale factor and a rotation $\mathbf{R}$ was extracted (the ambiguity was solved by using several frames). The results show the median of the three extracted translations and rotations.

The algorithm for MPT was tested with the ESM (MPT ESM), the ESM with a simple Huber robust approach (MPT ESM HUB) and the proposed approach for dealing with occlusion (MPT

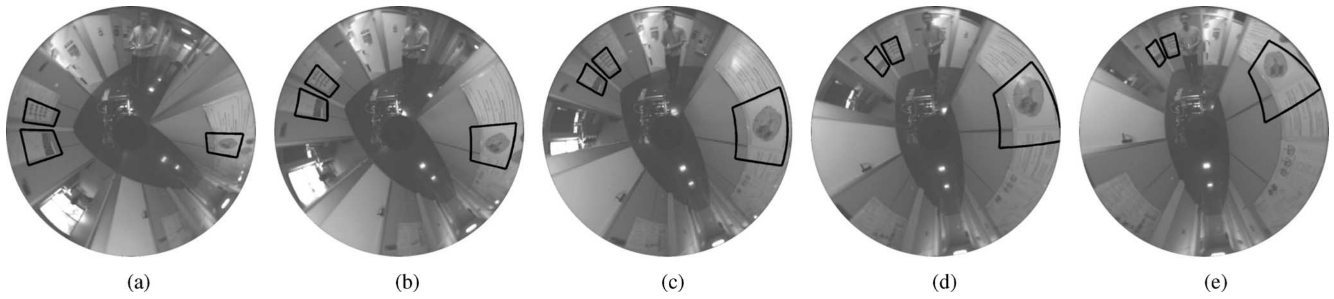


Fig. 11. Tracked templates in the image sequence. (a) Reference image. (b) Image 30. (c) Image 60. (d) Image 90. (e) Image 120.

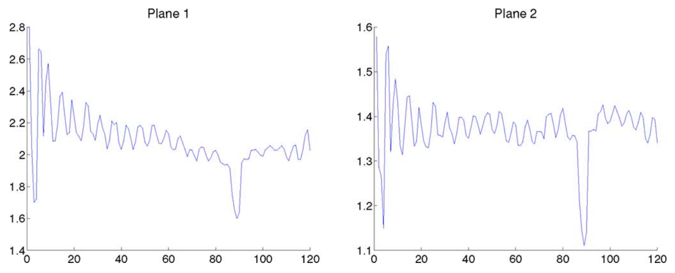


Fig. 12. Estimation of the plane distances for planes 1 and 2 with MPT_ESM_HUB_BL (in meters).

ESM HUB BL). The initial values given for the normals with depths was $[1; 0; 0]$ (the same results were obtained for values $[0; 1; 0]$, $[0; 0; 1]$, and $[0; 0; 1000]$). These initial values are far from the "real" values that can be deduced from Figs. 12 and 13 (with $d_3 = 0.5$ m): $[0.38; 0.31; 0]$, $[0.4; 0.6; 0]$, and $[1.2; 1.6; 0]$. The algorithm proved to be relatively insensitive to the initial values when an extra degree of freedom was given. This can be explained from the normals appearing in the homography as a product with the translation.

Fig. 11(a)–(e) shows the tracked regions during the sequence. Figs. 14(a)–15(d) show the errors obtained by using different methods on the left axis and the number of iterations needed to converge on the right axis with a dash-dot line (up to a maximum value of 40 iterations). The normals estimated online are represented in Fig. 13, and the distances estimated for planes 1 and 2 are detailed in Fig. 12 using the MPT_ESM_HUB_BL algorithm.

The first result that becomes clear is the improvement in terms of precision and robustness obtained by constraining the same motion for each tracked plane. As can be seen in Figs. 14(a) and 15(a), the patches did not have enough information to enable a good estimate of the motion. With a specularity appearing around image 70 on the stablest estimated plane, the motion estimation becomes erratic (Fig. 16). By imposing the same motion, the tracking algorithm does not fail even though the motion estimation still suffers from the nonrobust minimization approach [Figs. 14(b) and 15(b)]. Applying the robust Huber cost function improves the precision and robustness [Figs. 14(c) and 15(c)] but greatly reduces the speed of convergence. What is not visible in the figure is that the normal estimates become incorrect toward the end of the sequence. The proposed approach [Figs. 16, 17, 14(d), and 15(d)] not only gives more precise results but also avoids very slow convergence.

The estimation of the plane distances are quite noisy (Fig. 12). In the first few iterations, this is easily explained by the small baseline. In the subsequent frames, the distance estimate oscillates within a range of about 20 cm for plane 1 and 10 cm for plane 2. Plane 1 does not benefit from an important baseline as can be seen from the trajectory in Fig. 16 and the 3-D reconstruction in Fig. 17, which explains why the distance estimate is quite unstable. Furthermore, fixing the distance of plane 3 increases the effect of noise that is propagated to the estimates of planes 1 and 2.

V. Discussions and Conclusion

Large-field-of-view sensors can improve structure from motion by enabling regions of the image to be tracked over longer periods. The aim of this paper was to enable tracking with these types of sensors by taking into account the strong distortion induced by the nonlinear projections. We also studied the problems of efficiency and the proposed minimization techniques (ESM and variants) adapted to real-time frameworks for image-based and 3-D structure and motion estimation. We showed that given a fixed time, the ESM algorithm has better convergence properties than the IC. Several variants of the ESM with better computational properties were also presented. A simple outlier rejection algorithm was proposed that takes into account the spatial correlation of the error and does not reduce the convergence rate as much as the per-pixel application of M-estimators. This study applied to the motion estimation of a mobile robot shows that it is a valid approach for robotic tasks such as visual servoing or SLAM.

Appendix I

Efficient Solving of the Normal Equations

The Jacobian $\mathbf{J}$ appearing in the estimation of the pose between two views and the position of l planes (observed through q_j intensity values) has a sparse structure as in bundle adjustment [26] that can be used to efficiently solve the normal equations $\mathbf{J}\mathbf{x}_0 = -\mathbf{f}(\mathbf{0})$.

$\mathbf{J}$ has $r = \sum_{j=1}^{l} q_j$ rows for every pixel for every plane. $\mathbf{J}_{r\times(6+3l)} = [\mathbf{T}_{r\times 6}|\mathbf{P}_{r\times 3l}]$ with $\mathbf{T}$ being a dense matrix representing the Jacobian with respect to the transformation for every plane and $\mathbf{P}$ a block-diagonal matrix with l blocks $\mathbf{N}_j$ of q_j rows and three columns representing the Jacobian with respect to the normals

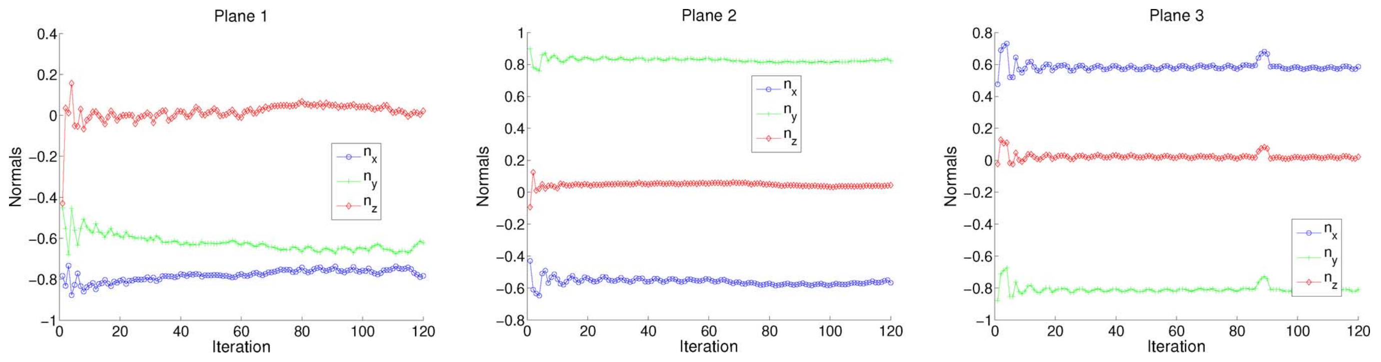


Fig. 13. Normals estimated for planes 1–3 with MPT_ESM_HUB_BL.

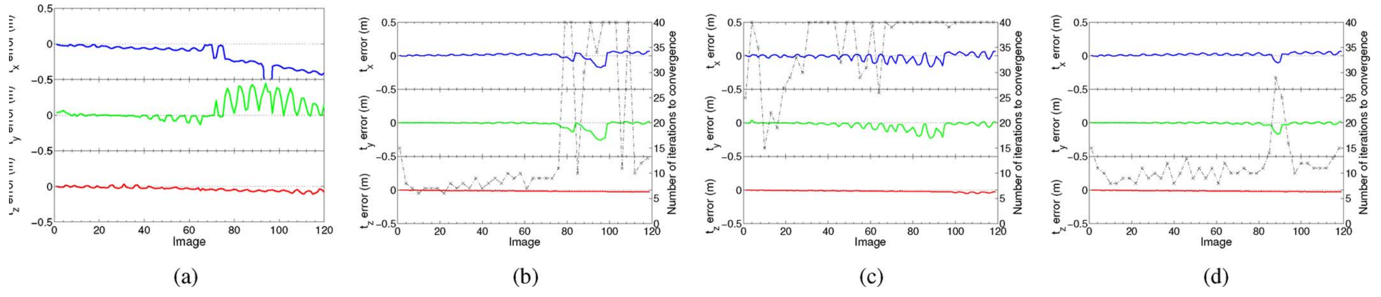


Fig. 14. Translation errors for the different algorithms. (a) Translation error with SPT (in meters). (b) Translation error with MPT_ESM (in meters). (c) Translation error with MPT_ESM_HUB (in meters). (d) Translation error with MPT_ESM_HUB_BL (in meters).

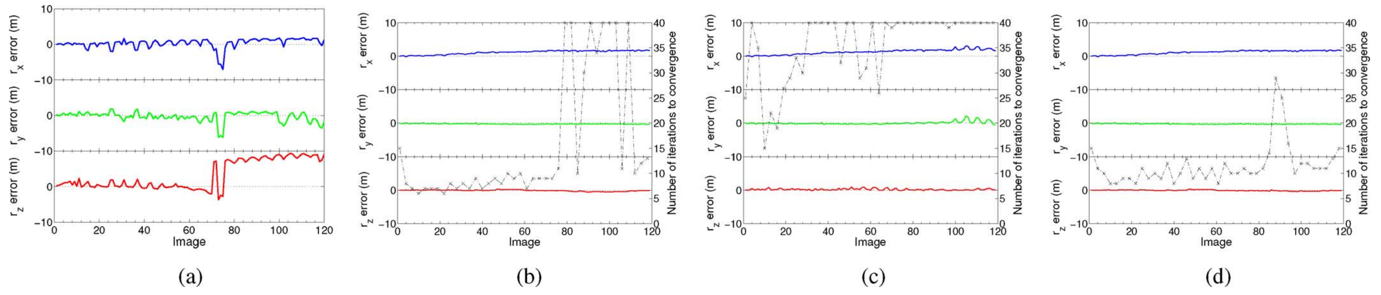


Fig. 15. Rotation errors for the different algorithms. (a) Rotation error with SPT (in degrees). (b) Rotation error with MPT_ESM (in degrees). (c) Rotation error with MPT_ESM_HUB (in degrees). (d) Rotation error with MPT_ESM_HUB_BL (in degrees).

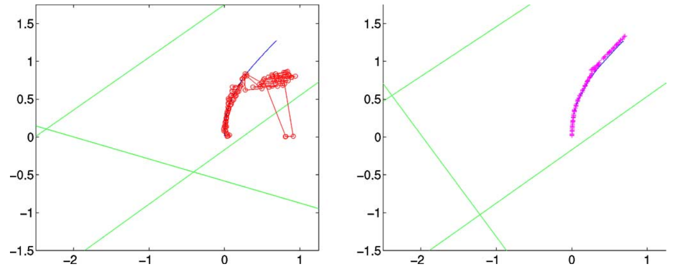
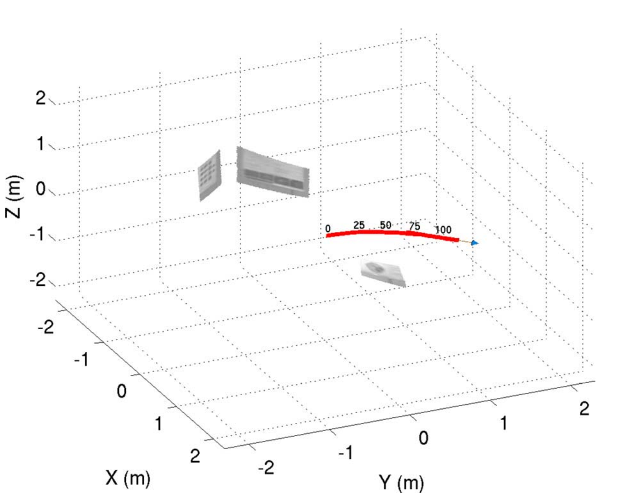


Fig. 16. Robot's motion in the *XY*-plane for SPT and MPT_ESM_HUB_BL.



$$\mathbf{P}_{r\times 3l} = \begin{bmatrix} \mathbf{N}_{1\ q_1\times 3} & \mathbf{0}_{q_1\times 3} & \cdots & \mathbf{0}_{q_1\times 3} \\ \mathbf{0}_{q_2\times 3} & \mathbf{N}_{2\ q_2\times 3} & \cdots & \mathbf{0}_{q_2\times 3} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_{q_l\times 3} & \mathbf{0}_{q_2\times 3} & \cdots & \mathbf{N}_{l\ q_l\times 3} \end{bmatrix}.$$

Fig. 17. Robot trajectory with 3-D reconstruction.

The normal equations can now be written as

$$\mathbf{J}\mathbf{x}_0 = [\mathbf{T}_{r\times 6}|\mathbf{P}_{r\times 3l}]\begin{bmatrix}\mathbf{x}_T\\ \mathbf{x}_P\end{bmatrix} = -\mathbf{f}(\mathbf{0}). \tag{31}$$

This leads to the following form:

$$\begin{bmatrix}\mathbf{T}^\top\mathbf{T} & \mathbf{T}^\top\mathbf{P}\\ \mathbf{P}^\top\mathbf{T} & \mathbf{P}^\top\mathbf{P}\end{bmatrix}\begin{bmatrix}\mathbf{x}_T\\ \mathbf{x}_P\end{bmatrix} = -\begin{bmatrix}\mathbf{T}^\top\mathbf{f}(\mathbf{0})\\ \mathbf{P}^\top\mathbf{f}(\mathbf{0})\end{bmatrix}. \tag{32}$$

By multiplying on the left by

$$\begin{bmatrix}\mathbf{I}_{6\times 6} & -\mathbf{T}^\top\mathbf{P}(\mathbf{P}^\top\mathbf{P})^{-1}\\ \mathbf{0}_{3l\times 6} & \mathbf{I}_{3l\times 3l}\end{bmatrix}$$

we eliminate the top right-hand block

$$\begin{aligned}(\mathbf{T}^\top\mathbf{T} - \mathbf{T}^\top\mathbf{P}(\mathbf{P}^\top\mathbf{P})^{-1}\mathbf{P}^\top\mathbf{T})\mathbf{x}_T\\ = -\mathbf{T}^\top\mathbf{f}(\mathbf{0}) + \mathbf{T}^\top\mathbf{P}(\mathbf{P}^\top\mathbf{P})^{-1}\mathbf{P}^\top\mathbf{f}(\mathbf{0}).\end{aligned} \tag{33}$$

The key observation is that $\mathbf{P}^\top\mathbf{P}$ is block-diagonal with blocks of size 3×3, so the complexity of the inversion is linear with the number of planes. The calculation of $\mathbf{P}^\top\mathbf{P}$ is the limiting factors with a complexity in $O(3^2lq)$ (3 because of the three parameters encoding the normal) with q depending on the number of pixels per plane.

Solving (33) by the inversion of the left-hand side (6×6 matrix), $\mathbf{x}_T$ can be obtained and used to find $\mathbf{x}_P$ by back-substitution (exploiting again the block-diagonal shape of $\mathbf{P}$)

$$\mathbf{x}_P = -(\mathbf{P}^\top\mathbf{P})^{-1}(\mathbf{P}^\top\mathbf{f}(\mathbf{0}) + \mathbf{P}^\top\mathbf{T}\mathbf{x}_T). \tag{34}$$

Written in this way, the computation of the normal equations is no longer in $O\left(\max[r(6+3l)^2,(6+3l)^3]\right)$ from the calculation or inversion of $\mathbf{J}^\top\mathbf{J}$, respectively, but in $O(3^2lq)$ (linear in the number of planes).

Appendix II

Lie Algebra Parameterization

A. Lie Groups and Lie Algebras

A Lie group is a group that locally has the topology of $\mathbb{R}^n$ everywhere (i.e., it is a smooth manifold). $\mathbb{SL}(3)$ (special linear group) and $\mathbb{SE}(3)$ (special Euclidean group) that are of interest in this paper are matrix Lie groups.

Let G be a matrix Lie group. The Lie algebra of G, denoted by $\mathfrak{g}$, can be introduced as the set of all matrices $\mathbf{X}$ such that $e^{t\mathbf{X}}$ (exponential map) is in G for all real numbers t.

An element of $g\in G$ can be expressed in terms of n independent elements of G with n being the dimension of the group ($n=8$ for $\mathbb{SL}(3)$ and $n=6$ for $\mathbb{SE}(3)$).

If g depends only on a parameter t_i

$$g(t_i) = \exp(t_i\mathbf{A}_i), \quad \text{by differentiating: } \mathbf{A}_i = \left.\frac{\partial g(t_i)}{\partial t_i}\right|_{t_i=0}$$

where $\mathbf{A}_i$ is referred to as a generator of the Lie algebra and is independent of t. If the group is connected, $\mathbf{A}_i$ can form a path to any matrix in the form of g starting from the identity. If we repeat the operation of differentiation around the identity for each independent element of the group, we obtain the set of all generators of the Lie algebra.

B. Lie Algebras for Optimization

With G being a matrix Lie group of dimension n, let

$$\begin{aligned}f: G &\longrightarrow \mathbb{R}\\ \mathbf{g} &\longmapsto f(\mathbf{g}).\end{aligned}$$

In this paper, g corresponds to a homography $\mathbf{H}$ for $G=\mathbb{SL}(3)$ or a transformation $\mathbf{T}$ for $G=\mathbb{SE}(3)$.

Consider the following minimization problem, with d being a differentiable distance and $\overline{\mathbf{f}}\in\mathbb{R}$

$$\overline{\mathbf{g}} = \min_{\mathbf{g}}\ d(f(\mathbf{g}),\overline{\mathbf{f}}).$$

When applying gradient descent to find a minimum, we start from an initial value $\widehat{\mathbf{g}}$, and at each step, add a value $\mathbf{g}_k$ calculated typically from the Jacobian $\widehat{\mathbf{g}}\leftarrow\widehat{\mathbf{g}}+\mathbf{g}_k$. As such, there is no guarantee that the new value of $\widehat{\mathbf{g}}$ will belong to the group G. To solve this problem, the new $\widehat{\mathbf{g}}$ is often projected onto the group manifold, but this can alter the convergence speed and the region of convergence.

Alternatively, we can define a new function h. With $\mathfrak{g}$ being the Lie algebra of G and $\times$ the group operation

$$\begin{aligned}h: \mathbb{R}^n &\longrightarrow \mathfrak{g} \longrightarrow \mathbb{R}\\ \mathbf{x} &\longmapsto G(\mathbf{x}) \longmapsto f(\widehat{\mathbf{g}}\times e^{G(\mathbf{x})})\end{aligned}$$

where h is only defined *locally* by the Lie algebra parameterization of G. If we apply a gradient descent approach to h, we will start at $\mathbf{x}=0$ (that corresponds to the initial value of f) and the update will be written as $\widehat{\mathbf{g}}\leftarrow\widehat{\mathbf{g}}\times e^{G(\mathbf{x}_k)}$. We are now sure that at each step, the new value of $\widehat{\mathbf{g}}$ belongs to the Lie group G. In this paper, the update operations are $\widehat{\mathbf{H}}\leftarrow\widehat{\mathbf{H}}\mathbf{H}(\mathbf{x}_k)$ and $\widehat{\mathbf{T}}\leftarrow\mathbf{T}(\mathbf{x}_k)\widehat{\mathbf{T}}$ for $\mathbb{SL}(3)$ and $\mathbb{SE}(3)$, respectively.

To be able to link $\widehat{\mathbf{g}}$ to $\overline{\mathbf{g}}$ by infinitesimal transformation, we must make sure that the values are path-connected (i.e., belong to the same component). This will always be the case for $\mathbb{SL}(3)$ and $\mathbb{SE}(3)$ as these groups have a single component.

Appendix III

Proof of the Lie Subgroup Property

Proposition 1: Let G be a matrix Lie group of dimension n, and $\mathbf{A}(\mathbf{x})$ be an $n\times n$ real matrix belonging to the associated Lie algebra and seen as a function of $\mathbf{x}\in\mathbb{R}^n$

$$\forall\mathbf{x}_0\in\mathbb{R}^n,\quad \left.\frac{d(e^{-\mathbf{A}(\mathbf{x}_0)}e^{\mathbf{A}(\mathbf{x})})}{d\mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_0}\mathbf{x}_0 = \left.\frac{de^{\mathbf{A}(\mathbf{x})}}{d\mathbf{x}}\right|_{\mathbf{x}=\mathbf{0}}\mathbf{x}_0. \tag{35}$$

Proof (This proof is valid for any Lie group, even with an implicit representation): Let g be a function of $\mathbb{R}^n$ with the property $g((1+t)\mathbf{x}_0) = g(\mathbf{x}_0)g(t\mathbf{x}_0)$ (property of the subgroup defined by $\mathbf{x}_0$). In the statement, $g(t\mathbf{x}) = e^{t\mathbf{A}(\mathbf{x})}$.

By differentiating the two expressions

$$\left.\frac{dg((1+t)\mathbf{x}_0)}{dt}\right|_{t=0} = \left.\frac{dg(\mathbf{x})}{d\mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_0}\mathbf{x}_0 \tag{36}$$

$$\left.\frac{dg(\mathbf{x}_0)g(t\mathbf{x}_0)}{dt}\right|_{t=0} = g(\mathbf{x}_0)\left.\frac{dg(\mathbf{x})}{d\mathbf{x}}\right|_{\mathbf{x}=0}\mathbf{x}_0. \tag{37}$$

Multiplying each equation by $g(\mathbf{x}_0)^{-1}$ and with the property $g(\mathbf{x}_0)^{-1} = g(-\mathbf{x}_0)$ (group property), (35) is obtained. ■

APPENDIX IV
JACOBIAN FOR TRACKING MULTIPLE PLANES

The translation can be obtained only up to a scale factor; so to avoid overparameterizing the system, the "first plane" can be estimated differently

$$\mathbf{H}_1 = \mathbf{R} + \frac{\mathbf{t}}{d_1}\mathbf{n}_1^\top, \qquad \mathbf{H}_i = \mathbf{R} + \frac{\mathbf{t}}{d_1}\left(\frac{d_1}{d_i}\mathbf{n}_i^\top\right), \qquad i > 1. \tag{38}$$

In other words, we can estimate only two parameters for $\mathbf{n}_1$ (for example, by normalizing) and three for $(d_1/d_i)\mathbf{n}_i$, $i > 1$.

For clarity, we will no longer indicate i, j, and d.

A. Current Jacobian

The current Jacobian can be written [dropping $\mathcal{I}^*(\mathbf{p}^*)$] as

$$\begin{aligned}\mathbf{J}(\mathbf{0}) &= \left[\nabla_\mathbf{x}\, \mathcal{I}\big(\Pi(\mathbf{w}\langle \mathbf{H}(\mathbf{T}(\mathbf{x})\widehat{\mathbf{T}}, \widehat{\mathbf{n}} + \mathbf{n}(\mathbf{x}))\rangle\langle \boldsymbol{\mathcal{S}}^*\rangle)\big)\right]_{\mathbf{x}=\mathbf{0}}\\ &= \mathbf{J}_\mathcal{I}\mathbf{J}_\Pi\ \mathbf{J}_w\,[\mathbf{J}_{H_T T}(\mathbf{0})\ \ \mathbf{J}_{H_n n}(\mathbf{0})].\end{aligned}$$

The group properties of $\mathbf{w}$ imply

$$\mathbf{w}\langle \mathbf{H}(\mathbf{T}(\mathbf{x})\widehat{\mathbf{T}}, \widehat{\mathbf{n}} + \mathbf{n}(\mathbf{x}))\rangle\langle \boldsymbol{\mathcal{S}}^*\rangle = \mathbf{w}\langle \mathbf{H}(\widehat{\mathbf{T}}, \widehat{\mathbf{n}})\rangle\langle \Pi^{-1}(\mathbf{q})\rangle$$

with $\mathbf{q} = \Pi(\mathbf{w}\langle \mathbf{H}(\widehat{\mathbf{T}}, \widehat{\mathbf{n}})^{-1}\mathbf{H}(\mathbf{T}(\mathbf{x})\widehat{\mathbf{T}}, \widehat{\mathbf{n}} + \mathbf{n}(\mathbf{x}))\rangle\langle \boldsymbol{\mathcal{S}}^*\rangle)$. In $\mathbf{0}$, $\mathbf{q} = \Pi(\mathbf{w}\langle \mathbf{I}\rangle\langle \boldsymbol{\mathcal{S}}^*\rangle)$; therefore, the first Jacobian $\mathbf{J}_\mathcal{I}$ is

$$\mathbf{J}_\mathcal{I} = \left[\nabla_\mathbf{q}\, \mathcal{I}\big(\Pi(\mathbf{w}\langle \mathbf{H}(\widehat{\mathbf{T}}, \widehat{\mathbf{n}})\rangle\langle \Pi^{-1}(\mathbf{q})\rangle)\big)\right]_{\mathbf{q}=\mathbf{p}}.$$

The pixel coordinates are transformed according to the current motion and normal estimates; $\mathbf{J}_\mathcal{I}$ is thus the Jacobian of the warped current image.

$\mathbf{J}_\Pi$ and $\mathbf{J}_w$ are constant Jacobians

$$\mathbf{J}_{H_T} = [\nabla_\mathbf{T}\ \mathbf{H}(\widehat{\mathbf{T}}, \widehat{\mathbf{n}})^{-1}\mathbf{H}(\mathbf{T}\widehat{\mathbf{T}}, \widehat{\mathbf{n}})]_{\mathbf{T}=\mathbf{I}} \tag{39}$$

$$\mathbf{J}_T(\mathbf{0}) = [\nabla_\mathbf{x}\ \mathbf{T}(\mathbf{x})]_{\mathbf{x}=\mathbf{0}}\,. \tag{40}$$

Let

$$\widehat{\mathbf{T}} = \begin{bmatrix}\widehat{\mathbf{R}} & \widehat{\mathbf{t}}\\ \mathbf{0} & 1\end{bmatrix}$$

and

$$\widehat{\boldsymbol{\tau}} = (\widehat{\tau}_x, \widehat{\tau}_y, \widehat{\tau}_z) = \frac{-\widehat{\mathbf{R}}^\top\widehat{\mathbf{t}}}{1 + \widehat{\mathbf{n}}^\top\widehat{\mathbf{R}}^\top\widehat{\mathbf{t}}}.$$

We also define $\mathbf{e}_1 = [1\ 0\ 0]^\top$, $\mathbf{e}_2 = [0\ 1\ 0]^\top$, and $\mathbf{e}_3 = [0\ 0\ 1]^\top$ (canonical vectors).

If we write $\mathbf{J}_{H_T T} = \mathbf{J}_{H_T}\mathbf{J}_T$, it can be shown by algebraic manipulation that

$$\mathbf{J}_{H_T T} = \begin{bmatrix}\widehat{\mathbf{n}}\ (\widehat{\tau}_x\widehat{\mathbf{n}} + \mathbf{e}_1)^\top\widehat{\mathbf{R}}^\top & \widehat{\mathbf{H}}^\top\,[\widehat{\tau}_x\widehat{\mathbf{n}} + \mathbf{e}_1]_\times\,\mathbf{I}\\ \widehat{\mathbf{n}}\ (\widehat{\tau}_y\widehat{\mathbf{n}} + \mathbf{e}_2)^\top\widehat{\mathbf{R}}^\top & \widehat{\mathbf{H}}^\top\,[\widehat{\tau}_y\widehat{\mathbf{n}} + \mathbf{e}_2]_\times\,\mathbf{I}\\ \widehat{\mathbf{n}}\ (\widehat{\tau}_z\widehat{\mathbf{n}} + \mathbf{e}_3)^\top\widehat{\mathbf{R}}^\top & \widehat{\mathbf{H}}^\top\,[\widehat{\tau}_z\widehat{\mathbf{n}} + \mathbf{e}_3]_\times\,\mathbf{I}\end{bmatrix} \tag{41}$$

$$\mathbf{J}_{H_n} = \left[\nabla_\mathbf{n}\ \mathbf{H}(\widehat{\mathbf{T}}, \widehat{\mathbf{n}})^{-1}\mathbf{H}(\widehat{\mathbf{T}}, \mathbf{n})\right]_{\mathbf{n}=\widehat{\mathbf{n}}} \tag{42}$$

$$\mathbf{J}_n(\mathbf{0}) = [\nabla_\mathbf{x}\ \widehat{\mathbf{n}} + \mathbf{n}(\mathbf{x})]_{\mathbf{x}=\mathbf{0}} = \mathbf{I} \tag{43}$$

$$\mathbf{J}_{H_n n} = \mathbf{J}_{H_n}\mathbf{J}_n(\mathbf{0}) = \begin{bmatrix}(\widehat{\tau}_x\widehat{\mathbf{n}} + \mathbf{e}_1)^\top\widehat{\mathbf{R}}^\top\widehat{\mathbf{t}}\mathbf{J}_i\\ (\widehat{\tau}_y\widehat{\mathbf{n}} + \mathbf{e}_2)^\top\widehat{\mathbf{R}}^\top\widehat{\mathbf{t}}\mathbf{J}_i\\ (\widehat{\tau}_z\widehat{\mathbf{n}} + \mathbf{e}_3)^\top\widehat{\mathbf{R}}^\top\widehat{\mathbf{t}}\mathbf{J}_i\end{bmatrix} \tag{44}$$

with

$$\mathbf{J}_i = \left[\nabla_\mathbf{n}\ \ \frac{\widehat{\mathbf{n}} + \mathbf{n}}{\|\widehat{\mathbf{n}} + \mathbf{n}\|}\right]_{\mathbf{n}=\mathbf{0}}$$

if $i = 1$ and $\mathbf{J}_i = \mathbf{I}_3$ for $i > 1$ (to take into account the normalization of $\mathbf{n}_1$).

B. Reference Jacobian

The reference Jacobian can be written [dropping $\mathcal{I}^*(\mathbf{p}^*)$] as

$$\begin{aligned}\mathbf{J}(\mathbf{x}_0) &= \left[\nabla_\mathbf{x}\, \mathcal{I}\big(\Pi(\mathbf{w}\langle \mathbf{H}(\mathbf{T}(\mathbf{x})\widehat{\mathbf{T}}, \widehat{\mathbf{n}} + \mathbf{n}(\mathbf{x}))\rangle\langle \boldsymbol{\mathcal{S}}^*\rangle)\big)\right]_{\mathbf{x}=\mathbf{x}_0}\\ &= \mathbf{J}_{\mathcal{I}^*}\ \mathbf{J}_\Pi\ \mathbf{J}_w\ \ [\mathbf{J}_{H_T^* T^*}(\mathbf{x}_0)\ \ \mathbf{J}_{H_n^* n^*}(\mathbf{x}_0)].\end{aligned}$$

The group properties of $\mathbf{w}$ imply

$$\mathbf{w}\langle \mathbf{H}(\mathbf{T}(\mathbf{x})\widehat{\mathbf{T}}, \widehat{\mathbf{n}} + \mathbf{n}(\mathbf{x}))\rangle\langle \boldsymbol{\mathcal{S}}^*\rangle = \mathbf{w}\langle \mathbf{H}(\overline{\mathbf{T}}, \overline{\mathbf{n}})\rangle\langle \Pi^{-1}(\mathbf{q})\rangle$$

with $\mathbf{q} = \Pi(\mathbf{w}\langle \mathbf{H}(\overline{\mathbf{T}}, \overline{\mathbf{n}})^{-1}\mathbf{H}(\mathbf{T}(\mathbf{x})\widehat{\mathbf{T}}, \widehat{\mathbf{n}} + \mathbf{n}(\mathbf{x}))\rangle\langle \boldsymbol{\mathcal{S}}^*\rangle)$. In $\mathbf{x}_0$, $\mathbf{q} = \Pi(\mathbf{w}\langle \mathbf{I}\rangle\langle \boldsymbol{\mathcal{S}}^*\rangle)$, so the first Jacobian $\mathbf{J}_{\mathcal{I}^*}$ is

$$\begin{aligned}\mathbf{J}_{\mathcal{I}^*} &= \left[\nabla_\mathbf{q}\ \mathcal{I}\left(\Pi(\mathbf{w}\langle \mathbf{H}(\overline{\mathbf{T}}, \overline{\mathbf{n}})\rangle\langle \Pi^{-1}(\mathbf{q})\rangle)\right)\right]_{\mathbf{q}=\mathbf{p}}\\ &= [\nabla_\mathbf{q}\ \mathcal{I}^*(\mathbf{q})]_{\mathbf{q}=\mathbf{p}}\,.\end{aligned} \tag{45}$$

This Jacobian is taken after applying the optimal transformation that transforms the pixel coordinates to the exact locations in the current image that correspond to the same image intensities as the reference template by hypothesis. Thus, this expression is the Jacobian of the reference image

$$\mathbf{J}_{H_T^*} = \left[\nabla_\mathbf{T}\ \mathbf{H}(\overline{\mathbf{T}}, \overline{\mathbf{n}})^{-1}\mathbf{H}(\mathbf{T}\overline{\mathbf{T}}, \overline{\mathbf{n}})\right]_{\mathbf{T}=\mathbf{I}} \tag{46}$$

$$\mathbf{J}_{T^*}(\mathbf{x}_0) = \left[\nabla_\mathbf{x}\ \mathbf{T}(\mathbf{x})\widehat{\mathbf{T}}\overline{\mathbf{T}}^{-1}\right]_{\mathbf{x}=\mathbf{x}_0} \tag{47}$$

$$\mathbf{J}_{H_n^*} = \left[\nabla_\mathbf{T}\ \mathbf{H}(\overline{\mathbf{T}}, \overline{\mathbf{n}})^{-1}\mathbf{H}(\widehat{\mathbf{T}}, \mathbf{n})\right]_{\mathbf{n}=\widehat{\mathbf{n}}} \tag{48}$$

$$\mathbf{J}_{n^*}(\mathbf{x}_0) = [\nabla_\mathbf{x}\ \widehat{\mathbf{n}} + \mathbf{n}(\mathbf{x})]_{\mathbf{x}=\mathbf{x}_0} = \mathbf{I} = \mathbf{J}_n(\mathbf{0}). \tag{49}$$

Trivially, we have $\mathbf{J}_n(\mathbf{0})\mathbf{x}_0 = \mathbf{J}_{n^*}(\mathbf{x}_0)\mathbf{x}_0$, and also, due to (35), $\mathbf{J}_{T^*}(\mathbf{x}_0)\mathbf{x}_0 = \mathbf{J}_T(\mathbf{0})\mathbf{x}_0$. Assuming that $\widehat{\mathbf{T}} \approx \overline{\mathbf{T}}$ and $\widehat{\mathbf{n}} \approx \overline{\mathbf{n}}$, $\mathbf{J}_{H_T^*} \approx \mathbf{J}_{H_T}$ and $\mathbf{J}_{H_n^*} \approx \mathbf{J}_{H_n}$, we obtain (27).

REFERENCES

[1] Y. Yagi, "Omnidirectional sensing and its applications," *IEICE Trans. Inf. Syst.*, vol. E82-D, no. 3, pp. 568–579, 1999.

[2] T. Pajdla, T. Svoboda, and V. Hlavac, "Epipolar geometry of central panoramic cameras," in *Panoramic Vision: Sensors, Theory, and Applications*. Berlin, Germany: Springer-Verlag, 2001, pp. 85–114.

[3] C. Geyer and K. Daniilidis, "Mirrors in motion: Epipolar geometry and motion estimation," in *Proc. Int. Conf. Comput. Vis.*, 2003, pp. 766–773.

[4] J. Barreto, F. Martin, and R. Horaud, "Visual servoing/tracking using central catadioptric cameras," in *Proc. Int. Symp. Exp. Robot.* (Advanced Robotics Series), 2002, pp. 863–869.

[5] H. Hadj-Abdelkader, Y. Mezouar, N. Andreff, and P. Martinet, "2 1/2 D visual servoing with central catadioptric cameras," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2005, pp. 3572–3577.

[6] G. Silveira, E. Malis, and P. Rives, "An efficient direct method for improving visual SLAM," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2007, pp. 4090–4095.

[7] A. I. Comport, E. Malis, and P. Rives, "Accurate quadrifocal tracking for robust 3D visual odometry," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2007, pp. 40–45.

[8] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proc. Int. Joint Conf. Artif. Intell.*, 1981, pp. 674–679.

[9] J. Shi and C. Tomasi, "Good features to track," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 1994, pp. 593–600.

[10] G. D. Hager and P. N. Belhumeur, "Efficient region tracking with parametric models of geometry and illumination," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 10, pp. 1025–1039, Oct. 1998.

[11] H. Y. Shum and R. Szeliski, "Construction of panoramic image mosaics with global and local alignment," *Int. J. Comput. Vis.*, vol. 16, no. 1, pp. 63–84, 2000.

[12] S. Baker and I. Matthews, "Equivalence and efficiency of image alignment algorithms," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2001, pp. 1090–1097.

[13] J. M. Buenaposada and L. Baumela, "Real-time tracking and estimation of planar pose," in *Proc. ICPR*, 2002, pp. 697–700.

[14] S. Benhimane and E. Malis, "Real-time image-based tracking of planes using efficient second-order minimization," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2004, pp. 943–948.

[15] S. Benhimane and E. Malis, "A new approach to vision-based robot control with omni-directional cameras," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2006, pp. 526–531.

[16] D. Cobzas and P. Sturm, "3D SSD tracking with estimated 3D planes," presented at the 2nd Can. Conf. Comput. Robot Vis., Victoria, BC, Canada, May 2005.

[17] S. Benhimane and E. Malis, "Homography-based 2D visual tracking and servoing," *Int. J. Robot. Res.*, vol. 26, no. 7, pp. 689–713, Jul. 2007.

[18] H. Jin, P. Favaro, and S. Soatto, "A semi-direct approach to structure from motion," *Vis. Comput.*, vol. 19, no. 6, pp. 377–394, 2003.

[19] N. Molton, A. Davison, and I. Reid, "Locally planar patch features for real-time structure from motion," presented at the Br. Mach. Vis. Conf., Kingston, ON, Canada, 2004.

[20] A. Bartoli, "A random samping strategy for piecewise planar scene segmentation," *Comput. Vis. Image Understanding*, vol. 105, no. 1, pp. 42–59, 2007.

[21] G. Silveira, E. Malis, and P. Rives, "Real-time robust detection of planar regions in a pair of images," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2006, pp. 49–54.

[22] M. Cummins and P. Newman, "Probabilistic appearance based navigation and loop closing," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2007, pp. 2042–2048.

[23] G. K. B. Williams and I. Reid, "Real-time SLAM relocalisation," in *Proc. Int. Conf. Comput. Vis.*, 2007, pp. 1–8.

[24] S. Baker, R. Patil, K. Cheung, and I. Matthews, "Lucas–Kanade 20 years on: Part 5," Carnegie Mellon Robot. Inst., Pittsburgh, PA, Tech. Rep., 2004.

[25] F. Jurie and M. Dhome, "Hyperplane approximation for template matching," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 996–1000, Jul. 2002.

[26] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge, U.K.: Cambridge Univ. Press, 2000.

[27] C. Mei and P. Rives, "Single view point omnidirectional camera calibration from planar grids," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 2007, pp. 3945–3950.

[28] C. Geyer and K. Daniilidis, "Catadioptric projective geometry," *Int. J. Comput. Vis.*, vol. 45, no. 3, pp. 223–243, 2001.

[29] J. P. Barreto and H. Araujo, "Geometric properties of central catadioptric line images and their application in calibration," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 8, pp. 1327–1333, Aug. 2005.

[30] E. Malis, "Improving vision-based control using efficient second-order minimization techniques," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2004, pp. 1843–1848.

[31] S. Baker, A. Datta, and T. Kanade, "Parameterizing homographies," Carnegie Mellon Robot. Inst., Pittsburgh, PA, Tech. Rep., Mar. 2006.

[32] C. Mei, S. Benhimane, E. Malis, and P. Rives, "Homography-based tracking for central catadioptric cameras," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, Oct. 2006, pp. 669–674.

[33] S. Baker, R. Patil, K. Cheung, and I. Matthews, "Lucas–Kanade 20 years on: Part 1," Carnegie Mellon Robot. Inst., Pittsburgh, PA, Tech. Rep., 2004.

[34] T. Ishikawa, I. Matthews, and S. Baker, "Efficient image alignment with outlier rejection," Carnegie Mellon Robot. Inst., Pittsburgh, PA, Tech. Rep., 2002.

Christopher Mei received the Postgraduate degree from the Institut d'Informatique d'Entreprise, Evry, France, in 2003, the M.S. degree in artificial intelligence and pattern recognition from Paris VI University, Paris, France, in 2003, and the Ph.D. degree from the Institut National de Recherche en Informatique et en Automatique (INRIA), Sophia Antipolis, France, in 2006.

Since 2007, he has been a Research Assistant with the Active Vision Group, University of Oxford, Oxford, U.K. His current research interests include simultaneous localization and mapping (SLAM), visual tracking, and omnidirectional vision.

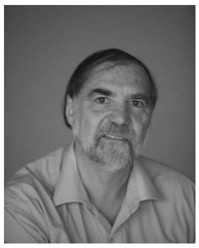
Selim Benhimane received the degree in engineering systems, automation, and vision from the National School of Higher Education in Physics, Strasbourg, France, in 2002, the Postgraduate Advanced Diploma in photonics, imaging, and cybernetics from Louis Pasteur University, Strasbourg, in 2002, and the Ph.D. degree from the École Nationale Supérieure des Mines de Paris, Paris, France.

Since 2005, he has been a Research Associate with the Technical University of Munich, Munich, Germany. His current research interests include computer vision, robotics, and vision-based control.

Dr. Benhimane received the Prize for the Best French Doctor Thesis in November 2007 from the Federation of Science and Information Technology Associations for 2005 and 2006 in the category Applied and Innovative Research.

Ezio Malis (A'04) was born in Gorizia, Italy, in 1970. He received the Postgraduate degree from both the University Politecnico di Milano, Milan, Italy, and the École Supérieure d'Electricité (Suplec), Paris, France, in 1995 and the Ph.D. degree from the University of Rennes, Rennes, France, in 1998.

For three years, he was with the Institut de Recherche en Informatique et Systèmes Aléatoires (IRISA)/Institut National de Recherche en Informatique et en Automatique (INRIA), Rennes, and the National French Company of Electricity Power (EDF). He was a Research Associate with the University of Cambridge, Cambridge, U.K., for two years. In 2000, he joined INRIA, Sophia Antipolis, France, as a Research Scientist. His current research interests include automatics, robotics, computer vision, and, in particular, vision-based control.



Patrick Rives (M'04) received the Ph.D. degree in robotics from the Université des Sciences et Techniques du Languedoc, Montpellier, France, in 1981 and the Habilitation à Diriger les Recherches degree from the Université de Nice, Nice, France, in 1991.

He was a Research Fellow with the Institut National de la Recherche Scientifique (INRS) Laboratory, Montreal, QC, Canada, for one year. In 1982, he joined the Institut National de Recherche en Informatique et Automatique (INRIA), Rennes, France. He is currently a Research Director at INRIA, Sophia Antipolis, France, where he is also the Head of the Advanced Robotics and Autonomous Systems (ARobAS) Project Team. His current research interests include mobile robot navigation, simultaneous localization and mapping (SLAM), visual servoing, sensing and scene modeling, and sensor-based control. He is also involved in the fields of aerial and underwater robotics and intelligent urban vehicles.