Comparing Optical Flow Algorithms Using 6-DOF Motion of Real-World Rigid Objects

Marco Mammarella, Giampiero Campa, *Member, IEEE*, Mario L. Fravolini, and Marcello R. Napolitano

***Abstract*—The application of optical flow algorithms to guidance and navigation problems has gained considerable interest in recent years. This paper summarizes the results of a comparative study on the accuracy of nine different optical flow (OF) algorithms using videos that are captured from an on-board camera during the flight of an autonomous aircraft model. The comparison among the algorithms relies on two formulas that are used both to calculate the ideal OF generated by the motion of a rigid body in the camera field of view and to estimate the linear and angular velocity from the OF.**

***Index Terms*—Computer vision, image processing, machine vision, optical flow (OF).**

I. Introduction

METHODS to calculate the optical flow (OF) associated with a sequence of images have been receiving increasing attention in recent years since it became clear that OF-based algorithms have promising capabilities for applications within a wide range of problems. In fact, OF can be used for a variety of purposes, such as motion detection and estimation, collision detection and avoidance, shape reconstruction, and object segmentation. In particular, the use of OF-based navigation and control solutions for autonomous vehicles [1]–[4] is becoming appealing since a significant amount of scientific evidence indicates that different kinds of insects utilize OF to perform quick and highly accurate navigation maneuvers in complex environments [5], [6]. However, significant research issues have yet to be addressed before wider scale applications of OF-based guidance, and navigation control schemes can be finally implemented. For example, to date most OF algorithms do not provide the necessary accuracy, along with the computational efficiency, necessary for real-time implementations.

Real-image sequences often present characteristics that are intrinsically difficult to model, such as motion discontinuities, complex 3-D surfaces, camera noise, specular highlights, shadows, transparencies, atmospheric effects, and other sources of disturbances. Therefore, it is critical to be able to assess the performance of different OF algorithms using real videos, as opposed to sequences of synthetically generated images.

Unfortunately, a quantitative evaluation of these algorithms in real-world settings is complicated by the fact that it is typically very difficult, and at times impossible, to establish "ground truth" baseline values for the OF generated by the complex motion of 3-D objects in the field of view. This is mostly due to the fact that such "ground truth" flow (hereafter also referred as "ideal" OF) depends on the position, the orientation, as well as the translational and rotational velocities of the objects with respect to the camera, which are not always easily measurable. Consequently, to date, most of the evaluation studies have relied either on standard sequences—generated by progressively translating, rotating, or distorting a real base image [7]—or on computer-generated image sequences [8].

In [9], a robot arm was used to move a camera along a predefined trajectory, which provided the ideal flow for very simple images of a scene consisting of polyhedral objects, while the case of OF generated by multiple objects was considered in [10]. In [11], a method to generate ground-truth motion fields from real sequences containing polyhedral objects was presented along with a test suite to benchmark OF algorithms. However, this method requires the user to select the corners of the polyhedral object for each image; therefore, the approach is impractical for longer sequences or for sequences without clearly defined polyhedral objects.

In [12], the ground-truth flow for real images was determined using hidden fluorescent texture painted on a scene, along with a computationally intensive approach to track small windows in a sequence of UV images.

The first contribution of this paper is a simplified approach to compute the ideal OF generated by a single rigid object that rotates and translates along all axes with respect to the camera (or by the camera moving and translating in a static scene). Both these cases are particularly significant for real-life robotic applications. To the author's knowledge, projection-based methods for calculating the ground-truth OF generated from a rigid body are limited to very specific simplified cases (e.g., translation only or rotation only), and a comprehensive formula that addresses simultaneous rotation and translation has rarely been considered.

The second original contribution is a static inversion formula to estimate the camera/rigid-object relative linear and angular velocities from the OF data. The use of stateless pseudoinversion techniques to estimate the velocities of a rigid object (subject to complex six degree-of-freedom (6-DOF) motion) from its OF field is, as far as the authors can tell, unprecedented within the classical OF literature.

Manuscript received November 9, 2011; revised June 7, 2012; accepted August 29, 2012. Date of current version December 17, 2012. This paper was recommended by Associate Editor T. Busch.

M. Mammarella is with GMV Aerospace and Defence, Space System B.U., Tres Cantos, Madrid 28760, Spain (e-mail: marco__mm@hotmail.com).

G. Campa is with MathWorks, Torrance, CA 90502 USA (e-mail: giampiero.campa@gmail.com).

M. L. Fravolini is with the Department of Electronics and Information Engineering, University of Perugia, Perugia 06123, Italy (e-mail: mario.fravolini@diei.unipg.it).

M. R. Napolitano is with the Department of Mechanical and Aerospace Engineering, West Virginia University, Morgantown, WV 26505 USA (email: marcello.napolitano@mail.wvu.edu).

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/TSMCC.2012.2218806

The final, but arguably at least equally important, original contribution of this study is the development of a quantitative performance metric (based on the developed formulae) to compare the accuracy of OF algorithms using images of rigid objects subject to rotational and translational motions (or, equivalently, images from a camera moving in a static scene). Specifically, the performance metric is composed of two parts: The first part is an extension of criteria first used in [9], which is based on the deviation between the OF field provided by a given algorithm and the ideal OF, calculated using the known rotational and translational object velocities. The second part relies on the comparison between the known (rotational and translational) velocities of a moving object, and the estimated velocities that are obtained using the information provided by the OF field.

The application of the formulas is illustrated using a sequence of real images, and the performance metric is used to compare nine different OF algorithms.

The rest of this paper is organized as follows. A basic review of the most important concepts is introduced in Section II. The novel formulas are derived in Sections III and IV. The experimental setups and the performance metrics are introduced in Section V, and the results are shown in Sections VI and VII.

II. Optical Flow

According to Horn and Schunck [13]: "The optical flow is a velocity field in the image that transforms one image into the next image in a sequence. As such it is not uniquely determined; the motion field, on the other hand, is a purely geometric concept, without any ambiguity—it is the projection into the image of three-dimensional (3-D) motion vectors." In this paper, we will use the general term "OF" to refer mostly to the "motion field," which, in general, can be calculated from a sequence of images by relying on three fundamental assumptions.

1) *Object brightness invariance:* The local changes in image intensity are caused only by the motion of a certain object with respect to the camera.
2) *Spatial coherence:* The motion is uniform over a small patch of pixels.
3) *Temporal persistence:* The image motion of a surface patch changes gradually over time.

$I(u, v, t)$ is defined as the intensity (i.e., brightness) of a pixel that has horizontal and vertical image plane coordinates u, v, and represents a feature which moves of δu, δv during the time δt. This leads to the equation $I(u, v, t) = I(u+\delta u, v+\delta v, t+\delta t)$. A derivation with respect to time leads to the following brightness conservation equation:

$$I_u \dot{u} + I_v \dot{v} + I_t = 0 \tag{1}$$

where I_u and I_v are the derivatives of I along the u and v image dimensions, calculated at the given pixel location u,v and time t. It is the temporal derivative of the image at the same location and time, and finally the terms $\dot{u}$ and $\dot{v}$ represent the "OF" at the point u, v at time t. Note that since cameras acquire images at a certain frame rate $1/T$—where T is the time interval in seconds between one image and the next one—the aforementioned time derivatives are commonly calculated using a first- or second-order discrete approximation.

In general, OF algorithms can be classified within the following broad classes:

1) "gradient" methods;
2) "phase" methods;
3) "region-based matching" methods;
4) "feature-based" methods.

A. Gradient Methods

Generally speaking, gradient methods attempt to solve (1) to calculate the unknowns $\dot{u}$ and $\dot{v}$. However, it can be noticed that for each pixel, there is one corresponding scalar equation (with known values of I_u and I_v and I_t), while there are two scalar unknowns $\dot{u}$ and $\dot{v}$, which lead to an analytically underdetermined algebraic system. This is also known as the "aperture problem." The methods that belong to the "gradient" class typically tackle this problem by including some constraints—usually based on some form of spatial or temporal coherence—in the algebraic system of equations to be solved. Within this effort, four algorithms that belong to this category have been analyzed: "gradient" [8], "Lucas–Kanade" [14], "Horn and Shunck" [13], and "Proesmans" [15].

The "gradient" algorithm—developed by the authors according to the guidelines that are presented in [8], and implemented as a MATLAB function—calculates the OF for each pixel that belongs to a predefined grid, assuming that $\dot{u}$ and $\dot{v}$ are constants within a certain spatial and temporal neighborhood of the pixel. Therefore, an overdetermined system of equations is assembled and solved—in the minimum square sense—for each considered pixel. Crucially, the system is solved only if its eigenvalues are greater than a given threshold. This allows discarding image areas, where derivatives are too close to zero or too similar to each other—e.g., due to a lack of motion or because there are no distinguishable features—and, at the same time, increasing computational efficiency by avoiding unnecessary calculations. The "Lucas–Kanade" and "Horn and Shunck" implementations are available in recent MATLAB versions as Simulink blocks.

The "Lucas–Kanade" algorithm is similar to the "gradient" implementation with the main difference being the assignment of numerical weights with the goal to give different weights to the neighborhoods as a function of their distances from the considered point. Both these algorithms lend themselves naturally to parallel implementations [16], [17].

The "Horn and Schunck" algorithm combines (1) with a global smoothness term λ with the goal to constrain the estimated velocity. This algorithm also features an iterative procedure, which is halted when the maximum number of iterations is reached.

The "Proesmans" method is conceptually similar to the "Horn and Schunck" method since it requires the minimization of a global energy functional; the main difference is that it takes into account the bias in the direction of motion due to correlations in the finite difference approximation [15]. The algorithm was originally developed in C++ at the University of Otago, New Zealand, and it was later revised by one of the authors.

Specifically, all the subfunctions were included within a single C++ file, and an interface that allowed the algorithm to be called from MATLAB was added.

The typical advantage of this class of algorithms is their computational speed. Their main disadvantages are that they need to cope with the aperture problem; additionally, the calculation of the spatial and temporal derivatives is usually very prone to errors—especially in situation of relatively fast motion [16], [18]—due to the presence of noise from different sources.

B. Phase Methods

Phase techniques are based on the idea that 2-D image velocity can be modeled as the phase behavior of a band-pass filter output [8]. The idea of using phase information for OF calculation purposes was originally developed by Fleet and Jepson [19]. The resulting algorithm is available from the MATLAB Central File Exchange Site [20]. As outlined in [21], the algorithm calculates the OF estimation using the following three sequential steps. First, a spatial filtering is obtained using the Gabor filter, and the temporal phase gradient is calculated using the estimation of the velocity components. Second, a component velocity is rejected if the corresponding filter pair's phase information is not linear over a given time span. Third, an interpolation is used to combine the partial velocities obtained using Gabor filters in order to achieve the OF in the u and v directions. These methods are known to work well for relatively slow motion [16]; however, they are not reliable when trying to estimate fast motion; in fact, their accuracy is significantly degraded due to temporal aliasing [18].

C. Region Matching-Based Techniques

For methods that belong to the "region-based matching" class, the OF vector $[\dot{u}, \dot{v}]$ is calculated for a given pixel by finding the displacement of a template around the pixel between two consecutive frames. The template matching between two consecutive frames is usually performed by minimizing a predefined function of the differences between the two templates. Within this effort, the "difference" and the "correlation" methods have been considered and implemented. The "difference" algorithm uses the sum of the absolute differences (SAD) among templates belonging to consecutive frames to find the best matching templates [8]. The "correlation" algorithm, instead, calculates the correlation among templates, which is used as the metric to select the matching. Both algorithms were developed by the authors and implemented in MATLAB. In general, algorithms that belong to this category are known to behave better than algorithms that belong to the "gradient" category for fast motion situations, but at the expense of being computationally more demanding. Specifically, their computation time increases quadratically with the maximum allowed object displacement.

D. Feature-Matching-Based Methods

These methods calculate $\dot{u}$ and $\dot{v}$ by measuring the displacements of certain image features—as detected by a feature-detection algorithm and later associated by a feature-matching algorithm—between two consecutive frames. Therefore, they implicitly rely on the assumption that the same image features can consistently be detected and tracked over different image frames. However, unlike algorithms that belong to other classes, they do not necessarily rely on the assumption that the distance between the positions of the same features between two consecutive image frames must be contained within predefined limits. It should also be emphasized that the image points for which these algorithms provide OF results not only do belong to an evenly spaced grid, but also change their position for different image frames.

In this effort, two different feature-based methods have been considered: Harris [22] and scale-invariant feature transform (SIFT) [23]. The Harris algorithm—coded in MATLAB by the authors using the guidelines that are provided in [22]—is a corner detection algorithm allowing the extraction of the position of specific corners with some robustness to real-world conditions such as changes in illumination. Next, an algorithm that was previously developed and coded in C [24] was used to match the corners detected to consecutive frames of an image sequence.

The other feature-detection algorithm is known as SIFT, and was developed with the goal to detect and associate the same features between different images. Specifically, features are detected using a filtering approach that identifies stable points in the scale space, and are then associated using a descriptor-based approach [23]. Empirical experience has shown that the precision of the OF methods belonging to this class strongly depends on the performance of the associated matching algorithm. While these algorithms also tend to be computationally demanding, parallel implementations are also possible [25].

III. Derivation of the Ideal Optical Flow

The comparison between the different OF algorithms is based on the calculation of the "ideal" OF (or ideal flow) generated by the motion of a (single) rigid object in space. Given any point on the image plane, the ideal flow can be computed from the position and the velocity—with respect to the camera—of the point in the field of view that generates—by projection—the OF.

Specifically, a "pinhole" mathematical model of the camera [26] is assumed:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \frac{f}{x_c} \begin{bmatrix} y_c \\ z_c \end{bmatrix} \tag{2}$$

where f is the camera focal length, and u and v are—as previously described—the horizontal and vertical coordinates of a point in the image plane resulting from the projection of the point ${}^{C}P = [x_c, y_c, z_c]^T$ on such plane.

Note that the left superscript "C" in ${}^{C}P$ indicates that the point is expressed with respect to a camera-fixed reference frame, which is centered in the camera plane and has its x-axis pointing in the direction of view, and its y- and z-axis pointing, respectively, in the directions of u and v of the image plane. Assuming that ${}^{C}P$ is part of a rigid body centered in ${}^{C}O_B$ and moving with respect to the camera reference frame with a linear

velocity ${}^{C}\mathrm{V}_{B/C}$ and angular velocity ${}^{C}\omega_{B/C}$, differentiating (2) with respect to time, and using standard kinematics relationships [27] to express the derivative of ${}^{C}P$ yields

$$\begin{bmatrix}\dot{u}\\ \dot{v}\end{bmatrix} = f\begin{bmatrix}-\dfrac{y_c}{x_c^2} & \dfrac{1}{x_c} & 0\\ -\dfrac{z_c}{x_c^2} & 0 & \dfrac{1}{x_c}\end{bmatrix} \times \left[{}^{C}V_{B/C} + {}^{C}\omega_{B/C} \otimes \left({}^{C}P - {}^{C}O_B\right)\right] \tag{3}$$

where $\dot{u}$ and $\dot{v}$ represent the *ideal* OF (at the image coordinates u and v) generated by the motion of ${}^{C}P$, and $\otimes$ indicates the 3-D cross product. Of course, while (3) is valid for all points that belong to the considered rigid body, only the points that belong to the visible part of the body surface contribute to the OF field on the image plane. In addition note that whenever the whole environment moves with respect to the camera (in other words, the camera moves in a static environment), the body reference frame can be selected to be centered in the camera reference frame, implying that the coordinates of ${}^{C}O_B$ are equal to zero.

IV. Extracting Velocity Information From the Optical Flow

The direct relationship in (3) can be used to extract relative velocity information from sequences of images. If the motion happens along one axis only (rotation or translation), the problem can be simplified as follows (see [28] for an alternative derivation and related experiments).

A. Rotational Motion Around One Axis

This section shows how the angular velocity of a known rigid object rotating along an axis that is parallel to the camera axis can be calculated using the OF. Specifically, for each OF vector the point ${}^{C}P$ in (3) is a point of the object subjected to the rotational motion, while the point ${}^{C}O$ is the center of rotation. Then, for each OF vector, a corresponding estimated angular velocity can be computed by setting to zero the other two components of the vector $\omega_{B/C}$ and the three components of the vector $V_{B/C}$, and then pseudoinverting (3). Specifically, zeroing out ${}^{C}\mathrm{V}_{B/C}$ and the last two components of $\omega_{B/C}$ in (3) yields

$$\begin{bmatrix}\dot{u}\\ \dot{v}\end{bmatrix} = f\begin{bmatrix}-\dfrac{y_c}{x_c^2} & \dfrac{1}{x_c} & 0\\ -\dfrac{z_c}{x_c^2} & 0 & \dfrac{1}{x_c}\end{bmatrix}\begin{bmatrix}0\\ -\left({}^{C}\omega_{B/C}\right)_x\left({}^{C}P - {}^{C}O_B\right)_z\\ \left({}^{C}\omega_{B/C}\right)_x\left({}^{C}P - {}^{C}O_B\right)_y\end{bmatrix} \tag{4}$$

where $({}^{C}\omega_{B/C})_x$ is the unknown angular velocity, while $({}^{C}\mathrm{P}-{}^{C}O_B)_y$ and $({}^{C}\mathrm{P}-{}^{C}O_B)_z$ are the second and third components of the vector $({}^{C}\mathrm{P}-{}^{C}O_B)$. Multiplying the terms on the right-hand side of (4) yields

$$\begin{bmatrix}\dot{u}\\ \dot{v}\end{bmatrix} = \frac{f}{x_c}\begin{bmatrix}-\left({}^{C}\omega_{B/C}\right)_x\left({}^{C}P - {}^{C}O_B\right)_z\\ \left({}^{C}\omega_{B/C}\right)_x\left({}^{C}P - {}^{C}O_B\right)_y\end{bmatrix}. \tag{5}$$

Since $\dot{u}$ and $\dot{v}$ are provided by the OF algorithm and are therefore known, it is possible to solve for $({}^{C}\omega_{B/C})_x$ in two different ways. Specifically, (5) can be solved by inverting the first equation in (5) (in the $\dot{u}$ component only) or by inverting the second equation in (5) (in the $\dot{v}$ component only). Both approaches can be pursued; however, better results are obtained when the two approaches are mixed together and the final value for the angular velocity for a given OF vector is obtained by averaging the two outcomes [28]:

$$\left({}^{C}\omega_{B/C}\right)_x = \frac{1}{2}\left(\frac{\dot{v}x_c}{f\left({}^{C}P - {}^{C}O_B\right)_y} - \frac{\dot{u}x_c}{f\left({}^{C}P - {}^{C}O_B\right)_z}\right). \tag{6}$$

The total angular velocity is then computed by averaging the estimated velocity over the section of the image that represents the rotating object. Note that if the object center of rotation is along the x-axis of the camera reference frame, then both components of ${}^{C}O_B$ along the x- and y-dimensions are zero; therefore, using (3) and (6) reduces to

$$\left({}^{C}\omega_{B/C}\right)_x = \frac{1}{2}\left(\frac{\dot{v}}{u} - \frac{\dot{u}}{v}\right) \tag{7}$$

which allows calculating the object angular velocity from the images, without knowing either f or CP. Formulas similar to (6) can be derived for the rotational motion around the other two axes.

B. Translational Motion Along One Axis

An estimate of the velocity of a rigid object translating in the field of view along a single axis can be easily calculated from the OF. Specifically, for each OF vector, a corresponding estimated velocity can be calculated by setting to zero both $\omega_{B/C}$ and the other two components of $V_{B/C}$ in (3) and pseudoinverting the formula. For example, if the object is translating along the x-axis,

$$\left({}^{C}V_{B/C}\right)_x = -\frac{x_c^2}{f}\begin{bmatrix}y_c\\ z_c\end{bmatrix}^{\dagger}\begin{bmatrix}\dot{u}\\ \dot{v}\end{bmatrix} \tag{8}$$

where the symbol "†" indicates the pseudoinverse operation. Clearly, a more accurate estimation of the object velocity can be calculated by averaging the estimated velocities over the portion of the image that represents the translating object.

C. Translational Motion Along Two Axes

Even the velocity of an object that translates along two axes can be estimated using OF. Specifically, for each OF vector, a corresponding velocity can be calculated by setting to zero both $\omega_{B/C}$ and the component of $V_{B/C}$ in (3) that does not have to be considered, and inverting the formula. For example, zeroing out ${}^{C}\omega_{B/C}$ and the first component of ${}^{C}\mathrm{V}_{B/C}$ leads to the estimation of the velocity in the components y and z:

$$\begin{bmatrix}\left({}^{C}V_{B/C}\right)_y\\ \left({}^{C}V_{B/C}\right)_z\end{bmatrix} = \frac{1}{f}\begin{bmatrix}x_c & 0\\ 0 & x_c\end{bmatrix}\begin{bmatrix}\dot{u}\\ \dot{v}\end{bmatrix}. \tag{9}$$

For each axis, the total velocity can be calculated by averaging the estimated velocities over the portion of the image representing the moving object.

D. Six Degree-of-Freedom Motion

In the following section, the problem of recovering the relative 6-DOF motion of the camera with respect to a static environment will be considered. This problem is also related to the more general "structure from motion" problem [29]–[31], which consists of estimating a sequence of relative positions and orientations from the motion of several points in the image. In particular, we focus on the possibility to recover from the OF the relative position and attitude of a camera moving over a planar terrain; this scenario is of particularly relevance for real-life aerial robotics applications.

Unlike the previous three simplified cases, for a general 6-DOF motion the velocities cannot be expressed directly—that is by directly inverting (3)—as a function of the instantaneous OF field in a single point in the image plane, because the problem is underdetermined (two known variables, namely the elements of the OF vector, and six unknowns, namely the elements of the rotational and translational velocity vectors). In other words, there is no additional *a priori* information about the structure of the motion (e.g., motion constrained along a certain known axis) that could be used to compensate for the information lost during the projection.

However, the overall information of all the OF vectors in the image plane can still be used to estimate both the translational and rotational velocities of the camera. Specifically, rearranging (3) by collecting the velocity terms in a 6×1 vector yields

$$\begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} = M(f, {}^{C}P, {}^{C}O_B) \begin{bmatrix} {}^{C}V_{B/C} \\ {}^{C}\omega_{B/C} \end{bmatrix} \tag{10}$$

with $M(f, {}^{C}P, {}^{C}O_B)$ being the following 2×6 matrix:

$$M(f, {}^{C}P, {}^{C}O_B) = \frac{f}{x_c} \begin{bmatrix} -\frac{y_c}{x_c} & 1 & 0 & z_o - z_c & \cdots \\ -\frac{z_c}{x_c} & 0 & 1 & y_c - y_o & \end{bmatrix} \begin{matrix} \cdots \\ \end{matrix} \begin{bmatrix} \frac{y_c(z_o - z_c)}{x_c} & \frac{y_c(y_c - y_o)}{x_c} + x_c - x_o \\ \frac{z_c(z_o - z_c)}{x_c} + x_o - x_c & \frac{z_c(y_c - y_o)}{x_c} \end{bmatrix} \tag{11}$$

where the rotation center of the object is expressed as ${}^{C}O_B = [x_o, y_o, z_o, 1]^T$, and the coordinates of a point ${}^{C}P = [x_c, y_c, z_c, 1]^T$ are calculated as a function of the coordinates (u and v) of its projection on the image plane (among other known variables) as shown in AppendixA.1. Collecting—in a column-wise fashion—the OF vectors $[\dot{u}_i \;\; \dot{v}_i]^T$ for each point (1...N) in the image plane for which the OF is available, along with their corresponding matrices $M(f, {}^{C}P, {}^{C}O_B)$, yields

$$\begin{bmatrix} \dot{u}_1 \\ \dot{v}_1 \\ \vdots \\ \dot{u}_N \\ \dot{u}_N \end{bmatrix} = \begin{bmatrix} M(f, {}^{C}P_1, {}^{C}O_B) \\ \vdots \\ M(f, {}^{C}P_N, {}^{C}O_B) \end{bmatrix} \begin{bmatrix} {}^{C}V_{B/C} \\ {}^{C}\omega_{B/C} \end{bmatrix}. \tag{12}$$

The pseudoinversion of (12) leads to

$$\begin{bmatrix} {}^{C}V_{B/C} \\ {}^{C}\omega_{B/C} \end{bmatrix} = \begin{bmatrix} M(f, {}^{C}P_1, {}^{C}O_B) \\ \vdots \\ M(f, {}^{C}P_N, {}^{C}O_B) \end{bmatrix}^{\dagger} \begin{bmatrix} \dot{u}_1 \\ \dot{v}_1 \\ \vdots \\ \dot{u}_N \\ \dot{v}_N \end{bmatrix} \tag{13}$$

which, assuming $N \geq 3$, yields an estimate of the translational and rotational velocities of the object at a certain time instant. For each OF image information, the velocities can then be computed according to (13).

V. Performance Metrics

Various methods to compare OF algorithms have been proposed [5], [8]. In this study, a novel comparison method is introduced that relies on both the possibility to calculate the ideal OF, given an object rotational and translational velocity (which is essentially done by applying (3)) and the capability to calculate an object's translational and rotational velocity from the OF vectors [which is done by applying (13), that is in a sense the inverse of (3)].

Knowing the camera focal length and the altitude above the ground, the coordinates of the point ${}^{C}P$ related to each given pixel $[u, v]$ can be calculated as shown in ppendix A [specifically using (16)], and successively used within (3)—along with the known translational and rotational velocity of the camera with respect to the terrain—to calculate the "ideal" OF at each point $[u, v]$. Note that, as described in Appendix A, formula (16) also provides a way to detect (and discard) any point on the image that does not correspond to a physical point that belongs to the terrain (e.g., a point in the sky).

Expressing in polar coordinates both the ideal and the detected OF vectors allows for the calculation of the errors in magnitude and angle for each detected OF vector.

Specifically, the mean (over the whole duration of the flight) of the absolute values of the magnitude and angular errors, expressed in pixels and radians, will be indicated, respectively, with the symbols μ_m and μ_a.

A second set of average errors can be calculated from (13). Specifically, selecting the OF components for all image points corresponding to terrain points and using (13) it is possible to estimate the translational and rotational velocity vectors of the terrain with respect to the camera for the particular time instant at which the image is taken.

These estimated velocity vectors can be compared with the known translational and rotational velocities that are available from the simulation and/or IMU/GPS data, therefore yielding two error vectors for each acquired image. Averaging the norms of these two error vectors over all images acquired during the simulation (or during the flight, for the second experiment) yields the velocity errors μ_v and μ_ω.

The final performance criterion uses the four previously described parameters (each divided by the range of the quantity it represents) to express the overall error of the OF calculated by a given algorithm.

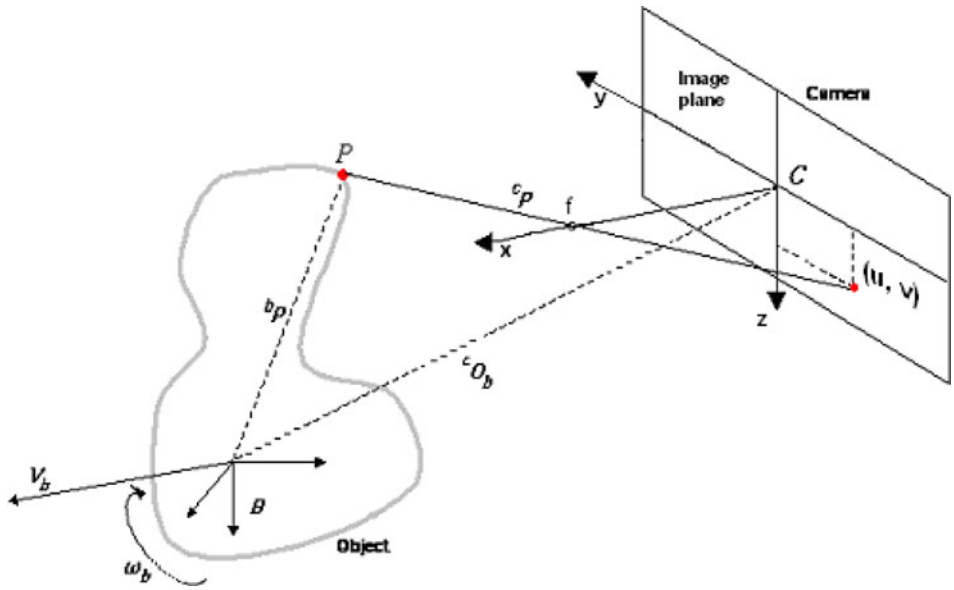

Fig. 1. Pinhole model.

The first two components of J rely on direct measurements of the OF components; therefore, the quality of the produced OF is evaluated by direct comparison with the "true" OF. The second two components rely on inverse measurements of the OF; therefore, the quality of the produced OF is evaluated according to the accuracy of the corresponding linear and angular velocities, with respect to the "true" (or measured) linear and angular velocities. It is important to notice that the second part of the formula also implicitly evaluates the performance of an algorithm in terms of the density of the produced OF field, which is very important for some applications

$$J = \frac{\mu_a}{\pi} + \frac{\mu_m}{\max\left(m_{id}\right)} + \frac{\mu_v}{\max\|V\|} + \frac{\mu_\omega}{\max\|\omega\|} \tag{14}$$

where $\max(m_{id})$ is the maximum magnitude of the ideal OF generated during the flight, and $\max\|V\|$ and $\max\|\omega\|$ represent, respectively, the maximum norms of the translational and rotational velocities reached during the flight.

The final performance metric to judge the quality of the OF produced by an algorithm is, therefore, J where low values of this metric correspond to good performance. Note that, for OF applications that are not related to reconstructing object motion, the proposed performance metrics could be easily modified so that its last two terms are discarded; conversely, for applications in which motion reconstruction is critical, the first two terms of the performance metric could be discarded.

VI. Flight Experiment and Results

In this experiment, the algorithms were applied on a sequence of images from a video recorded during one of the flight tests performed for the WVU YF-22 Autonomous Formation Flight Program [32]. Fig. 2 shows a typical image from this video. The reader is referred to [28] for a preliminary comparison using only-translational or only-rotational motion. The parameter settings for the various algorithms are described in Appendix A.2.

Fig. 2. Image from the flight video.

The aircraft was equipped with an Inertial Measurement Unit (Crossbow IMU VG400), which allowed the acquisition of the accelerations along the x-, y-, and z-directions and the angular rates p, q, and r. The vertical gyro (Goodrich-VG34) provided measurements for the aircraft Euler's angles, and the GPS (Novatel-OEM4) provided the translational position and velocity measurements x, y, z, V_x, V_y, V_z with respect to the earth reference frame. Since the data from the IMU and the GPS were used as ground truth, the accuracy and the performance of these sensors are important. The IMU Crossbow VG400 operates at 50 Hz and has a range of ± 200 °/s with 0.05 °/s resolution for the rate sensors and ± 10 g with 1.25 mg resolution for the accelerometers. The accuracy—gauged in terms of random walk (RW)—is $1.7\,^\circ/\sqrt{\mathrm{h}}$ and $0.5\,(\mathrm{m/s})/\sqrt{\mathrm{h}}$ for the angular rates and the accelerations, respectively. The Novatel-OEM4 GPS unit operates at 20 Hz with accuracy in terms of circular error probability (CEP) of 1.8 m. Finally, the sensors in the nose probe provided measurements for the α, β angles, while absolute and differential pressure sensors were used to measure H and V [32].

TABLE I
STATISTICS FOR THE FLIGHT EXPERIMENT

	μ_a (deg)	σ_a (deg)	μ_m (pixel/ frame)	σ_m (pixel /frame)	μ_v (m/s)	σ_v (m/s)	μ_ω (deg/s)	σ_ω (deg/s)	J
SIFT	**32.6**	**31.7**	**6.09**	**4.72**	**15.40**	**14.42**	**7.63**	**6.31**	**0.899**
Correlation	**35.7**	**32.7**	**6.42**	**4.97**	**25.31**	**57.20**	**8.45**	**8.03**	**1.156**
Difference	**47.4**	**25.9**	**6.40**	**4.58**	**39.96**	**49.23**	**9.26**	**7.95**	**1.550**
Harris	55. 5	22.6	6.93	5.10	38.01	33.25	10.16	7.66	1.596
Proesmans	40.1	31.5	9.12	6.30	39.39	37.14	10.22	8.03	1.597
Gradient	61.2	20.3	11.86	6.98	31.47	14.69	12.20	7.76	1.684
Zero	**64.8**	**25.7**	**12.31**	**7.00**	**31.81**	**64.61**	**12.38**	**25.74**	**1.728**
Phase	78.4	26.8	10.48	7.05	33.14	19.87	12.29	8.63	1.781
L -K	82.2	7.37	10.86	6.68	32.44	15.22	12.30	7.72	1.798
H - S	82.6	8.0	10.56	6.57	32.80	15.44	12.31	7.70	1.800

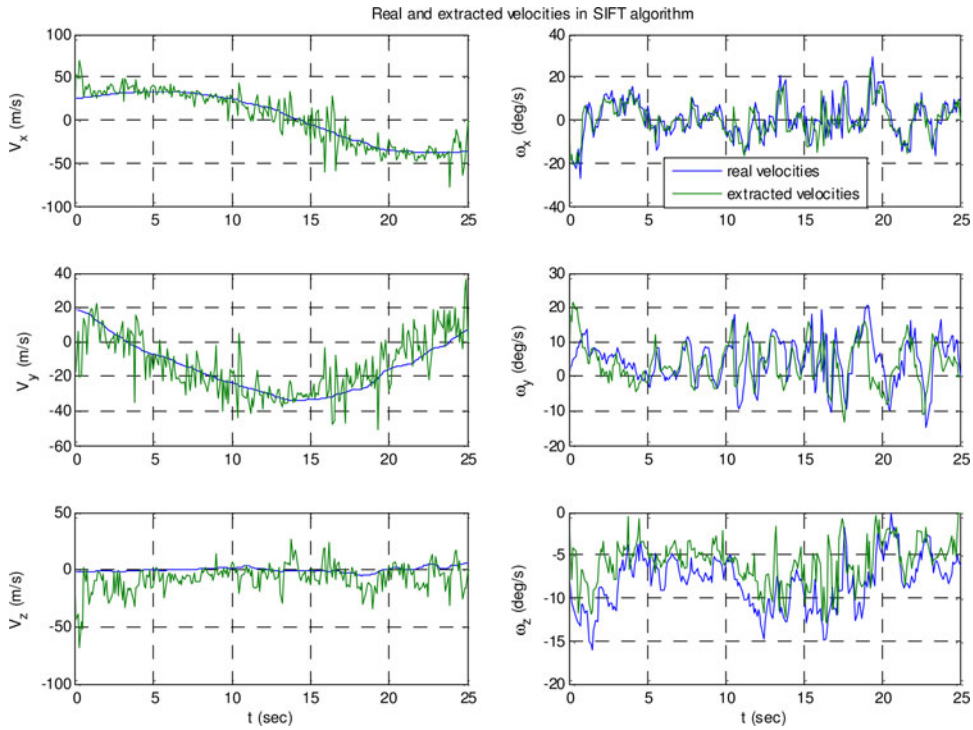

Fig. 3. SIFT—estimated translational and rotational velocities.

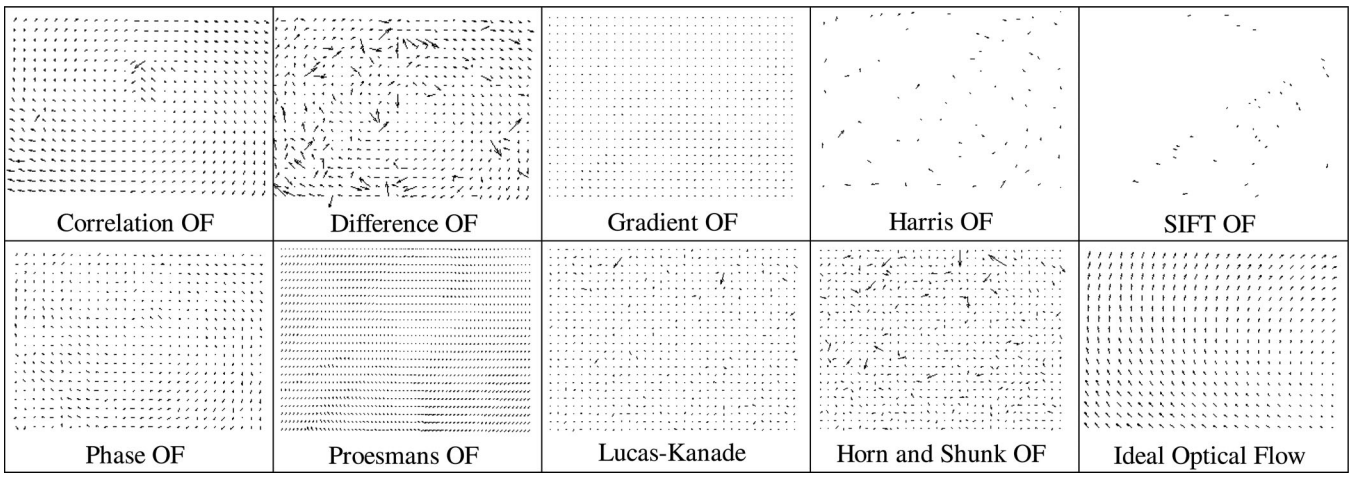

Fig. 4. Behavior of the different OF algorithms.

The camera was placed at exactly 1 m in front of the aircraft center of gravity, with orientation with respect to the aircraft body frame that consists of yaw and pitch angles of –45° and –14.5°, respectively.

To perform the OF experiment, a 25-s 320 × 240 video—acquired at the rate of 10 frames/s—was extracted from an original 607-s video.

Applying basic trigonometry to the pinhole model leads to

$$\frac{S_v}{2} = f \tan\left(\frac{\Phi_v}{2}\right) \tag{15}$$

where Φ_v is the vertical field of view, and S_v is the vertical size (length) of the image sensor (e.g., the charge-coupled device). Using the vertical length of a pixel on the image sensor as an (arbitrary) unit of length, it can be stated that $S_v = 240$ pixels. A focal length value of $f = 847.5$ pixels can then be calculated from (15). As can be seen for example from (3), this implies that the OF is measured in units of pixel/sec. A multiplication by $T = 0.1$ s can then be used to express the OF in units of pixel/frame.

The nine OF algorithms were continuously executed for each couple of consecutive images. The ideal OF was computed using the method described in previous sections, relying on the (translational and rotational) position and velocity data acquired during the flight session. The maximum translational velocity reached in the used flight section was 39.04 m/s, and the maximum angular velocity was 0.549 rad/s. The ideal OF vectors had a mean value of 12.32 pixel/frame, while the largest OF vector had a magnitude of 48.25 pixel/frame.

Table I shows the results, in terms of the previously defined performance metric, for this experiment. Furthermore, the performance of a fictitious "ZERO" algorithm, which always produces an OF having both angle and magnitude equal to zero, is also reported purely as a benchmark.

Fig. 3 show the comparison between the real (as measured by the sensors) translational and rotational velocities, and the translational and rotational velocities estimated using the OF field provided by the SIFT algorithm. The estimated velocities—and especially the angular velocities—exhibit a desirable correlation with the velocities measured by the GPS.

This fact suggests the possibility to use the proposed technique to aid velocity estimation in applications, where a camera and some processing power are available.

Finally, Fig. 4 shows the OF fields resulting from each algorithm corresponding to the image in Fig. 2. The last image of Fig. 4 shows the ideal OF computed using (3). Note that the assumption that all the features correspond to points that have the same altitude has been used in calculating the ideal OF.

The algorithms that provide the best performance are SIFT, correlation, and difference. The particularly bad performance of the gradient, phase, Lucas–Kanade, and Horn and Shunck methods originates from the high relative velocity between camera and terrain, especially during aircraft turns. On the other hand, patch-based methods and feature-based methods seem more flexible in detecting large intraframe displacements and for this reason tended to exhibit better performance.

VII. Conclusion

This paper has described the results of a comparative analysis, performed using a novel performance metric, of nine OF algorithms that belong to different classes. A novel method to compute the ground-truth flow and a method to extract the linear and angular velocities from the OF were presented. The algorithms were implemented and tested using real images that represent a complex rigid 6-DOF motion in 3-D. The analysis indicates that the correlation and SIFT feature-detection algorithms provide overall the best performance, using the performance metric based on the presented formulas. This is in part due to the fact that these methods are better equipped to handle both the large and small displacements that may be present in complex 3-D motion. In addition to defining performance metrics for OF algorithms, the presented formulas could also find use for navigation purposes, since results indicate that an acceptable estimation of the angular velocity was produced from the OF field alone.

Appendix

The first section of this appendix explains the technique used to calculate the coordinates, in camera frame, of any given terrain point, while the second section gives a brief overview of the different parameters that need to be set, and their chosen values, for the nine OF algorithms.

A. Calculating the Coordinates of a Terrain Point in Camera Frame

Using (3) to relate the relative terrain motion to its corresponding OF requires the calculation of the coordinates of each terrain point P in the camera reference frame, that is CP. Assuming that the terrain is flat and has a known constant altitude z_e^* (equal to 100 m in the examples), the homogeneous coordinates [27] of the point P with respect to the earth reference frame are ${}^E P = [x_e, y_e, z_e^*, 1]^T$. The homogeneous coordinates of P in camera reference frame are given by ${}^C P = {}^C\mathrm{T}_E(\psi, \theta, \varphi, {}^E O_C){}^E P$, where ${}^C T_E(\psi, \theta, \varphi, {}^E O_C) = {}^E T_C(\psi, \theta, \varphi, {}^E O_C)^{-1}$ is the 4 × 4 matrix that transforms earth-frame coordinates into camera-frame coordinates. This transformation matrix is a function of the Euler angles ψ, θ, and φ (expressing the orientation of the camera reference frame with respect to the earth reference frame), and of the vector ${}^E O_C = [{}^c x, {}^c y, {}^c z, 1]^T$ (expressing the position of the origin of the camera reference frame with respect to the earth reference frame).

The coordinates x_e and y_e can then be determined by setting the projection on the image plane of the point ${}^C P = {}^C T_E(\psi, \theta, \varphi, {}^E O_C){}^E P$, equal to the image point coordinates $[u, v]$, from which the considered OF vector originates. The MATLAB Symbolic Math Toolbox [33] was used to obtain a formula yielding the earth frame coordinates x_e and y_e of a generic terrain point ${}^E P$, as a function of the image plane coordinates u, v, as well as ψ, θ, and φ, and ${}^E O_C$. Once the coordinates x_e and y_e were found as functions of the variables *f, u*, v, ψ, θ, φ, ${}^E O_C$, the point ${}^C P$ was calculated using the formula

${}^{C}P = {}^{C}T_{E}(\psi, \theta, \varphi, {}^{E}O_{C}) {}^{E}P(f,u,v, \psi, \theta, \varphi, {}^{E}O_{C})$, resulting in

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \frac{n_p}{\cos(\theta)^2 d_p} \begin{bmatrix} f \\ u \\ v \end{bmatrix} \tag{16}$$

with

$$n_p = u(c_z - z_e^*)\cos(\theta)\sin(\varphi) + v(c_z - z_e^*)\cos(\theta)\cos(\varphi) + f(c_z - z_e^*)\sin(\theta) \tag{17}$$

and

$$d_p = u^2(\cos(\varphi)^2 - 1) - 2uv\sin(\varphi)\cos(\varphi) - v^2\cos(\varphi)^2 + f^2\tan(\theta)^2. \tag{18}$$

Note that points of the image that do not correspond to points on the terrain (for example points representing the sky, above the horizon line) provide a negative x coordinate in the camera frame, and therefore they can be automatically discarded when calculating the ideal OF. As a consequence, this also allows discarding the OF generated by any algorithm for points that do not belong to the terrain.

In general, (16) can also be valuable in determining the position of an interesting object on the ground from the image captured by the on-board camera.

B. Parameter Settings for the OF Algorithms

This section will briefly describe the various parameters used by the different algorithms, along with their selected values. The reader is referred to the previously mentioned references for a more detailed explanation of the inner workings of such algorithms.

1) *Gradient:* In this algorithm, three parameters need to be selected: "step," "neighborhood," and "threshold." The "step" parameter sets the distance in pixels between each OF vector in the output OF matrix; this parameter was set to 5 pixels. The "neighborhood" parameter sets the half-size of the window used for solving the aperture problem; this parameter was set to 8 pixels. The "threshold" parameter sets the minimum eigenvalue acceptable to compute the OF, as explained in Section II-A; this parameter was set to 5000.
2) *Lucas–Kanade:* In this algorithm, two parameters need to be set: "temporal gradient filter" and "threshold." The "temporal gradient filter" characterizes how the temporal gradient is computed; this parameter was set for a "difference filter", that is, [−1 1]. The "threshold" represents the minimum eigenvalue acceptable to compute the OF; this parameter was set to 0.01 in the Simulink block.
3) *Horn and Schunck:* This algorithm requires setting two parameters: "smoothness factor" and "stopping criteria." The "smoothness factor" is a positive constant that has to be large if the motion between two consecutive frames is large; this parameter was set to 10. The "stopping criteria" is the maximum number of iterations for the method, and it was set to 10.
4) *Phase:* This algorithm requires three parameters to be set: the "number of OF vectors in the u (horizontal) direction," the "linearity threshold," and the "minimum value of valid component velocities." The "number of OF vectors in the horizontal direction" was set to 25. The "linearity threshold" is a limit on the error between the phase estimated by using the filter and the actual phase of the OF; this parameter was set to 0.01 radians. The "minimum value of valid component velocities" allows discarding points in which the filter output is located in a phase singularity neighborhood; this parameter was set to 7.
5) *Correlation:* In this algorithm, three parameters need to be set: "step," "window," and "template." The "step" parameter sets the distance in pixel between each OF vector in the output OF matrix, and it was set to 5 pixels. The "window" parameter sets the size of the (square) window used to search for the closest correlation value, and it was set to 30 pixels. The "template" parameter sets the width of the image portion on which the correlation has to be computed; this parameter was set to 20 pixels.
6) *Difference:* This algorithm also has three parameters that need to be set, which are very similar to the ones used for the correlation algorithm: "step," "window," and "template." The "step" parameter sets the distance in pixel between each OF vector in the output OF matrix, and it was set to 5 pixels. The "window" parameter sets the half-size of the (square) window used to search for the closest SAD value; this parameter was set to 15 pixels. The "template" parameter sets the width of the image portion on which the SAD has to be computed; and it was set to 5 pixels.
7) *Harris:* This algorithm requires five parameters: "sigma," "threshold," "size," "maximum number of OF vectors," and "window." The "sigma" parameter is the standard deviation of the Gaussian filter applied to the image before the corner detection; this parameter was set to 5. The "threshold" is the minimum considered value for the maxima of the Harris coefficient; this parameter was set to 50. The "size" is the dimension of the gray-scale dilation; this parameter was set to 1. The "maximum number of OF vectors" limits the number of vectors of the resulting OF matrix; this parameter was set to 300. Finally, "window" sets the maximum distance in pixels that can be used by the point-matching algorithm to search for the position of the matching corner in the previous image; this parameter was set to 100 pixels.
8) *SIFT:* In this algorithm, three parameters need to be set: the "descriptor distance," the "maximum number of OF vectors," and the "window." The first parameter sets the maximum distance ratio between two SIFT descriptors for being considered the same feature; and it was set to 0.4. The second parameter limits the number of vectors of the OF; and it was set to 300. The "window" parameter is completely equivalent to the one for the Harris algorithm, and it was set to 100 pixels.

It is important to highlight that in the selection of the parameters for the aforementioned algorithms, the designer has to face fundamental tradeoffs between camera frame rate, allowed

velocity of the objects in the scene, and computational requirements. The selection of the parameters for region-matching techniques such as difference and correlation is where these tradeoffs become clearly evident.

In our study, a preliminary analysis of the two flights was made, and the mean and standard deviation values of the ideal OF vectors (over the whole flight) were established. These values were then used to select an appropriate value for the "window" parameter for the correlation and difference algorithms. A smaller value of this parameter would have resulted in an unacceptable loss in accuracy, while a larger value would have resulted in unnecessarily longer computation times.

For both feature-matching methods, the maximum allowed feature distance was set to be 100 pixels (this value was essentially chosen as a protective measure against mismatching). Finally, while a detailed parameter tuning of the gradient and phase methods was attempted, their performance in estimating larger OF vectors remained somewhat poorer than the performance of the other algorithms. As previously pointed out, this was not surprising since these methods are essentially local in nature.

REFERENCES

[1] G. Barrows and C. Neely, "Mixed-mode VLSI optic flow sensors for in-flight control of a micro air vehicle," *Proc. SPIE*, vol. 4109, pp. 52–63, 2000.

[2] K. Souhila and A. Karim, "Optical Flow based robot obstacle avoidance," *Int. J. Adv. Robot. Syst.*, vol. 4, no. 1, pp. 13–16, 2007.

[3] E. C. Cho, G. Seetharaman, R. J. Holyer, and M. Lybanon, "Velocity vectors for features of sequential oceanographic images," *IEEE Trans. Geosci. Remote Sens. E*, vol. 36, no. 3, pp. 985–998, 1998.

[4] K. Kanatani and K. Watanabe, "Reconstruction of 3-D road geometry from images for autonomous land vehicles," *IEEE Trans. Robot. Autom.*, vol. 6, no. 1, pp. 127–132, 1990.

[5] F. F. Khalil and P. Payeur, "Optical flow techniques in biomimetic UAV Vision," in *Proc. IEEE Int. Workshop Robot. Sensors Environ.*, 2005, pp. 14–19.

[6] G. Zen and E. Ricci, "Earth mover's prototypes: A convex learning approach for discovering activity patterns in dynamic scenes," in *Proc. Computer Vis. Pattern Recognit.*, Colorado Springs, CO, Jun. 21–23, 2011.

[7] B. Galvin, B. McCane, K. Novins, D. Mason, and S. Mills, "Recovering motion fields: An evaluation of eight optical flow algorithms," in *Proc. 9th Brit. Mach. Vis. Conf.*, 1998, pp. 195–204.

[8] J. L. Barron, D. J. Fleet, and S. S. Beauchemin, "Performance of optical flow techniques," *Int. J. Comput. Vis.*, vol. 12, pp. 43–77, 1994.

[9] M. Otte and H. H. Nagel, "Estimation of optical flow based on higher-order spatiotemporal derivatives in interlaced and non-interlaced image sequences," *Artif. Intell.*, vol. 78, pp. 5–43, 1995.

[10] G. Adiv, "Inherent ambiguities in recovering 3-D motion and structure from a noisy flow field," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, no. 5, pp. 477–489, 1989.

[11] B. McCane, K. Novins, D. Crannitch, and B. Galvin, "On benchmarking optical flow," *Comput. Vis. Image Underst.*, vol. 84, no. 1, pp. 126–143, 2001.

[12] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski, "A database and evaluation methodology for optical flow," in *Proc. 11th Int. Conf. Comput. Vis.*, Oct. 14–21, 2007, pp. 1–8.

[13] B. K. P. Horn and B. G. Schunck, "Determining optical flow," *Artif. Intell.*, vol. 17, pp. 185–204, 1981.

[14] B. Lucas and T. Kanade, "An iterative image restoration technique with an application to stereo vision," in *Proc. DARPA Image Underst. Workshop*, 1981, pp. 121–130.

[15] M. Proesman, L. Van Gool, E. Pauwels, and A. Oosterlinck, "Determination of optical flow and its discontinuities using non-linear diffusion," in *Proc. 3rd Eur. Conf. Computer Vis.*, 1994, vol. 2, pp. 295–304.

[16] J. Díaz, E. Ros, R. Agís, and J. L. Bernier, "Superpipelined high-performance optical-flow computation architecture," *Comput. Vis. Image Underst.*, vol. 112, no. 3, pp. 262–273, Dec. 2008.

[17] M. Durkovic, M. Zwick, F. Obermeier, and K. Diepold, "Performance of optical flow techniques on graphics hardware," in *Proc. IEEE Int. Conf. Multimedia Expo*, 2006, pp. 241–244.

[18] S. Lim, J. G. Apostolopoulos, and A. E. Gamal, "Optical flow estimation using temporally over-sampled video," *IEEE Trans. Image Process.*, vol. 14, no. 8, pp. 1074–1087, Aug. 2005.

[19] D. J. Fleet and A. D. Jepson, "Computation of component image velocity from local phase information," *Int. J. Comput. Vis*, vol. 5, pp. 77–104, 1990.

[20] T. Gautama (2002). "Phase-based optical flow," MATLAB Central. [Online]. Available: http://www.mathworks.com/matlabcentral/fileexchange/2422

[21] T. Gautama and M. M. Van Hulle, "A phase-based approach to the estimation of the optical," *IEEE Trans. Neural Netw.*, vol. 13, no. 5, pp. 1127–1136, 2002.

[22] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proc. 4th Alvey Vis. Conf.*, 1988, pp. 147–151.

[23] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proc. Int. Conf. Computer Vision*, 1999, pp. 1150–1157.

[24] M. Mammarella, G. Campa, M. R. Napolitano, and M. L. Fravolini, "Comparison of point matching algorithms for the UAV aerial refueling problem," *Mach. Vis. Appl.*, vol. 21, no. 3, pp. 241–251, Apr. 2010.

[25] H. Feng, E. Li, Y. Chen, and Y. Zhang, "Parallelization and characterization of SIFT on multi-core systems," in *Proc. IEEE Int. Symp. Workload Charact.*, Seattle, WA, 2008, pp. 14–23.

[26] S. Hutchinson, G. Hager, and P. Corke, "A tutorial on visual servo control," *IEEE Trans. Robot. Autom.*, vol. 12, no. 5, pp. 651–670, Oct. 1996.

[27] L. Sciavicco and B. Siciliano, *Modeling and Control of Robot Manipulators*. New York, NY: McGraw-Hill, 1996.

[28] M. Mammarella, G. Campa, M. L. Fravolini, Y. Gu, B. Seanor, and M. R. Napolitano, "A comparison of optical flow algorithms for real time aircraft guidance and navigation," in *Proc. AIAA Guid., Navig. Control Conf. Exhibit*, Honolulu, HI, Aug. 2008, pp. 18–21.

[29] A. Chiuso, P. Favaro, H. Jin, and S. Soatto, "Structure from motion causally integrated over time," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 4, pp. 523–535, Apr. 2002.

[30] G. Seetharaman, "Estimation of 3D motion and orientation of rigid objects from an image sequence: A region correspondence approach," Ph.D. dissertation, Univ. Miami, Coral Gables, Miami, FL, Aug. 1988.

[31] K. Kanatani, "Computational projective geometry," *CVGIP: Image Underst.*, vol. 54, no. 3, pp. 333–348, Nov. 1991.

[32] G. Campa, Y. Gu, B. Seanor, M. R. Napolitano, L. Pollini, and M. L. Fravolini, "Design and flight testing of nonlinear formation control laws," *Control Eng. Pract.*, vol. 15, no. 9, pp. 1077–1092, 2007.

[33] *Symbolic Math Toolbox*™, *User's Guide, 1993–2010,* MathWorks Inc., Natick, MA.

Marco Mammarella was born in Milan, Italy. He received the M.S. degree in automation and robotic engineering in 2005 from the University of Pisa, Italy, and the Ph.D. degree in aerospace engineering from West Virginia University, Morgantown, in 2008.

He is currently Project Manager at GMV Aerospace and Defense in the Space System Business Unit and specifically in the GNC Division. His research interests include machine vision navigation system for UAVs and space missions, particularly, in entry descent and landing scenarios. Furthermore, Marco research interests are related with trajectory design and spacecraft control, sensors fusion, nonlinear and hybrid control systems, neural networks, and real-time embedded computing.

Giampiero Campa (M'99) was born in Taranto, Italy. He received both the "Laurea" degree in electrical engineering and the Ph.D. degree in robotics and automation from the University of Pisa, Pisa, Italy, in 1996 and 2000, respectively.

In 1995, he was with the Industrial Control Centre, Strathclyde University, U.K., and in 1999, he was with the Department of Aerospace Engineering, Georgia Institute of Technology, Atlanta. From 2000 to 2008, he has served as a faculty with the Flight Control Group, Department of Aerospace Engineering, West Virginia University (WVU), Morgantown. His research interests at WVU include adaptive and nonlinear control, system identification, fault tolerant systems, sensor fusion, and machine vision, with UAVs being the typical application. Since January 2009, he was with MathWorks, Torrance, CA, as the Technical Evangelist for southern CA.

Mario L. Fravolini was born in Perugia, Italy. He received the Ph.D. degree in electronic engineering from the University of Perugia, Perugia, in 2000.

He is currently a Research Assistant with the Department of Electronics and Information Engineering, University of Perugia. In 1999, he was with the Control Group, School of Aerospace Engineering, Georgia Institute of Technology, Atlanta. He has been a Visiting Research Assistant Professor with the Department of Mechanical and Aerospace Engineering, West Virginia University, Morgantown, for several years. He teaches courses in the area of feedback control systems at Perugia and Terni University. His research interests include fault diagnosis, intelligent and adaptive control, predictive control, optical feedback, active control of structures, and feedback control of type-1 diabetes.

Marcello R. Napolitano was born in Pomigliano d'Arco, Italy. He received the M.S. degree from the University of Naples, Naples, Italy, and the Ph.D. degree from Oklahoma University, Norman, in 1985 and 1989, respectively, both in aeronautical engineering.

In 1990, he joined the Department of Mechanical and Aerospace Engineering, West Virginia University, Morgantown, where he is currently a Full Professor and Director of the Flight Control Systems Laboratory. He is the author of the textbook titled *Aircraft Dynamics: From Modeling to Simulation* (Wiley). His current research interests include flight control systems, unmanned aerial vehicles, and fault tolerant systems.