Fast Visual Odometry for 3-D Range Sensors

Mariano Jaimez, *Student Member, IEEE*, and Javier Gonzalez-Jimenez, *Member, IEEE*

***Abstract*—This paper presents a new dense method to compute the odometry of a free-flying range sensor in real time. The method applies the range flow constraint equation to sensed points in the temporal flow to derive the linear and angular velocity of the sensor in a rigid environment. Although this approach is applicable to any range sensor, we particularize its formulation to estimate the 3-D motion of a range camera. The proposed algorithm is tested with different image resolutions and compared with two state-of-the-art methods: generalized iterative closest point (GICP) [1] and robust dense visual odometry (RDVO) [2]. Experiments show that our approach clearly overperforms GICP which uses the same geometric input data, whereas it achieves results similar to RDVO, which requires both geometric and photometric data to work. Furthermore, experiments are carried out to demonstrate that our approach is able to estimate fast motions at 60 Hz running on a single CPU core, a performance that has never been reported in the literature. The algorithm is available online under an open source license so that the robotic community can benefit from it.**

***Index Terms*—Range sensors, real time, visual odometry.**

I. Introduction

FAST and accurate 6 degrees of freedom (DOF) visual odometry (VO) is gaining importance in current robotics where increasingly demanding applications are pursued. Two clear examples are terrestrial vehicles that must operate on uneven terrains and UAVs, which often need their 3-D pose to be tracked in order to fly autonomously. The alternative to VO in these cases is applying inertial navigation based on IMUs, but they accumulate too much error over time due to their inability to cancel gravity with enough exactitude [3]. On the other hand, traditional solutions like wheel odometry or GPS navigation simply cannot replace VO as they are not able to provide 3-D pose estimates. Another important advantage of VO is that the required sensorial data (provided by cameras or laser scanners) is also exploited by other robotic modules, both for navigation (SLAM, obstacle avoidance, etc.) and for scene understanding.

The emergence of RGB-D cameras has given rise to new and promising prospects in VO but has also posed some challenges. These sensors are able to provide RGB and depth images simultaneously at 30–60 Hz, which is a huge amount of data to process. For this reason, novel VO approaches struggle to maintain a good computational performance while trying to make the most of all these incoming data and frequently tend to focus on either RGB or depth images. In any case, not many VO methods can actually run at 30 Hz, and very few of them reach the maximum frame rate of 60 Hz that some RGB-D cameras offer.

In this paper, we introduce a novel VO method called DIFODO (DIFferential ODOmetry), which takes 3-D range images (or scans) to estimate the linear and angular velocity of the sensor. Its formulation is founded on the spatial and temporal linearization of a range function (the so-called range flow constraint equation [4], [5]), which is imposed pixel-wise in a coarse-to-fine scheme in order to cope with the estimation of large motions. A distinct feature of DIFODO is that the same input data (range measurements) are exploited both to obtain the camera motion at each level of the coarse-to-fine pyramid and to perform the warping after the level transitions. Thus, although here we particularize its formulation for depth cameras, DIFODO could be easily adapted to work with any range sensor. Another key characteristic of DIFODO is that, in contrast to other VO methods, it relies on a closed-form solution and runs in real time on a single CPU core. As a consequence, DIFODO is able to estimate fast motions and finer trajectories as it can work at the highest camera frame rate (60 Hz). As a downside, it shares the same weakness than other geometry-based VO approaches, namely: the estimation problem becomes underdetermined when the observations of the scene lack of enough geometric information, which similarly occurs for VO approaches relying on photometric data when observing low-textured areas. The idea of our proposal arose from [5] and [6] which, inspired by the concept of optical flow, presented an algorithm to estimate 2-D motion from laser scans

In order to validate our method, extensive experimentation has been carried out. First, DIFODO is tested with different resolutions to analyze how its performance changes with the number of points and levels considered. Secondly, it is compared with two prominent methods: Generalized-ICP [1] and the robust dense VO algorithm proposed by Kerl *et al.* [2] (RDVO from here onwards). The former is one of the most widespread VO strategies based on geometry, and hence, it is the best candidate to compare with given that it uses the same input data. The latter is one of the highest performance methods published recently and, conversely to our approach, exploits both geometric and photometric data to estimate the camera motion. Results show that DIFODO is about 30 times faster than generalized iterative closest point (GICP) and 2–3 times more accurate, whereas it achieves a performance similar to robust dense visual odometry (RDVO) [2] with less input data (only depth). Finally, quantitative and qualitative results are presented to demonstrate that DIFODO is able to estimate fast and real motions, which makes it suitable for real-time applications.

Manuscript received December 18, 2014; accepted April 24, 2015. Date of publication June 8, 2015; date of current version August 4, 2015. This paper was recommended for publication by Associate Editor R. Eustice and Editor D. Fox upon evaluation of the reviewers' comments. This work was supported by the Spanish Government under project DPI1011-25483, the EU FP7 Program under the GiraffPlus Project (Contract #288173), and the Spanish Grant program FPI-MICINN 2012.

M. Jaimez is with the Department of System Engineering and Automation, University of Málaga, 29071 Málaga Spain, and also with the Computer Vision Group, Department of Computer Science, Technical University of Munich, 80333 Munich, Germany (e-mail: marianojt@uma.es).

J. Gonzalez-Jimenez is with the Department of System Engineering and Automation, University of Málaga, 29071 Málaga, Spain.

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.2015.2428512

The implemented code has been added to MRPT [7] and is available under an open-source license. An illustrative video of our approach, together with the code, can be found here: http://mapir.isa.uma.es/mjaimez.

II. Related Work

Although the term "visual odometry" was first introduced by Níster in 2004 [8], the problem of estimating motion from visual inputs has been addressed from different perspectives during the last 30 years. Traditionally, VO systems have been developed for grayscale images coming from one camera or a stereo pair [9]. This general approach usually relies on detecting and matching visual features, having to deal with the problem of data association and outliers [8], [10], [11]. Most solutions resort to RANSAC [12] to solve this limitation and, although the presence of outlier matches is an inherent limitation for these strategies, great results have been achieved (e.g., in planetary exploration [10]). Most recently, the semidense approach of Engel *et al.* [13] obtains very accurate motion estimates by imposing photoconsistency at image regions with non-negligible gradients and estimating depth within a probabilistic framework. However, an important drawback of these methods based only on grayscale images is that their performance deteriorates considerably if the illumination conditions are poor.

Alternatively, range data in the form of 2-D scans have proven to be suitable to estimate planar motion. General point registration methods, most of which are variants of ICP [14], have been extensively utilized to find the homogeneous transformation between consecutive scans and have been applied not only in VO but also in Localization, Mapping, or SLAM [15], [16]. On the other hand, some methods were specifically conceived to work with 2-D range scans. In [17], a probabilistic framework is proposed to find the rigid-body transformation that maximizes the probability of having observed a scan given the previous one. Rather than trusting a local search to find the global maximum, a multiresolution CPU implementation is proposed to perform a search over the entire space of plausible rigid-body transformations, obtaining good results in simulation. Because of its relation with the proposal here, we have to mention the work of Gonzalez [5], who introduced the concept of "range image flow" and particularized the range flow constraint equation to 2-D scans to estimate the scanner motion in a very straightforward manner. Yet, its applicability was only tested in simulated and simple environments.

Recently, the advent of the new and affordable RGB-D cameras has revolutionized research in robotics. The huge amount of geometric data contained in the depth images, along with the fact that it can be easily combined with traditional RGB ones, have given rise to a fair number of new VO approaches. Some of them are similar to those relying on visual features but they incorporate depth either to directly calculate the geometric transformation between matched features or to improve the outlier rejection stage [18], [19]. Likewise, in [20], visual features are detected and fused to a global model of features against which every new set is registered using ICP. A completely different alternative consists in minimizing an energy function that is usually related to the photometric or geometric error, i.e., the differences between consecutive RGB or depth images when one of them is reprojected or "warped" against the other. This idea was first presented in [21], although in this case the authors used grayscale images generated by stereo pairs. In [22], this concept is first applied to RGB-D images and both photometric and geometric errors are minimized in a variational framework, yet lacking implementation details and quantitative results. Similarly, in [23], the focus is on the photometric error and the authors employ the depth images only to construct the warping function between consecutive intensity images. This work was extended in [2] by introducing a new probabilistic formulation that produced impressive results. However, despite their high accuracy and robustness, most of these proposals require registered geometric and photometric data to work, which in practice restricts their applicability to systems or robots equipped with RGB-D cameras.

There is another group of methods which, as we do, exclusively make use of the 3-D geometric data provided by range sensors. Either ICP or some of its variants have been employed in this regard but it is probably Generalized-ICP [1] which has demonstrated the best performance, being commonly taken as a reference for comparison [23]–[25]. On the other hand, other strategies were specifically designed to work with the range images generated by Kinect-like cameras. A remarkable case is KinectFusion [26], which introduces a signed distance function (SDF) to represent the observed scene and applies ICP to align the depth frames against this scene model. In a similar way, the work presented in [27] defines a truncated SDF (TSDF) and registers data directly to the TSDF model, rather than using it to obtain denoised depth images from a virtual sensor (as KinectFusion does). Apart from the great qualitative and quantitative results that these methods have achieved, all of them share two weaknesses. First, they are computationally very expensive, an issue that is typically overcome by developing complex parallelized CPU or GPU implementations. Second, their performance degrades if the scene does not present sufficient geometric-distinctive features.

Up to date, very few methods truly exploit both depth and RGB images jointly to estimate motion. A recent example is Kintinuous [28], the extension of KinectFusion, which added FOVIS [18] to complement ICP in areas with lack of geometric features. Additionally, in the work of Whelan *et al.* [29] different combinations of [1], [2], and [18] were implemented and tested, reporting good qualitative and quantitative results, and low runtimes by virtue of an exhaustive GPU implementation. The aforementioned work of Tykkala *et al.* [22] could also be considered to belong to this category.

III. Velocity Constraint Derived From the Range Flow Equation

This section describes how the 3-D motion of a range camera can be estimated from two consecutive frames by applying the range flow constraint and the restriction it imposes to the velocities of the observed points. Under the common assumption of a rigid scene, we formulate the point velocities in terms of

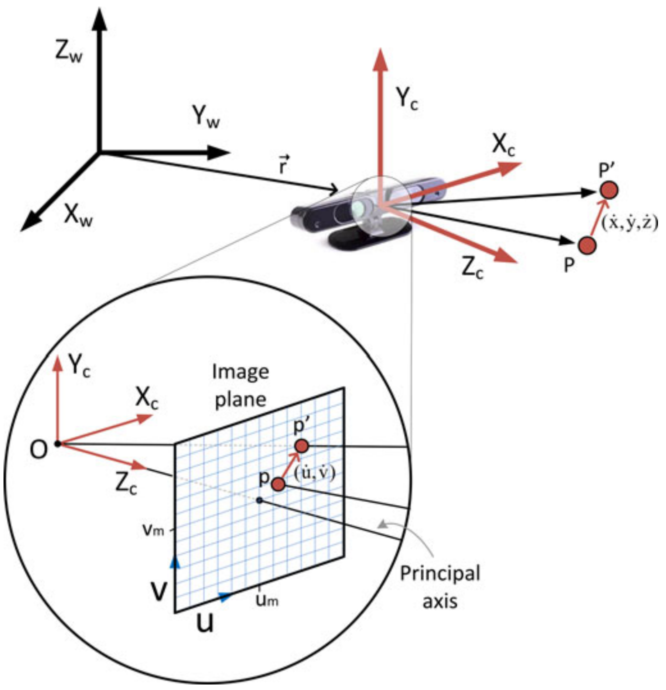

Fig. 1. Global reference frame $\{XYZ\}_w$ and camera reference frame $\{XYZ\}_C$. The Pin-Hole model defines the relationship between the motion of a point P and its optical flow $(\dot{u}, \dot{v})$ in the image plane.

the camera motion and, hence, the latter can be calculated if a sufficient number of points are considered.

Let $Z : (\Omega \in \mathbb{N}^2) \to \mathbb{R}$ be a depth image provided by a 3-D range camera where Ω is the image domain. According to the work of Spies *et al.* [4], the range flow constraint equation for range cameras reads

$$\dot{Z} = \frac{\partial Z}{\partial t} + \frac{\partial Z}{\partial u}\dot{u} + \frac{\partial Z}{\partial v}\dot{v} + O(\Delta t, \dot{u}, \dot{v}) \Rightarrow \tag{1}$$

$$\dot{Z} \simeq Z_t + Z_u \dot{u} + Z_v \dot{v} \tag{2}$$

where $\boldsymbol{w} = (u, v)$ are the pixel coordinates. Equation (2) reflects that the total derivative of the depth can be calculated from the optical flow $\dot{\boldsymbol{w}} = (\dot{u}, \dot{v})$ and the partial derivatives of Z with respect to time t, u, and v (Z_t, Z_u, and Z_v, respectively). Since (2) is derived from a first-order Taylor series expansion (1), it is exact only when the higher order terms $O(\Delta t, \dot{u}, \dot{v})$ are negligible. In practice, this condition is fulfilled if either the displacement between consecutive images is small or the observed points belong to planar patches where the linearization holds.

The three partial derivatives of Z can be directly calculated from the depth images, but $\dot{Z}$, $\dot{u}$, and $\dot{v}$ are unknowns and should be expressed in terms of the camera velocity. Let $\boldsymbol{\xi} = (v_x, v_y, v_z, \omega_x, \omega_y, \omega_z)^T$ be the camera velocity and $\boldsymbol{P_c} = (x, y, z)$ the spatial coordinates of an arbitrary point P of the scene, both described in the reference frame of the camera. The relationships between $\dot{Z}, \dot{\boldsymbol{w}}$, and $\boldsymbol{P_c}$ can be deduced from the "pin-hole model" assuming that the pixel and spatial coordinates of P are time-varying (see Fig. 1):

$$v = f_y \frac{y}{z} + v_m \Rightarrow \dot{v} = f_y \left(\frac{\dot{y}\, z - \dot{z}\, y}{z^2} \right) \tag{3}$$

$$u = f_x \frac{x}{z} + u_m \Rightarrow \dot{u} = f_x \left(\frac{\dot{x}\, z - \dot{z}\, x}{z^2} \right) \tag{4}$$

where (u_m, v_m) is the image center (principal point), and f_x, f_y are the focal length values, all expressed in pixels.

With respect to the camera reference frame, and under the assumption of a rigid and static scene, all observed points move with a velocity that is equal to the camera velocity but opposite in sign. Thus, the velocity of any 3-D point with respect to the camera can be expressed as

$$\dot{\boldsymbol{P}}_{\boldsymbol{c}} = \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{pmatrix} = \begin{pmatrix} -v_x - z\omega_y + y\omega_z \\ -v_y + z\omega_x - x\omega_z \\ -v_z - y\omega_x + x\omega_y \end{pmatrix}. \tag{5}$$

As an intermediate step, we express the optical flow in (2) in terms of the 3-D velocity of P, according to (3) and (4), as

$$-Z_t = -\dot{z} + Z_u f_x \left(\frac{\dot{x}\, z - \dot{z}\, x}{z^2} \right) + Z_v f_y \left(\frac{\dot{y}\, z - \dot{z}\, y}{z^2} \right). \tag{6}$$

Notice that Z has been replaced by z because they are equivalent: the capital letter has been used to denote the depth image, while the lower case letter directly refers to the spatial coordinate of depth. Finally, rigidity (5) is imposed in (6) as

$$\begin{aligned} -Z_t = &\left(1 + \frac{x f_x}{z^2} Z_u + \frac{y f_y}{z^2} Z_v \right) (+v_z + y\omega_x - x\omega_y) \\ &+ \frac{f_x}{z} Z_u \left(-v_x + y\omega_z - z\omega_y \right) \\ &+ \frac{f_y}{z} Z_v \left(-v_y - x\omega_z + z\omega_x \right). \end{aligned} \tag{7}$$

Equation (7) is a linear restriction that the velocity of a point of the scene (with respect to the camera) has to fulfill and, consequently, imposes a restriction to the camera velocity. Therefore, we can build a solvable algebraic system if at least six linearly independent restrictions are available. Notice that not every point will add new information to the system and, as will be described in Section V, the problem could be ill-posed depending on the spatial distribution of the scene points.

The linearization applied to derive the depth flow equation assumes differentiability of the depth images, and either small displacement of the scene or constant depth gradients. As a consequence, in the first place, points on edges must be ruled out since the depth field is not differentiable at the object borders. Second, the depth image gradients may not be constant (having higher order derivatives) which implies that, in the most general case, (7) only holds for small displacements. Within our formulation, "small displacements" means that the motion of the scene points projected onto the image plane is smaller than the neighborhood where the image gradients are computed, typically

$$\begin{aligned} |\dot{u}\Delta t| &< 1 \\ |\dot{v}\Delta t| &< 1 \end{aligned} \tag{8}$$

where Δt is the time interval between consecutive frames. Hence, the hypothesis of small displacements involves both the image resolution and the actual 3-D motion of the points.

The solution to the "small displacements" restriction was proposed by Brox *et al.* [30] and consists in utilizing a coarse-to-fine scheme. Within this strategy, a Gaussian pyramid is built for the input images and the optical flow is solved from coarser to finer levels, capturing large displacements at the coarsest levels that are subsequently refined throughout the pyramid. At each level, the previously obtained solutions are employed to warp one of the images against the other, leading to image pairs that present less differences than the original pair and for which the hypothesis of small displacements is fulfilled. This strategy can be applied as well to estimate the camera motion, as explained in [2] and [23]. In this case, the warping is not performed in the image plane but in the 3D space, and the geometric data captured by range sensors is exploited to warp a given image or measurement according to a spatial transformation. This is the reason why the methods presented in [2] and [23] require both geometric and photometric data to work: they impose photoconsistency to estimate motion but they need the geometry of the scene to perform the warping. In our study, we adopt the same warping strategy as [2] or [23], but since DIFODO relies only on geometric constraints to estimate the camera motion, it can be considered as a purely geometry-based method and can be generalized to work with any range sensor.

Warping has been extensively applied in computer vision, and it is not a contribution of this paper. Therefore, its mathematical formulation is not presented here; more information and details can be found in [2], [23], or [30].

IV. Solving the Camera Motion

A. Least-Square Solution

As previously mentioned, at least six points bearing linearly independent restrictions are required to compute the camera velocity at each level of the pyramid. In practice, however, a much higher number of points (N) are considered to make the solution robust to noise and errors, which leads to an overdetermined linear system that can be solved by weighted least squares:

$$WA\boldsymbol{\xi} = WB \rightarrow \boldsymbol{\xi} = \left(A^T WA\right)^{-1} A^T WB = MB \tag{9}$$

where $A \in \mathbb{R}^{N\times 6}$ contains the coefficients that multiply $\boldsymbol{\xi}$ in (7), $B \in \mathbb{R}^{N\times 1}$ contains the temporal derivative of depth for each pixel (inverted in sign), and $W \in \mathbb{R}^{N\times N}$ is a diagonal matrix containing the weights associated with the uncertainty of each equation. The $M \in \mathbb{R}^{6\times 6}$ matrix in (9) is symmetric and positive definite /semidefinite and, therefore, a Cholesky LDLT factorization (as implemented in Eigen [31]) can be used to compute the solution. Since such factorization applies to a fixed-size matrix (6 × 6), it does not condition the computational cost of our method. Although not all the steps of the algorithm have been detailed yet (see Section VI), the only operations whose complexity is quadratic with the number of points are the products $A^T WA$ and $A^T WB$ (using dense algebra). Nonetheless, the algebraic system (9) can be rewritten in a way that these products do not need to be computed: by multiplying both sides of (7) by the square root of the corresponding weight and solving the resulting equation system with the pseudoinverse matrix

$$\begin{aligned} A^w\boldsymbol{\xi} &= B^w \rightarrow \boldsymbol{\xi} = \left((A^w)^T A^w\right)^{-1} (A^w)^T B^w \\ &= MB \end{aligned} \tag{10}$$

$$A^w = \begin{pmatrix} \sqrt{w_1}a_{11} & \cdots & \sqrt{w_1}a_{16} \\ \vdots & \ddots & \vdots \\ \sqrt{w_N}a_{N1} & \cdots & \sqrt{w_N}a_{N6} \end{pmatrix},$$

$$B^w = \begin{pmatrix} \sqrt{w_1}b_1 \\ \vdots \\ \sqrt{w_N}b_N \end{pmatrix}. \tag{11}$$

Thus, the new formulation allows us to recover the motion parameters with a computational time that grows only linearly with the number of points.

B. Weighting Function

Weights are necessary to adjust the contribution of every point to the overall motion estimate according to the uncertainty or error associated with its range flow equation. Without loss of generality, this error can be expressed as the addition of two terms: the measurement error, which measures how the sensor noise affects the range equation, and the linearization error from the first-order approximation in (2). The former term is typically modeled by a zero-mean Gaussian distribution and can be calculated propagating the measurement error to the whole equation. The latter does not follow a Gaussian model and, in general, it is more complex to estimate because it involves studying the second-order derivatives of depth to evaluate how significant the neglected terms in (1) actually are. In this study, we address the analysis of both sources of error and present a weighting function that encompasses information about the camera and the geometry of the scene from which to weight the image pixels accordingly.

In order to estimate the measurement error, we need to take into consideration every stochastic variable or parameter that appears in (7). Assuming that the parameters of the camera are exactly known, (7) can be rearranged and expressed as a function of all noise-prone variables, that is, those dependent on the depth:

$$R\left(x, y, z, Z_t, Z_u, Z_v\right) = 0 \tag{12}$$

being x, y, z linearly related for a given pixel (see (3) and (4)).

To propagate the error of these depth-dependent variables to the range flow equation, their covariance matrix $\Sigma_d \in \mathbb{R}^{6\times 6}$ has to be built as a preliminary step:

$$\Sigma_d = \left(\begin{array}{c|c} \Sigma_{xyz} & \Sigma_{(xyz)(Z_{t,u,v})} \\ \hline \Sigma^T_{(xyz)(Z_{t,u,v})} & \Sigma_{Z_{t,u,v}} \end{array}\right) \tag{13}$$

where $\Sigma_{xyz}, \Sigma_{(xyz)(Z_{t,u,v})}, \Sigma_{Z_{t,u,v}} \in \mathbb{R}^{3\times 3}$. The mathematical derivation of the submatrices in Σ_d is based on the characteristics of Kinect-like cameras and is presented in the Appendix I. Knowing Σ_d, the variance of (7) associated with the measurement errors can be computed as

$$\sigma_m^2 = \nabla R \cdot \Sigma_d \cdot \nabla R^T \tag{14}$$

where ∇R is the gradient of R with respect to x, y, z, Z_t, Z_u and Z_v. Given that R also depends on $\boldsymbol{\xi}$, an approximation of ∇R is computed using the $\boldsymbol{\xi}$ estimated at the previous time step.

On the other hand, to analyze the error derived from the linearization in (1), we must rewrite that equation including the second-order terms

$$\begin{aligned} \dot{Z} =& \; Z_t + Z_u \dot{u} + Z_v \dot{v} + Z_2(\Delta t, \dot{\boldsymbol{w}}) + O\left(\Delta t^2, \dot{\boldsymbol{w}}\right), \\ Z_2(\Delta t, \dot{\boldsymbol{w}}) =& \; \frac{\Delta t}{2}\left(Z_{tt} + Z_{tu}\dot{u} + Z_{tv}\dot{v} + Z_{uu}\dot{u}^2\right. \\ &\left. + Z_{vv}\dot{v}^2 + 2Z_{uv}\dot{u}\dot{v}\right). \end{aligned} \tag{15}$$

It can be observed that, neglecting third or higher order terms, the error is a function of all the second-order derivatives and the optical flow. To estimate how (15) deviates from linearity, we can approximate the depth second-order derivatives from the depth images, but the optical flow is not known in advance. One possible solution to this problem would consist in computing the optical flow from the previous $\boldsymbol{\xi}$. However, this strategy loses its sense when it has to be applied at finer levels of the Gaussian pyramid where the previous solution gives us no information about the optical flow with respect to warped images. For that reason, we choose a quadratic expression that penalizes the second derivatives homogeneously for all pixels as

$$\delta_l^2 = k_l \left[\Delta t^2 \left(Z_{tu}^2 + Z_{tv}^2\right) + Z_{uu}^2 + Z_{vv}^2 + Z_{uv}^2\right]. \tag{16}$$

The constant k_l weights the linearization error against the measurement error, and the time increment multiplies the temporal derivatives so that all terms are of the same order of magnitude (depth differences). In practice, it can be noticed that a high penalization of the second-order derivatives might discard some points or areas of the scene that, despite its inaccuracy, are useful to constrain the motion estimate. A deeper study of the scene geometry could be performed to detect beforehand how the range equation of every point would constrain the velocity estimate and adapt k_l accordingly. However, this procedure would significantly increase the computational cost of the algorithm, and hence, it is not adopted in this study. Besides, the second temporal derivative of Z is not considered in (16) because it cannot be estimated at the different pyramid levels. With the exception of the coarsest level, the others involve warped images which are timeless and for which the concept of second temporal derivative makes no sense.

If N points are considered to build (9), for each point i with $i \in 1, 2 \ldots N$, its corresponding weight is inversely related to the uncertainty associated with the range flow equation

$$W_{ii} = \frac{1}{\sigma_{m,i}^2 + \delta_{l,i}^2}. \tag{17}$$

V. Scene Geometry, Covariance Analysis, and Velocity Filtering

Depending on the spatial distribution of the points used to build the algebraic system (10), the problem can be well or ill-posed. If the points contain sufficient information about how the scene has changed in the three directions of space, then the M matrix will be positive definite and (10) will stand for a well-posed configuration; otherwise, the M matrix will be rank deficient (or close to). Excluding the degenerated singular cases of some surfaces of revolution (i.e., the camera in the center of a sphere), sufficient information is guaranteed if the normals of the observed surfaces can make up a 3-D vector basis, that is

$$\operatorname{rank}\left([\boldsymbol{n}_1, \boldsymbol{n}_2, \ldots, \boldsymbol{n}_N]\right) = 3 \tag{18}$$

where $\boldsymbol{n_i} \in \mathbb{R}^{3\times 1}$ is the surface normal vector at a given point i. The unfulfillment of this condition leads to the well-known sliding problem of ICP [32] and frequently occurs when the whole point cloud comes from one or two planes (a wall, the floor, etc.).

If the M matrix is positive definite, the points of the scene provide enough geometric constraints to estimate the 6 DOF of the motion. On the contrary, if M has not full rank, some linear or angular velocity terms cannot be estimated, and the solution that the solver provides for these variables is meaningless. This casuistry can be detected analyzing the covariance matrix $\Sigma_\xi \in \mathbb{R}^{6\times 6}$ associated to the least-squares solution

$$\Sigma_\xi = \frac{1}{N-6}\sum_{i=1}^{N} r_i^2 \left((A^w)^T A^w\right)^{-1} \tag{19}$$

where r_i are the residuals of the least squares solution. The diagonal elements of the covariance matrix reflect the variance of the components of the estimated $\boldsymbol{\xi}$ in the reference frame of the camera. In general, it is more meaningful to express the matrix Σ_ξ in a diagonal form where the eigenvalues indicate which combinations of motions (eigenvectors) are constrained and which are undetermined. This way, any uncertainty in the velocity estimate that may appear due to poor geometric information will be revealed: its corresponding eigenvalue will reflect how high or low this uncertainty is while its eigenvector will contain the velocity terms affected by it. This information can be employed to neglect those velocity terms whose variance (eigenvalue) is too high. Thus, if data from other sources like IMUs or wheel odometry were available they could be fused with the VO estimation to generate a more robust solution. When this is not the case, a suitable option consists of applying a smooth filter based on the current and previous estimates, as explained next.

Let $E = \{\boldsymbol{ev_1}, \ldots, \boldsymbol{ev_6}\}$ be an orthogonal basis comprising the eigenvectors ($\boldsymbol{ev_{1\ldots 6}} \in \mathbb{R}^6$) of Σ_ξ, and $D \in \mathbb{R}^{6\times 6}$ the covariance diagonal matrix containing the associated eigenvalues. At a given interval of time $[t, t+1]$, the solution provided by the solver $\boldsymbol{\xi}^{t,s}$ and the previous velocity estimate $\boldsymbol{\xi}^{t-1}$ have to be expressed in the basis E^t, which is always computed from the last covariance matrix Σ_ξ^t. Then, the filtered velocity $\boldsymbol{\xi}_E^t$ in the basis E^t is obtained as a weighted sum of the current

Algorithm 1. DIFODO

```
Inputs: New depth frame $-Z^t$
        Previous Gaussian pyramid $-GP(Z^{t-1})$
        Previous velocity estimate $-\boldsymbol{\xi}^{t-1}$
1. Build the new Gaussian pyramid $\rightarrow GP(Z^t)$
for $i = 1$ to number of pyramid levels do:
    2. if $(i > 1)$ then Perform warping $\rightarrow Z^t_{w,i}$
       else then $Z^t_{w,1} = Z^t_1$
    3. if $(Z^t_i == 0) \,||\, (Z^t_{w,i} == 0)$
         Discard pixels (null measurements)
    4. Compute depth derivatives $\rightarrow \frac{\partial Z}{\partial t}^t, \frac{\partial Z}{\partial u}^t, \frac{\partial Z}{\partial v}^t$
    5. Compute the weighting function $\rightarrow W^t$
    6. Solve weighted least squares $\rightarrow \boldsymbol{\xi}^{t,s}_i, \Sigma^t_{\xi,i}$
    7. Filter level solution $\rightarrow \boldsymbol{\xi}^t_i$
end for
8. Compute overall solution and update pose $\rightarrow \boldsymbol{\xi}^t$
```

estimate and the previous velocity

$$\left[(1 + k_1)\, I + k_2 D^t\right] \boldsymbol{\xi}^t_E = \boldsymbol{\xi}^{t,s}_E + \left(k_1 I + k_2 D^t\right) \boldsymbol{\xi}^{t-1}_E. \quad (20)$$

Equation (20) represents a low-pass filter with dynamic smoothing, where k_1 helps to soften the estimated trajectory and k_2 controls how the eigenvalues in D affect the final estimate. A high value of k_2 implies that those velocity terms with uncertainty will be approximated to their previous value (interval $[t-1, t]$), whereas the current estimate (interval $[t, t+1]$) will have a higher importance if k_2 is low.

VI. DIFODO Framework and Implementation

Overall, the algorithm carries out a sequence of steps that are depicted in Algorithm 1. It receives as inputs the new depth image, the previous pyramid of depth images and the last motion estimate, and yields the average linear and angular camera velocities during the last interval of time, from which the camera pose will be updated. Important aspects and implementation details of this algorithm are explained below.

A. Gaussian Pyramid

The Gaussian pyramid is built by downsampling the depth images with a standard 5×5 Gaussian kernel. However, a pure Gaussian smoothing would create artifacts in the depth images because it would mix very dissimilar depth values at the object borders. For that reason, a bilateral filter is applied instead to preserve the geometry of the scene. In our study, the Gaussian pyramid starts to be built at a QVGA resolution (240×320). As will be discussed in Section VII, in order to estimate fast motions it might be advantageous to run DIFODO with a lower resolution of 120×160 (QQVGA), in which case the image pyramid should be built from this resolution onwards, saving computational time.

B. Warping

At every new level of the Gaussian pyramid, one of the two depth images must be warped against the other. To perform the warping, all the motion estimates from the previous levels must be integrated to obtain the overall transformation accumulated up to the present level. A special case is, of course, the first level where no warping is needed. In our formulation, the new frame is always warped against the old frame, and the motion estimates of every level are expressed in the same reference frame: the last known pose of the camera.

C. Depth Image Gradients and Weights

The implementation of DIFODO requires a discrete formulation that estimates the average camera velocity between two consecutive depth frames. Conversely to the color of a scene, whose gradient remains almost constant for all perspectives from which the scene is observed, the depth gradients change as the camera moves. For this reason, the depth gradients should not be computed from either the initial or the final depth images at a given interval of time. As an alternative, we demonstrate that a better choice to approximate the depth gradients consists in applying a trapezoidal solution that averages values from both images.

If $\Delta \boldsymbol{w} = (\dot{u}\, \Delta t, \dot{v} \Delta t)$ is the motion of a given point projected onto the image domain, its depth motion is given by

$$\dot{Z}(\boldsymbol{w})\Delta t = Z^t\left(\boldsymbol{w} + \Delta\boldsymbol{w}\right) - Z^{t-1}\left(\boldsymbol{w}\right). \quad (21)$$

Equation (21) is the general and nonlinear expression of geometric consistency from which (2) is derived. Instead of substituting the term $Z^t(\boldsymbol{w} + \Delta\boldsymbol{w})$ by the standard Taylor linearization, we employ a more accurate approximation that weights the initial and final depth gradients at a given time instant as

$$\begin{aligned} Z^t\left(\boldsymbol{w} + \Delta\boldsymbol{w}\right) &= Z^t\left(\boldsymbol{w}\right) \\ &\quad + \nabla\left(\frac{Z^t\left(\boldsymbol{w}\right) + Z^t\left(\boldsymbol{w} + \Delta\boldsymbol{w}\right)}{2}\right) \cdot \Delta\boldsymbol{w}. \end{aligned} \quad (22)$$

At the right-hand side of (22), the term $Z^t(\boldsymbol{w} + \Delta\boldsymbol{w})$ is still present and needs to be expressed as a function of the previous depth image and the depth motion, according to (21), as

$$\begin{aligned} &Z^t\left(\boldsymbol{w} + \Delta\boldsymbol{w}\right) \\ &= Z^t\left(\boldsymbol{w}\right) + \nabla\left(\frac{Z^t\left(\boldsymbol{w}\right) + Z^{t-1}\left(\boldsymbol{w}\right) + \dot{Z}^t\left(\boldsymbol{w}\right)\Delta t}{2}\right) \cdot \Delta\boldsymbol{w}. \end{aligned} \quad (23)$$

As commonly done in variational methods, we can impose regularization over the depth motion and assume that the depth motion field is locally constant as

$$\nabla \dot{Z}^t(\boldsymbol{w}) \sim 0. \quad (24)$$

Finally, this smoothness constraint is imposed in (23) which, in turn, is substituted into (21) to obtain a more accurate expression of the range flow constraint equation

$$\dot{Z}^t(\boldsymbol{w}) \simeq \frac{Z^t(\boldsymbol{w}) - Z^{t-1}(\boldsymbol{w})}{\Delta t} + \nabla\left(\frac{Z^t(\boldsymbol{w}) + Z^{t-1}(\boldsymbol{w})}{2}\right) \cdot \dot{\boldsymbol{w}}. \tag{25}$$

On the other hand, the image gradients must be approximated by some finite difference formula, given that Ω is not a continuous domain but a discrete one. Most of the times, this aspect does not receive much attention in the literature and a certain constant kernel is applied over the whole image. However, this is not the best strategy to compute the depth gradients because it leads to very high values at the object borders which does not reflect the real gradients of the surfaces of these objects. As an alternative, we make use of an adaptive formula of finite differences which, at a point P observed at a pixel $\boldsymbol{w}$, computes the depth gradients taking into account those surrounding pixels that observe points close to P. Mathematically, it implies that two nearness functions $r_u, r_v : (\Omega \in \mathbb{N}^2) \to \mathbb{R}$ must be computed as

$$r_u(u,v) = \frac{1}{\|X(u+1,v) - X(u,v), Z(u+1,v) - Z(u,v)\|} \tag{26}$$

$$r_v(u,v) = \frac{1}{\|Y(u,v+1) - Y(u,v), Z(u,v+1) - Z(u,v)\|} \tag{27}$$

where $\|\bullet\|$ is the Euclidean norm and $X, Y : (\Omega \in \mathbb{N}^2) \to \mathbb{R}$ represent the x, y coordinates of the points of the scene. The depth gradients are calculated then from the depth images and the nearness functions:

$$\begin{aligned} Z_u(u,v) &= \frac{r_u(u,v)\, Z_u^+(u,v) + r_u(u-1,v)\, Z_u^-(u,v)}{r_u(u,v) + r_u(u-1,v)} \\ Z_u^+(u,v) &= Z(u+1,v) - Z(u,v) \\ Z_u^-(u,v) &= Z(u,v) - Z(u-1,v) \end{aligned} \tag{28}$$

$$\begin{aligned} Z_v(u,v) &= \frac{r_v(u,v)\, Z_v^+(u,v) + r_v(u,v-1)\, Z_v^-(u,v)}{r_v(u,v) + r_v(u,v-1)} \\ Z_v^+(u,v) &= Z(u,v+1) - Z(u,v) \\ Z_v^-(u,v) &= Z(u,v) - Z(u,v-1). \end{aligned} \tag{29}$$

For simplicity's sake, the temporal superscripts have been omitted. These expressions always upweight the closest points in space to approximate the depth gradients, hence avoiding the assignment of huge values at the object borders. If they are evaluated at pixels which do not lie on borders of objects, then the right and left derivatives ($Z^+$and Z^-) will be similarly weighted, which results in a standard centered approximations of the image gradients.

Regarding the weighting function, the constant k_l that weights the linearization error against the measurement error is set to 5×10^{-6}. The second-order derivatives are approximated by constant centered formulas (using the stencil $[-1\ 0\ 1]$) applied over the first-order derivatives.

D. *Filter Velocity and Update Pose*

The velocity estimate must be filtered at each level of the pyramid because each level can suffer from the lack of geometric distinctive features. However, the last velocity estimate cannot be used directly to filter the solution of every level because all the previous levels have already estimated some motion which should be subtracted from it. The sequential steps that the filter performs at each level of the pyramid are as follows.

1) Compute the overall velocity estimate $\boldsymbol{\xi}_i^{t,\text{acu}}$ accumulated up to the present level i.
2) Subtract $\boldsymbol{\xi}_i^{t,\text{acu}}$ from the last velocity estimate $\boldsymbol{\xi}^{t-1}$ to obtain $\boldsymbol{\xi}_i^{t,\text{sub}}$.
3) Compute the covariance matrix $\Sigma_{\xi,i}^t$.
4) Calculate the eigenvalues D_i^t and the eigenvectors E_i^t.
5) Express the least-squares solution $\boldsymbol{\xi}_i^{t,s}$ and $\boldsymbol{\xi}_i^{t,\text{sub}}$ in the base E_i^t.
6) Apply (20) with $\boldsymbol{\xi}_{i,E}^{t,s}$ and $\boldsymbol{\xi}_{i,E}^{t,\text{sub}}$ as inputs, and obtain $\boldsymbol{\xi}_{i,E}^t$.
7) Transform $\boldsymbol{\xi}_{i,E}^t$ to the reference frame of the camera.

When the process is finished in all the pyramid levels, the final motion estimate $\boldsymbol{\xi}^t$ is computed and integrated over the last time interval to update the camera pose.

Although the same filtering process is followed in every level, there is an important aspect that should be taken into account: the coarsest levels capture the trend of the motion, while the last levels refine it to get an accurate estimate. As a consequence, the coarsest levels are likely to get solutions which are similar to the previous velocity of the camera, whereas the estimates of the finer levels are probably quite independent of it. Therefore, our filter should give a weight to the last motion estimate that decreases from coarser to finer levels. To this end, and to smooth the trajectory estimates, we have empirically chosen the following weighting functions:

$$k_1 = 0.5e^{-(l-1)}, k_2 = 0.05e^{-(l-1)} \tag{30}$$

where l is the pyramid level that ranges from 1 (coarsest) to the number of levels considered. These functions have heuristically proved to be a good tradeoff between smoothness of the estimated trajectory and capability to accommodate motion changes in a huge variety of scenes (as shown in the experiments and results of the next section).

VII. Experiments and Results

This section is divided into four parts corresponding to four distinct series of experiments. First, DIFODO is tested with different resolutions to analyze how its performance changes with the number of pyramid levels considered. Second, several experiments are conducted to compare DIFODO, GICP [1], and RDVO [2] focusing on their speed and accuracy. Third, we demonstrate the suitability of our approach to estimate fast motions, comparing results of increasing camera velocities and analyzing its performance using QVGA at 30 Hz against QQVGA at 60 Hz. Fourth, qualitative results will be shown in the form of 3-D maps built purely from odometry pose estimations.

For the first three groups of experiments, we have utilized some data series from the TUM datasets [24], given that they

TABLE I
DATASET CHARACTERISTICS

Datasets	Duration (s)	Avg. trans. vel (m/s)	Avg. rot. vel (deg/s)	Est. traj. length (m)
f1-desk	23.4	0.413	23.33	9.66
f1-desk2	24.86	0.426	29.31	10.59
f1-teddy	50.82	0.315	21.32	16.01
f1-plant	41.53	0.365	27.89	15.16
f1-room	48.9	0.334	29.88	16.33
f2-desk	99.36	0.193	6.34	19.18
f2-deskwp	142.08	0.121	5.34	17.19

provide the ground truth to calculate the estimate error. The selected datasets reproduce varied camera motions at different speeds and include scenes that contain sufficient geometric and photometric information (the latter is necessary for RDVO). The datasets chosen, according to the authors of [24], belong to three different categories:

1) *Handheld SLAM:* It includes general camera motions in office-like or house-like environments. Within this category, we use "Freiburg1/desk" (f1-desk), "Freiburg1/desk2" (f1-desk2), "Freiburg1/room" (f1-room) and "Freiburg2/desk" (f2-desk).
2) *Three-Dimensional Object Reconstruction*: It includes trajectories around certain objects. Within this category, we use "Freiburg1/teddy" (f1-teddy) and "Freiburg1/plant" (f1-plant).
3) *Dynamic Objects*: In these scenes, there are one or more moving objects and, hence, the rigid scene hypothesis is not fulfilled. Within this category, we use "Freiburg2/desk with person" (f2-deskwp).

Relative and absolute pose errors will be considered, as described in [24], to study the accuracy of our approach. For the relative pose error, both translational and rotational deviations per second will be evaluated with the root-mean-squared error (RMSE). Besides, we will evaluate the RMSE and the maximum values of the absolute trajectory errors (translational) to analyze the robustness of these approaches to reproduce 3-D trajectories faithfully. A brief summary of the datasets considered is presented in Table I. Because of the lack of ground truth at some trajectory stretches, the trajectory length is estimated using the datasets duration and their average translational speed. For all the experiments, the test platform used is a standard desktop PC running Ubuntu 12.04 with an Intel Core i7-3820 CPU at 3.6 GHz.

A. DIFODO with Different Resolutions

Several tests have been carried out to analyze how the accuracy and speed of DIFODO are affected by the maximum resolution of the depth images employed in the coarse-to-fine scheme. A total amount of 28 experiments have been performed, four for each dataset, with maximum resolutions of 30×40, 60×80, 120×160, and 240×320. In every case, the coarsest level of the pyramid had a resolution of 15×20, implying that the number of levels in the aforementioned experiments were 2, 3, 4, and 5, respectively. No results are presented with a resolution of 15×20 because it is a too low resolution to produce decent estimates. Initially, relative translational and rotational errors are compared (see Fig. 2). As expected, accuracy increases with higher resolutions. It can be noticed that the RMSE of each dataset varies as accuracy does not only depend on the VO method but also on the camera speed and the geometry of the observed scenes. Likewise, in Fig. 3, the RMSE and maximum absolute translational errors are plotted. It can be observed that, on average, the longest datasets present the highest absolute error since the errors of every iteration are propagated over a longer period of time. It is worth mentioning that the highest resolution (QVGA) does not provide significantly improved estimates respect to those of the previous level (QQVGA). Although it is logical that the solution refinement becomes less and less noticeable with higher resolutions, the main factor that

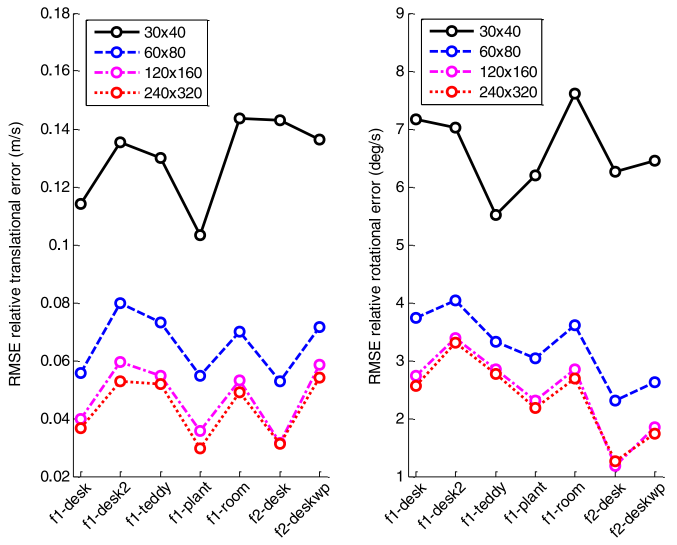

Fig. 2. Comparison of translational (left) and rotational (right) drifts per second with DIFODO at different resolutions.

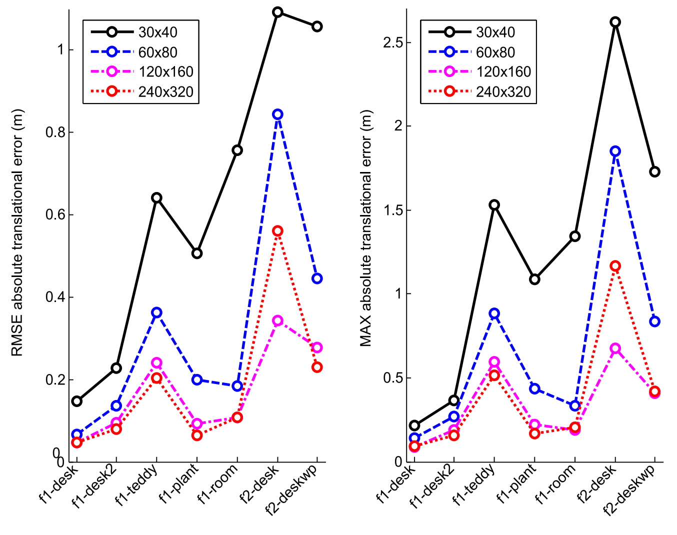

Fig. 3. Comparison of the RMSE (left) and MAX (right) absolute translational errors with DIFODO for different resolutions.

TABLE II
DIFODO RUNTIME (ms)

Dataset	Resolution			
	30×40	60×80	120×160	240×320
f1-desk	3.79	5.12	10.09	28.51
f1-desk2	3.78	5.13	10.03	28.3
f1-teddy	3.94	5.30	10.18	28.99
f1-plant	3.91	5.27	10.17	28.57
f1-room	3.87	5.12	9.92	28.58
f2-desk	3.85	5.23	10.14	29.19
f2-deskwp	3.78	5.09	9.73	28.14
Average	3.85	5.18	10.04	28.61

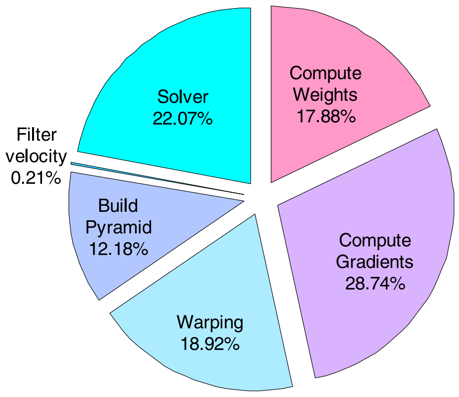

Fig. 4. Percentage computational load of the main steps that DIFODO performs to estimate motion with QVGA resolution.

explains this result is the fact that images at QVGA resolution are not filtered because they represent the finest level of the pyramid. As a consequence, the noise of the measurements affects the derivatives calculation more drastically than at any other level. At the finest level, points are closer to each other, and the differences between their depth values are caused not only by the geometry of the scene but also by the noise of the measurements. Hence, the depth gradient approximation becomes imprecise and does not perfectly reflect the real gradients of the observed surfaces. If necessary, this effect could be partially alleviated by applying a previous bilateral filter to the finest level, but at the expense of a higher computational cost. Overall, Figs. 2 and 3 show that it is beneficial to set QVGA as the maximum resolution although there might be some exceptions like the dataset f2-desk.

DIFODO computational time is reported in Table II. As DIFODO provides a closed-form solution, its runtime is essentially the same for all the datasets and the exact values might slightly differ depending on how many pixels present null measurements. In Fig. 4, we show how the runtime of DIFODO at QVGA is distributed among the main steps that comprise it (see Algorithm 1). The velocity filter is significantly faster than any other step because it is the only one which does not perform pixel-wise operations. Besides, one aspect should be remarked. In all cases, the Gaussian pyramid was built starting from a resolution of 240×320, which on average takes 3.5 ms. However, if DIFODO is configured to work with any inferior resolution, the

TABLE III
RUNTIME COMPARATIVE

Dataset	Average runtime (ms)			
	$\text{DIFODO}_{\text{LR}}$	$\text{DIFODO}_{\text{HR}}$	RDVO	GICP
f1-desk	10.09	28.51	28.96	838.24
f1-desk2	10.03	28.3	27.00	877.53
f1-teddy	10.18	28.99	27.66	862.37
f1-plant	10.17	28.57	41.82	824.11
f1-room	9.92	28.58	27.39	856.73
f2-desk	10.14	29.19	42.85	769.81
f2-deskwp	9.73	28.14	32.43	677.74
Average	10.04	28.61	32.43	815.22

Gaussian pyramid can be built from 120×160 upwards, which takes less than 1 ms. As a consequence, DIFODO runtime can be actually smaller than it is shown in Table II for any resolution inferior to QVGA. This aspect makes QQVGA resolution particularly appropriate for fast robotic applications because it can provide quite accurate results (see Figs. 2 and 3) with a very low runtime.

B. Comparison Between DIFODO, GICP, and RDVO

In this section, we compare our approach with two state-of-the-art methods: GICP and RDVO. For every method, we will consider QVGA resolution, although results of DIFODO with QQVGA will be also included to analyze how its performance deteriorates if a faster version of it is needed. To refer to them, we will use the names $\text{DIFODO}_{\text{HR}}$ and $\text{DIFODO}_{\text{LR}}$ which stand for high and low resolution, respectively. The results for RDVO have been generated with the code that the authors published online for ROS [33]. Regarding GICP, we make use of the implementation included in PCL [34]. For a fair comparison, a bilateral filter with a 5×5 mask is applied to the depth images before running GICP, which smoothes the raw data and leads to better results. The GICP parameters have been chosen as follows.

1) The maximum correspondence distance is set to 0.5 m to cover all the translations and rotations of the datasets where the scenes are sampled at 30 Hz and the maximum observed distance is about 5 m.
2) The maximum number of iterations is set to 10, although we observed in the experiments that the method almost always converges at the third–fourth iteration.
3) Transformation epsilon is set to 10^{-5} which marks the minimum difference between consecutive transformations to assume that the algorithm has converged. It is actually applied separately to translations and rotations (with this same value).

First, Table V presents the execution time statistics for each algorithm and dataset. From the results in Table V, we can conclude that our proposal is almost 30 times faster than GICP and as fast as RDVO, although our runtime is more deterministic since DIFODO is not iterative. On the other hand, we also compare how precise these methods are in estimating the camera motion. Relative pose errors in the form of translational and rotational deviations per second are analyzed and shown in

TABLE IV
RELATIVE ERRORS: TRANSLATIONAL AND ROTATIONAL DEVIATIONS PER SECOND

Datasets	RMSE Translational deviations (m/s)				RMSE Rotational deviations (deg/s)			
	DIFODO$_{\mathrm{LR}}$	DIFODO$_{\mathrm{HR}}$	RDVO	GICP	DIFODO$_{\mathrm{LR}}$	DIFODO$_{\mathrm{HR}}$	RDVO	GICP
f1-desk	0.0398	**0.0366**	0.0408	0.1017	2.731	2.562	**2.176**	6.88
f1-desk2	0.0596	**0.0528**	0.0645	0.1166	3.393	**3.311**	3.547	6.66
f1-teddy	0.0547	**0.0518**	0.0967	0.1273	2.85	2.766	**2.455**	4.706
f1-plant	0.0356	**0.0298**	0.0341	0.1117	2.306	2.179	**1.245**	4.783
f1-room	0.0531	**0.0489**	0.0622	0.0993	2.846	2.688	**2.617**	4.366
f2-desk	0.0317	0.0313	**0.0239**	0.0768	1.182	1.259	**1.019**	2.975
f2-deskwp	0.0585	0.0542	**0.0312**	0.0719	1.85	1.741	**0.897**	2.885
Average	0.0476	**0.0436**	0.0505	0.1008	2.451	2.358	**1.994**	4.751

TABLE V
ABSOLUTE TRANSLATIONAL ERRORS

Datasets	RMSE (m)				Maximum (m)			
	DIFODO$_{\mathrm{LR}}$	DIFODO$_{\mathrm{HR}}$	RDVO	GICP	DIFODO$_{\mathrm{LR}}$	DIFODO$_{\mathrm{HR}}$	RDVO	GICP
f1-desk	**0.0466**	0.0476	0.0636	0.1829	**0.0877**	0.094	0.119	0.3638
f1-desk2	0.0937	**0.0793**	0.0846	0.2275	0.1873	**0.1554**	0.427	0.5808
f1-teddy	0.2412	**0.2041**	0.271	0.2849	0.5983	0.5147	**0.4823**	0.5598
f1-plant	0.0917	0.0645	**0.0456**	0.2444	0.2203	0.1667	**0.0962**	0.5392
f1-room	0.1094	**0.1088**	0.3319	0.3386	**0.1918**	0.2037	0.567	0.6238
f2-desk	**0.3424**	0.5602	0.3451	1.343	0.676	1.1663	**0.614**	2.848
f2-deskwp	0.2774	**0.2289**	0.259	0.6788	**0.4105**	0.4223	0.5001	1.5156
Average	**0.1718**	0.1848	0.2001	0.4714	**0.3388**	0.3890	0.4008	1.0044

Table III. It can be noticed that DIFODO, with both QVGA and QQVGA resolutions, is the most accurate method to estimate translations, whereas RDVO provides the best results for rotations. GIPC, on the other hand, is always considerably less accurate than the other two approaches. Moreover, absolute trajectory errors are presented in Table IV, where it can be seen that DIFODO is the method that estimates the whole trajectories more faithfully. Curiously, DIFODO$_{\mathrm{LR}}$ provides, on average, the best estimated trajectories although DIFODO$_{\mathrm{HR}}$ performs better locally. This apparent contradiction is caused by one single dataset: "f2-desk," where DIFODO$_{\mathrm{HR}}$ produces a particularly bad overall result.

A special case is the dataset "f2-deskwp" because it is the only one that purposely incorporates moving objects, breaking the rigid scene assumption on which DIFODO relies. RDVO includes in its formulation a weighting function to mitigate (but not eliminate) errors derived from moving objects and, hence, presents a considerable smaller relative RMSE than DIFODO for this dataset. Nevertheless, but for the stretches where the moving person appears, DIFODO performs better than RDVO and, surprisingly, is able to produce the best overall trajectory estimate even for this dataset where RDVO was expected to significantly outperform any other method.

C. DIFODO and Fast Motions

In this section, we study how the performance of our approach varies when the camera velocity increases. To this end, we simulate faster motions by time-decimating the input data from the TUM datasets. In this set of experiments, which have been carried out with QVGA resolution, the input image sequences are decimated by a factor of 2 and 3, representing velocities that are two and three times faster than the original ones. Relative and absolute pose errors are shown in Tables VI and VII, respectively. It can be noticed that the accuracy of our method decreases when a faster motion is simulated on the "f1 datasets," whereas the opposite occurs for the "f2 datasets." To understand these disparate results, we have to take into account that the average camera velocities at each dataset are quite different. In particular, the average camera velocities at the "f2 datasets" are about three times slower than those at the "f1 datasets," which explains why the decimated sequences of the "f2 datasets" provide good

TABLE VI
RELATIVE POSE ERRORS WITH THE ORIGINAL AND THE TIME-DECIMATED SEQUENCES

Datasets	RMSE Translational deviations (m/s)			RMSE Rotational deviations (deg/s)		
	Original	$\times 2$	$\times 3$	Original	$\times 2$	$\times 3$
f1-desk	**0.037**	0.050	0.089	**2.562**	4.275	6.259
f1-desk2	**0.053**	0.089	0.177	**3.311**	3.587	3.999
f1-teddy	**0.052**	0.077	0.109	**2.766**	4.410	7.145
f1-plant	**0.030**	**0.030**	0.035	2.179	**2.127**	2.413
f1-room	**0.049**	**0.049**	0.090	**2.688**	2.69	3.766
f2-desk	0.031	0.025	**0.024**	1.259	0.963	**0.883**
f2-deskwp	0.054	0.051	**0.049**	1.741	1.567	**1.472**
Average	**0.044**	0.053	0.082	**2.358**	2.803	3.705

TABLE VII
ABSOLUTE TRAJECTORY ERRORS WITH THE ORIGINAL AND THE TIME-DECIMATED SEQUENCES

Datasets	RMSE Translational deviations (m/s)			Maximum Translational deviations (deg/s)		
	Original	× 2	× 3	Original	× 2	× 3
f1-desk	**0.048**	0.171	0.236	**0.094**	0.418	0.464
f1-desk2	**0.079**	0.131	0.271	**0.155**	0.305	0.647
f1-teddy	**0.204**	0.256	0.350	**0.515**	0.564	0.613
f1-plant	**0.065**	0.065	0.090	0.167	**0.151**	0.216
f1-room	**0.109**	0.182	0.388	**0.204**	0.358	0.762
f2-desk	0.560	0.352	**0.262**	1.166	0.704	**0.468**
f2-deskwp	0.229	0.189	**0.178**	0.422	0.346	**0.328**
Average	**0.185**	0.192	0.254	**0.389**	0.406	0.500

results. On the contrary, the worst estimates are found for the accelerated sequences of “f1-desk” and “f1-desk2,” which present the fastest camera motions. It is worth noting that accuracy improves for the “f2 datasets” when the sequences are decimated or accelerated, effect that can be explained by analyzing the measurement noise of Kinect-like cameras. The depth images provided by this kind of cameras present a flickering or trembling over time that grows quadratically with depth [35]. This implies that no scene looks perfectly static to the camera even if the camera is still. For this reason, every new execution of the algorithm introduces some error in the trajectory estimation which, in the case of small motions, can be partially prevented with a lower sampling rate. This is essentially what causes the improved performance of the decimated “f2-sequences”.

In view of these results, we can conclude that DIFODO with QVGA provides very accurate estimates for camera velocities up to 0.7 m/s. If a faster motion needs to be estimated, QQVGA should be chosen to work at 60 Hz which would double the velocity estimation range without compromising the accuracy significantly (see comparative in Section VII-A). For very slow motions, it would be advantageous to reduce the frequency of the estimates, according to the results of the “f2 datasets”.

D. DIFODO in Real Time and Map Building

Finally, we have performed real experiments with a handheld camera to demonstrate the accuracy of our approach to estimate geometrically consistent trajectories. To this purpose, DIFODO has been utilized to build 3-D maps of two different scenarios (a living room and our lab). These maps are built as a concatenation of point clouds along the trajectory that DIFODO estimates, without resorting to global consistency or any other mapping strategy. The images are taken with a PrimeSense Carmine 1.08 at 30 Hz with QVGA resolution. A brief summary of the estimated trajectories is presented in Table VIII. The maximum velocity values have been obtained after applying a median filter to the sequence of velocities to reject possible outliers. Figs. 5 and 6 depict the generated maps from different views. In both cases, the real geometry of the mapped environments is preserved quite accurately: flat surfaces remain flat in the map (see the floor in Fig. 5), walls remain perpendicular to each other, etc. However, the scene colors are not consistent because the shutter speed of the camera was automatic, and therefore, the object colors vary depending on the average brightness of the scene. In any case, we do not aim to address the map building problem but to show that DIFODO is able to provide very accurate estimates not only locally but for full trajectories. Color has simply been added to the maps to enhance their appearance but, as has been repeatedly said throughout the paper, it is not employed to estimate motion.

TABLE VIII
REAL TRAJECTORIES ESTIMATED WITH DIFODO

	Lab	Living room
Duration (s)	33.66	20.31
Length (m)	18.65	8.611
Aver. trans. velocity (m/s)	0.554	0.424
Aver. rot. velocity (deg/s)	16.91	20.55
Max. trans. velocity (m/s)	0.841	0.770
Max. rot. velocity (deg/s)	47.95	49.32

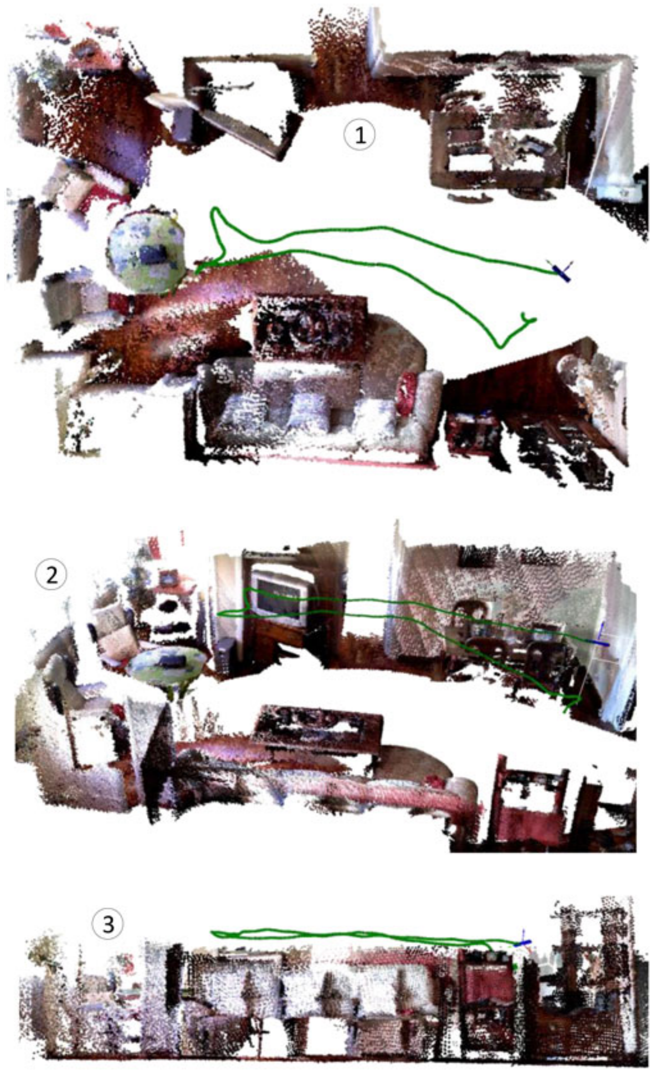

Fig. 5. Three-dimensional map of a living room generated with an RGB-D camera from DIFODO motion estimates. The map is shown in top (1), perspective (2) and front (3) views. The camera trajectory is depicted with a green line.

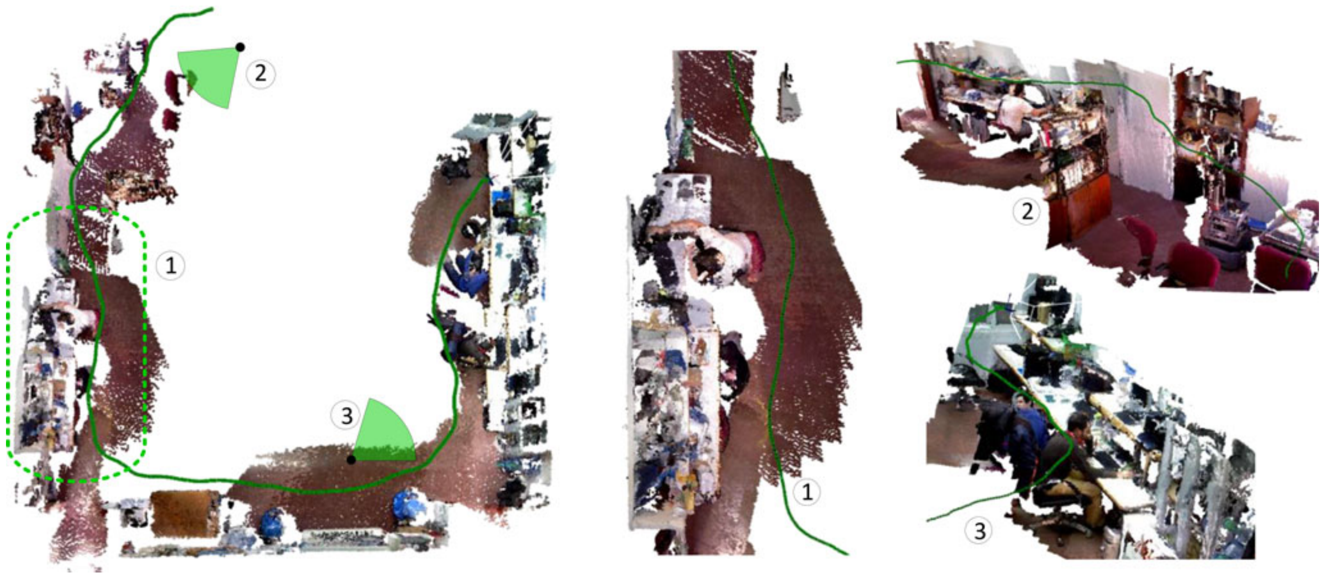

Fig. 6. Three-dimensional map of our lab generated with an RGB-D camera from DIFODO motion estimates. An overall top view is shown (left) together with some local views: a detailed top view (1) and two perspective views (2 and 3). The camera trajectory is depicted with a green continuous line.

VIII. Conclusion and Future Work

In this paper, we have introduced a novel visual odometry algorithm based on geometric data, and have detailed its formulation to work with range cameras. Within a coarse-to-fine scheme, the camera motion is estimated by assuming rigid motion of the scene with respect to the camera and finding the rigid body velocity that best describes the velocity of all the points of the scene. A velocity filter is proposed to detect and mitigate wrong estimates under cases of geometric uncertainty, and a detailed study on how to approximate the depth gradients accurately is presented.

In terms of speed and precision, our approach has been compared with GICP [1] and RDVO [2]. With respect to GICP, which also estimates motion from geometry, our method is about 30 times faster and more than 2 times more accurate. Regarding RDVO, which needs geometric and photometric data to work, similar results (or even slightly better) are obtained from purely geometric inputs. Maps of two different scenarios have been built from real-time odometry estimates to demonstrate qualitatively that DIFODO is able to reproduce full trajectories consistently. Furthermore, DIFODO has proved to provide accurate motion estimates with low image resolutions (120×160) which makes it suitable for real-time robotic applications that might involve fast motions and demand a higher frame rate (60 Hz).

On the other hand, there are some factors that limit the performance of our approach. First, the currently available range cameras are not very precise and generate depth images that considerably deform the real geometry of the scenes they observe. This defect will surely be alleviated at the upcoming new generations of range sensors given the attention that big companies like Microsoft, Apple or Samsung are paying to 3-D sensing. Second, its accuracy worsens when the rigid scene hypothesis is not fulfilled. We believe that the real solution to this problem will be given by scene flow algorithms that estimate the dense 3-D motion field of the scene points. Particular solutions (like that of [2]) can help to deal with moving objects if a very high percentage of the scene is still rigid, but would fail to estimate motion when the number of moving objects increases or the moving object itself represents a substantial part of the whole scene. Therefore, a more general solution that could be generalized to any arbitrary scene composed of any arbitrary number of moving objects should be found. Last, similarly to GICP or other geometry-based VO methods, DIFODO is unable to estimate some linear or angular velocity components when the scene does not present sufficient geometric-distinctive features. Although the proposed velocity filter helps to mitigate this limitation, a more robust solution should incorporate RGB or inertial information to effectively tackle the problem of ill-posed configurations.

Appendix I

This appendix derives the expression of the matrix Σ_d which, according to (13), is composed of the following blocks:

$$\Sigma_{xyz} = \begin{pmatrix} \sigma_x^2 & \sigma_{xy} & \sigma_{xz} \\ \sigma_{xy} & \sigma_y^2 & \sigma_{yz} \\ \sigma_{xz} & \sigma_{yz} & \sigma_z^2 \end{pmatrix},$$

$$\Sigma_{Z_{t,u,v}} = \begin{pmatrix} \sigma_{Z_t}^2 & \sigma_{Z_t Z_u} & \sigma_{Z_t Z_v} \\ \sigma_{Z_t Z_u} & \sigma_{Z_u}^2 & \sigma_{Z_u Z_v} \\ \sigma_{Z_t Z_v} & \sigma_{Z_u Z_v} & \sigma_{Z_v}^2 \end{pmatrix},$$

$$\Sigma_{(xyz)(Z_{t,u,v})} = \begin{pmatrix} \sigma_{xZ_t} & \sigma_{xZ_u} & \sigma_{xZ_v} \\ \sigma_{yZ_t} & \sigma_{yZ_u} & \sigma_{yZ_v} \\ \sigma_{zZ_t} & \sigma_{zZ_u} & \sigma_{zZ_v} \end{pmatrix}. \quad (31)$$

Assuming that the only source of error is the inaccuracy of the depth measurements, we first characterize this error according to the work of Khoshelham and Elberink [35]:

$$\sigma_{zm} = \frac{\sigma_d}{fb} z^2 \sim 1.4 \cdot 10^{-5} z^2 \tag{32}$$

where σ_{zm} is the standard deviation of the depth measured at a given pixel, f is the focal distance of the IR camera, b is the baseline between the IR camera and projector, and σ_d is the disparity measurement error. Since the depth images are filtered with a Gaussian mask of 5×5, the actual standard deviation of z should take it into account:

$$\sigma_z = \frac{\sigma_d}{5fb} z^2 = k_z z^2 \sim 2.8 \cdot 10^{-6} z^2. \tag{33}$$

Regarding the Pin-Hole model (see (3) and (4)), it can be deduced that

$$\sigma_x = \frac{(u - u_m)}{f_x} \sigma_z = k_z x z \tag{34}$$

$$\sigma_y = \frac{(v - v_m)}{f_y} \sigma_z = k_z y z. \tag{35}$$

On the other hand, the standard deviation of the depth derivatives should be computed as the difference of Gaussians but, for simplicity's sake, it is calculated as if both the subtrahend and minuend of the finite difference expressions were equal to the value of z at the corresponding pixel, i.e.,

$$z^{t+1}(u,v) \sim z^t(u \pm 1, v) \sim z^t(u, v \pm 1) \sim z^t(u,v) \Rightarrow$$

$$\sigma_{Z_t} = \frac{k_z z^2}{\sqrt{2}\Delta t}, \sigma_{Z_u} = \frac{k_z z^2}{2\sqrt{2}}, \sigma_{Z_v} = \frac{k_z z^2}{2\sqrt{2}}. \tag{36}$$

The covariance terms in (31) can be obtained analytically, but only those which are not null will be explicitly derived. The null terms correspond to independent variables (Z_t, Z_u, and Z_v respect to each other and x, y, and z) and, hence, do not need further consideration.

$$\begin{aligned}\sigma_{xy} &= E\left[(x - E[x])(y - E[y])\right] \\ &= \frac{(u - u_m)}{f_x}\frac{(v - v_m)}{f_y} E\left[(z - E[z])^2\right] \\ &= k_z^2 x y z^2 \qquad (37)\\ \sigma_{xz} &= E\left[(z - E[z])(x - E[x])\right] \\ &= \frac{(u - u_m)}{f_x} E\left[(z - E[z])^2\right] = k_z^2 x z^3 \qquad (38)\\ \sigma_{yz} &= E\left[(z - E[z])(y - E[y])\right] \\ &= \frac{(v - v_m)}{f_y} E\left[(z - E[z])^2\right] = k_z^2 y z^3 \qquad (39)\\ \sigma_{zZ_t} &= \sigma_{zZ_u} = \sigma_{zZ_v} = \sigma_{xZ_t} = \sigma_{xZ_u} = \sigma_{xZ_v} = 0 \\ \sigma_{yZ_t} &= \sigma_{yZ_u} = \sigma_{yZ_v} = \sigma_{Z_t Z_u} = \sigma_{Z_t Z_v} \\ &= \sigma_{Z_u Z_v} = 0. \qquad (40)\end{aligned}$$

This is all the information required to build the matrix Σ_d. A similar study is presented in [20] but based on different assumptions and applied to a different set of variables. It focuses on the 3-D uncertainty of a point (variables x, y, and z above) and, although they also model the pixel coordinates u and v as normal distributions, their results are similar to ours if the variance of u and v are set to zero.

ACKNOWLEDGMENT

The authors would like to thank J. L. Blanco Claraco and J. R. Ruiz Sarmiento for their help and support in the process of code implementation, and also E. Fernández Moral for fruitful discussions. They would also like to thank C. Kerl regarding the use of his code.

REFERENCES

[1] A. Segal, D. Haehnel, and S. Thrun, "Generalized-ICP," presented at the Robot., Sci. Syst., Seattle, WA, USA, 2009.

[2] C. Kerl, J. Sturm, and D. Cremers, "Robust odometry estimation for RGB-D cameras," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2013, pp. 3748–3754.

[3] O. J. Woodman, "An introduction to inertial navigation," Computer Laboratory, University of Cambridge, Cambridge, U.K., Tech. Rep. UCAMCL-TR-696, 2007.

[4] H. Spies, B. Jähne, and J. L. Barron, "Range flow estimation," *Comput. Vision Image Understand.*, vol. 85, no. 3, pp. 209–231, 2002.

[5] J. Gonzalez, "Recovering motion parameters from a 2D range image sequence," in *Proc. IEEE Int. Conf. Pattern Recog.*, 1996, pp. 433–440.

[6] J. Gonzalez and R. Gutierrez, "Direct motion estimation from a range scan sequence," *J. Robot. Syst.*, vol. 16, no. 2, pp. 73–80, 2009.

[7] "The Mobile Robot Programming Toolkit. (2015, May 16). (MRPT)," [Online]. Available: http://www.mrpt.org/

[8] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry," in *Proc. IEEE Comput. Soc. Conf. Comput. Vision Pattern Recog.*, 2004, pp. 652–659.

[9] D. Scaramuzza and F. Fraundorfer, "Visual odometry [tutorial]," *IEEE Robot. Autom. Mag.*, vol. 18, no. 4, pp. 80–92, Dec. 2011.

[10] M. Maimone, Y. Cheng, and L. Matthies, "Two years of visual odometry on the mars exploration rovers," *J. Field Robot.*, vol. 24, no. 3, pp. 169–186, 2007.

[11] K. Konolige, M. Agrawal, and J. Sola, "Large-scale visual odometry for rough terrain," *Robot. Res.*, vol. 66, pp. 201–212, 2011.

[12] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[13] J. Engel, J. Sturm, and D. Cremers, "Semi-dense visual odometry for a monocular camera," presented at the IEEE Int. Conf. Comput. Vision, Sydney, Australia, 2013.

[14] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 2, pp. 239–256, 1992.

[15] D. M. Cole and P. M. Newman, "Using laser range data for 3D SLAM in outdoor environments," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2006, pp. 1556–1563.

[16] J. S. Gutmann and K. Konolige, "Incremental mapping of large cyclic environments," in *Proc. Int. Symp. Comput. Intell. Robot. Autom.*, 1999, pp. 318–325.

[17] E. B. Olson, "Real-time correlative scan matching," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2009, pp. 4387–4393.

[18] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy, "Visual odometry and mapping for autonomous flight using an RGB-D camera," presented at the Int. Symp. Robot. Res., Flagstaff, AZ, USA, 2011.

[19] M. Fiala and A. Ufkes, "Visual odometry using 3-dimensional video input," in *Proc. Can. Conf. Comput. Robot Vision*, 2011, pp. 86–93.

[20] I. Dryanovski, R. G. Valenti, and J. Xiao, "Fast visual odometry and mapping from RGB-D data," presented at the IEEE Int. Conf. Robot. Autom., Karlsruhe, Germany, 2013.

[21] 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.

[22] T. Tykkala, C. Audras, and A. I. Comport, "Direct iterative closest point for real-time visual odometry," in *Proc. IEEE Int. Conf. Comput. Vision Workshops*, 2011, pp. 2050–2056.

[23] F. Steinbrucker, J. Sturm, and D. Cremers, "Real-time visual odometry from dense RGB-D images," in *Proc. IEEE Int. Conf. Comput. Vision Workshops*, 2011, pp. 719–722.

[24] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *Proc. Int. Conf. Intell. Robot Syst.*, 2012, pp. 573–580.

[25] V. Panwar, P. Jasiobedzki, and M. Greenspan, "Interest point sampling for range data registration in visual odometry," Ph.D. dissertation, Dept. Electr. Comput. Eng., Queen's Univ., Belfast, U.K., 2011.

[26] R. A. Newcombe, A. J. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim, and A. Fitzgibbon, "KinectFusion: Real-time dense surface mapping and tracking," in *Proc. IEEE Int. Symp. Mixed Augmented Reality*, 2011, pp. 127–136.

[27] D. R. Canelhas, T. Stoyanov, and A. J. Lilienthal, "SDF Tracker: A parallel algorithm for on-line pose estimation and scene reconstruction from depth images," presented at the Int. Conf. Intell. Robots Syst., Tokyo, Japan, 2013.

[28] T. Whelan, J. McDonald, M. Kaess, M. Fallon, H. Johannsson, J. Leonard, and J. McDonald, "Kintinuous: Spatially extended kinectfusion," presented at the 3rd RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras, Sydney, Australia, July 2012.

[29] T. Whelan, H. Johannsson, M. Kaess, J. J. Leonard, and J. McDonald, "Robust real-time visual odometry for dense RGB-D mapping," presented at the IEEE Int. Conf. Robot. Autom., Karlsruhe, Germany, 2013.

[30] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert, "High accuracy optical flow estimation based on a theory for warping," presented at the Eur. Conf. Comput. Vision, Prague, Czech Republic, 2004.

[31] G. Guennebaud, B. Jacob, and Others. (2010). Eigen v3. [Online]. Available: http://eigen.tuxfamily.org

[32] N. Gelfand, L. Ikemoto, S. Rusinkiewicz, and M. Levoy, "Geometrically stable sampling for the ICP algorithm," in *Proc. IEEE 4th Int. Conf. 3-D Digital Imag. Model.*, 2003, pp. 260–267.

[33] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Ng, "ROS: An open-source robot operating system," presented at the Int. Conf. Robot. Autom. Workshop Open Source Softw., Kobe, Japan, 2009.

[34] B. R. Rusu and S. Cousins, "3d is here: Point cloud library (PCL)," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2011, pp. 1–4.

[35] K. Khoshelham and S. O. Elberink, " Accuracy and resolution of kinect depth data for indoor mapping applications," *Sensors*, vol. 12, no. 2, pp. 1437–1454, 2012.

Mariano Jaimez (S'15) received the B.S. and M.S. degrees in mechatronics from University of Málaga, Málaga, Spain, in 2010 and 2012, respectively. Since 2013, he has been working toward the Ph.D. degree with the Department of Ingeniería de Sistemas y Automática, University of Málaga and, since February 2015, he has also been a joint Ph.D. student with the Computer Vision Group, Technical University of Munich, Munich, Germany.

His research interests include motion estimation with range sensors and RGB-D cameras, mainly visual odometry, and scene flow.

Javier Gonzalez-Jimenez (M'92) received the B.S. degree in electrical engineering from University of Seville, Seville, Spain, in 1987. He received the Ph.D. degree from the Department of Ingeniería de Sistemas y Automática, University of Málaga, Málaga, Spain, in 1993.

In 1990–1991, he was with the Field Robotics Center, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA, working on mobile robots as part of his Ph.D. Since 1996. he has been leading Spanish and European projects on mobile robotics and perception. He is currently a Professor with the University of Málaga and Head of the Machine Perception and Intelligent Robotics Group. His research interest includes mobile robot navigation, olfactory robotics, and computer vision. In these fields, he has published three books and more than 150 papers.