JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 33, No. 6, November–December 2010

Navigation Aiding Based on Coupled Online Mosaicking and Camera Scanning

Vadim Indelman,* Pini Gurfil,† and Ehud Rivlin‡
Technion–Israel Institute of Technology, 32000 Haifa, Israel
and
Hector Rotstein§
Rafael, 31021 Haifa, Israel

DOI: 10.2514/1.48134

This paper presents a new method for vision-aided navigation of airborne platforms. The method is based on online mosaicking using images acquired by an onboard gimballed camera, which scans ground regions in the vicinity of the flight trajectory. The coupling of the scanning and mosaicking processes improves image-based motion estimation when operating in challenging scenarios such as narrow field-of-view cameras observing low-texture scenes. These improved motion estimations are fused with an inertial navigation system. The mosaic used for navigation is constructed in two levels. A small mosaic based on recently captured images is computed in real-time, and a larger mosaic including all the images is computed in a background process. The low-level mosaic is used for immediate motion estimation, while the higher-level mosaic is used for global navigation. The correlation terms between the navigation system and the mosaic construction process are not maintained in the proposed approach. The advantage of this architecture is the low computational load required for navigation aiding. However, the accuracy of the proposed method could be compromised compared with bearing-only simultaneous localization and mapping. The new method was examined using statistical simulation runs and experiments based on Google Earth imagery, showing its superior performance compared with traditional methods for two-view navigation aiding.

I. Introduction

DURING the last few decades, considerable research efforts have been directed toward developing methods for navigation aiding of airborne platforms. Navigation aiding deals with improving the performance of some basic inertial navigation system by fusing measurements from auxiliary sensors or additional, possibly exogenous, sources of information. In recent years, substantial attention has been devoted to using vision sensors such as onboard cameras for navigation aiding. Indeed, vision-aided navigation (VAN) has been extensively studied. A typical VAN algorithm uses the information extracted from an image registration process, along with the information available from other sensors, for estimating the platform's states and possibly additional navigation parameters. For example, [1] proposed integrating the velocity-to-height vision-based estimation with additional onboard sensors; [2] applied the subspace-constraint approach [3] to partially estimate an airborne platform's states based on measurements from an image registration process injected into an implicit extended Kalman filter; and [4] used epipolar constraints formulated for each pair of matching features to aid the inertial navigation of a ground vehicle. All the preceding methods rely only on information from inertial navigation sensors and an onboard camera, without assuming any a priori available information or additional external sensors. This is also the approach adopted in the current work.

Various methods for vision-aided navigation have been proposed assuming some additional external sensors and a priori information. Roumeliotis et al. [5] assumed that altimeter measurements are used for scaling the imaging sensors to improve state estimation during the landing phase of a space probe. References [6,7] showed that absolute pose and motion estimation is possible when assuming that a digital terrain model is available. Another fusion approach is map-based navigation, which assumes that a map of the operational area is given and that the vehicle navigates by fusing inertial measurements, images of the environment and a map [8–10]. For example, [9,10] proposed vision-based navigation for an unmanned underwater vehicle relying on a previously constructed mosaic image of the ocean floor.

This paper describes a method for vision-aided navigation of an airborne platform using the information contained in the online construction of a mosaic from images that are captured by an onboard camera. It is assumed that the airborne platform is equipped with a standard, possibly low-quality, inertial navigation system and a camera mounted on gimbals. No additional external sensors are assumed to exist, and no a priori additional information is necessarily available. Consequently, the mosaic image that represents the map of the overflown region has to be constructed during the flight. If an additional source of information, e.g., Global Positioning System, is partially available, then the information of this sensor can be used to provide an absolute geo-reference for the mosaic.

This approach is related to simultaneous localization and mapping (SLAM) [11–18], in which the construction of the observed environment representation and the estimation of the platform's parameters are performed simultaneously. The general approach for solving the SLAM problem is to augment the platform's state vector with parameters describing the observed environment (e.g., feature locations). When processing a measurement, the augmented state vector yields an update both in the platform's states (e.g., position, velocity) and in the environment model. Consequently, correlation between the platform's states and the environment parameters is consistently maintained. However, the computational requirements are constantly increasing as the state vector grows in size. Several works have proposed to overcome this computational bottleneck by neglecting

Presented as Paper 2009-6219 at the AIAA Guidance, Navigation, and Control Conference, Chicago, IL, 10–13 August 2009; received 17 November 2009; revision received 15 July 2010; accepted for publication 19 July 2010.

*Doctoral Student, Faculty of Aerospace Engineering; ivadim@tx.technion.ac.il.

†Associate Professor, Faculty of Aerospace Engineering; pgurfil@technion.ac.il. Associate Fellow AIAA.

‡Professor, Department of Computer Science; ehudr@cs.technion.ac.il.

§Chief Control Systems Engineer; hector@rafael.co.il.

low-correlation bonds in the augmented state vector [17] or maintaining only currently visible features in the state vector [18].

This work suggests a variation of the SLAM framework for coping with the aforementioned challenge: Separating the environment recovery, represented in the current work by a mosaic, from motion estimation. To do this, two types of mosaics are constructed: a *temporary* or *local* mosaic and a *main* or *global* mosaic. The parameter estimation is based on the first type of mosaic image, which is a small temporary mosaic constructed based on recent incoming camera-captured images. Once the temporary mosaic image reaches a certain size, its contents are removed and used for updating the main mosaic image. The update of the main mosaic image is performed in a background process, using various algorithms [19–26] depending on the available computational resources.

The correlation terms between the navigation system and the mosaic construction process are not maintained in the proposed approach. The advantage of this architecture is the low computational load required for parameter estimation, because it involves processing a constant-size state vector and only a portion of the temporary mosaic image. However, the accuracy of the proposed method could be inferior to the accuracy of bearing-only SLAM methods.

Image registration and image-based motion estimation are important constituents in all VAN methods. The existence of overlapping regions between processed images is the common assumption to all vision-based motion estimation techniques. A large overlapping region between two images should allow a more accurate motion estimation relying on two-view geometry methods. If a mutual overlapping region for more than two images can be found, the performance may be further enhanced by applying multiview geometry-based methods.

The two-view geometry-based methods include, for example, [27], in which the relative motion between two given views is extracted from an estimated essential matrix [26]. The motion parameters are then used for estimating the state vector, which is an augmented vector composed of the vehicle's current pose and past poses for each captured image. When the observed scene is planar, the motion parameters can be calculated by estimating the homography matrix [10,28–30]. Having in mind the requirements for real-time performance and a low computational load, in this work the estimated camera motion is related to a constant-size state vector composed of the vehicle's current parameters only (in contrast to [27]).

The multiview geometry-based methods, in contrast to two-view geometry, use connections among several images, assuming that a common overlapping region exists. This results in an increased observability and better estimation of the platform states. For example, in [31], the authors derive constraints relating features that are observed in several consecutive images, thereby claiming to achieve optimal exploitation of the available information in the observed scene. Shakernia et al. [32] proposed using multiple-view geometry for estimating the motion of an unmanned helicopter during its landing phase. However, assuming that an overlapping region among several images exists may be invalid in many airborne applications. Violating this assumption usually degenerates the multiview methods into the two-view methods discussed above.

Another important factor in the context of motion estimation is the camera field-of-view (FOV). Wide-FOV cameras encompass large ground regions; thus, assuming that features are distributed evenly over the whole FOV, gives rise to a benign geometry for the estimation problem. On the other hand, wide-FOVs also give rise to some hard problems that are difficult to compensate for, such as increased optical image distortion and reduced resolution. One is tempted to assume that reduced resolution is a nonissue for high-pixel sensors, but this is only partially true because FOV, resolution and computing power are parameters that must be traded-off when designing a practical system. It is worth mentioning that many current airborne platforms use narrow-FOV cameras of up to a few degrees to obtain large-zoom images with high resolution and feasible requirements on computational power. The constellation of narrow-FOV cameras and compromised geometry presents a challenge for estimating camera motion, and therefore for VAN; moreover, the typical trajectories and maneuvers of these airborne platforms render multiview geometry methods useless. The current work focuses on motion estimation and VAN with narrow-FOV cameras. To mitigate the difficulties mentioned before, a method for increasing the overlapping regions between images, based on a camera scanning pattern and an online mosaic construction, is developed.

Consequently, the main contribution of this paper is the development of a coupling mechanism between a camera scanning process and a temporary mosaic image construction, which provides motion estimation with improved accuracy for narrow-FOV cameras, while allowing mapping of extended regions. Moreover, the proposed architecture, decoupling the construction of the main mosaic image from the platform's parameter estimation process, requires less computational resources compared with the SLAM approach. We also show that the improvement in motion estimation precision is achieved without imposing any constraints on the platform's trajectory.

II. Method Overview

Figure 1 shows the main components of the architecture under consideration. The specific system assumed in this work is an airborne platform equipped with a gimballed camera and an inertial navigation system (INS). Throughout this work, a narrow-FOV camera is assumed, since it is more realistic than wide-FOV camera for many cases of practical interest. As mentioned in the Introduction, a narrow-field makes the VAN and the image-based motion estimation problems more challenging. However, the proposed method is not restricted to cameras with narrow FOV, and is valid for other cameras as well. In addition, it is assumed that the observed ground area is sufficiently close to planar, or alternatively, that the flight altitude above ground level is high relative to ground changes in elevation.¶

The INS consists of an inertial measurement unit (IMU), a strapdown algorithm and a navigation Kalman filter. The strapdown algorithm integrates the accelerations and angular rates (or rather, the velocity and angular increments) from the IMU to produce a navigation solution, which is composed of platform position, velocity and attitude. Because of the unavoidable errors of the IMU sensors, the computed navigation parameters develop errors which increase unboundedly over time. It is well-known that for relatively low-grade inertial sensors, errors grow proportionally to time cubed, and hence an uncompensated inertial solution becomes useless in a relatively short period of time.

During the flight, an onboard camera captures images of ground regions according to a scanning procedure. The acquired images are directed to the image-processing module that is accountable for mosaic image construction and for relative motion estimation. While all the images are used for updating the mosaic image, the motion estimation is performed at a lower frequency, using only some of the images.

The mosaic construction is coupled with the camera scanning procedure, and is processed in two phases:

1) The camera-captured images are used for constructing a small temporary mosaic image. This temporary mosaic image is used for motion estimation at appropriate time instances.

2) After each motion estimation event, the temporary mosaic image is emptied and initialized to the most recent camera-captured image, while the removed images from the temporary mosaic image are used to update the main mosaic image in a background process.

The image-based motion estimation is reformulated into measurements, which are then injected into the navigation Kalman filter to update the navigation system and thereby arrest the

¶This assumption is made due to the construction process of the mosaic image, which is based on the homography transformation. However, the proposed approach for fusing image-based motion estimations and navigation data may be also applied without constructing a mosaic image, in which case nonplanar scenes can be handled as well [33].

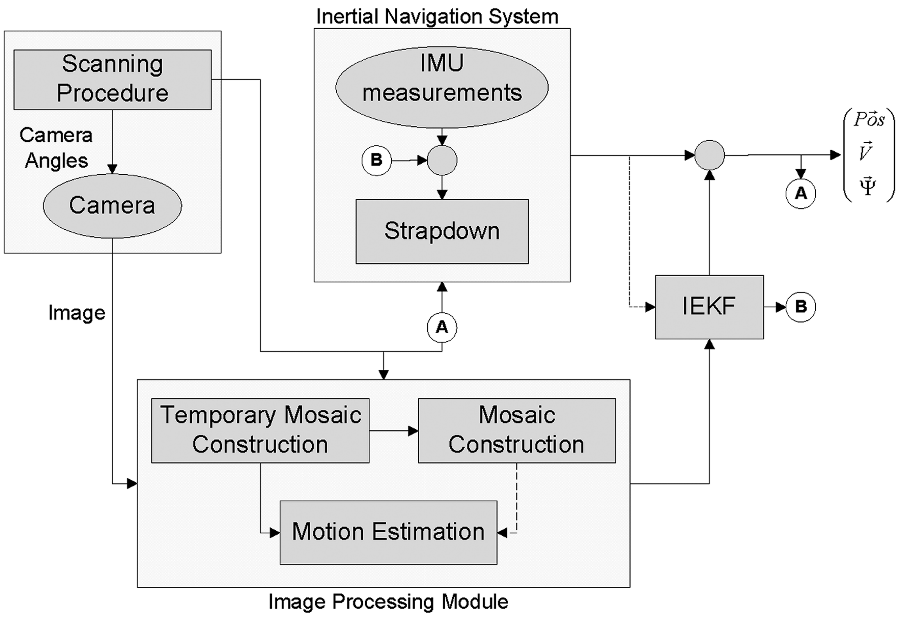


Fig. 1 Overview of the system concept.

development of inertial navigation errors. In this way, the platform can navigate for long periods of time even with low-grade sensors.

Throughout this paper, the following coordinate systems are used:

1) E: Earth-fixed reference frame, also known as an Earth-centered, Earth-fixed (ECEF) coordinate system. Its origin is set at the center of the Earth, the Z_E axis coincides with the axis of rotation, X_E goes through the point latitude 0°, longitude 0°, and Y_E completes a Cartesian right-hand system.

2) L: Local-level, local-north (LLLN) reference frame, also known as a north-east-down coordinate system. Its origin is set at the platform's center-of-mass. X_L points north, Y_L points east, and Z_L is aligned with the plumb-bob vertical to complete a Cartesian right-hand system.

3) B: Body-fixed reference frame. Its origin is set at the platform's center-of-mass. X_B points toward the nose tip, Y_B points toward the right wing and Z_B completes the setup to yield a Cartesian right-hand system.

4) C: Camera-fixed reference frame. Its origin is set at the camera center-of-projection. X_C points toward the FOV center, Y_C points toward the right half of the FOV and Z_C completes the setup to yield a Cartesian right-hand system, as shown in Fig. 2b.

Notice that the camera is mounted on gimbals and performs pan and tilt movements with respect to the platform; the yaw and pitch angles between B and C are denoted by ψ_C and θ_C, respectively.

III. Camera Scanning Procedure and Mosaic Construction Method

This section presents a detailed description of the camera scanning and mosaic construction procedures. Each procedure by itself is simple and several variations have appeared in the literature before. The section hence focuses on the coupling between scanning and mosaic image construction, in particular on the aspects that allow improving the accuracy of the motion estimation in challenging scenarios such as narrow-FOV cameras and low-texture scenes. In addition, it will be shown that relatively large ground areas may be represented in the mosaic image without restricting the trajectory of the platform. More sophisticated camera scanning methods may be considered, for instance those exploiting the coupling with the platform trajectory or maximizing the ground coverage, but are left for future studies.

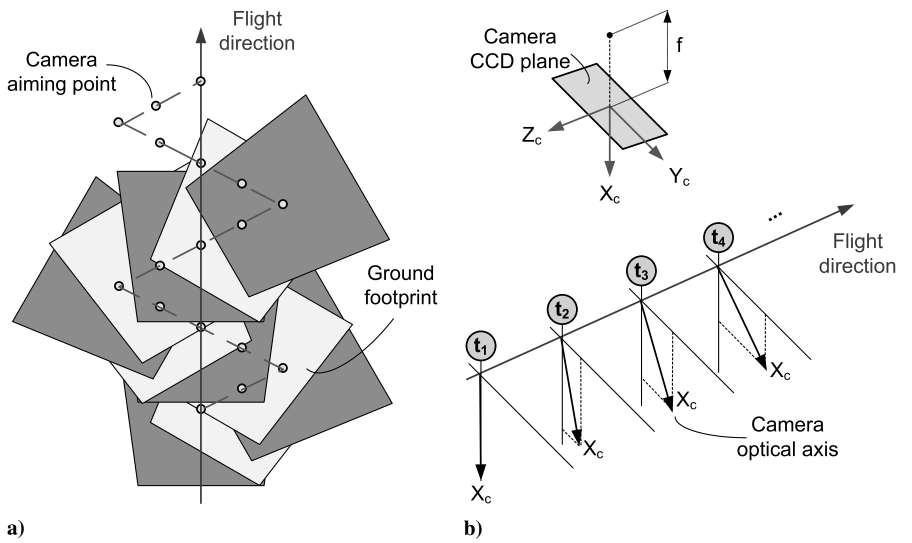


Fig. 2 Schematic illustrations of a) the scanning procedure, and b) camera angles during the scan procedure, and a definition of the camera coordinate system.

A. Scanning Procedure

During flight, the onboard camera captures images of the ground according to commands either from a human operator, an autonomous tracking algorithm of some features on the ground, or a scanning procedure. Figure 2a shows a schematic illustration of the implemented scan procedure. When captured, each new frame is processed and used to update the mosaic image of the flight area. A detailed discussion of the online mosaic construction appears in Sec. III.B.

As can be seen, each image partially overlaps the preceding image as well as images from the previous scan stripe. The existence of overlapping regions is essential for performing image matching between captured images. In addition, and as opposed to most motion-from-structure methods, the additional overlapping region, provided by the camera scanning procedure, enables enhancement of motion estimation, as will be seen in Sec. IV. The proposed scan pattern also allows implementation of improved mosaic construction methods.

We assume that the scanning procedure modifies the pan angle of the camera, ψ_c, while keeping the camera tilt angle constant, as shown in Fig. 2b. Given camera angles at the current time instant, the calculation of camera angles for the next time instant is performed in two steps. First, the line-of-sight vector for the next camera aiming point in the body-fixed reference frame, $\hat{\mathbf{r}}^B$, is determined according to

$$\hat{\mathbf{r}}^B = T_B^C(\psi_c)\frac{[f, d\cdot \mathrm{CCD}_{\mathbf{Y}_C}/2, 0]^T}{\|[f, d\cdot \mathrm{CCD}_{\mathbf{Y}_C}/2, 0]^T\|} \tag{1}$$

where $T_B^C(\psi_c)$ is the directional cosines matrix (DCM) transforming from the camera reference frame to the body frame, computed based on current camera angles; f is the camera focal length; d is the scan direction, so that $d = 1$ for increasing the camera pan angle and $d = -1$ for decreasing the camera pan angle; and $\mathrm{CCD}_{\mathbf{Y}_C}$ is the size of the camera charged coupled device in pixels along the $\mathbf{Y}_C$ axis.

The next step is to compute the new camera angles from $\hat{\mathbf{r}}^B$. The DCM transforming from B to C can be written as

$$T_C^B(\psi_c) = \begin{bmatrix} 0 & \sin\psi_c & \cos\psi_c \\ 0 & \cos\psi_c & -\sin\psi_c \\ -1 & 0 & 0 \end{bmatrix} \tag{2}$$

Because the aiming point vector in C is, by definition, $[1 \quad 0 \quad 0]^T$, one can write

$$\hat{\mathbf{r}}^B = T_B^C(\psi_c)[1 \quad 0 \quad 0]^T = [0 \quad \sin\psi_c \quad \cos\psi_c]^T \tag{3}$$

hence

$$\psi_c = \tan^{-1}\left[\frac{\hat{\mathbf{r}}^B(2)}{\hat{\mathbf{r}}^B(3)}\right] \tag{4}$$

The scanning direction d is switched once the camera pan angle ψ_c reaches a certain prespecified level; this level is constrained by the corresponding gimbal limit but may be smaller than this mechanical limit.

For simplicity, it is assumed implicitly that the velocity-over-height ratio and the camera sampling frequency provide sufficient overlapping regions between each two adjacent images along the flight direction. Thus, the proposed scan methodology moves the camera only in a direction perpendicular to the flight trajectory. Notice that in a practical application this imposes a complex tradeoff among flying altitude over ground, platform speed, FOV, scanning slant angle and resolution. However, the method presented here may be adjusted to work with a relaxed version of the assumption. Note also that no additional or a priori information is required.

B. Mosaic Construction Method

We shall now present a detailed description of the mosaic construction method using the camera scanning method described in Sec. III.A. During the scan, images are captured using varying camera angles. While all the images contribute to the construction of a mosaic, in the current implementation only images taken while the camera was pointing downward are used for motion estimation. These images are referred to as *downward-looking images*.

Figure 3 provides a block diagram of the implemented mosaic construction process. Two mosaic representations are constructed in the proposed approach: a temporary mosaic image that is used for motion estimation, and the main mosaic image which is the final mosaic image constructed based on the captured images.

The temporary mosaic image is initialized to a downward-looking image, once such an image is captured, and is updated with new nondownward-looking images. When a new downward-looking image is captured, it is matched to a relevant region in the temporary

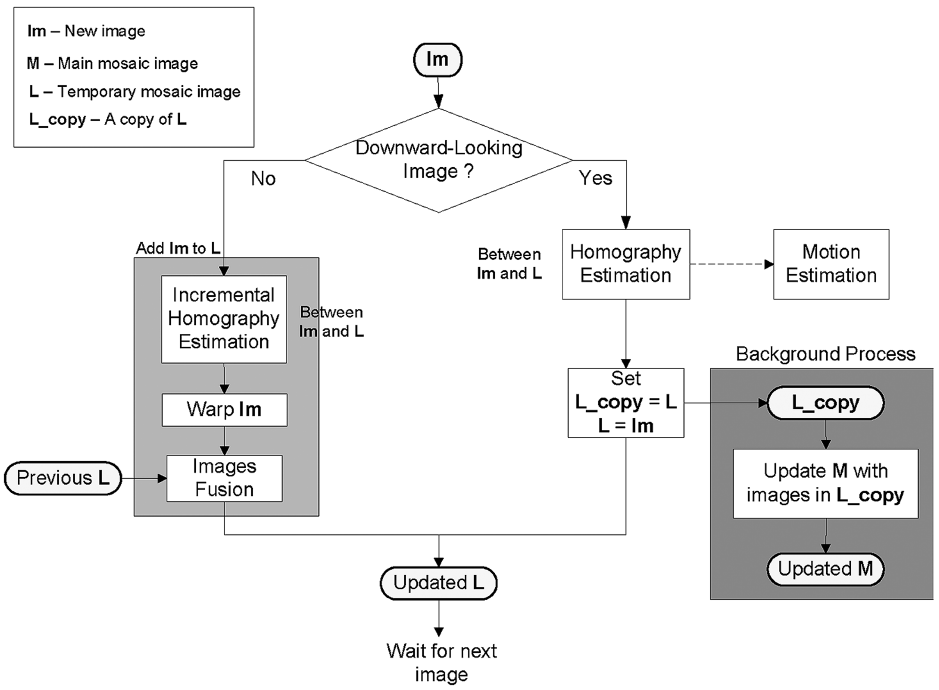
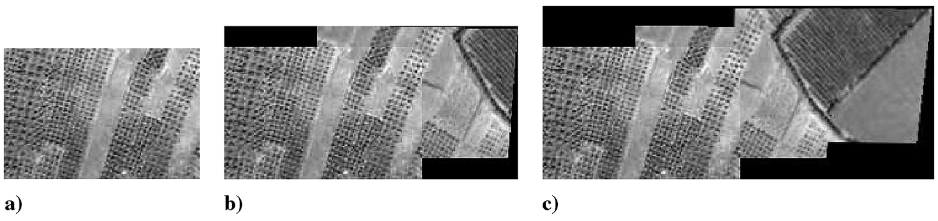


Fig. 3 Diagram of the proposed mosaic construction method.



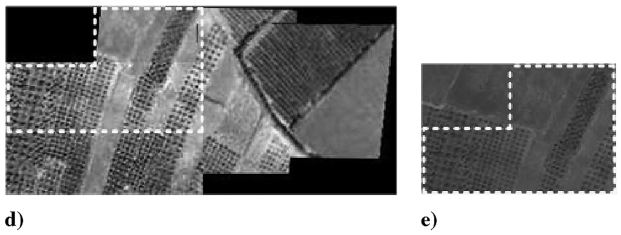


Fig. 4 Mosaic images construction example. Parts a–d temporary mosaic image construction, and part e) a new downward-looking image. An enlarged overlapping area between this image and the temporary mosaic image is shown in (d) and (e).

mosaic image, which is calculated using information from the navigation system. Next, motion estimation is performed, as will be discussed in Sec. IV.

The temporary mosaic image is expressed in the preceding downward-looking image system, defined as the coordinate system C of the previous downward-looking image. Therefore, the estimated motion describes the relative motion performed by the camera between two adjacent downward-looking images. This estimation will be used to correct developing inertial navigation errors (cf. Sec. V).

Because of the coupling between the scanning procedure and the mosaic construction process, an enlarged overlapping area between the new downward-looking image and the temporary mosaic image is achieved. This, and the quality of the constructed temporary mosaic image, are the two factors that allow better motion estimation in certain scenarios, as will be demonstrated in Sec. VI.A.

After motion estimation is performed, the temporary mosaic image is reset and initialized to the new downward-looking image. The images that were removed from the temporary mosaic image are then used for updating the main mosaic image. Because the main mosaic image is not used for motion estimation, it may be updated in a background process. This may be performed by applying various algorithms [19–26,34], depending on the available computational resources.

It should be noted that loop scenarios may be also handled in this background process yielding an improved main mosaic image. In case of a loop in the trajectory, motion estimation and navigation aiding are performed based on the temporary mosaic image, following the method suggested herein. However, a different approach is required for using the full potential of the available information in such an event (e.g., three overlapping images), which is the subject of an ongoing research.

An example of the mosaic image construction process, based on real images acquired using the scanning procedure described above, is given in Figs. 4 and 5. The images were extracted from Google Earth, as detailed in Sec. VI. Figures 4a–4d show the construction of the temporary mosaic image, involving a camera scanning procedure that is composed of two nondownward-looking images in each direction. The temporary mosaic image is initialized with a downward-looking image (Fig. 4a) and is updated with images until a new downward-looking image is acquired (Fig. 4e). One can easily notice the enlarged overlapping region between this new downward-looking image and the temporary mosaic image (Figs. 4d and 4e). Figures 5a and 5b show the update of the main mosaic image, based on images from the temporary mosaic image, once a new downward-looking image was captured: Fig. 5a is the main mosaic image before the update; Fig. 5b shows the main mosaic image after the update.

The following sections further elaborate on several aspects of the mosaic images construction process. The first step is to briefly review a standard method for estimating a homography matrix; this is followed by additional implementation details of the mosaic construction process.

1. Homography Matrix Estimation

The homography matrix [26] is a transformation that relates two images of a planar scene for a general camera motion. The homography relation is also valid for a three-dimensional scene if the camera performs a pure rotation, although this is hardly relevant for fixed-wing aerial platforms. Given some point $\mathbf{x}$ in the first image and a matching point $\mathbf{x}'$ in the second image, both expressed in homogeneous coordinates [26], the following constraint can be written for the homography matrix, H: $\mathbf{x}'_i \cong H\mathbf{x}_i$.

The homography matrix is used for updating the mosaic with each new image frame, and also for performing relative motion estimation

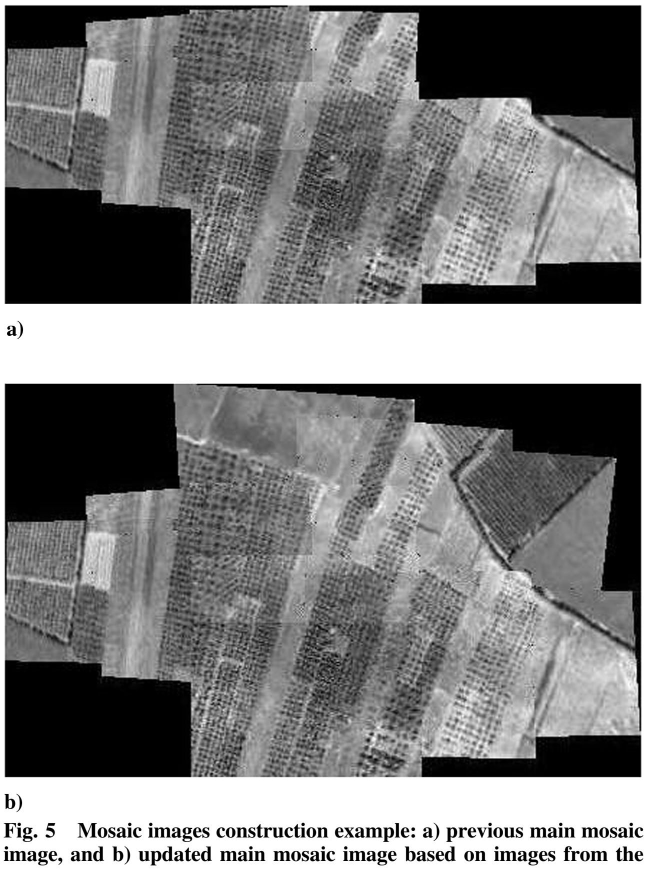


Fig. 5 Mosaic images construction example: a) previous main mosaic image, and b) updated main mosaic image based on images from the temporary mosaic image (Fig. 4d) and the new downward-looking image (Fig. 4a).

between these images. There are various methods for estimating the homography matrix given two partially overlapping images. The method used herein relies on [26]: first, scale invariant feature transform (SIFT) [35] features and their descriptors are computed for each of the images. If one of the images is a mosaic image, an overlapping area between the two images is estimated based on information extracted from the navigation system, and the computation of SIFT features is performed only on this part of the mosaic image. Next, features from the two images are matched based on the minimum Euclidean distance of their descriptor vectors, which yields a set of matched points $\mathcal{S}$.

As the set $\mathcal{S}$ may contain wrong matches (outliers), a robust estimation technique is applied, which provides a refined set of matched points, $\mathcal{R} \subseteq \mathcal{S}$. This is performed by applying the random sample consensus (RANSAC) algorithm [36] for robust estimation of the homography matrix [26]. The final step in the homography estimation algorithm is to perform least-squares (LS) homography estimation based on the subset $\mathcal{R}$ of feature matches [26].

2. *Nondownward-Looking Images*

If the new image is a non downward-looking image, the homography estimation is *incremental*, as described below. The purpose of the incremental estimation is the reduction of the accumulated alignment errors in the temporary mosaic image, while updating the mosaic with new non downward-looking images.

The method proposed here is an adaptation of the procedure suggested by [29,37] for the camera scanning method used herein. Denote by r the index of the most recent downward-looking image, defining the reference frame in which the current temporary mosaic image is expressed. Each new image I_k, which is a nondownward-looking image, is matched against the previous image I_{k-1}, yielding a homography between these two images $H_{k\to k-1}$. The next step is to calculate an intermediate homography matrix between the new image I_k and the current temporary mosaic image, relying on information computed for the previous image I_{k-1}:

$$H^I_{k\to r} = H_{k-1\to r} \cdot H_{k\to k-1} \tag{5}$$

where $H_{k-1\to r}$ is the homography matrix transforming from the previous image I_{k-1} to the current temporary mosaic image. This homography matrix was calculated and saved while processing image I_{k-1}. Once this homography is available, the new image I_k is warped toward the temporary mosaic image using the homography matrix $H^I_{k\to r}$, yielding the warped image $\tilde{I}^r_k$.

Ideally, the warped image and the temporary mosaic image should be aligned; however, this is usually not true in practice due to homography estimation errors. To improve the estimation, a *correction homography* between the warped image $\tilde{I}^r_k$ and the current temporary mosaic image, is estimated by applying the standard homography estimation technique discussed in Sec. III.B.1 on these two images. This homography, H_{corr}, is used to correct the estimated intermediate homography matrix between the new image and the temporary mosaic image

$$H_{k\to r} = H_{\text{corr}} \cdot H^I_{k\to r} \tag{6}$$

Finally, the new image I_k is warped toward the current temporary mosaic image using the improved homography matrix, $H_{k\to r}$, followed by an integration of the two images into an updated mosaic, using one of the available techniques [19,26]. In addition, $H_{k\to r}$ is saved for future use with new nondownward-looking images. The process repeats for an each new image that was not taken when the camera was looking downward.

3. *Downward-Looking Images*

Once a new downward-looking image I_d is captured, a direct estimation of the homography matrix (cf. Sec. III.B.1) relating this new image to the current temporary mosaic image is performed. During this process, only part of the temporary mosaic image is used, based on the current platform heading and altitude (known from the navigation system). The estimated homography matrix is then used for motion estimation (cf. Sec. IV). The next step is to remove all the images from the temporary mosaic image and to initialize it with I_d.

Let r denote the index of the previous downward-looking image. Now, the images $\{I_i\}^d_{i=r+1}$ should be used for updating the main mosaic image. This may be performed in a background process, using various approaches, since the main mosaic image is not required for motion estimation. The approach that was implemented in this work is to use the incremental homography estimation technique, discussed above, for adding the images $\{I_i\}^d_{i=r+1}$ to the main mosaic image.

IV. Image-Based Motion Estimation

In this section, we discuss a method for image-based motion estimation based on a previously estimated homography matrix. The camera motion between any two views of a planar scene, related by a homography matrix H, is encoded in H according to [28]:

$$H = K'\left[R - \frac{\mathbf{t}}{z}\mathbf{n}^T\right]K^{-1} \tag{7}$$

where K', K are the calibration matrices at two image time instances, assumed to be known, $\mathbf{t}$ is the translation vector, R is the rotation matrix, z is the scene depth and $\mathbf{n}$ is a unit vector normal to the scene plane. Because the process of motion estimation is based only on the information provided by the camera, the translation motion between two images can be estimated only up to scale (i.e., only the translation direction can be estimated).

A method for extracting the motion parameters from Eq. (7) was suggested in [28], where it was proven that there are at most two valid sets of solutions $(\mathbf{t}, R, \mathbf{n})$. The correct solution out of these two alternatives can be chosen based on $\mathbf{n}$ while relying on previous estimates [28]. The implementation of the estimation process in this work involves yet another phase, which will be described in the next section. This phase allows improved precision motion estimation assuming a standard approach for estimating the homography matrix (cf. Sec. III.B.1).

It should be noted that any two views of a *non*-planar scene are related through the more complex epipolar geometry, from which relative motion parameters may be extracted as well [26] via, e.g., the fundamental matrix. However, when assuming a narrow-FOV camera, the epipolar geometry method tends to become ill-conditioned, due to the limited ground information captured by the camera, resulting in a semiplanar scene.

A. Implementation of Motion Estimation Assuming a Planar Scene

As mentioned in Sec. III.B.1, the homography estimation process involves the RANSAC algorithm for robust outliers rejection. The output of this algorithm is a subset $\mathcal{R} = \{(\mathbf{x}_i, \mathbf{x}'_i)\}^{N_\mathcal{R}}_{i=1}$ of feature matches that are considered to be inliers. These are then used for LS estimation of the homography matrix. When considering ideal features, this process yields the same results when executed several times. However, the solution varies from one execution to another for noisy data (for a given threshold value), because each execution may yield a different features subset, and hence a different estimation of the homography matrix (and motion parameters).

More specifically, assume that the extracted SIFT features image coordinates are corrupted by noise. As a consequence, the computed set of all point matches $\mathcal{S} = \{(\mathbf{x}_i, \mathbf{x}'_i)\}^{N_\mathcal{S}}_{i=1}$, $\mathcal{R} \subseteq \mathcal{S}$, is also corrupted by noise, and in addition may contain false matches (outliers). In each iteration of the RANSAC algorithm, four point matches are drawn and used to compute a homography matrix, which is then used to construct a subset $\mathcal{T}$ of point matches that are consistent with this homography matrix. Thus, a pair of point correspondences $(\mathbf{x}, \mathbf{x}')$ is chosen if

$$d_E(\mathbf{x}, H^{-1}\mathbf{x}')^2 + d_E(\mathbf{x}', H\mathbf{x})^2 < t^2_{\text{threshold}} \tag{8}$$

where $d_E(.,.)$ is the Euclidean distance between two points in the same image, and $t_{\text{threshold}}$ is a predefined threshold.

Algorithm 1 Sequential Estimation of the Motion Parameters

```
1:  Run $N$ times the homography estimation routine, given in Sec. III.B.1, and calculate the solution set from the estimated homography matrices: $\{(\mathbf{t}_i, R_i, \mathbf{n}_i)\}_{i=1}^{N}$.
2:  if At least one image was already processed then
3:      Compute a score $s_i$ for each solution, based on Eqs. (9) and (10).
4:      Reject solutions whose score is lower than $s_\mu - s_\sigma$, where $(s_\mu, s_\sigma)$ are the mean and standard deviation values of the computed set of scores $\{s_i\}_{i=1}^{N}$.
5:  end if
6:  Construct a translation matrix $\Lambda$ based on Eq. (11) and examine each of its columns for outliers. Solutions that contain outliers are rejected, yielding a refined set $\{(\mathbf{t}_i, R_i, \mathbf{n}_i)\}_{i=1}^{N_2}$ of solutions.
7:  if At least one image was already processed then
8:      Choose a solution $(\mathbf{t}, R, \mathbf{n}) \in \{(\mathbf{t}_i, R_i, \mathbf{n}_i)\}_{i=1}^{N_2}$ with the highest score.
9:  else
10:     Choose a solution $(\mathbf{t}, R, \mathbf{n}) \in \{(\mathbf{t}_i, R_i, \mathbf{n}_i)\}_{i=1}^{N_2}$ which maximizes $\langle \mathbf{n}, \mathbf{n}_\mu \rangle$, where $\mathbf{n}_\mu$ is computed according to Eq. (12).
11: end if
```

Consider such two different iterations yielding the subsets $\mathcal{T}_1$ and $\mathcal{T}_2$, and assume that these subsets do not contain any false matches. In each of these subsets, the homography matrix was computed based on a different set of drawn 4 point matches. These two homography matrices, H_1 and H_2, are expected to be different, despite the fact that they were computed based on inlier point matches, since all the features in $\mathcal{S}$, and in particular the drawn features, are corrupted by noise.

In the next step of the RANSAC algorithm, all point matches in $\mathcal{S}$ are checked for consistency with the homography matrix, H_i, $i = \{1, 2\}$, according to Eq. (8). Only point matches that agree with the condition (8) are added to the subset $\mathcal{T}_i$. Because both homography matrices are legitimate but different, it is obvious from Eq. (8) that the two subsets $\mathcal{T}_1$ and $\mathcal{T}_2$ will be identical for a sufficiently large value of $t_{\text{threshold}}$. However, decreasing the value of $t_{\text{threshold}}$ will yield different subsets $\mathcal{T}_1$, $\mathcal{T}_2$, starting from some critical value. This critical value is influenced by the image features noise characteristics, and is therefore a function of the observed scene: high-texture scenes are likely to be corrupted with less noise compared with low-texture scenes, since in the former case the features can be localized with improved precision. Thus, a specific value of $t_{\text{threshold}}$ might yield identical subsets $\mathcal{T}_1$, $\mathcal{T}_2$ in certain scenarios, and different subsets in other scenarios.

The above conclusion is valid also for the output from the RANSAC algorithm (the inliers subset, $\mathcal{R}$), as it is merely one of the subsets constructed during its iterations. Thus, sequential activation of the RANSAC algorithm might give different subsets of $\mathcal{R}$, meaning that the LS homography estimation will yield a number of different homography matrices $\{H_i\}$, and consequently, a set of motion parameters extracted from each homography matrix H_i. This process of homography matrix estimation, involving a sequential execution of the RANSAC algorithm, is denoted in this section as *sequential homography estimation*.

Given the set of motion parameters, $\{(\mathbf{t}_i, R_i, \mathbf{n}_i)\}_{i=1}^{N}$, obtained from the sequential homography estimation process, one can employ different logic for automatically choosing the most accurate motion estimation. The logic implemented in this work consists of the following steps.

Denote $|\mathbf{q}| = |[q_1, \ldots, q_n]^T| \triangleq [|q_1|, \ldots, |q_n|]^T$. Define the mean unit vector normal to a scene plane, based on the normal unit vectors $\{\mathbf{n}_i^{\text{prev}}\}_{i=1}^{N_{\text{prev}}}$ from estimates of N_{prev} previous images, as

$$\mathbf{n}_\mu^{\text{prev}} = \frac{\Sigma_{i=1}^{N_{\text{prev}}} |\mathbf{n}_i^{\text{prev}}|}{\| \Sigma_{i=1}^{N_{\text{prev}}} |\mathbf{n}_i^{\text{prev}}| \|} \tag{9}$$

Compute a score for each available solution $(\mathbf{t}_i, R_i, \mathbf{n}_i)$ based on the proximity of its normal unit vector $\mathbf{n}_i$ to the mean unit vector $\mathbf{n}_\mu^{\text{prev}}$:

$$s_i = |\langle \mathbf{n}_\mu^{\text{prev}}, \mathbf{n}_i \rangle| \tag{10}$$

where $\langle ., . \rangle$ is the inner product operator. Calculate the mean and the standard deviation (s_μ, s_σ) of the set $\{s_i\}_{i=1}^{N}$, and reject all the solutions whose score is lower than $s_\mu - s_\sigma$. Denote by N_1 the number of remaining solutions.

Next, a translation matrix $\Lambda = (\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \boldsymbol{\lambda}_3)$ is constructed from the absolute values of the translation motion estimation vectors in the set of remaining solutions ($\Lambda \in \mathbb{R}^{N_1 \times 3}$):

$$\Lambda = (\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \boldsymbol{\lambda}_3) \equiv \begin{bmatrix} |\mathbf{t}_1^T| \\ |\mathbf{t}_2^T| \\ \vdots \\ |\mathbf{t}_{N_1}^T| \end{bmatrix} \tag{11}$$

Each of the Λ columns is examined for outliers based on the distribution of its values. More specifically, a histogram of the vector $\boldsymbol{\lambda}_i$ is computed over N_1 slots in the range $[\min(\boldsymbol{\lambda}_i), \max(\boldsymbol{\lambda}_i)]$, followed by a rejection of entries in $\boldsymbol{\lambda}_i$ which do not appear in clusters. Denote by N_2 the number of remaining solutions after this step was applied on all three columns of Λ.

Finally, a solution is chosen among the remaining-solutions set $\{(\mathbf{t}_i, R_i, \mathbf{n}_i)\}_{i=1}^{N_2}$, whose normal is the closest to $\mathbf{n}_\mu^{\text{prev}}$, i.e., a solution with the highest score s_i.

If the mean normal vector from previous images is unavailable, a solution $(\mathbf{t}, R, \mathbf{n})$ is chosen whose normal vector $\mathbf{n}$ is the closest to the mean normal vector of all the other solutions in $\{(\mathbf{t}_i, R_i, \mathbf{n}_i)\}_{i=1}^{N_2}$, i.e., a solution i that maximizes $\langle \mathbf{n}_i, \mathbf{n}_\mu \rangle$ where $\mathbf{n}_\mu$ is defined as

$$\mathbf{n}_\mu = \frac{\Sigma_{i=1}^{N_2} |\mathbf{n}_i|}{\| \Sigma_{i=1}^{N_2} |\mathbf{n}_i| \|} \tag{12}$$

The sequential estimation process of the motion parameters described above is summarized in Algorithm 1. The improvement in the estimation precision is demonstrated in Sec. VI.A.

V. Fusion of Image-Based Relative Motion Estimation with a Navigation System

In this section we present a method for fusing the image-based estimated camera relative motion with a navigation system of an aerial platform. A measurement model is developed that relates the image-based estimated relative motion with the accumulating navigation errors of a standard inertial navigation system (INS). The data fusion is performed using an indirect implicit extended Kalman filter (IEKF) [38] that estimates the navigation parameter errors instead of the parameters themselves. These estimated errors are then used for correcting the navigation parameters. The state vector to be estimated is composed of navigation errors and of inertial sensor errors. The state vector used in this work is defined as

$$\mathbf{X} = [\, \Delta\mathbf{P}^T \quad \Delta\mathbf{V}^T \quad \Delta\boldsymbol{\Psi}^T \quad \mathbf{d}^T \quad \mathbf{b}^T \,]^T \tag{13}$$

where $\Delta\mathbf{P} \in \mathbb{R}^3$, $\Delta\mathbf{V} \in \mathbb{R}^3$, $\Delta\boldsymbol{\Psi} = (\Delta\phi, \Delta\theta, \Delta\psi)^T \in [0, 2\pi] \times [0, \pi] \times [0, 2\pi]$ are the position, velocity and attitude errors, respectively, and $(\mathbf{d}, \mathbf{b})$ is the parametrization of errors in the inertial sensor measurements: $\mathbf{d} \in \mathbb{R}^3$ is the gyro drift, and $\mathbf{b} \in \mathbb{R}^3$ is the accelerometer bias. The first 9 components of $\mathbf{X}$ are given in LLLN coordinates, while the last 6 are written in a body-fixed reference frame.

Once an estimate of the state vector is available, the vector is used both to correct the output of the navigation system, and to provide a feedback to the inertial sensors (cf. Fig. 1): the estimated position, velocity and attitude errors are used to correct the INS output. These components in the state vector are then reset, since they have been incorporated into the navigation system, i.e., the first nine components of the a posteriori estimation at some time instant t_k, $\hat{\mathbf{X}}_{k|k}$, are set to zero. The covariance matrix is not changed, so it represents the uncertainty in the platform's navigation parameters estimation, i.e., the navigation errors. In addition, the IMU measurements readings are corrected with the most recent available estimations of drift and bias parameters at the frequency of the inertial sensors readings,** which is much higher than the frequency of the IEKF updates.

It is worth stressing that the state vector is a constant-size vector, $\mathbf{X} \in \mathbb{R}^{15}$. Another possible approach is to use a direct data-fusion technique and relate the vision-based estimated motion to an augmented state vector composed of the platform current parameters (e.g., position, velocity) and past poses for each captured image [27]. Yet, as in the case of SLAM methods, this approach requires increasing computational resources (because the state vector size increases with time), and therefore the use of the indirect fusion approach with a constant-size state vector is preferred.

It is assumed here that the relative motion parameters between each two image time instances, $t = t_1$ and $t = t_2$, were already extracted by the image-processing module. Thus, the camera relative rotation matrix, $R_{C_1}^{C_2}$, transforming from the camera axes at time t_2 to the camera axes at time t_1, is known. In addition, the relative translation, $\mathbf{t}_{1\to 2}^{C_2}$, is known up to some scale, γ.

The estimated relative motion is reformulated into residual measurements $\mathbf{z}_{\text{translation}}$, $\mathbf{z}_{\text{rotation}}$, which are injected into the filter:

$$\mathbf{z}_{\text{translation}} = [\mathbf{Pos}_{\text{Nav}}(t_2) - \mathbf{Pos}_{\text{Nav}}(t_1)]^{L_2} \times T_{L_2,\text{Nav}}^{C_2}\hat{\mathbf{t}}_{1\to 2}^{C_2} \quad (14a)$$

$$\mathbf{z}_{\text{rotation}} = \begin{pmatrix} \tan^{-1}\left[\frac{D(3,2)}{D(3,3)}\right] \\ -\sin^{-1}[D(1,3)] \\ \tan^{-1}\left[\frac{D(1,2)}{D(1,1)}\right] \end{pmatrix} \qquad D \doteq T_{C_1,\text{Nav}}^{C_2}[\hat{R}_{C_1}^{C_2}]^T \quad (14b)$$

where $T_{L_2}^{C_2}$ is the directional cosines matrix (DCM) transforming from C to LLLN at the time instance $t = t_2$; $T_{C_1}^{C_2}$ is the DCM transforming from C at $t = t_2$ to C at $t = t_1$; and $\mathbf{Pos}$ is the platform's position. The subscript Nav denotes the parameters that are taken from the navigation data.

The state vector and the residual measurements are related via a measurement equation

$$\mathbf{Z} \doteq \begin{pmatrix} \mathbf{z}_{\text{translation}} \\ \mathbf{z}_{\text{rotation}} \end{pmatrix} = H\mathbf{X} + \begin{pmatrix} \mathbf{v}_{\text{tr}} \\ \mathbf{v}_{\text{rot}} \end{pmatrix} \quad (15)$$

where H is the measurement matrix

$$H = \begin{bmatrix} 0_{3\times 3} & H_{\Delta V}^{\text{tr}} & H_{\Delta\Psi}^{\text{tr}} & H_d^{\text{tr}} & H_b^{\text{tr}} \\ 0_{3\times 3} & 0_{3\times 3} & H_{\Delta\Psi}^{\text{rot}} & H_d^{\text{rot}} & 0_{3\times 3} \end{bmatrix} \quad (16)$$

$\mathbf{v}_{\text{tr}}$ is given by

$$\mathbf{v}_{\text{tr}} = \left[\mathbf{Pos}_{\text{True}}(t_2) - \mathbf{Pos}_{\text{True}}(t_1)\right]^{L_2} \times \left[T_{L_2,\text{Nav}}^{C_2}\hat{\mathbf{t}}_{1\to 2}^{C_2} - T_{L_2,\text{True}}^{C_2}\mathbf{t}_{1\to 2,\text{True}}^{C_2}\right] \quad (17)$$

and $\mathbf{v}_{\text{rot}}$ represents rotation motion estimation errors and linearization errors. Note that because an IEKF formulation is used, the measurement noise terms are not necessarily white [2]. The development of the above measurement equation and the explicit expression for H are given in Appendix A.

The estimated state vector is initialized to zero, since the actual initial navigation errors are unknown, while the estimation error covariance matrix is set to the believed levels of navigation errors. Although these values are usually known from the performance specifications of the inertial sensors, in all practical applications the initial covariance and process noise covariance matrices are adjusted during the tunning process.

The propagation step involves computation of an a priori covariance matrix $P_{k+1|k}$ according to

$$P_{k+1|k} = \Phi_d(k+1,k)P_{k|k}\Phi_d^T(k+1,k) + Q_k \quad (18)$$

where $\Phi_d(k+1,k)$, $P_{k|k}$, Q_k are the process discrete system matrix, a posteriori covariance matrix, and the process noise covariance matrix, respectively. The discrete system matrix Φ_d is computed using $\Phi_d = e^{\Phi_c \Delta t}$, where Δt is the propagation step time interval, and Φ_c the continuous system matrix [33].

The propagation of the state vector is given by

$$\hat{\mathbf{X}}_{k+1|k} = \Phi_d(k+1,k)\hat{\mathbf{X}}_{k|k} \quad (19)$$

However, since the first 9 components of $\hat{\mathbf{X}}_{k|k}$ are used for correcting the strapdown integration (see above) after the update step and then reset, and the other 6 components are random constants (i.e., $\hat{\mathbf{X}}_{k|k} = [\mathbf{0}_{1\times 9} \quad \mathbf{d}^T \quad \mathbf{b}^T]^T$), Eq. (19) is equivalent to $\hat{\mathbf{X}}_{k+1|k} = \hat{\mathbf{X}}_{k|k}$.

After performing the propagation step, a measurement update is performed given the motion estimation ($\mathbf{t}_{1\to 2}^{C_2}$, $R_{C_1}^{C_2}$) from the image-processing module. First, the Kalman filter gain matrix is computed according to

$$K_{k+1} = P_{k+1|k}H_{k+1}^T\left[H_{k+1}P_{k+1|k}H_{k+1}^T + R_{k+1}\right]^{-1} \quad (20)$$

The matrix R_{k+1} in Eq. (20) is a measurement noise covariance matrix, which is of the following block-diagonal form:

$$R_{k+1} = \begin{bmatrix} R_{k+1}^{\text{tr}} & 0_{3\times 3} \\ 0_{3\times 3} & R_{k+1}^{\text{rot}} \end{bmatrix} \quad (21)$$

where R^{tr}, R^{rot} are the translation and rotation measurement noise covariance matrices, respectively. While R^{rot} is a constant matrix, an adaptive translation measurement noise covariance matrix R^{tr} is calculated based on Eq. (17):

$$R^{\text{tr}} = -\left[\mathbf{Pos}_{\text{Nav}}^{L_2}(t_2) - \mathbf{Pos}_{\text{Nav}}^{L_2}(t_1)\right]^{\wedge} R^{\text{est}}\left[\mathbf{Pos}_{\text{Nav}}^{L_2}(t_2) - \mathbf{Pos}_{\text{Nav}}^{L_2}(t_1)\right]^{\wedge} \quad (22)$$

where R^{est} is a 3×3 tuning matrix that represents the level of accuracy in the vision-based estimation of the translation direction and $(.)^{\wedge}$ denotes the matrix cross-product equivalent. For example, in the experiments with real imagery presented in Sec. VI.B, it was assumed to be close to $I_{3\times 3}$. It should be noted that the matrices R^{est}, R^{rot} may also be estimated as part of the image-based motion estimation procedure [30].

Once the gain matrix K is available, a posteriori values of the state vector and covariance matrix are computed using the standard IEKF formulas [2,38]:

$$\hat{\mathbf{X}}_{k+1|k+1} = \hat{\mathbf{X}}_{k+1|k} + K_{k+1}\mathbf{Z}_{k+1} \quad (23)$$

$$P_{k+1|k+1} = [I - K_{k+1}H_{k+1}]P_{k+1|k}[I - K_{k+1}H_{k+1}]^T + K_{k+1}R_{k+1}K_{k+1}^T \quad (24)$$

Some of the errors in the image-based relative motion estimation may be projected onto the unobservable states and yield poor estimation performance even when compared with the pure inertial

**Another possible variation of this is to estimate the residual bias and drift values, while maintaining the estimations of actual bias and drift parameters outside the filter. In this case the whole state vector should be reset after each update step [33].

case. To mitigate this phenomenon, several heuristic methods may be considered. For the current implementation, a fictitious ideal velocity measurement was used in addition to the relative motion measurements to overcome the scaling ambiguity, so that

$$(\mathbf{V}^L_{\text{true}})^T \Delta\mathbf{V} = \mathbf{0} \tag{25}$$

Namely, the velocity errors in the direction of the flight are assumed to be zero, and hence errors from the image-processing block are essentially not projected onto this direction. The term $\mathbf{V}^L_{\text{true}}$ refers to the true value of the platform velocity in the LLLN system. Because this velocity is unknown, it is replaced by the platform velocity $\mathbf{V}^L$ taken from the navigation system.

A Kalman filter gain matrix, K, is computed according to Eq. (20) based on an a priori covariance matrix $P_{k+1|k}$, an augmented measurement matrix, $H_{\text{aug}} = [H^T, H_v^T]^T$, and an augmented measurement noise covariance matrix, R_{aug}, where

$$H_v = [\,\mathbf{0}_{1\times 3} \quad (\mathbf{V}^L)^T \quad \mathbf{0}_{1\times 3} \quad \mathbf{0}_{1\times 3} \quad \mathbf{0}_{1\times 3}\,] \tag{26}$$

and H is the measurement matrix of Eq. (16).

The augmented measurement noise covariance matrix R_{aug} is given by

$$R_{\text{aug}} = \begin{bmatrix} R & 0 \\ \mathbf{0}_{1\times 3} & R_v \end{bmatrix} \tag{27}$$

where R is given in Eq. (22) and R_v is the fictitious velocity (FV) measurement noise covariance matrix, which constitutes a tuning parameter. Small-valued entries in R_v indicate that this additional measurement is reliable, and therefore other measurements will have a minor influence on the entries of the gain matrix K, corresponding to position and velocity along the flight heading. This, in turn, prevents from erroneous image-based relative motion measurements to affect the unobservable states.

Once K is computed, the column related to the fictitious velocity measurement is discarded; in this way, the measurement limits the corrections in the direction of the flight but does not render the problem inconsistent, since the measurement is not actually performed. The advantage of using the FV measurement is demonstrated in [33]; thus, all the results of VAN presented in this paper were obtained with the FV measurement active. Moreover, because the FV measurement is homogeneous, it can be weighted into the Kalman filter formulas, thereby providing an additional design parameter to tradeoff estimation quality with projections onto the unobservable subspace. Note that the FV measurement does not limit the platform's motion to any specific type. This is in contrast, for example, to nonholonomic constraints that may be applied only for land vehicles [39]. In addition, due to the varying quality of the image measurements (cf. Sec. VI.A), a measurement-rejection mechanism must also be used to avoid fusion of low-quality measurements or other outliers.

A. Computational Requirements

The overall computational requirements of the proposed navigation aiding architecture consist of applying a standard Kalman filter for constant-size state and measurement vectors of 15 and 6 elements, respectively, and of the image-processing phase. The latter consists of a temporary mosaic image construction, which is limited to a few images (four in the current example), and of motion estimation. The main mosaic image is not used in the navigation aiding scheme.

As discussed earlier, the inclusion of a new image into a temporary mosaic image involves the computation of SIFT features, estimation of the homography matrix, warping the new image and fusing the two images. Only a partial area in the temporary mosaic image is used for calculating SIFT features. These operations require modest computational resources and pose no difficulties for real-time operation.

In conventional SLAM, as opposed to the proposed method, the state vector is augmented with relevant data from each captured image, and thus the whole state vector needs to be updated each time. Denoting by d the number of elements that are added to the state vector once a new image is acquired, SLAM generates, after 100 s of flight (assuming the same image sampling frequency of 5 Hz) a state vector of $15 + 500d$ elements that should be processed in real time, which is far more demanding than applying a Kalman filter to a 15-element state vector and constructing a temporary mosaic image[††] (cf. Sec. III.B.1), as suggested in the new approach.

VI. Results

This section contains simulation results of the developed mosaic-aided navigation method. The simulation is composed of the following modules: A navigation module, a camera scanning module, and an image-processing module.

The navigation phase consists of the following steps: 1) trajectory generation; 2) velocity and angular velocity increments extraction from the created trajectory; 3) IMU error definition and contamination of pure increments by noise; and 4) strapdown calculations. The strapdown mechanism provides, at each time step, the calculated position, velocity and attitude of the platform. In parallel to the strapdown calculations, at a much slower rate, Kalman filter calculations are performed based on the available measurements. At the end of each filter cycle, the strapdown output is updated with the estimated state vector of the filter.

The camera scanning module provides camera angle commands that yield a continuous scan, according to the camera scanning procedure discussed in Sec. III.A.

The image-processing module constructs mosaic images and performs motion estimation each time a downward-looking image is acquired (cf. Secs. III.B and IV). The inputs to this module are real images obtained from Google Earth[‡‡] based on the proposed camera scanning procedure. In addition, the module is capable of calculating an ideal real motion based on the true platform trajectory taken from the navigation module, without actually using any real images. Naturally, in this mode of operation the mosaic images are not constructed. The ideal motion estimations are used as baseline for evaluating the best possible performance of the proposed method, since motion estimation based on real images will be imperfect.

The experiments are based on real image sequences acquired using Google Earth, which contains 3-D geo-data of Earth based on real imagery, i.e., a 3-D environment based on real images and a digital terrain model. For this purpose, we developed an interface that bridges between the navigation simulation and Google Earth, allowing to obtain images from Google Earth at specified camera position and attitude. Further details regarding the interface to Google Earth are provided in Appendix B.

It should be noted that any two images of the same ground region observed from different viewpoints will yield a correct relative image transformation. However, this approach does not mimic real-world images perfectly, since it lacks the effect of lighting variations when some region is observed from different directions. Yet, because the trajectories presented in this paper do not involve loops, the implemented camera scanning procedure will take different images of the same ground region under similar conditions. Thus, the effect of varying lighting conditions is expected to be marginal.

The experiments presented in this paper were conducted while the platform performed a straight-and-level north-heading trajectory, whose initial conditions are given in Table 1. The observed scene along this trajectory is of a planar nature with about 50 m elevation above sea level.

We begin our presentation of the results by demonstrating an improved image-based motion estimation for a narrow-FOV camera

[††]The construction of the main mosaic image in a background process may involve different algorithms. However, while some of them may be computationally expensive (such as global optimization), they are to be applied *only* in case of a loop in a trajectory, or in some low frequency. This is in contrast to the constantly increasing high-computational requirements of the update step in the SLAM approach.

[‡‡]Additional data available at http://earth.google.com/index.html [accessed June 2009].

Table 1 Trajectory parameters

Parameter	Description	Value	Units
λ	Initial latitude	32.8285005298	deg
Λ	Initial longitude	35.1479222075	deg
alt	Initial altitude above sea level	1500	m
$\mathbf{V}^L$	Velocity in LLLN system	$(100, 0, 0)^T$	m/s
Ψ	Platform attitude	$(0, 0, 0)^T$	deg

observing a low-texture scene, when applying the newly proposed coupling algorithm between the camera scanning and mosaic construction processes. The following section then provides results of mosaic-aided navigation. Additional results are given in [33,40].

A. Mosaic-Based Motion Estimation

Before presenting the results for mosaic-based motion estimation, we demonstrate the improvement in the precision of motion estimation when applying the sequential estimation procedure, summarized in Algorithm 1. In the current implementation, this routine was executed with $N = 10$.

Figure 6 presents the results of translation motion estimation when a low-texture scene is observed by a narrow-FOV camera. A pair of such images are shown in Figs. 6a and 6b.

The improvement in the estimation precision is clearly evident in Fig. 6c, where the same pair of images was used to perform motion estimation with and without the sequential estimation procedure. The figure presents a cumulative distribution function (CDF) of errors in the estimation of the translation direction; the x-axis values represent different thresholds of errors (in degrees), while the y-axis represents the percentage of estimations with an estimation error lower than the threshold values.

The results in Fig. 6c were obtained by retrieving motion parameters from a conventionally estimated homography matrix (cf. Sec. III.B.1) and by applying the sequential estimation procedure of motion parameters (Algorithm 1). Both of the methods were executed 100 times on a pair of low-texture images taken with a $7° \times 4°$-FOV camera (Figs. 6a and 6b). The advantage of the sequential estimation method is significant. For example, nearly 80% of the estimation errors were below 20° when applying sequential estimation, compared with only 50% with a standard homography estimation.

Next, we examine the performance of the proposed mosaic-based motion estimation method for cameras with a narrow FOV, compared with estimations of a standard two-view method, in which the motion estimation is based on camera-captured images, without our new camera scanning and mosaicking procedures. In both cases the motion parameters are estimated using the proposed sequential estimation method. Image sequences were acquired from Google Earth, using the same trajectory, for each of the examined motion estimation methods: Images for the traditional two-view motion estimation method were captured using a constant downward-looking camera at a 1 Hz frequency, while images for the mosaic-based motion estimation method were captured according to the camera scanning procedure at a 5 Hz frequency. Among all the images acquired during the camera scanning, the downward-looking images were captured every second, and therefore motion estimation in the mosaic-based method was also applied at a 1 Hz frequency (cf. Figure 3).

The results are presented in Fig. 7, showing the CDF of the translation direction estimation error and of the rotation estimation error. The shown rotation error is the maximum value of the error in the estimated rotation vector, i.e.,

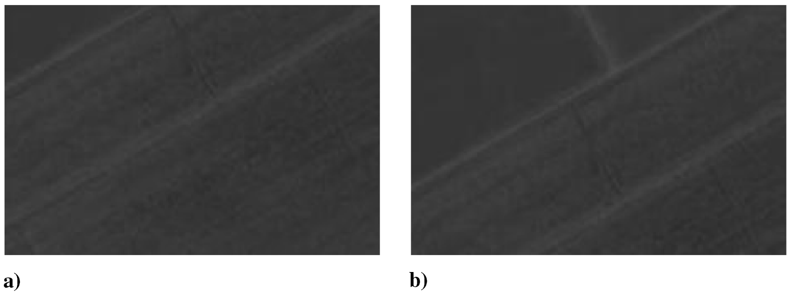
$$\Delta\eta \triangleq \max(|\Delta\phi|, |\Delta\theta|, |\Delta\psi|) \tag{28}$$

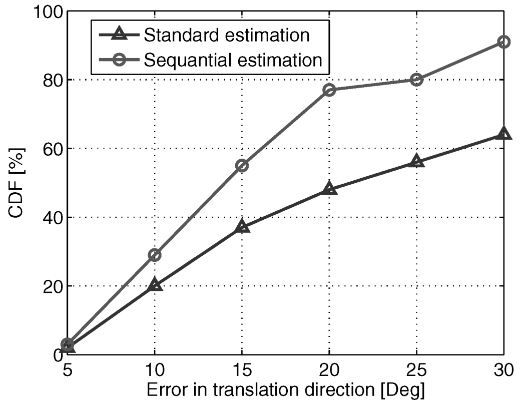
where $\Delta\phi$, $\Delta\theta$, $\Delta\psi$ are the Euler angle errors of the estimated rotation matrix, computed from the DCM R_{err}:

$$R_{\text{err}} \equiv R_{\text{true}} \cdot R^T \tag{29}$$

Here R_{true} and R are the true and estimated values of the rotation matrix, respectively.

It is important to understand when the mosaic-based method is expected to outperform the two-view-based method. In the context of


a) b)



c)

Fig. 6 Images of a low-texture scene captured from Google Earth by a 7° × 4° FOV camera are shown in parts a and b. Motion sequential estimation vs standard estimation over the pair of images presented in parts a and b is shown in part c: CDF of the translation direction estimation error. Significantly improved estimation accuracy in favor of sequential estimation.

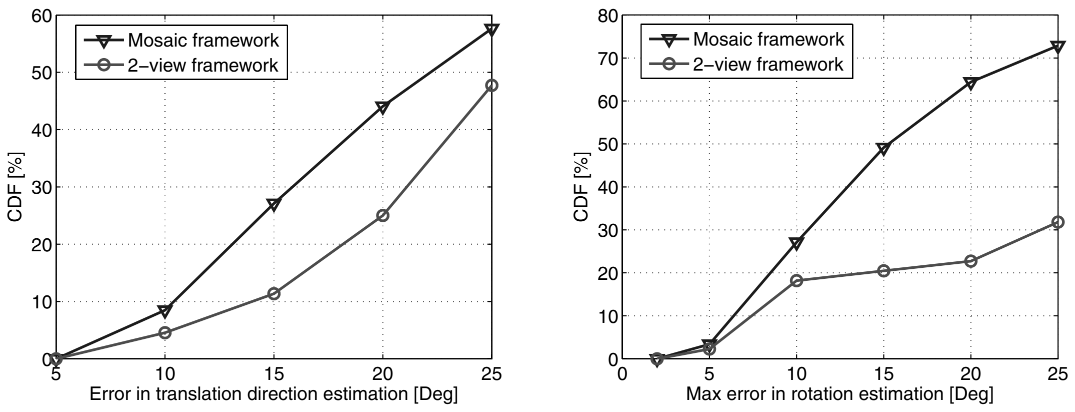


Fig. 7 Image-based motion estimation accuracy for a low-texture scene and a narrow-FOV camera of 5° x 3° (CDF). The mosaic framework significantly improves the estimation accuracy compared to a traditional two-view method.

motion estimation, the two methods differ only in the size of the image overlap region. Because of the camera scanning process, the constructed mosaic image contains an enlarged overlapping region compared with the overlapping region between two regular images. This region is composed of the original overlapping area between two regular images and *an additional* overlapping region, see a schematic illustration in Fig. 2a and a mosaic example image in Figs. 4 and 5. However, because the mosaic construction process is by itself affected by errors, features from the additional overlapping area tend to be of lower-quality compared with those from the original overlapping region, while features from the original overlapping region are of the same quality in both cases (the camera-captured image and the mosaic image), due to the mosaic construction process (cf. Sec. III.B). Thus, there is an inherent tradeoff: On one hand, the mosaic provides an increased number of features, while on the other hand, part of the features are of a lower quality. Hence, the performance of the mosaic-based method is expected to be superior over the two-view framework in difficult scenarios, in which either the overlapping region between two captured images yields a small number of high-quality features or the geometry underlined by the line-of-sight direction to the features is close to singular. In other words, it is expected that the mosaic will outperform the two-view method for the narrow-FOV camera and low-texture scenes.

The above observation is clearly evident in Fig. 7, which describes the scenario of a narrow-FOV camera ($5° \times 3°$) and a low-texture scene. This relatively small FOV is common in many airborne applications. It can be seen that the mosaic-based motion estimation yields considerably better results compared with the two-view motion estimation. For example, in case of translation direction estimation (Figs. 7a), 50% of the estimates using the mosaic method are provided with an accuracy better than 15°, compared with only 20% using the two-view method. As will be seen in the next section, these motion estimations can be effectively used for improving the performance of navigation systems.

B. Mosaic-Aided Navigation

This section contains simulation results of the developed mosaic-aided navigation method following the fusion technique discussed in Sec. V. First, we present navigation results assuming an ideal motion estimation (without using any real images). Next, results of mosaic-aided navigation are presented. These results are compared with two-view aided navigation. We note that the simulation runs were performed without a captive flight stage.

The assumed 1σ values of IMU errors and initial navigation errors are given in Table 2. Actual values of initial navigation errors and IMU errors in the statistical simulation runs are determined by drawing samples from a zero-mean normal distribution with a standard deviation σ, that is, the value of some parameter s_i is drawn according to $s_i \sim N(0, \sigma_{s_i})$.

The IMU errors are then used for contaminating the pure IMU readings, while the initial input to the strapdown module is set according to initial platform parameters and initial navigation errors (cf. Sec. VI).

1. *Navigation Performance Using Ideal Motion Estimation*

Figures 8 and 9 show Monte Carlo results for a straight and level north-heading trajectory, in which the measurements based on an ideal motion estimation were injected into a Kalman filter at a 1 Hz frequency. Each figure contains 4 curves: mean (μ), mean + standard deviation ($\mu + \sigma$), and the square root of the filter covariance, defined for the ith component in the state vector $\mathbf{X}$ as $\sqrt{P(i,i)}$, where P is the a posteriori covariance matrix. In addition, a comparison is provided to an inertial scenario ($\mu + \sigma$ inertial).

The velocity errors are presented in Fig. 8b. Velocity errors normal to the flight heading are significantly reduced relative to the inertial scenario; however, they are not nullified due to errors introduced by expressing the translation measurement in the LLLN system (cf. Appendix A). It can also be seen that these errors are constant and do not grow with time. As a consequence, position errors (Fig. 8a) normal to the flight heading are considerably reduced compared with an inertial scenario. Velocity errors and position errors along the flight heading are not reduced due to lack of observability.

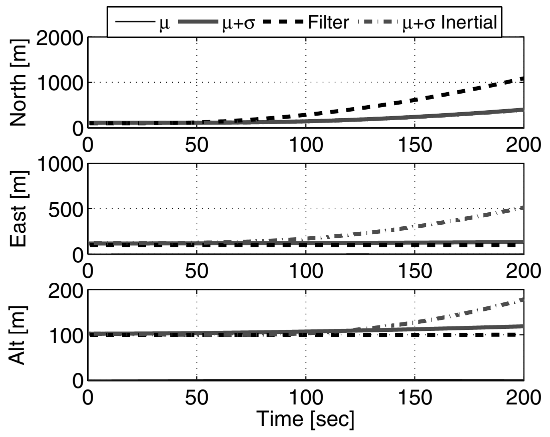
The roll angle error $\Delta\Phi$ (Fig. 9a) is partially estimated and is kept constant relative to the increasing error obtained in an inertial scenario. While the pitch and yaw angles errors ($\Delta\Theta$, $\Delta\Psi$) are not estimated, the error growth is restrained relative to the inertial scenario. This is due to a precise estimation of the drift state $\mathbf{d} = (d_x, d_y, d_z)^T$ (Fig. 9b), that was obtained since ideal rotation motion estimation is used. However, the precision of rotation motion estimation when real images are used is insufficient for estimating the drift state [33]. The bias state, $\mathbf{b} = (b_x, b_y, b_z)^T$, is estimated in the z-direction (Fig. 9b). In general, the filter covariance is consistent with the actual 1σ errors.

2. *Navigation Performance Using Mosaic-Based Motion Estimation*

This section demonstrates the superior performance of mosaic-aided navigation over two-view motion estimation. The examined scenario consists of a narrow-FOV camera ($5° \times 3°$) and a

Table 2 Initial navigation errors and IMU errors

Parameter	Description	Value	Units
$\Delta\mathbf{P}$	Initial position error (1σ)	$(100, 100, 100)^T$	m
$\Delta\mathbf{V}$	Initial velocity error (1σ)	$(0.3, 0.3, 0.3)^T$	m/s
$\Delta\mathbf{\Psi}$	Initial attitude error (1σ)	$(0.1, 0.1, 0.1)^T$	deg
$\mathbf{d}$	IMU drift (1σ)	$(1, 1, 1)^T$	deg /hr
$\mathbf{b}$	IMU bias (1σ)	$(1, 1, 1)^T$	mg


a) Position errors

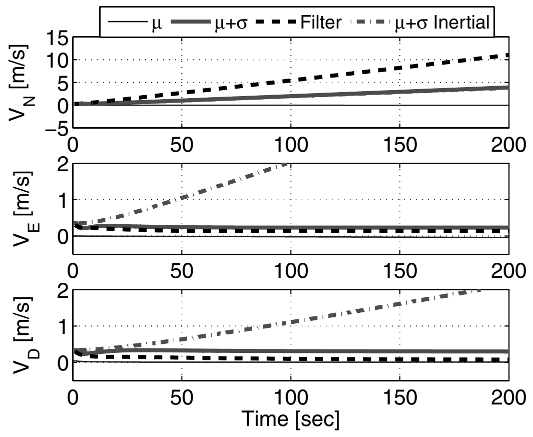
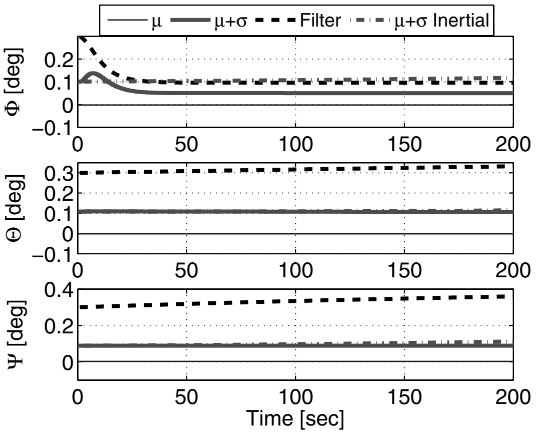

b) Velocity errors

Fig. 8 Navigation errors statistics vs filter covariance (Monte Carlo run) using ideal motion estimation. Part a shows position errors. Errors normal to the flight heading are reduced, errors along the flight heading are not diminished due to lack of observability. Part b shows velocity errors. Errors normal to the flight heading are significantly reduced relative to the inertial scenario, and are kept constant.


a) Euler angle errors

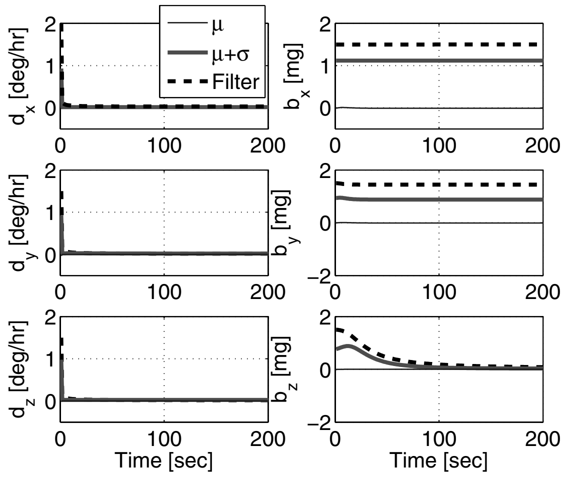
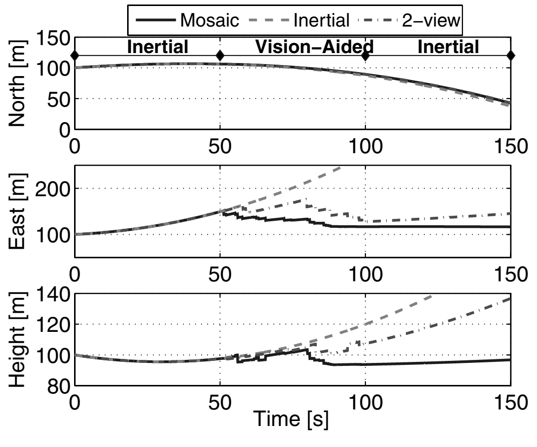

b) Drift and Bias estimation errors

Fig. 9 Navigation errors statistics vs filter covariance (Monte Carlo run) using ideal motion estimation. Part a shows Euler angle errors. Roll angle error, $\Delta\Phi$, is partially estimated and is kept constant relative to the inertial scenario; pitch and yaw angles errors ($\Delta\Theta$, $\Delta\Psi$), development is arrested. Part b shows drift and bias estimation errors. Full drift estimation due to ideal relative rotation measurement. The bias in the z direction is estimated after about 50 s.


a) Position errors

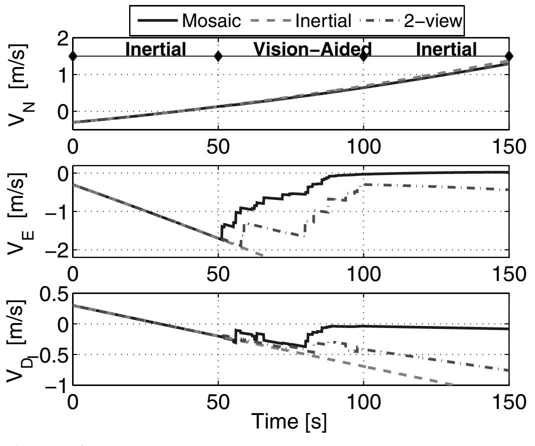
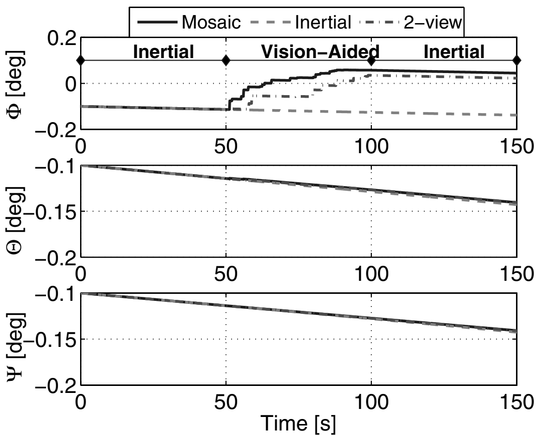
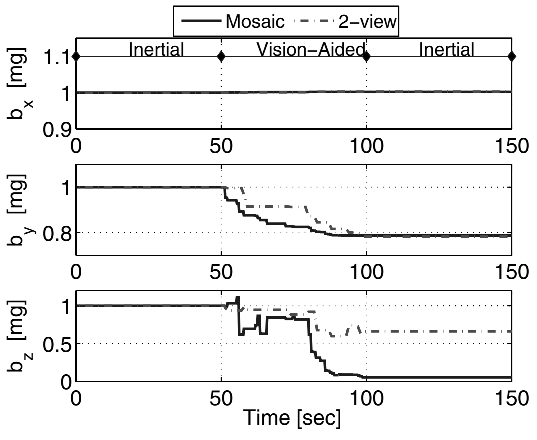

b) Velocity errors

Fig. 10 Vision-aided navigation: mosaic aiding vs two-view framework. Parts a and b show position and velocity errors, respectively. Inertial error development in the north direction due to lack of observability. Reduced errors in the east and down directions, with a significant improvement in favor of the mosaic aiding.



a) Euler angle errors



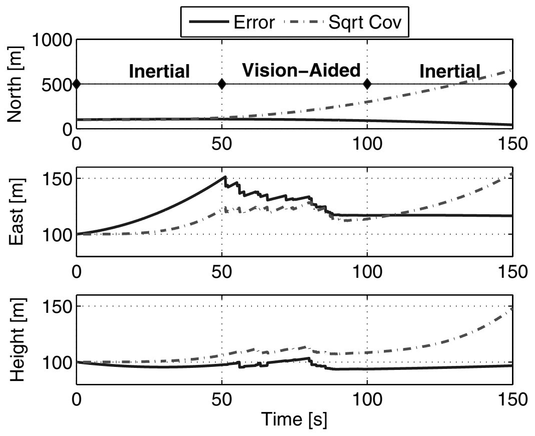
b) Bias estimation errors

Fig. 11 Vision-aided navigation: mosaic aiding vs two-view framework. Part a shows Euler angle errors. Roll angle error estimation for both motion estimation methods. Pitch and yaw angles errors are not reduced due to lack of observability. Part b shows bias estimation errors. Considerably improved b_z estimation in favor of mosaic-aided navigation.

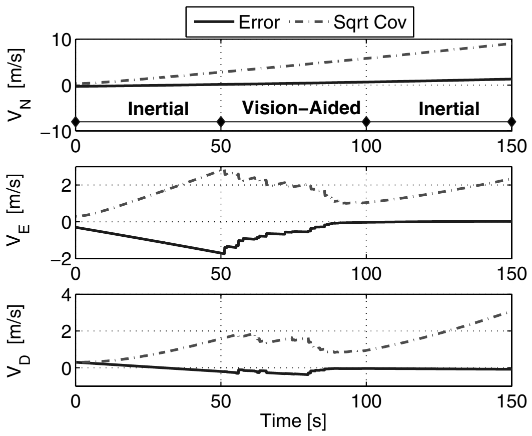
low-texture scene. The platform performs a straight and level north-heading flight, as described in Sec. VI.

The experiment consisted of 50 s of inertial flight, followed by a 50 s of vision-aided phase, during which the mosaic- and two-view-based motion estimations were injected into the navigation system. The last phase is another inertial navigation flight segment for 50 s. Figures 10 and 11 provide the experimental results, comparing the navigation performance for the two examined methods (mosaic and two-view). In addition, the development of inertial navigation errors is given for reference.

The enhanced performance of mosaic-aided navigation can be clearly seen. During the vision-aided phase, the position and velocity

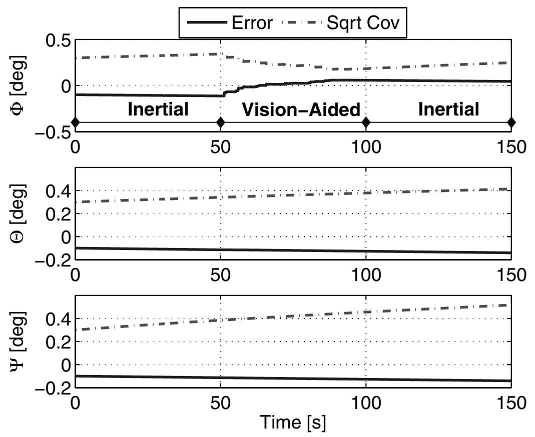


a)

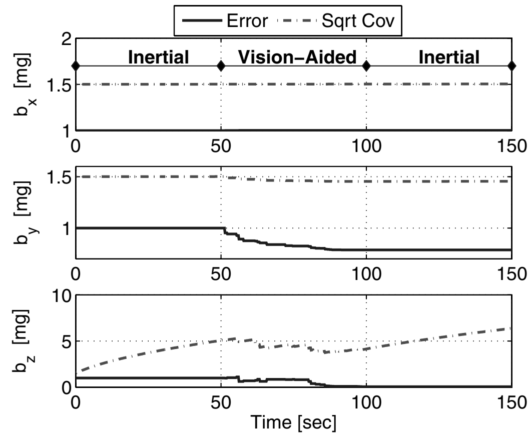


b)

Fig. 12 Actual navigation errors vs filter covariance: mosaic aiding. Position errors, part a. Velocity errors, part b. A consistent overall behavior of filter covariances, compared with the actual errors.



a)



b)

Fig. 13 Actual navigation errors vs filter covariance: mosaic aiding. Euler angles errors, part a. Bias estimation errors, part b. A consistent overall behavior of filter covariances compared with the actual errors.

errors (Figs. 10a and 10b) perpendicular to the flight heading are significantly reduced. The mosaic-based aiding yields better results than two-view-based aiding, due to more accurate vision-based motion estimation. It can be concluded from these graphs that the number of measurements accepted by the filter is considerably higher in case of the mosaic framework (between 60 and 80 sec, all the measurements in the two-view method were rejected by the filter). As for the roll angle error (Fig. 11a), although this error is smaller with the two-view method, it is expected to reach higher values if more measurements were accepted by the filter.

When examining the behavior of navigation errors in an inertial segment (after the vision-aided phase), one can notice the slow development of inertial errors when using mosaic aiding. The reason for this is the improved bias estimation compared with the estimation using the two-view method, as shown in Fig. 11b: b_z is almost exactly estimated and thus it does not contribute to the growth of inertial position and velocity errors in the down axis. The drift state was not estimated at all, because all the relative rotation measurements were rejected by the filter due to their low quality.

The relative motion measurements have another interesting effect: Although the position error does not have a representation in the translation measurement matrix [cf. Eq. (A8)], the measurements still reduce the position errors (Fig. 10a), due to the developing cross-covariance terms in the covariance matrix of the state vector.

Figures 12 and 13 compare the filter covariance to the actual developed errors. As seen, the covariance is consistent. However, in the last segment of the inertial flight (after $t = 100$ s), the covariance development rate does not match the actual rate of the developing inertial navigation errors. After the vision-aided segment, part of the IMU error parameters are estimated by the filter (e.g., b_z) and are used to correct the actual IMU measurements. As a consequence, the actual IMU measurements injected into the navigation system are corrupted by the residual IMU parameters only, resulting in a much slower development of navigation errors. One possible alternative to account for this behavior is to perform a dynamic adjustment of the filter noise covariance matrix Q as a function of the actual covariance values of the estimated IMU states.

VII. Conclusions

This paper presented a method for vision-aided navigation for an airborne platform equipped with an inertial navigation system and a camera with a relatively small field-of-view. The camera was mounted on gimbals so that it could scan the overflown ground regions during flight. The images captured by the camera were used for both constructing a mosaic image and performing motion estimation. Motion estimation was fused with the inertial navigation using an Implicit Extended Kalman filter.

The main idea of the new method was to combine camera scanning and online mosaic construction, which yielded enlarged overlapping areas. Because of the imperfectness of the mosaic construction process, features from the additional overlapping area tend to be of a lower quality compared with those from the original overlapping area. Consequently, the proposed method allowed to obtain improved-precision image-based motion estimation when the original overlapping area between the captured images contains only a small set of high-quality features, which is the case when observing planar low-texture ground regions.

Two types of mosaic images were constructed. The first type is a small mosaic image that was used for motion estimation and constructed in real time. The second type is the main mosaic image, constructed in a background process from all the captured images. This architecture breaks the curse of dimensionality of simultaneous localization and mapping methods, and allows to reduce the computational load in a significant manner.

The proposed method was examined using a statistical simulation study assuming ideal motion estimations, and experiments involving realistic scenarios based on real imagery from Google Earth. These experiments included implementation of camera scanning and mosaic construction. Superior performance was demonstrated compared with traditional two-view methods for motion estimation and navigation aiding for challenging scenarios, such as cameras with narrow field-of-views and low-texture scenes. In particular, it was shown that estimation of position and velocity errors normal to the flight heading, as well as of the roll angle, can be significantly improved.

Appendix A: Development of the Measurement Equation

This appendix presents a development of a measurement model that relates between the image-based estimated camera relative motion $\mathbf{t}_{1\to2}^{C_2}$, $R_{C_1}^{C_2}$ and the developing navigation errors of a standard INS and the parameters that model the IMU errors. The state vector is defined in Eq. (13):

$$\mathbf{X} = [\,\Delta\mathbf{P}^T \quad \Delta\mathbf{V}^T \quad \Delta\mathbf{\Psi}^T \quad \mathbf{d}^T \quad \mathbf{b}^T\,]^T \tag{A1}$$

Under ideal conditions, viz. when there are no navigation errors and $\mathbf{t}_{1\to2}^{C_2}$, $R_{C_1}^{C_2}$ are perfectly estimated, the following can be written:

$$\mathbf{Pos}_{\text{True}}^{L_2}(t_2) - \mathbf{Pos}_{\text{True}}^{L_2}(t_1) = \gamma T_{L_2,\text{True}}^{C_2}\mathbf{t}_{1\to2,\text{True}}^{C_2} \tag{A2a}$$

$$T_{C_1,\text{True}}^{C_2} = R_{C_1,\text{True}}^{C_2} \tag{A2b}$$

where $T_{L_2}^{C_2}$ is the directional cosines matrix (DCM) transforming from C to LLLN at the time instance $t = t_2$; $T_{C_1}^{C_2}$ is the DCM transforming from C at $t = t_2$ to C at $t = t_1$; and $\mathbf{Pos}^{L(t_2)}(t_1)$ is the platform's position at $t = t_1$ expressed in the LLLN system at $t = t_2$, so that $\mathbf{Pos}^{L(t_2)}(t_1) = T_{L(t_2)}^{L(t_1)}\mathbf{Pos}^{L(t_1)}(t_1)$. The subscript $(\cdot)_{\text{True}}$ in Eq. (A2) indicates ideal conditions as defined above.

The DCM $T_{L_2}^{C_2}$ is required since the extracted translation $\mathbf{t}_{1\to2}^{C_2}$ is given in the camera reference frame, while the left side of Eq. (A2a) is expressed in the LLLN system.

I. Translation Measurement Equation

In an ideal situation, with no navigation and image-processing errors, the two sides of Eq. (A2a) constitute parallel vectors. Thus, this equation yields the following constraint:

$$\left[\mathbf{Pos}_{\text{True}}^{L_2}(t_2) - \mathbf{Pos}_{\text{True}}^{L_2}(t_1)\right] \times T_{L_2,\text{True}}^{C_2}\mathbf{t}_{1\to2,\text{True}}^{C_2} = \mathbf{0} \tag{A3}$$

In reality, there are navigation errors that increase with time, moreover, the estimated camera matrix contains errors due to image noise. Thus, Eq. (A3) no longer holds. Denoting by Nav parameters that are taken from the navigation data and by $\hat{\mathbf{t}}_{1\to2}^{C_2}$ the actual estimated translation vector obtained from the image-processing module, Eq. (A3) becomes

$$\left[\mathbf{Pos}_{\text{Nav}}^{L_2}(t_2) - \mathbf{Pos}_{\text{Nav}}^{L_2}(t_1)\right] \times T_{L_2,\text{Nav}}^{C_2}\hat{\mathbf{t}}_{1\to2}^{C_2} = \mathbf{z}_{\text{translation}} \tag{A4}$$

where $\mathbf{z}_{\text{translation}}$ denotes the residual measurement vector.

Taking into account the fact that $\mathbf{Pos}_{\text{Nav}}^{L_2}(.) = \mathbf{Pos}_{\text{True}}^{L_2}(.) + \Delta\mathbf{P}^{L_2}(.)$ and subtracting (A3) from (A4) results in

$$[\Delta\mathbf{P}(t_2) - \Delta\mathbf{P}(t_1)]^{L_2} \times T_{L_2,\text{Nav}}^{C_2}\hat{\mathbf{t}}_{1\to2}^{C_2} + \mathbf{v}_{\text{tr}} = \mathbf{z}_{\text{translation}} \tag{A5}$$

where $\mathbf{v}_{\text{tr}} = [\mathbf{Pos}_{\text{True}}(t_2) - \mathbf{Pos}_{\text{True}}(t_1)]^{L_2} \times \left[T_{L_2,\text{Nav}}^{C_2}\hat{\mathbf{t}}_{1\to2}^{C_2} - T_{L_2,\text{True}}^{C_2}\mathbf{t}_{1\to2,\text{True}}^{C_2}\right]$. The vector $\mathbf{v}_{\text{tr}}$ is due to imperfect translation measurements and navigation errors. One may verify that in ideal conditions this term is nullified.

The inertial position error for a sufficiently small $\Delta t = t_2 - t_1$ or for a straight and level flight is given by [33] (the Nav subscript is omitted for simplicity from here on; thus, all parameters are computed based on the navigation system data, unless otherwise specified):

$$\Delta\mathbf{P}(t_2) = T_{L_2}^{L_1}\Big[-\frac{1}{6}A_s(t_1)T_{L_1}^{B_1}\mathbf{d}\cdot(\Delta t)^3 + \frac{1}{2}[A_s(t_1)\Delta\mathbf{\Psi}(t_1) + T_{L_1}^{B_1}\mathbf{b}](\Delta t)^2 + \Delta\mathbf{V}(t_1)\Delta t + \Delta\mathbf{P}(t_1)\Big] \tag{A6}$$

Note that a transformation matrix, $T_{L_2}^{L_1}$, was added to express the position error at $t = t_2$ in LLLN coordinates.

Substituting Eq. (A6) into Eq. (A5), canceling position errors at $t = t_1$ and denoting $\hat{\mathbf{t}}_{1\to 2}^{L_2} \equiv T_{L_2,\mathrm{Nav}}^{C_2}\hat{\mathbf{t}}_{1\to 2}^{C_2}$ yields

$$\Big[T_{L_2}^{L_1}\Big[-\frac{1}{6}A_s(t_1)T_{L_1}^{B_1}\mathbf{d}\cdot(\Delta t)^3 + \frac{1}{2}[A_s(t_1)\Delta\mathbf{\Psi}(t_1) + T_{L_1}^{B_1}\mathbf{b}](\Delta t)^2 + \Delta\mathbf{V}(t_1)\Delta t\Big]\Big]^{\wedge}\hat{\mathbf{t}}_{1\to 2}^{L_2} + \mathbf{v}_{\mathrm{tr}} = \mathbf{z}_{\mathrm{translation}} \tag{A7}$$

where $(.)^{\wedge}$ denotes the matrix cross-product equivalent. One can see from Eq. (A7) that the translation measurement equation is of the form $\mathbf{z}_{\mathrm{translation}} = H^{\mathrm{tr}}\mathbf{X} + \mathbf{v}_{\mathrm{tr}}$, where

$$H^{\mathrm{tr}} = [\,0_{3\times 3} \quad H_{\Delta V}^{\mathrm{tr}} \quad H_{\Delta\Psi}^{\mathrm{tr}} \quad H_{d}^{\mathrm{tr}} \quad H_{b}^{\mathrm{tr}}\,] \tag{A8}$$

After some algebraic manipulations (cf. Appendix A in [33]), the submatrices of H^{tr} can be rendered into

$$H_{\Delta V}^{\mathrm{tr}} = -[\hat{\mathbf{t}}_{1\to 2}^{L_2}]^{\wedge}T_{L_2}^{L_1}\Delta t$$

$$H_{\Delta\Psi}^{\mathrm{tr}} = -\frac{1}{2}[\hat{\mathbf{t}}_{1\to 2}^{L_2}]^{\wedge}T_{L_2}^{L_1}A_s(t_1)(\Delta t)^2$$

$$H_{d}^{\mathrm{tr}} = \frac{1}{6}[\hat{\mathbf{t}}_{1\to 2}^{L_2}]^{\wedge}T_{L_2}^{L_1}A_s(t_1)T_{L_1}^{B_1}(\Delta t)^3$$

$$H_{b}^{\mathrm{tr}} = -\frac{1}{2}[\hat{\mathbf{t}}_{1\to 2}^{L_2}]^{\wedge}T_{L_2}^{L_1}T_{L_1}^{B_1}(\Delta t)^2$$

II. Rotation Measurement Equation

Recall Eq. (A2b), written under the assumption of ideal conditions: $T_{C_1,\mathrm{True}}^{C_2} = R_{C_1,\mathrm{True}}^{C_2}$. When taking into account navigation errors and errors in the estimated rotation matrix, this equation no longer holds. Instead, we define a residual error angle vector, $\mathbf{z}_{\mathrm{rotation}}$. Under the assumption of small angles, this vector can be written as

$$I - \mathbf{z}_{\mathrm{rotation},A}^{\wedge} = T_{C_1,\mathrm{Nav}}^{C_2}\Big[\hat{R}_{C_1}^{C_2}\Big]^T \tag{A9}$$

Here $T_{C_1,\mathrm{Nav}}^{C_2}$ denotes the DCM transforming from C at $t = t_1$ to C at $t = t_2$, computed by the navigation system of the platform. This matrix differs from the true DCM due to platform navigation errors. The matrix $\hat{R}_{C_1}^{C_2}$ is the estimated rotation matrix. One can verify that under ideal conditions, $T_{C_1,\mathrm{True}}^{C_2} = R_{C_1,\mathrm{True}}^{C_2}$, and thus the rotation error angle $\mathbf{z}_{\mathrm{Rotation}}$ is equal to zero. We omit the subscript (Nav) for simplicity, and write simply $T_{C_1}^{C_2}$.

In general, $T_{C_1}^{C_2}$ can be written as follows:

$$T_{C_1}^{C_2} = T_{C_1}^{B_1}T_{B_1}^{E}T_{E}^{B_2}T_{B_2}^{C_2} \tag{A10}$$

where the matrices $T_{C_1}^{B_1}$ and $T_{B_2}^{C_2}$ are assumed to be known precisely, or at least with much more precision compared with the developing attitude errors. Thus, $T_B^C = T_{B,\mathrm{True}}^C$.

The errors in the ECEF-to-body rotation matrix stem from two sources: position errors and attitude errors. Denote by L_C the correct LLLN system at the platform estimated position, and by L the LLLN system estimated by the navigation system. Thus $T_B^E = T_B^L T_{L_C}^E$, where T_B^L is erroneous due to attitude errors and $T_{L_C}^E$ is erroneous due to position errors. When these errors are not present, $L = L_C = L_{\mathrm{True}}$, where L_{True} is the true LLLN system.

The errors in $T_{L_C}^E$ are assumed to be negligible, since they depend on the position errors, which are small relative to Earth's radius. Thus, $L_C = L_{\mathrm{True}}$ and therefore $T_{L_C}^E = T_{L_{\mathrm{True}}}^E$. However, the attitude errors do not allow a perfect estimation of the DCM transforming from LLLN to B, since the estimated LLLN system is erroneous. Hence, $T_B^L = T_B^{L_C}T_{L_C}^L$.

Assuming small attitude errors, we write $\mathbf{\Psi}_{\mathrm{Nav}} = \mathbf{\Psi}_{\mathrm{True}} + \Delta\mathbf{\Psi}$ to obtain $T_{L_C}^L = I - \Delta\mathbf{\Psi}^{\wedge}$. Taking all the above into consideration, Eq. (A10) turns into

$$T_{C_1}^{C_2} = T_{C_1}^{B_1}T_{B_1}^{L_{C_1}}[I - \Delta\mathbf{\Psi}^{\wedge}(t_1)]T_{L_{C_1}}^{E}T_{E}^{L_{C_2}}[I + \Delta\mathbf{\Psi}^{\wedge}(t_2)]T_{L_{C_2}}^{B_2}T_{B_2}^{C_2} \tag{A11}$$

For a sufficiently small $t - t_0$ or for a straight and level flight, one can use the approximation (cf. Appendix A in [33]) $\Delta\mathbf{\Psi}(t_2) = -T_{L_1}^{B_1}\mathbf{d}\Delta t + \Delta\mathbf{\Psi}(t_1)$. Substituting this into Eq. (A11), ignoring second-order terms and carrying out some additional algebraic manipulations we get

$$T_{C_1}^{C_2} = T_{C_1}^{B_1}T_{B_1}^{L_{C_1}}T_{L_{C_1}}^{E}T_{E}^{L_{C_2}}T_{L_{C_2}}^{B_2}T_{B_2}^{C_2} + T_{C_1}^{B_1}T_{B_1}^{L_{C_1}}\Big[T_{L_{C_1}}^{E}T_{E}^{L_{C_2}}\Big(\Delta\mathbf{\Psi}(t_1) - T_{L_1}^{B_1}\mathbf{d}\Delta t\Big)^{\wedge} - \Delta\mathbf{\Psi}^{\wedge}(t_1)T_{L_{C_1}}^{E}T_{E}^{L_{C_2}}\Big]T_{L_{C_2}}^{B_2}T_{B_2}^{C_2} \tag{A12}$$

As was mentioned before, the rotation matrix that was estimated by the image-processing module differs from the true matrix. Let $T_{R_{\mathrm{Err}}}$ be the DCM transforming between the true rotation matrix and the estimated one: $\hat{R}_{C_1}^{C_2} = T_{R_{\mathrm{Err}}}R_{C_1,\mathrm{True}}^{C_2}$.

Multiplying Eq. (A12) by $[\hat{R}_{C_1}^{C_2}]^T$ from the right and noting that $T_{C_1}^{B_1}T_{B_1}^{L_{C_1}}T_{L_{C_1}}^{E}T_{E}^{L_{C_2}}T_{L_{C_2}}^{B_2}T_{B_2}^{C_2}$ is the nominal value of $T_{C_1,\mathrm{True}}^{C_2}$ and hence also of $R_{C_1,\mathrm{True}}^{C_2}$ yields

$$T_{C_1}^{C_2}\Big[\hat{R}_{C_1}^{C_2}\Big]^T = \Big\{I + T_{C_1}^{B_1}T_{B_1}^{L_{C_1}}\Big[T_{L_{C_1}}^{E}T_{E}^{L_{C_2}}\Big(\Delta\mathbf{\Psi}(t_1) - T_{L_1}^{B_1}\mathbf{d}\Delta t\Big)^{\wedge} - \Delta\mathbf{\Psi}^{\wedge}(t_1)T_{L_{C_1}}^{E}T_{E}^{L_{C_2}}\Big]T_{L_{C_2}}^{B_2}T_{B_2}^{C_2}R_{C_2,\mathrm{True}}^{C_1}\Big\}T_{R_{\mathrm{err}}}^T \tag{A13}$$

Assuming small estimation rotation errors $\mathbf{v}_{\mathrm{rot}}$, one can write $T_{R_{\mathrm{err}}}^T = I - \mathbf{v}_{\mathrm{rot}}^{\wedge}$. Thus, substituting Eq. (A13) into Eq. (A9) yields

$$z_{\mathrm{rotation}}^{\wedge} = \mathbf{v}_{\mathrm{rot}}^{\wedge} + T_{C_1}^{B_1}T_{B_1}^{L_{C_1}}\Big[-T_{L_{C_1}}^{E}T_{E}^{L_{C_2}}\Big(\Delta\mathbf{\Psi}(t_1) - T_{L_1}^{B_1}\mathbf{d}\Delta t\Big)^{\wedge} + \Delta\mathbf{\Psi}^{\wedge}(t_1)T_{L_{C_1}}^{E}T_{E}^{L_{C_2}}\Big]T_{L_{C_2}}^{B_2}T_{B_2}^{C_2}\Big[\hat{R}_{C_1}^{C_2}\Big]^T \tag{A14}$$

Using Eq. (A10), one can write the following two relations:

$$T_{C_1}^{B_1}T_{B_1}^{L_{C_1}}T_{L_{C_1}}^{E}T_{E}^{L_{C_2}} = \hat{R}_{C_1}^{C_2}T_{C_2}^{B_2}T_{B_2}^{L_{C_2}}$$
$$T_{C_1}^{B_1}T_{B_1}^{L_{C_1}} = \hat{R}_{C_1}^{C_2}T_{C_2}^{B_2}T_{B_2}^{L_{C_2}}T_{L_{C_2}}^{E}T_{E}^{L_{C_1}} \tag{A15a}$$

$$T_{C_1}^{B_1}T_{B_1}^{L_{C_1}} = \hat{R}_{C_1}^{C_2}T_{C_2}^{B_2}T_{B_2}^{L_{C_2}}T_{L_{C_2}}^{E}T_{E}^{L_{C_1}} \tag{A15b}$$

Substituting Eqs. (A15a) and (A15b) into Eq. (A14) and using the fact that for any matrix Λ and any vector $\boldsymbol{\xi}$, $\Lambda\boldsymbol{\xi}^{\wedge}\Lambda^T = (\Lambda\boldsymbol{\xi})^{\wedge}$, Eq. (A14) transforms into

$$z_{\mathrm{rotation}} = \hat{R}_{C_1}^{C_2}T_{C_2}^{B_2}T_{B_2}^{L_{C_2}}\Big(T_{L_{C_2}}^{E}T_{E}^{L_{C_1}} - I\Big)\Delta\mathbf{\Psi}(t_1) + \hat{R}_{C_1}^{C_2}T_{C_2}^{B_2}T_{B_2}^{L_{C_2}}T_{L_1}^{B_1}\mathbf{d}\Delta t + \mathbf{v}_{\mathrm{rot}} \tag{A16}$$

One can see that Eq. (A16) is of the form $\mathbf{z}_{\mathrm{rotation}} = H^{\mathrm{rot}}\mathbf{X} + \mathbf{v}_{\mathrm{rot}}$, where

$$H^{\mathrm{rot}} = [\,0_{3\times 3} \quad 0_{3\times 3} \quad H_{\Delta\Psi}^{\mathrm{rot}} \quad H_{d}^{\mathrm{rot}} \quad 0_{3\times 3}\,]$$
$$H_{\Delta\Psi}^{\mathrm{rot}} = \hat{R}_{C_1}^{C_2}T_{C_2}^{B_2}T_{B_2}^{L_{C_2}}\Big(T_{L_{C_2}}^{E}T_{E}^{L_{C_1}} - I\Big)$$
$$H_{d}^{\mathrm{rot}} = \hat{R}_{C_1}^{C_2}T_{C_2}^{B_2}T_{B_2}^{L_{C_2}}T_{L_1}^{B_1}\Delta t \tag{A17}$$

Appendix B: Google Earth Interface

Given a platform trajectory and measurement settings (such as measurement frequency), a command is sent to Google Earth throughout the interface to display a region at a specified position (latitude, longitude, and altitude) and inertial orientation. These are computed based on the current platform position, attitude, and camera angles.

Because the current version of Google Earth allows changing only the camera heading and tilt angles, special care was taken to allow roll motion in Google Earth, which is required for implementing the camera scanning procedure. This was achieved by shifting the yaw angle by 90° relative to the flight heading angle and adjusting the camera system accordingly. As a result, camera/platform roll motion is obtained through tilt motion (handled automatically by the interface).

In the current implementation, the image acquisition through Google Earth is performed offline, i.e., this command is sent according to the measurement's frequency and the acquired images are saved into some repository. The images are injected into the image-processing module in the simulation at appropriate instants. Examples of images acquired from Google Earth are given throughout the paper (e.g., Fig. 4). See also Appendix A in [40].

References

[1] Merhav, S., and Bresler, Y., "On-Line Vehicle Motion Estimation from Visual Terrain Information Part 1: Recursive Image Registration," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 22, No. 5, 1986, pp. 583–587.
doi:10.1109/TAES.1986.310725

[2] Gurfil, P., and Rotstein, H., "Partial Aircraft State Estimation from Visual Motion Using the Subspace Constraints Approach," *Journal of Guidance, Control and Dynamics*, Vol. 24, No. 5, July 2001, pp. 1016–1028.
doi:10.2514/2.4811

[3] Soatto, S., and Perona, P., "Recursive 3-D Visual Motion Estimation Using Subspace Constraints," *International Journal of Computer Vision*, Vol. 22, No. 3, 1997, pp. 235–259.
doi:10.1023/A:1007930700152

[4] Diel, D., DeBitetto, P., and Teller, S., "Epipolar Constraints for Vision-Aided Inertial Navigation," *Proceedings of the IEEE Workshop on Motion and Video Computing*, Vol. 2, Jan. 2005, pp. 221–228.
doi:10.1109/ACVMOT.2005.48

[5] Roumeliotis, S., Johnson, A., and Montgomery, J., "Augmenting Inertial Navigation with Image-Based Motion Estimation," *IEEE International Conference on Robotics and Automation*, Vol. 4, 2002, pp. 4326–4333.
doi:doi: 10.1109/ROBOT.2002.1014441

[6] Merhav, S., and Bresler, Y., "On-Line Vehicle Motion Estimation from Visual Terrain Information Part 2: Ground Velocity and Position Estimation," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 22, No. 5, 1986, pp. 588–604.
doi:10.1109/TAES.1986.310726

[7] Lerner, R., Rivlin, E., and Rotstein, H., "Pose and Motion Recovery from Feature Correspondences and a Digital Terrain Map," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 28, No. 9, Sept. 2006, pp. 1404–1417.
doi:10.1109/TPAMI.2006.192

[8] Sim, D., Park, R., Kim, R., Lee, S., and Kim, I., "Integrated Position Estimation Using Aerial Image Sequences," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 24, No. 1, Jan. 2002, pp. 1–18.
doi:10.1109/34.982881

[9] Gracias, N., Zwaan, S., Bernardino, A., and Santos-Victor, J., "Results on Underwater Mosaic-Based Navigation," *Oceans*, Vol. 3, 2002, pp. 1588–1594.
doi:10.1109/OCEANS.2002.1191872

[10] Gracias, N., and Santos-Victor, J., "Underwater Video Mosaics as Visual Navigation Maps," *Computer Vision and Image Understanding*, Vol. 79, No. 1, 2000, pp. 66–91.
doi:10.1006/cviu.2000.0848

[11] Garcia, R., "A Proposal to Estimate the Motion of an Underwater Vehicle Through Visual Mosaicking," Ph.D. Thesis, Univ. of Girona, Spain, 2002.

[12] Garcia, R., Puig, J., Ridao, P., and Cufi, X., "Augmented State Kalman Filtering for AUV Navigation," *IEEE International Conference on Robotics and Automation*, Vol. 4, 2002, pp. 4010–4015.
doi:10.1109/ROBOT.2002.1014362

[13] Davison, A. J., Reid, I. D., Molton, N. D., and Stasse, O., "MonoSLAM: Real-Time Single Camera SLAM," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 29, No. 6, 2007, pp. 1052–1067.
doi:10.1109/TPAMI.2007.1049

[14] Bryson, M., and Sukkarieh, S., "Active Airborne Localisation and Exploration in Unknown Environments using Inertial SLAM," *IEEE Aerospace Conference*, IEEE Publications, Piscataway, NJ, 2006.

[15] Bryson, M., and Sukkarieh, S., "Bearing-Only SLAM for an Airborne Vehicle," *Australasian Conference on Robotics and Automation*, ACRA, Australia, 2005.

[16] Kim, J., and Sukkarieh, S., "6DoF SLAM aided GNSS/INS Navigation in GNSS Denied and Unknown Environments," *Journal of Global Positioning Systems*, Vol. 4, Nos. 1–2, 2005, pp. 120–128.

[17] Thrun, S., Liu, Y., Koller, D., Ng, A. Y., Ghahramani, Z., and Durrant-Whyte, H., "Simultaneous Localization and Mapping with Sparse Extended Information Filters," *The International Journal of Robotics Research*, Vol. 23, Nos. 7–8, 2004, pp. 693–716.
doi:10.1177/0278364904045479

[18] George, M., and Sukkarieh, S., "Inertial Navigation Aided by Monocular Camera Observations of Unknown Features," *IEEE International Conference on Robotics and Automation*, IEEE Publications, Piscataway, NJ, April 2007, pp. 3558–3564.

[19] Szeliski, R., "Image Alignment and Stitching: A Tutorial," Microsoft Research Tech. Rept. MSR-TR-2004-92, 2005.

[20] Fleischer, S., Wang, H., Rock, S., and Lee, M., "Video Mosaicking Along Arbitrary Vehicle Paths," *Proceedings of the 1996 Symposium on Autonomous Underwater Vehicle Technology*, IEEE Publications, Piscataway, NJ, 1996, pp. 293–299.

[21] Gracias, N., Costeira, J., and Santos-Victor, J., "Linear Global Mosaic for Underwater Surveying," *Symposium on Intelligent Autonomous Vehicles*, IEEE Publications, Piscataway, NJ, 2004.

[22] Shum, H., and Szeliski, R., "Systems and Experiment Paper: Construction of Panoramic Image Mosaics with Global and Local Alignment," *International Journal of Computer Vision*, Vol. 36, No. 2, 2000, pp. 101–130.
doi:10.1023/A:1008195814169

[23] Richmond, K., and Rock, S., "An Operational Real-Time Large-Scale Visual Mosaicking and Navigation System," *Oceans*, Sept. 2006, pp. 1–6.
doi:10.1109/OCEANS.2006.306897

[24] Ferrer, J., Elibol, A., Delaunoy, O., Gracias, N., and Garcia, R., "Large-Area Photo-Mosaics Using Global Alignment and Navigation Data," *Oceans : [record]*, Sept. 2007, pp. 1–9.
doi:10.1109/OCEANS.2007.4449367

[25] Zhang, P., Milios, E. E., and Gu, J., "Graph-based Automatic Consistent Image Mosaicking," *IEEE International Conference on Robotics and Biomimetics*, IEEE Publications, Piscataway, NJ, 2004, pp. 558–563.

[26] Hartley, R., and Zisserman, A., *Multiple View Geometry*, Cambridge Univ. Press, Cambridge, MA, 2000.

[27] Eustice, R. M., Pizarro, O., and Singth, H., "Visually Augmented Navigation for Autonomous Underwater Vehicles," *IEEE Journal of Oceanic Engineering*, Vol. 33, No. 2, 2008, pp. 103–122.
doi:10.1109/JOE.2008.923547

[28] Tsai, R., Huang, T., and Zhu, W., "Estimating Three-Dimensional Motion Parameters of a Rigid Planar Patch, II: Singular Value Decomposition," *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol. 30, No. 4, Aug. 1982, pp. 525–534.

[29] Caballero, F., Merino, L., Ferruz, J., and Ollero, A., "Improving Vision-based Planar Motion Estimation for Unmanned Aerial Vehicles through Online Mosaicing," *IEEE International Conference on Robotics and Automation*, IEEE Publications, Piscataway, NJ, May 2006, pp. 2860–2865.

[30] Caballero, F., Merino, L., Ferruz, J., and Ollero, A., "Vision-Based Odometry and SLAM for Medium and High Altitude UAVs," *Journal of Intelligent and Robotic Systems: Theory and Applications*, Vol. 54, Nos. 1–3, March 2009, pp. 137–161.
doi:10.1007/s10846-008-9257-y

[31] Mourikis, A., and Roumeliotis, I., "A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation," *International Conference on Robotics and Automation*, IEEE Publications, Piscataway, NJ, April 2007, pp. 3565–3572.

[32] Shakernia, O., Vidal, R., Sharp, C., Ma, Y., and Sastry, S., "Multiple View Motion Estimation and Control for Landing an Unmanned

Aerial Vehicle," *IEEE International Conference on Robotics and Automation*, IEEE Publications, Piscataway, NJ, May 2002, pp. 2793–2798.

[33] Indelman, V., Gurfil, P., Rivlin, E., and Rotstein, H., "Real-Time Mosaic-Aided Aircraft Navigation: II. Sensor Fusion," *AIAA Guidance, Navigation and Control Conference*, AIAA, Reston, VA, 2009.

[34] Caballero, F., Merino, L., Ferruz, J., and Ollero, A., "Homography Based Kalman Filter for Mosaic Building. Applications to UAV Position Estimation," *IEEE International Conference on Robotics and Automation*, IEEE Publications, Piscataway, NJ, April 2007, pp. 2004–2009.

[35] Lowe, D., "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, Vol. 60, No. 2, Nov. 2004, pp. 91–110.
doi:10.1023/B:VISI.0000029664.99615.94

[36] Fischler, M., and Bolles, R., "Random Sample Consensus: a Paradigm for Model Fitting with Application to Image Analysis and Automated Cartography," *Communications of the Association for Computing Machinery*, Vol. 24, 1981, pp. 381–395.

[37] Negahdaripour, S., and Xu, X., "Mosaic-Based Positioning and Improved Motion-Estimation Methods for Automatic Navigation of Submersible Vehicles," *IEEE Journal of Oceanic Engineering*, Vol. 27, No. 1, Jan. 2002, pp. 79–99.
doi:10.1109/48.989892

[38] Soatto, S., Frezza, R., and Perona, P., "Motion Estimation via Dynamic Vision," *IEEE Transactions on Automatic Control*, Vol. 41, No. 3, March 1996, pp. 393–413.
doi:10.1109/9.486640

[39] Dissanayake, G., Sukkarieh, S., Nebot, E., and Durrant-Whyte, H., "The Aiding of a Low-Cost Strapdown Inertial Measurement Unit Using Vehicle Model Constraints for Land Vehicle Applications," *IEEE Transactions on Robotics and Automotation*, Vol. 17, No. 5, Oct. 2001, pp. 731–747.
doi:10.1109/70.964672

[40] Indelman, V., Gurfil, P., Rivlin, E., and Rotstein, H., "Real-Time Mosaic-Aided Aircraft Navigation: I. Motion Estimation," *AIAA Guidance, Navigation and Control Conference*, AIAA, Reston, VA, 2009.