Real-Time Mosaic-Aided Aerial Navigation: I. Motion Estimation

Vadim Indelman [*] Pini Gurfil [†] Ehud Rivlin [‡]

Technion - Israel Institute of Technology Haifa 32000, Israel

Hector Rotstein [§]

RAFAEL - Advanced Defense Systems Limited

This work presents a method for real-time mosaic-aided aircraft navigation. The method utilizes an on-line mosaic image construction process based on images acquired by a gimballed camera attached to an airborne platform, which scans ground regions in the vicinity of the flight trajectory. The images captured by the camera are used to update the mosaic image while simultaneously estimating the platform's motion. The work is divided into two parts: The current paper addresses mosaic-based motion estimation using a scanning camera and a concomitant mosaic construction procedure. Part II focuses on fusion of the motion estimation with a standard navigation system. An extensive performance evaluation of the proposed method was carried out, involving real imagery and an implementation of the camera scanning and mosaic image construction processes. The current paper, concerned with the motion estimation procedure, shows a significant improvement of motion estimation in challenging scenarios, such as using a narrow field-of-view camera and tracking low-texture scenes. The approach proposed in this work is an alternative to Simultaneous Localization and Mapping (SLAM) in the following sense: We assume that the mosaic image construction is an independent process, to be utilized for improving the navigation system. Therefore, the proposed architecture alleviates some of the computational load associated with SLAM.

[*]Doctoral Student, Faculty of Aerospace Engineering. email: ivadim@tx.technion.ac.il

[†]Senior Lecturer, Faculty of Aerospace Engineering, Associate Fellow AIAA. email: pgurfil@technion.ac.il

[‡]Professor, Department of Computer Science. email: ehudr@cs.technion.ac.il

[§]Chief Control Systems Engineer. email: hector@rafael.co.il

I. Introduction

Over the last few decades, considerable research efforts have been directed towards 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. One of the prevalent approaches for navigation aiding is using vision-based sensors such as on-board cameras.

The current work describes a method for vision-aided navigation of an airborne platform utilizing information from an on-line construction of a mosaic of images. We assume that the airborne platform is equipped with a standard inertial navigation system and a gimballed camera. During the flight, the camera scans ground regions in the vicinity of the flight trajectory. The captured images are processed on-line to form a mosaic, and are simultaneously used for motion estimation. The mosaic image construction process is accomplished by augmenting each image with a previously-constructed mosaic. The proposed camera scanning pattern and the resulting mosaic construction process yield an increased overlap between each new image and the preceding mosaic, which is key for improving motion estimation. The presentation of the proposed method is divided into two parts: The current paper focuses on mosaic-based motion estimation, while Part II[1] discusses fusion of motion estimation with a standard navigation system.

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 multi-view-geometry-based methods.

The two-view-geometry-based methods include Ref. 4, in which the velocity-to-height ratio is estimated by real-time processing of visual cues captured by a stabilized camera. Ref. 5 estimates the epipolar geometry between two given views, and expresses it in terms of the fundamental matrix. This matrix is then used for extracting the motion parameters.[6] Ref. 7 utilized epipolar constraints between tracked features of consecutive images. When the observed scene is assumed planar, motion parameters can be calculated by estimating the homography matrix.[8–10]

The multi-view-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 Ref. 11, 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. Ref. 12 proposed using multiple-view geometry for estimating the motion of an unmanned helicopter during its landing phase. However, assuming that a mutual overlapping region among several images exists, may be invalid in many airborne applications. Violating this assumption usually degenerates the multi-view 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 larger ground regions, which endows the images with richer details, thereby allowing better motion estimation. This is crucial when flying over regions that produce low-texture images. However, many airborne platforms use narrow-FOV cameras; moreover, the typical trajectories and maneuvers of these platforms render multi-view geometry methods useless. Ground footprints of narrow-FOV cameras can often be considered planar, resulting in a homography relation between the captured images, due to the fact that epipolar geometry is invalid in planar scenes.[6] The current work focuses on motion estimation using narrow-FOV cameras. To that end, a method for enlarging the overlapping regions between images, based on a camera scanning pattern and on real-time mosaic construction, is developed.

The mosaic construction may be performed using various algorithms.[13–25] A basic mosaic construction process is performed according to the following steps: i) Motion model - establishment of a mathematical relationships that map pixel coordinates from one image to another (e. g. pure translation, rotation+translation, affine transformation, perspective transformation); ii) Motion model state estimation - the most common approaches are *direct methods* and *feature-based methods*. In the first approach, the images are shifted one relative to another until a minimum of a predefined cost function is achieved. The second approach involves feature extraction from each image and subsequent matching of these features to establish the transformation between the images (i. e. motion parameters). Robust estimation of the motion parameters is performed using one of the available random sampling algorithms (e. g. RANSAC[26] and LMedS[27]); iii) Image blending - integration of the images into a single image according to the estimated motion model and by following one of the available blending techniques.[14,28]

In many applications, a mosaic of the operational environment might not be available *a-priori*, so the mosaic must be constructed during the flight. Thus, it is required to construct a representation of the mission environment (e. g. construct a mosaic) and also utilize this process for enhancing the performance of an existing navigation system. This approach is known as *Simultaneous Localization and Mapping* (SLAM).[29–35] The general approach for solving the SLAM problem is to augment the platform's state vector with parameters that are used to describe the observed environment (e. g. features locations). When processing a measurement, the augmented state vector yields an update both in the platform states (e. g.

position, velocity) and in the environment model. However, the SLAM framework has several disadvantages: The most conspicuous is computational load, which does not allow real-time performance once the augmented state has reached some critical size. Another difficulty is feature initialization, which requires either metric information (e. g. an altimeter[32]) or at least two significantly distinct camera positions,[34] which constrains the platform trajectory.

Since there are various applications in which the aerial mosaic construction is the main goal, in this work we assume that the mosaic image construction is an *independent* process, to be utilized for improving the performance of an existing navigation system. Thus, our architecture alleviates the computational load required by SLAM, as the state vector is not augmented by the environment model representation. We propose to exploit the camera scanning procedure and the mosaic construction process in order to improve vision-based motion estimation, focusing on difficult scenarios of narrow-FOV cameras and low-texture scenes, which was not discussed in previous studies.[10] In addition, we design a method for utilizing the mosaic-based motion estimation for reducing the accumulating inertial navigation errors of airborne platforms, thus estimating several platform states.

The rest of this paper is organized as follows: Section II overviews the main components of the proposed method; Section III elaborates upon the camera scan and real-time mosaic construction; Section IV develops motion estimation techniques and provides experimental results for mosaic-based motion estimation; and Section V concludes the discussion. The second part of this work[1] discusses fusion of the mosaic-based motion estimation with a standard navigation system.

II. Method Overview

Figure 1 shows the main components of the proposed architecture. The specific system assumed in this work is an airborne platform equipped with a gimballed camera and a standard inertial navigation system (INS).

The INS consists of an inertial measurement unit (IMU) and a strapdown block. The latter processes measurements from the IMU and produces a navigation solution, which is comprised of platform position, velocity and attitude. Due to the imperfectness of the IMU sensors, the computed navigation parameters develop errors over time.

During the flight, an on-board camera captures images of ground regions according to a scanning procedure. The acquired images are directed to the image processing module, where they are used to construct the mosaic image, while simultaneously using some of the images for relative motion estimation.

The motion estimation is reformulated into measurements, which are then injected into the Kalman filter in order to update the navigation system and thereby arrest the development of inertial navigation errors. This is the main focus of Part II of this work.[1]

Throughout this paper, the following coordinate systems are used:

- E - Earth-fixed inertial reference frame, also known as an Earth-centered, Earth-fixed (ECEF) coordinate system. Its origin is set at an arbitrary point on the Earth's surface at sea level. X_E points north, Y_E points east and Z_E completes the setup to yield a Cartesian right hand system.

- L - Local-level, local-north (LLLN) reference frame, also known as a north-east-down (NED) coordinate system. Its origin is set at the platform's center-of-mass. X_L points north, Y_L points east and Z_L completes the setup to yield a Cartesian right hand system.

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

- 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 Figure 2(b).

The camera is rigidly attached to the platform and performs pan and tilt movements, which are denoted by ψ_C and θ_C, respectively.

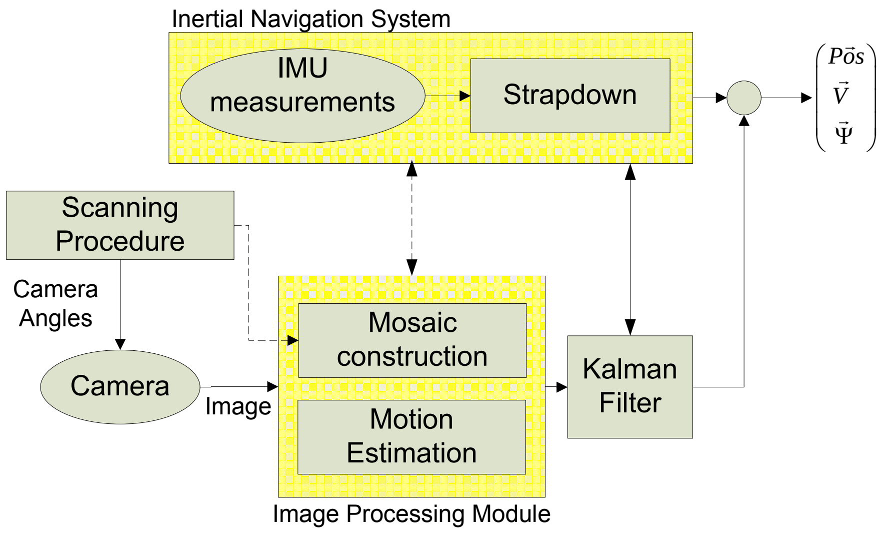


Figure 1. Overview of the system concept.

III. Camera Scanning Procedure and Mosaic Construction Method

In this section, a detailed presentation of the camera scanning and mosaic construction procedures is given. Experimental results of mosaic-aided navigation, which involve implementation of these procedures on real imagery are given in Part II of this work.[1]

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 feature on the ground, or a scanning procedure. This section elaborates upon the scanning procedure that was implemented in this work using real imagery. Each new captured image is processed and used to update the mosaic image of the flight area. A detailed discussion of the on-line mosaic construction appears in §III-B.

Figure 2(a) shows a schematic illustration of the implemented scan procedure. 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. However, the additional overlapping region, provided by the camera scanning procedure, enables enhancement of motion estimation, as will be seen in Section III-B. The proposed scan pattern also allows implementation of improved mosaic construction methods.

We assume that the scanning procedure modifies the camera's pan angle, ψ_c, while keeping the camera tilt angle constant, as shown in Figure 2(b). 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 (LOS) vector for the next camera aiming point in the body-fixed reference frame, $\hat{\mathbf{r}}^B$, is determined according to

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

where $T_B^C(\psi_c)$ is the directional cosines matrix (DCM) transforming from 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 $CCD_{\mathbf{Y}_C}$ is the size of the camera charged coupled device (CCD) 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

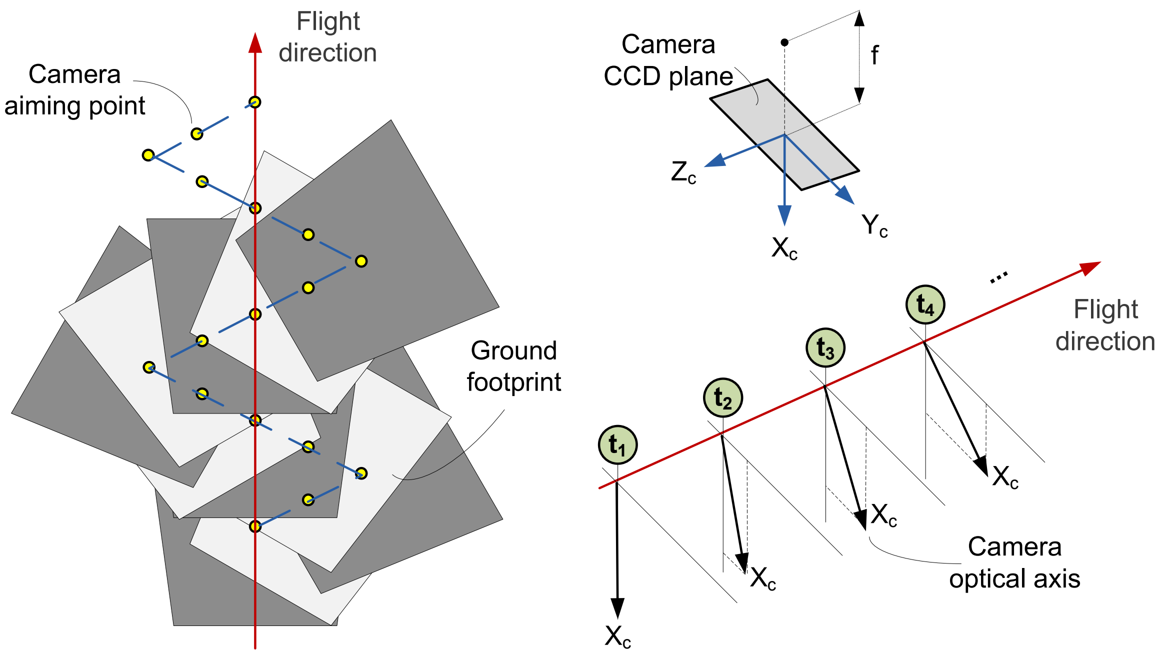


Figure 2. (a) A schematic illustration of the scanning procedure. (b) Definition of camera coordinate system and a schematic illustration of camera angles during the scan procedure.

from B to C can be written as

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

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

$$\hat{\mathbf{r}}^B = T_B^C(\psi_c)\begin{bmatrix}1 & 0 & 0\end{bmatrix}^T = \begin{bmatrix}0 & \sin\psi_c & \cos\psi_c\end{bmatrix}^T \tag{3}$$

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 to -1 once the camera pan angle, ψ_c, reaches a certain pre-specified limit.

The proposed scan methodology moves the camera in a direction perpendicular to the flight trajectory. We implicitly assume that the velocity-over-height ratio and the camera sampling frequency provide sufficient overlapping regions between each two adjacent images along the flight direction. The proposed scanning methodology differs the one suggested in Ref. 3, since in the proposed method there is no need for any external information, whereas in Ref. 3 it was assumed that the altitude was given. An example of real mosaic images, constructed based on images acquired during the camera scanning procedure, is given in Figure 4.

B. Mosaic Construction Method

Mosaic construction has several merits. It is capable of showing the whole flight region in a single image, a feature that constitutes an important aid to surveillance, communication and mission operation. Moreover, as will be shown herein, the mosaic image and its construction process can be utilized for enhancing the precision of image-based motion estimation. We shall now present a detailed description of the mosaic construction method using the scanning method described in Section III-A.

1. *Homography Matrix Estimation*

Given a newly-captured image and the current mosaic, the first step in the mosaic construction process is finding how one image is aligned relative to the other. This may be achieved by estimating the homography matrix between the two images, assuming that the observed scene is planar. The homography relation is also valid for a three-dimensional scene if the camera performs a pure rotation. However, this is not typical to fixed-wing aerial platforms.

The homography matrix is used for updating the mosaic image given a new image, and also to perform relative motion estimation between these images, as will be explained in the sequel. To define the homography matrix, we first recall the definition of *homogenous coordinates*: A homogeneous representation of a point $(x, y) \in \mathbb{R}^2$ is the vector $\mathbf{x} = [x_1, x_2, x_3]^T \in \mathbb{R}^3$, which is defined up to scale. The homogeneous point (x_1, x_2, x_3) represents the point $(x_1/x_3, x_2/x_3) \in \mathbb{R}^2$. In particular, the homogeneous point $(x, y, 1)$ represents the point (x, y). Homogeneous points with $x_3 = 0$ represent points which lie on a plane at infinity.[6]

Given some point $\mathbf{x}$ in the first image and a matching point $\mathbf{x}'$ in the second image, both expressed in homogeneous coordinates, the following constraint can be written for the homography matrix, H:

$$\mathbf{x}'_i \cong H\mathbf{x}_i \tag{5}$$

Explicitly,

$$\begin{bmatrix} x'_1 \\ x'_2 \\ x'_3 \end{bmatrix} \cong \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{6}$$

where $\cong$ denotes equality up to scale. The entries of H are related to the observed scene-plane parameters and to the translational and rotational motion.[8] Assuming that $\mathbf{x} = [x, y, 1]^T$, the second image coordinates (x', y') may be computed based on the inhomogeneous form of Eq. (6) as follows:[6]

$$x' = \frac{x'_1}{x'_3} = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}} \tag{7}$$

$$y' = \frac{x'_2}{x'_3} = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}} \tag{8}$$

There are various methods for estimating the homography matrix given two partially overlapping images. The method used herein relies on Ref. 6 and is summarized in Algorithm 1: First, Scale Invariant Feature Transform (SIFT)[36] 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. The motivation to use SIFT and not the Harris detector with correlation correspondence stems from the fact that the camera may perform unconstrained relative motion between two images, especially in a scan scenario. This is better handled by SIFT (a brief description of SIFT is given in Appendix B).

Next, the features from the two images are matched based on the minimum Euclidean distance of their descriptor vectors, which yields a set of matched points, $\mathcal{S}$. Features with duplicate image coordinates in this set are removed. Each SIFT feature is represented by 4 parameters $\{x, y, \sigma, \theta\}$, where (x, y) are the image coordinates of the feature, and (σ, θ) are the feature scale and orientation values, respectively, computed by the SIFT algorithm (cf. Appendix B). After the matching has been performed, the features image coordinates are the only parameters calculated in upcoming computations, and thus features with the same image coordinates and with other varying parameters, e.g. $\{x, y, \sigma_1, \theta_1\}$, $\{x, y, \sigma_2, \theta_2\}$, are in fact the same representation of a single feature, and are therefore discarded.

Algorithm 1 Homography Estimation Algorithm

```
1: Extract features from both images using the SIFT algorithm:$\{\mathbf{x}_i\}_{i=1}^{N}$ and $\{\mathbf{x}'_i\}_{i=1}^{N'}$.
2: Match features from both images based on their descriptor vectors, to produce the set
   $\mathcal{S} = \{(\mathbf{x}_i, \mathbf{x}'_i)\}_{i=1}^{N_\mathcal{S}}$.
3: Apply Algorithm 2 for robust rejection of outliers in $\mathcal{S}$, yielding a set $\mathcal{R} = \{(\mathbf{x}_i, \mathbf{x}'_i)\}_{i=1}^{N_\mathcal{R}}$
   of inliers matched points.
4: Perform homography least-squares (LS) estimation based on the matched points set $\mathcal{R}$.
```

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,[26] which is summarized in Algorithm 2 for the case of homography matrix estimation.[6]

First, four feature matches are drawn from the available features set $\mathcal{S}$, based on which the homography matrix, H, is calculated. This homography matrix is then used to choose a subset of feature matches $\mathcal{T} \subseteq \mathcal{S}$ that lie within a predefined threshold, $t_{threshold}$. More specifically, a pair of point correspondences $(\mathbf{x}, \mathbf{x}')$ is chosen if

$$d_E(\mathbf{x}, H^{-1}\mathbf{x}')^2 + d_E(\mathbf{x}', H\mathbf{x})^2 < t^2_{threshold} \tag{9}$$

where $d_E(.,.)$ is the Euclidean distance between two points in the same image. The number of iterations, M, should be high enough to guarantee with some probability p that at least one of the subsets $\{\mathcal{T}_i\}_{i=1}^{M}$ is free from outliers (usually $p = 0.99$). Let w be the probability that any chosen feature match is an inlier. Taking into account the fact that in each iteration 4 feature matches are drawn, the following equation may be written:

$$\left(1 - w^4\right)^M = 1 - p \tag{10}$$

Let $\epsilon = 1 - w$, i. e. ϵ is the probability that any chosen feature match is an outlier. Substituting ϵ into the above equation and performing some algebraic operations yields an expression for M:

$$M = \frac{\log{(1-p)}}{\log{\left(1 - (1-\epsilon)^4\right)}} \tag{11}$$

Since ϵ is unknown, it is evaluated at each iteration based on the believed number of inliers, which is the cardinality the subset $\mathcal{T}$, and the overall number of matched points (the cardinality of the set $\mathcal{S}$). Let the respective number of matched points in $\mathcal{T}$ and $\mathcal{S}$ be $L_{\mathcal{T}}$ and $L_{\mathcal{S}}$. Thus, ϵ is calculated at each iteration according to

$$\epsilon = 1 - \frac{L_{\mathcal{T}}}{L_{\mathcal{S}}} \tag{12}$$

and is then used to update the parameter M based on Eq. (11).

After M iterations, the subset with the maximum number of point matches is chosen among $\{\mathcal{T}_i\}_{i=1}^{M}$. This subset, denoted by $\mathcal{R}$, contains the set of point matches that were identified by the RANSAC algorithm as inliers.

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. Assuming $\mathbf{x}'_i = (x'_i, y'_i, w'_i)$ and $\mathbf{x}_i$ are a pair of matched points in $\mathcal{R}$, and denoting by $\mathbf{h}^{jT}$ the j-th row of the homography

Algorithm 2 RANSAC Algorithm for Homography Estimation

```
1: Initialization: $M = \infty$, $Counter = 0$.
2: while $M > Counter$ do
3:     $\mathcal{T} = \phi$.
4:     Draw 4 feature matches from $\mathcal{S}$ and compute the homography matrix $H$ based on
       these matches.
5:     Construct a subset $\mathcal{T}$ of feature matches which are consistent with $H$: Each feature
       match, $(\mathbf{x}, \mathbf{x}') \in \mathcal{S}$, for which Eq. (9) holds is added to $\mathcal{T}$.
6:     Calculate $\epsilon$ according to Eq. (12).
7:     Update $M$ based on the calculated $\epsilon$ according to Eq. (11). Assume $p = 0.99$.
8:     $Counter \leftarrow Counter + 1$.
9: end while
10: Choose a subset $\mathcal{R} \in \{\mathcal{T}_i\}_{i=1}^{M}$ which has a maximum number of feature matches among
    $\{\mathcal{T}_i\}_{i=1}^{M}$.
```

matrix H, Eq. (5) yields the following two independent linear equations:[6]

$$A_i \mathbf{h} = \mathbf{0} \tag{13}$$

where

$$A_i = \begin{bmatrix} \mathbf{0}^T & -w_i' \mathbf{x}_i^T & y_i' \mathbf{x}_i^T \\ w_i' \mathbf{x}_i^T & \mathbf{0}^T & -x_i' \mathbf{x}_i^T \end{bmatrix} \tag{14}$$

and $\mathbf{h} = \begin{bmatrix} \mathbf{h}^{1T} & \mathbf{h}^{2T} & \mathbf{h}^{3T} \end{bmatrix}^T$. Thus, from the $N_{\mathcal{R}}$ point matches of the subset $\mathcal{R}$, a system of linear equations can be written, $A\mathbf{h} = \mathbf{0}$, where the matrix A is composed from the rows of A_i, $A \in \mathbb{R}^{2N_{\mathcal{R}} \times 9}$.

The LS estimation of H is accomplished by performing a singular value decomposition of the matrix A:

$$A = UDV^T \tag{15}$$

where U and V are orthogonal matrices and D is a diagonal matrix containing the singular values of the matrix A. Assuming the singular values are arranged in descending order in the matrix D, $\mathbf{h}$ is set to the last column of V and is then used to construct the homography matrix H (cf. Ref. 6).

It is also possible to estimate the *fundamental matrix*, $F \in \mathbb{R}^{3\times 3}$, given a set of matched points. The fundamental matrix will be used in one of the examined methods for motion estimation (Section IV) based on regular camera captured images (and not mosaic images). The fundamental matrix represents the *epipolar constraint*, expressed for a pair of corresponding points $\mathbf{x} = (x, y, 1)^T$ and $\mathbf{x'} = (x', y', 1)^T$ as

$$\mathbf{x}'^T F \mathbf{x} = 0 \tag{16}$$

or in the explicit form[6]

$$x'xf_{11} + x'yf_{12} + x'f_{13} + y'xf_{21} + y'yf_{22} + y'f_{23} + xf_{31} + yf_{32} + f_{33} = 0 \tag{17}$$

where $f_{ij} = F(i, j)$.

By letting $\mathbf{f} \in \mathbb{R}^9$ denote a 9-dimensional vector constructed from the entries of F, Eq. (17) can be written as

$$[x'x, x'y, x', y'x, y'y, y', x, y, 1]\,\mathbf{f} = 0 \tag{18}$$

When n matched points are available, the following set of linear equation may be formed

$$\widetilde{A}\mathbf{f} = \mathbf{0} \tag{19}$$

where

$$\widetilde{A} = \begin{bmatrix} x'_1x_1 & x'_1y_1 & x'_1 & y'_1x_1 & y'_1y_1 & y'_1 & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x'_nx_n & x'_ny_n & x'_n & y'_nx_n & y'_ny_n & y'_n & x_n & y_n & 1 \end{bmatrix} \tag{20}$$

Since each pair of corresponding points contributes a single equation, a minimum of 8 feature matches is required for calculating the fundamental matrix (in contrast to 4 feature matches in the case of homography computation).

The estimation process of F is similar to the estimation process of H, which is summarized in Algorithm 1, with the following exceptions:[6]

- In Step 4 of Algorithm 2, 8 feature matches instead of 4 matches are drawn.
- Eq. (9) used in Step 5 of Algorithm 2, is replaced with

  $$\frac{(\mathbf{x}'^T F\mathbf{x})^2}{(F\mathbf{x})_1^2 + (F\mathbf{x})_2^2 + (F\mathbf{x'})_1^2 + (F\mathbf{x'})_2^2} < t^2_{threshold} \tag{21}$$

  where $(F\mathbf{x})_j^2$ is the j-th entry of the vector $F\mathbf{x}$.
- The LS estimation in Step 4 of Algorithm 1 is computed based on the matrix $\widetilde{A}$ defined in Eq. (20).

It is important to note that the fundamental matrix cannot be estimated if one of the two examined images is a homography-constructed mosaic, because any two views of a scene are related by a homography relation (Eq. (5)) only in the case of a planar scene or a pure rotation; in these scenarios, however, the epipolar geometry is undefined.[6]

2. Mosaic Construction Logic

The mosaic construction process and the camera scanning procedure are coupled. During the scan, images are captured using varying camera angles. While all the images contribute to the construction of a mosaic, only images taken while the camera was pointing downwards 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. The mosaic image is expressed in the current downward-looking image coordinate system, defined as the coordinate system C of the most current downward looking image. Thus, a new non downward-looking image is warped towards the mosaic image based on the homography estimation between these two images, whereas if the new image is a downward-looking image, the mosaic image is warped towards the new image. The latter process is required since the observations that are derived from the homography matrix describe camera motion between time instances of successive downward-looking images (cf. Ref. 1). The increased overlap between the warped mosaic image and the new downward-looking image (cf. Figs. 2(a) and 4), and the quality of the estimated homography matrix, are the two factors that allow better motion estimation in certain scenarios, as will be demonstrated in Section IV-B.

Warping the mosaic image is a computationally-extensive operation; the computational effort increases with time due to growth of the mosaic image size. Other variations of this process may be considered to alleviate the computational load. One possible approach is to construct the mosaic image in a constant reference frame, and to warp relevant regions from the mosaic towards the previous downward-looking image frame once motion estimation is required. After the relevant image has been warped, the two images are blended using one of the available image integration techniques.[13,14,23]

An example of the mosaic image construction process, based on real images acquired using the scanning procedure described above, is given in Fig. 4. The images were captured using Google Earth, as detailed in Section IV-B. One can easily notice the increased overlapping region in the mosaic image (Figure 4(b)) and the new downward-looking image (Figure 4(c)).

Non Downward-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 reducing the accumulation of alignment errors in the mosaic image, while updating it with new non downward-looking images.

The proposed method is an adaptation of the procedure suggested by Refs. 37,10 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 mosaic image is expressed. Each new image I_k, which is a *non*-downward-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

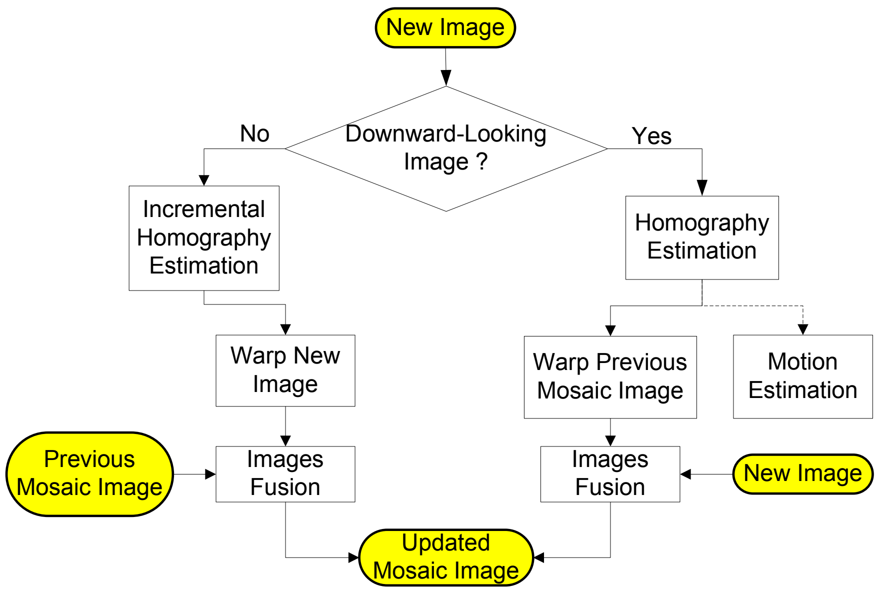


Figure 3. Block diagram of the implemented mosaic construction method.

calculate an intermediate homography matrix between the new image I_k and the current mosaic image, relying on information computed for the previous image I_{k-1}:

$$H^I_{k\to r} = H_{k-1\to r} \cdot H_{k\to k-1} \tag{22}$$

where $H_{k-1\to r}$ is the homography matrix from the previous image, I_{k-1}, to the current 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 towards the mosaic image using the homography matrix $H^I_{k\to r}$: Every point $\mathbf{x}_i$ in I_k is transformed to $\mathbf{x}'_i = H^I_{k\to r} \cdot \mathbf{x}_i$ while maintaining its intensity level, $I(\mathbf{x}'_i) = I(\mathbf{x}_i)$. The set of these transformed points, along with the intensity levels, $\{\mathbf{x}'_i, I(\mathbf{x}'_i)\}$, constitute a warped image $\tilde{I}^r_k$.

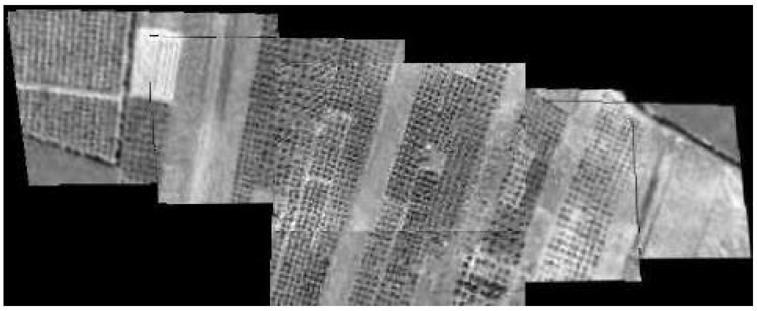
Ideally, the warped image and the mosaic image should be in perfect alignment; however, due to homography estimation errors, this does not happen. To improve the estimation, a *correction homography* between the warped image, $\tilde{I}^r_k$, and the current mosaic image, is estimated by applying the standard homography estimation technique given in Algorithm 1 on these two images. This homography, H_{corr}, is used to correct the estimated intermediate homography matrix between the new image and the mosaic image,

$$H_{k\to r} = H_{corr} \cdot H^I_{k\to r} \tag{23}$$

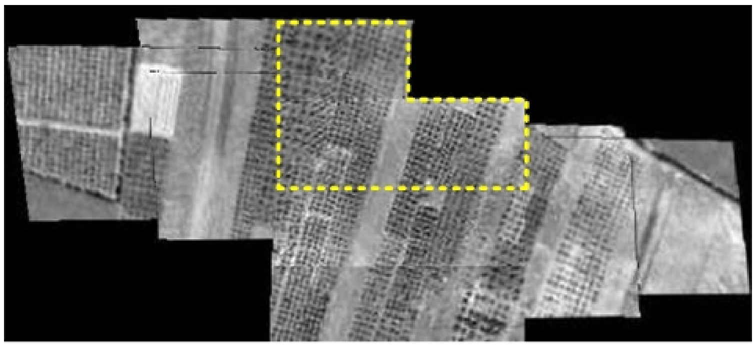
Finally, the new image, I_k, is warped towards the current mosaic image using the improved homography matrix, $H_{k\to r}$, followed by an integration of the two images into an updated mosaic. In addition, $H_{k \to r}$ is saved for future use with new non downward-looking images. The process repeats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

Downward-Looking Images

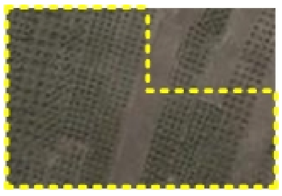
Once a new downward-looking image, I_d, is captured, a direct estimation of the homography matrix relating this new image to the previous mosaic is performed. The mosaic image is warped towards the new image using this homography, and the two images are integrated into an updated mosaic. As a result of the warping operation, the new mosaic image is expressed in the coordinate system of I_d (i.e. the reference index r is changed to d), and the homography matrix that relates the current image to the mosaic is set to the identity matrix, i.e. $H_{r \to r} = I$. From here on, each new non downward-looking image will be matched against mosaic images expressed in the updated reference frame, until another downward-looking image is received.



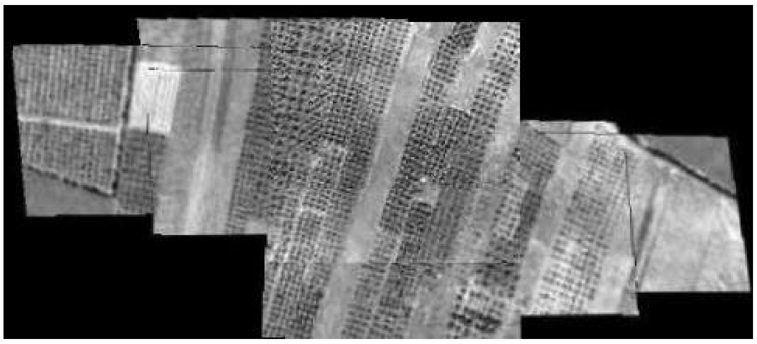
(a)



(b)



(c)



(d)

Figure 4. Mosaic Image Incremental Construction and Camera Scan Example. The images were acquired from Google Earth with a narrow FOV camera of $5^o \times 3^o$. The camera scan procedure is comprised of taking two non downward-looking images in each direction (cf. Figure 2(a)), while the images that are located in the middle of the mosaic image are the downward-looking images. (a) A mosaic image from previous steps. (b) Inclusion of a new non downward-looking image into the mosaic image. (c) A new downward-looking image. (d) Inclusion of a new downward-looking image (c) into the mosaic image (b), and relative motion estimation between these two images. Notice the increased overlapping region (marked by a dotted line) between the previous mosaic image (b) and the new downward-looking image (c).

IV. Image-Based Motion Estimation

In this section, two methods for motion estimation are discussed. In both cases, it is assumed that the calibration matrices are known, and that there are no additional sensors or information except the information provided by the camera. Thus, the translation motion between two images can be estimated only up to scale (i. e. only the translation direction can be estimated).

The first method assumes some general scene and a translation motion of the camera, a scenario that can be described by means of epipolar geometry (see, e. g. Ref. 6). Estimation of the fundamental matrix, F, (cf. Section III-B.1) enables extraction of the motion parameters according to the following relationship:

$$F = K'^{-T}\mathbf{t}^{\wedge}RK^{-1} \tag{24}$$

where K', K are the calibration matrices at two images time instances, $\mathbf{t}$ is the translation vector, R is the rotation matrix and $(\cdot)^{\wedge}$ is the matrix cross-product equivalent defined for some vector $\mathbf{a} = [a_1, a_2, a_3]^T$ as

$$\mathbf{a}^{\wedge} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \tag{25}$$

The calibration matrix of a CCD camera, assuming square pixels, can be written as[6]

$$K = \begin{bmatrix} f & s & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \tag{26}$$

Here f is the camera focal length, p_x, p_y are the principal point coordinates relative to the camera system origin, and s is a skew parameter. The extraction procedure of the motion parameters from Eq. (24), given the matrices F, K, K', is given in Chapter 8.6 of Ref. 6.

When a planar scene or a pure rotation motion is considered, the method described above cannot be applied because the epipolar geometry is undefined. As explained by Ref. 6, the two views are related in this case by a homography matrix, H, which can be written as:[8]

$$H = K'\left[R - \frac{\mathbf{t}}{z}\mathbf{n}^T\right]K^{-1} \tag{27}$$

where z is the scene depth and $\mathbf{n} = (a, b, c)^T$ is a unit vector normal to the scene plane (both are unknown). A 3D point $(x, y, z)^T$ is located on the scene plane if

$$ax + by + cz = 1 \tag{28}$$

A method for extracting the motion parameters from Eq. (27) was suggested in Ref. 8, 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.[8] If one of the processed images is a mosaic image, the principal point coordinates in its calibration matrix (Eq. (26)) should be updated with the location of the most current downward-looking image in the mosaic. The implementation of the estimation process in this work involves yet another phase, which will be described in the next section.

The experimental results of motion estimation, which are presented in Section IV-B, are based on both methods discussed above. 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 semi-planar scene. In addition, estimation of the fundamental matrix cannot be performed in a mosaic-based framework (cf. Section III-B.1). Thus, in case of a narrow-FOV, only the homography-based motion estimation method is relevant. This method was applied on two motion estimation frameworks: The traditional two-view framework, in which the two images are regular camera-captured images, and the mosaic framework, in which one of the images is a mosaic image from the previous time step.

The next section describes the implemented method for motion estimation in a planar scene. Section IV-B focuses on the performance of the motion estimation techniques described above, using the mosaic framework viz-à-viz the traditional two-view framework. First, results of motion estimation based on the two-view framework are given and analyzed for the case of a wide-FOV camera. Second, the performance of the mosaic framework and the two-view framework is evaluated for a narrow-FOV camera and various scene types. It is shown that the mosaic framework yields considerably better results in the case of low-texture scenes.

A. Implementation of Motion Estimation Assuming a Planar Scene

As mentioned in Section III-B.1, the homography estimation process involves the RANSAC algorithm for robust outliers rejection. The output from this algorithm is a subset $\mathcal{R}$ 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 does differ from one execution to another for noisy data (for a given threshold value), since each execution may yield a different features subset group, and hence a different estimation of the homography matrix (and the motion parameters).

More specifically, assume that the extracted SIFT features image coordinates are corrupted with some noise. As a consequence, the computed set $\mathcal{S}$ of all point matches (step 2 in Algorithm 1) is also corrupted with 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 utilized to construct a subset $\mathcal{T}$ of point matches that are consistent with this homography matrix (Step 5 in Algorithm 1).

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. (9):

$$d_E(\mathbf{x}, H_i^{-1}\mathbf{x}')^2 + d_E(\mathbf{x}', H_i\mathbf{x})^2 < t^2_{threshold} \tag{29}$$

Only point matches that agree with the above condition are added to the subset $\mathcal{T}_i$. Since both homography matrices are legitimate (as they were calculated based on inlier point matches) but different, it is obvious from Eq. (29) that the two subsets $\mathcal{T}_1$ and $\mathcal{T}_2$ will be identical for a sufficiently large value of $t_{threshold}$ (which should still reject false matches). However, decreasing the value of $t_{threshold}$ 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: Images of high-texture scenes are likely to be corrupted with less noise compared to images of low-texture scenes, since in the former case the features are localized with improved precision. Thus, a specific value of $t_{threshold}$ 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 (cf. Algorithm 2). Thus, sequential activation of the RANSAC algorithm might give different subsets of $\mathcal{R}$, implicating that the LS homography estimation (Step 4 in Algorithm 1) 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 above described 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 $|\mathbf{q}| = \big|[q_1, \ldots, q_n]^T\big| \triangleq [|q_1|, \ldots, |q_n|]^T$. Define the mean unit vector normal to a scene plane, based on the normal unit vectors $\{\mathbf{n}_i^{prev}\}_{i=1}^{N_{prev}}$ from estimates of N_{prev} previous images, as

$$\mathbf{n}_\mu^{prev} = \frac{\Sigma_{i=1}^{N_{prev}} |\mathbf{n}_i^{prev}|}{\left\|\Sigma_{i=1}^{N_{prev}} |\mathbf{n}_i^{prev}|\right\|} \tag{30}$$

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 $\mathbf{n}_\mu^{prev}$:

$$s_i = | < \mathbf{n}_\mu^{prev}, \mathbf{n}_i > | \tag{31}$$

where $< .,. >$ 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 ($\Lambda \in \mathbb{R}^{N_1 \times 3}$):

$$\Lambda = (\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \boldsymbol{\lambda}_3) \equiv \begin{bmatrix} |\mathbf{t}_1^T| \\ |\mathbf{t}_2^T| \\ \vdots \\ |\mathbf{t}_{N_1}^T| \end{bmatrix} \tag{32}$$

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 $\mathbf{n}_\mu^{prev}$, 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 $< \mathbf{n}_i, \mathbf{n}_\mu >$ where $\mathbf{n}_\mu$ is defined as

$$\mathbf{n}_\mu = \frac{\Sigma_{i=1}^{N_2} |\mathbf{n}_i|}{\left\|\Sigma_{i=1}^{N_2} |\mathbf{n}_i|\right\|} \tag{33}$$

The sequential homography estimation process described above is summarized in Algorithm 3. In the current implementation, this routine was executed with $N = 10$, i. e. a single execution of the sequential homography estimation procedure consists of running the standard homography estimation given in Algorithm 1 10 times. The improvement in the precision of estimation is clearly evident in Figure 5(c), where the same pair of images was used to perform motion estimation with and without a sequential homography estimation. 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.

Algorithm 3 Sequential Homography Estimation Algorithm

```
1: Run $N$ times Algorithm 1 for homography matrix estimation, 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. (30,31).
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. (32) 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 $< \mathbf{n}, \mathbf{n}_\mu >$, where $\mathbf{n}_\mu$ is
      computed according to Eq. (33).
11: end if
```

The results in Figure 5(c) were achieved by executing a standard homography estimation (Algorithm 1) and a sequential homography estimation (Algorithm 3) 100 times on a pair of low-texture images taken with a $7^o \times 4^o$-FOV camera (Figures 5(a) and 5(b)), i. e., executing the homography estimation procedure described in Section III-B.1 100 times, and executing the sequential homography estimation procedure 100 times. The advantage of the sequential homography estimation is significant. For example, nearly 80% of the estimation errors were below 20^o when applying sequential estimation, compared to only 50% with a standard homography estimation.

It is important to note that the sequential estimation procedure was applied for *both* the mosaic-based and two-view motion estimation methods. Furthermore, implementation of more sophisticated methods for homography estimation (such as bundle adjustment), if enough computational resources are available, are expected to improve the estimation accuracy and also to eliminate the need in the sequential estimation procedure described in this section. However, the sequential estimation procedure provides significant improvement in precision, given the standard estimation technique given in Algorithm 1 is used.

B. Image-Based and Mosaic-Based Motion Estimation - Performance Evaluation for Different Camera FOVs

This section presents a performance evaluation of the mosaic-based motion estimation method, summarized in Algorithm 4, for different camera FOVs and different scene types. The results of the mosaic-based estimation are compared to the traditional two-view motion estimation. The experiments are based on real image sequences acquired using Google Earth,[38] which contains 3D geo-data of Earth based on real imagery and thus may be used as a substitute for complicated test setups in which the camera captures 3D scenes. Further details regarding the interface to Google Earth are provided in Appendix A.

Algorithm 4 Mosaic-Based Motion Estimation

```
1: Input: Image $I_k$ and previous mosaic image $M_{k-1}$.
2: if Image $I_k$ is a downward-looking image then
3:     Perform motion estimation between image $I_k$ and mosaic image $M_{k-1}$ following Algorithm 3.
4: end if
5: Calculate camera angles for next image according to Section III-A.
6: Update the mosaic image $M_{k-1}$ with image $I_k$ according to Section III-B.
```

Due to the different accuracy levels of the estimated motion parameters given different images, and the need to evaluate the overall performance, the results are presented in terms of a cumulative distribution of the estimated motion parameters error along the image sequences that were used during the examinations.

We begin by examining the accuracy of the translation direction estimation in the case of a wide-FOV camera. Given a non-flat scene and a camera with a wide FOV, one can estimate the fundamental matrix, since the epipolar geometry is well defined (as long the camera performs some translation motion), and then extract the motion parameters (rotation and up-to-scale translation). Figure 6 shows the accuracy of the translation direction estimation for the two-view motion estimation method. It can be seen that more than 70% of the estimates yield accuracy better than 5^o, and that more than 90% of the estimates provide accuracy better than 15^o.

The more interesting and challenging scenarios, however, are those involving a camera with a narrow FOV and low-texture scenes, which result in low-quality and very few features. Such scenarios are common in many airborne applications. The combination of a narrow FOV and low-texture scenes raises significant hurdles when trying to estimate the fundamental matrix, because the epipolar geometry is ill-conditioned. A common remedy in these cases is to assume a planar scene and to perform homography matrix estimation[8,12]) followed by a motion parameters extraction (see Section IV).

Next, we examine the performance of the proposed mosaic-based motion estimation method for cameras with a narrow FOV, compared to estimations of the two-view framework. In both cases the motion parameters are extracted from an estimated homography matrix. However, in the mosaic framework only the downward-looking images participate in the motion estimation (cf. Algorithm 4).

To evaluate the performance of the proposed mosaic framework for motion estimation, several experiments were carried out using real image sequences acquired from Google Earth.

An aerial platform equipped with a gimbaled camera was assumed to perform a straight-and-level north-heading trajectory[a], whose details are given in Table 1. The observed scene along this trajectory is of a planar nature with about 50 m elevation above sea level. 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, discussed in Section III-A, at a 5 Hz frequency.

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

The experiments are different in the camera FOV ($7^o \times 4^o$ and $5^o \times 3^o$) and in the scene type captured by the camera. Two scene types were examined. In the first scene type, the captured images were usually high-textured and thus had numerous high-quality features, as opposed to the second scene type, which is characterized by low-texture images and a small number of features. Examples of typical images of these two scene types are given in Figure 7.

Figures 8 and 9 present the experimental results: Figure 8 depicts the cumulative dis-

[a]The same trajectory and image sequences were used in experiments of mosaic-aided navigation, presented in Part II of this work.[1]

tribution of the translation direction estimation error, while Figure 9 shows the cumulative distribution of the rotation estimation error. The shown rotation error is the maximum value of the error in the estimated rotation vector, i. e.

$$\Delta\eta \triangleq \max(|\Delta\phi|, |\Delta\theta|, |\Delta\psi)| \tag{34}$$

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

$$R_{err} \equiv R_{true} \cdot R^T \tag{35}$$

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

Each of these figures describe the performance of the examined motion estimation methods for the following four scenarios:

- FOV $7^o \times 4^o$, high-texture scene.
- FOV $7^o \times 4^o$, low-texture scene.
- FOV $5^o \times 3^o$, high-texture scene.
- FOV $5^o \times 3^o$, low-texture scene.

The platform performed the same trajectory for each scene type. Thus images that were taken by the two examined camera FOVs in high-texture scenes are different only due to the change in the camera FOV only, and not because of a different platform trajectory (the same applies to low-texture scenes as well).

As expected, for both methods the estimated motion precision deteriorates when a smaller FOV camera is assumed. For example, the percentage of translation direction estimations with estimation errors below 15^o is around 60% in case of a $7^o \times 4^o$ FOV (Figure 8(d)), dropping to 20% for the narrow FOV of $5^o \times 3^o$ (Figure 8(b)).

The same applies to rotation estimations. Around 90% of the estimates are provided with an estimation error below 10^o in the case of a $7^o \times 4^o$ FOV (Figure 9(d)), compared to 20%-40% in the other case (Figure 9(b)). As will be seen in Part II of this work,[1] these motion estimations can be effectively utilized for improving the performance of navigation systems.

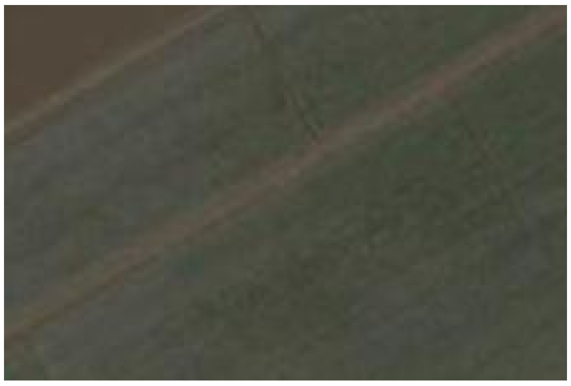
The precision of motion estimation is better for the high-texture scenes compared to low-texture scene due to the larger number of high-quality features This can be seen by comparing Figure 8(d) to Figure 8(c). However, the low-texture scenes are common in many airborne applications; it is in these scenarios that the mosaic provides improved motion estimation.

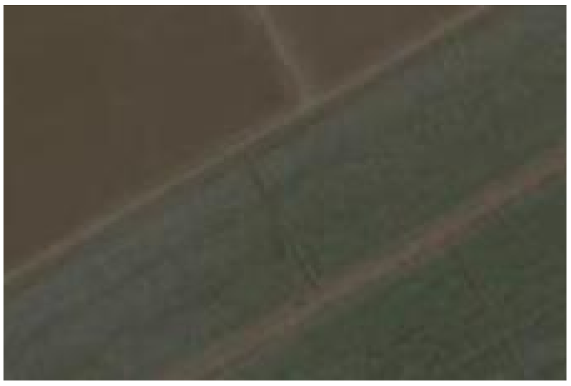
It is important to understand when the mosaic-based method outperform the two-view-based method. In the context of motion estimation, the two methods differ only in the size of the image overlap region. Due to the camera scanning process, the constructed mosaic image contains an enlarged overlapping region compared to the overlapping region between two regular images. This region is comprised of the original overlapping area between two regular images and *an additional* overlapping region - see a schematic illustration in Figure 2(a) and a mosaic example image in Figure 4. However, since the mosaic construction process is imperfect, features from the additional overlapping area tend to be of lower-quality compared to 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. Section III-B). Thus, there is an inherent tradeoff: On the 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 the overlapping region between two captured images cannot yield a large number of high-quality features, i. e., a narrow-FOV camera and low-texture scenes.

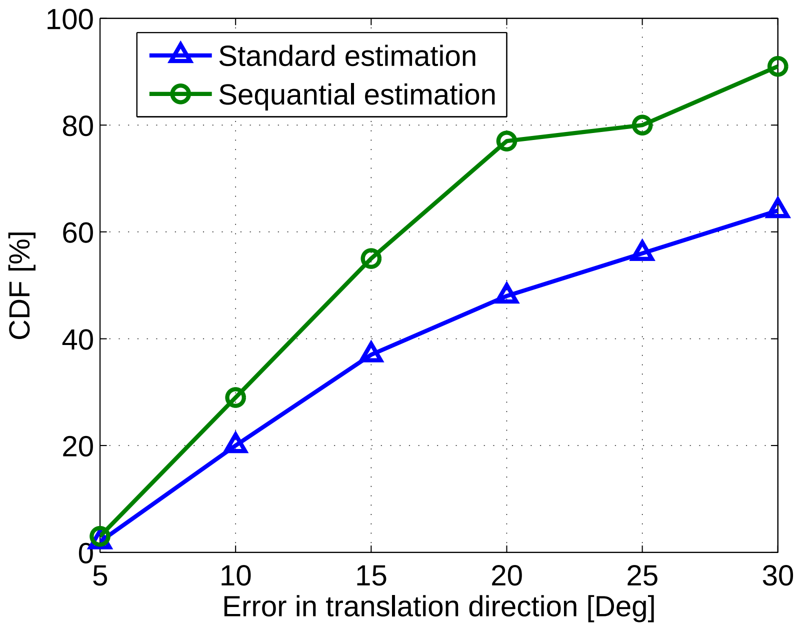
The above finding clearly evident in Figures 8(a) and 9(a), which describe the scenario of a narrow-FOV camera ($5^o \times 3^o$) and a low-texture scene. It can be seen that the mosaic-based motion estimation yields considerably better results compared to the two-view motion estimation. For example, in case of translation direction estimation (Figures 8(a)), 50% of the estimates using the mosaic method are provided with an accuracy better than 15^o, compared to only 20% using the two-view method. In the case of rotation estimation, the mosaic-based motion estimation yields improved results for the narrow FOV camera in both scene types (Figures 9(a) and 9(b)). As for the other examined scenarios, there is some advantage in favor of the two-view framework, since in these cases the set of high-quality features that can be extracted from the overlapping region between the two captured images is sufficiently large, while the inclusion of lower-quality additional features only deteriorates the estimation precision.

The performance of the mosaic-based motion estimation method can be further enhanced by implementing more sophisticated methods of mosaic construction, such as adjusting the estimated homography matrices of images from consecutive scan stripes by means of global optimization. This should not be a very expensive operation due to the small number of features in each image (because of the assumed narrow-FOV camera and a low-texture scene) and the limited number of images that constitute each scan stripe (5 images in the current implementation). In addition, the performance of both methods can be improved by following more powerful homography estimation methods. The reader is referred to Ref. 6

for further details.


(a)


(b)



(c)

Figure 5. (a),(b) Images of a low-texture scene captured from Google Earth by a $7^o \times 4^o$ FOV camera. (c) Motion sequential estimation vs. standard estimation over the pair of images presented in (a),(b): CDF of the translation direction estimation error. The improvement in favor of sequential estimation can be clearly seen. For example, nearly 80% of the estimations errors were below 20^o when applying sequential estimation, compared to only 50% in case sequential estimation is not applied.

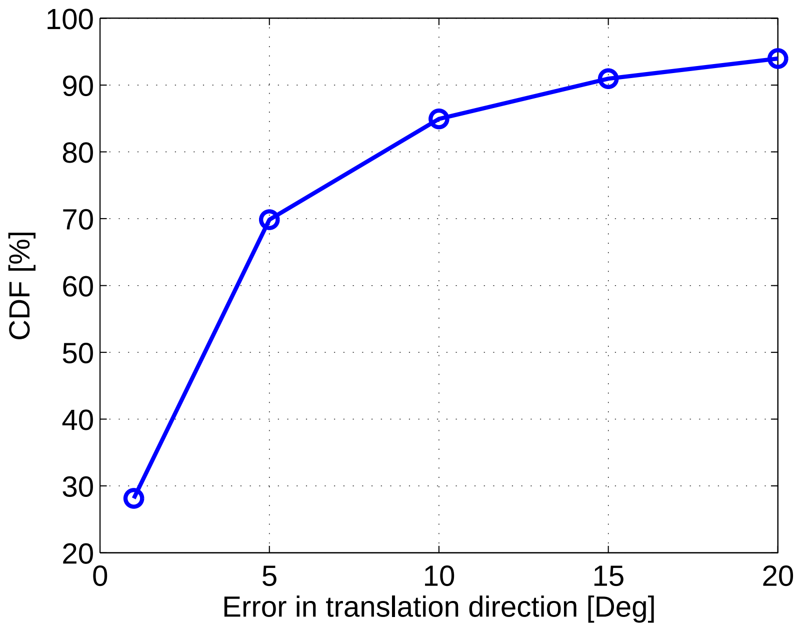


Figure 6. CDF of the translation direction estimation error based on two-view framework, wide camera FOV (about 30^o). The translation motion was extracted from an estimated fundamental matrix. More than 90% of the estimates are at least 15^o-accurate.

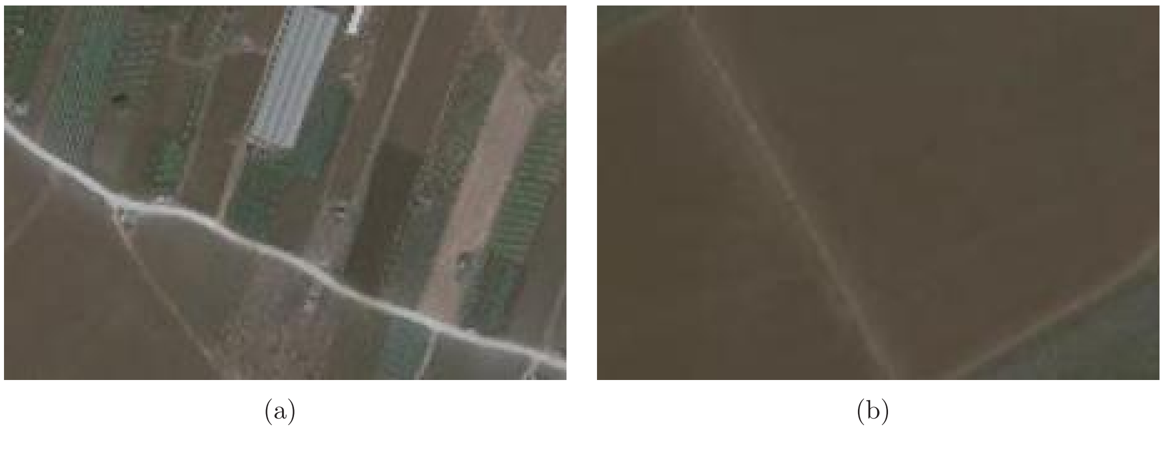
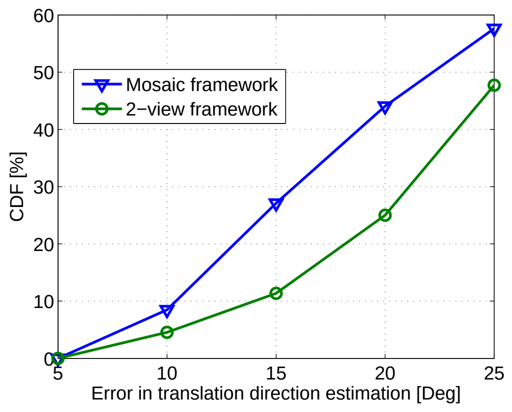
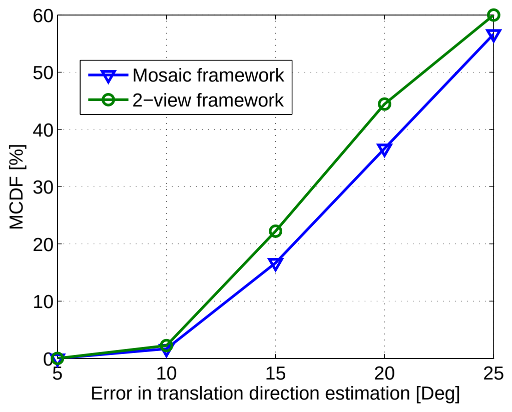


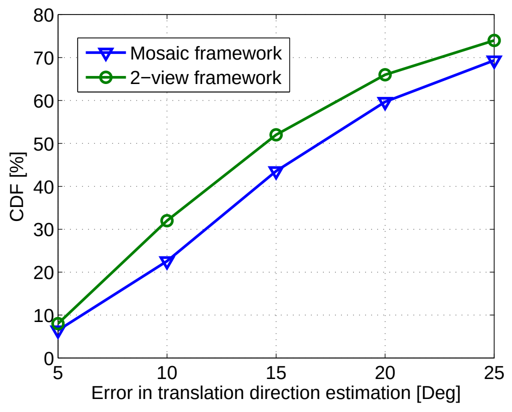
Figure 7. Images taken by a camera with a $7^o \times 4^o$ FOV: (a) High-texture scene; (b) Low-texture scene.



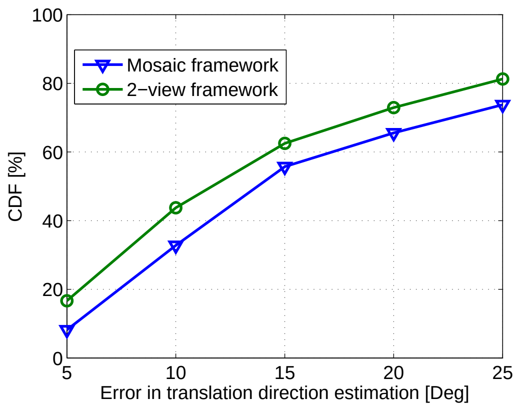
(a) FOV $5^o \times 3^o$, low-texture scene type



(b) FOV $5^o \times 3^o$, high-texture scene type

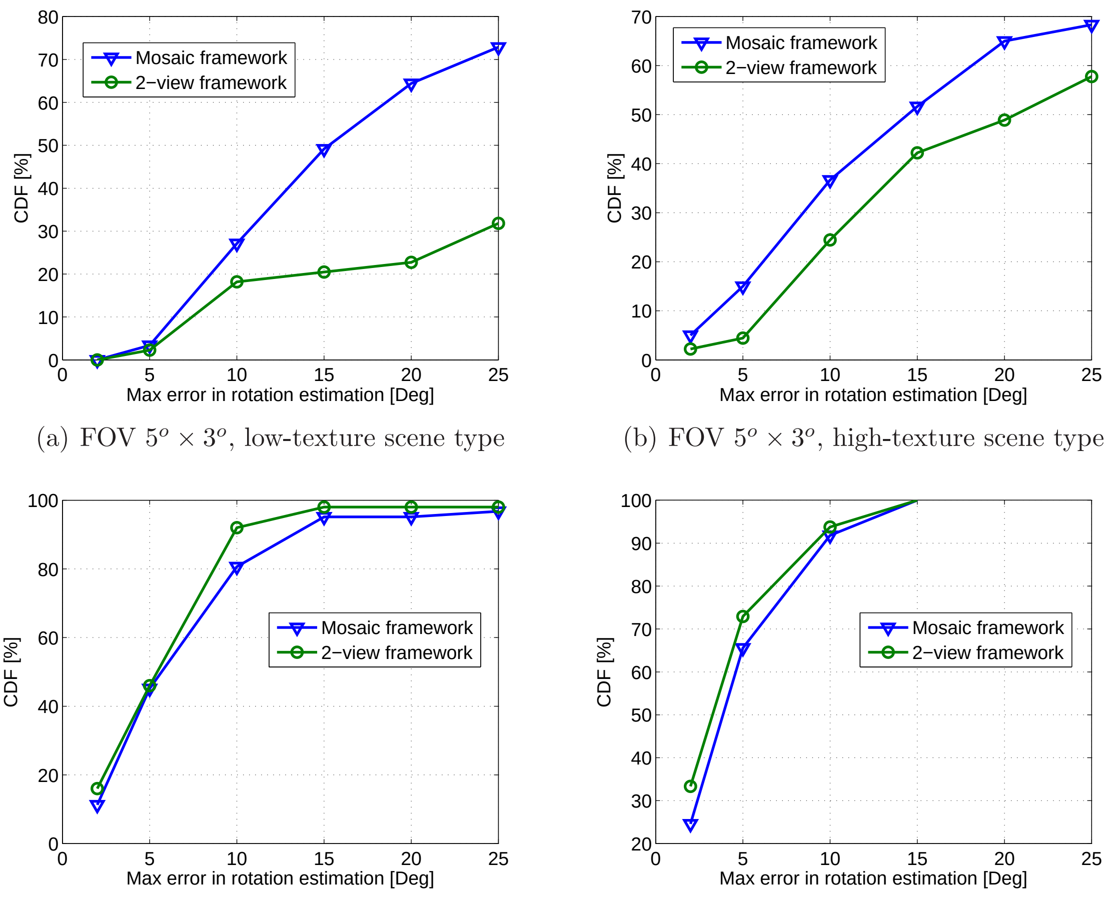


(c) FOV $7^o \times 4^o$, low-texture scene type



(d) FOV $7^o \times 4^o$, high-texture scene type

Figure 8. Translation direction estimation accuracy for different scene types and camera field of views (CDF). The mosaic framework improves the accuracy of translation direction estimation for a narrow FOV of $5^o \times 3^o$, while dealing with low-texture type scenes.



(a) FOV $5^o \times 3^o$, low-texture scene type

(b) FOV $5^o \times 3^o$, high-texture scene type

(c) FOV $7^o \times 4^o$, low-texture scene type

(d) FOV $7^o \times 4^o$, high-texture scene type

Figure 9. Rotation estimation accuracy for different scene types and camera field of views (CDF). The mosaic framework improves the accuracy of rotation estimation for a narrow FOV of $5^o \times 3^o$, in both low and high texture scene types. Similar estimation accuracy in both methods for camera with $7^o \times 4^o$ FOV, with some advantage in favor of two-view framework.

V. Conclusions

This paper developed a method for improving vision-based motion estimation utilizing an on-line mosaic image construction process. The system considered in this work consisted of an airborne platform equipped with a standard navigation system and a gimballed camera, which scanned ground regions during flight. The camera-captured images were processed on-line by the mosaic construction procedure into a single mosaic image. In parallel, part of these image were also used for motion estimation.

The improvement in precision of mosaic-based motion estimation was due to the enlarged overlapping regions between the camera-captured image and the most recent mosaic image. The proposed mosaic-based method for motion estimation was examined using real image sequences acquired from Google Earth, which demonstrated its superiority over the traditional two-view motion estimation method for difficult scenarios, such as cameras with narrow field-of-views and low-texture scenes.

In particular, a $5^o \times 3^o$ FOV camera was examined in a low-texture scene scenario, for which it was shown that about 50% of the translation estimations achieved using the mosaic-based motion estimation method were with a better accuracy than 15^o, compared to only 20% when a traditional two-view method was applied.

Appendix A: Google Earth Interface

Figure 10 schematically depicts the interface to Google Earth. 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. In addition, special care was taken to allow roll motion in Google Earth, as this type of motion is not supported by the current version of Google Earth.

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. Figure 7).

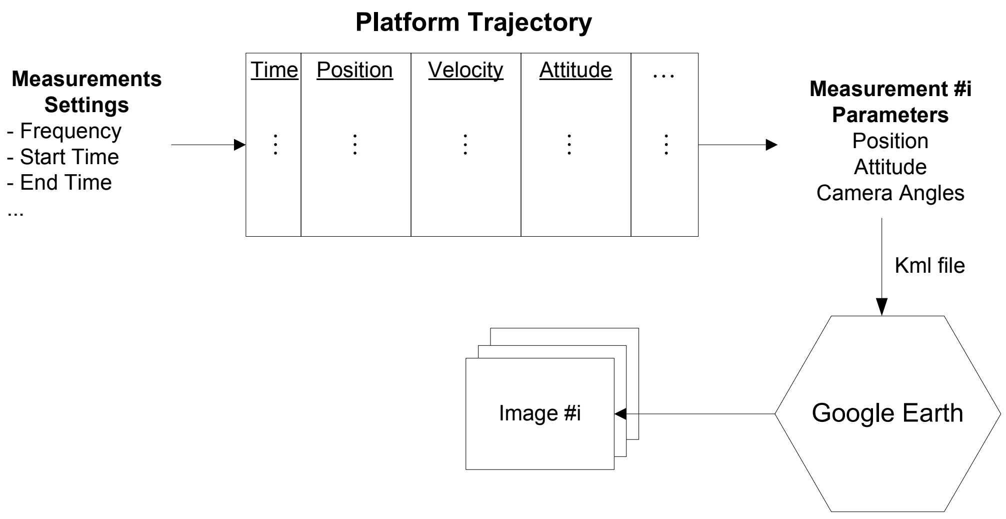


Figure 10. A schematic illustration of an interface between the platform trajectory and Google Earth

Appendix B: Scale Invariant Feature Transform (SIFT)

Scale Invariant Feature Transform (SIFT) [36] is a method for extracting features from images, which are then can be used for matching the same scene observed from different views. The features are invariant to image scaling and rotation, and provide robust matching to a variety of other transformations that an image object might undergo (such as change of illumination, affine transformations, change in viewpoint).[36] They are also highly distinctive, i. e., a specific feature can be correctly found among a large set of different features.

The computation of SIFT features is performed in several steps. First, a scale-space representation of the image is generated by smoothing the original image with a Gaussian kernel, followed by an extrema detection of this scale-space representation. This step yields a set of candidate points $\{(x, y, \sigma)\}$ for SIFT features, where (x, y) are the image coordinates of the feature, and σ is the width of the Gaussian kernel for which the extrema were found. Low-contrast candidate points and the candidate points that are positioned along edges are rejected for improved stability, yielding a reduced set of features. The next step is to assign orientation θ based on local image properties for each feature, which ensures invariance of the feature to image rotation. In the last step of the SIFT algorithm, each extracted SIFT feature is attached with a descriptor vector, which encodes information regarding this feature and is used to distinguish this specific feature from other features. In a default SIFT implementation,[39] this vector contains 128 elements.

Thus, a SIFT feature is represented by 4 parameters $\{x, y, \sigma, \theta\}$, and by a 128-length descriptor vector, which is used for matching a specific SIFT feature with other SIFT features (from other images). The matching of a specific feature i with a set of other features is based on computation of Euclidean distances between the descriptor vector of feature i and the descriptor vectors of the features in the set. The matched feature is chosen as the feature in the set for which a minimum Euclidean distance was obtained.

References

[1]Indelman, V., Gurfil, P., Rivlin, E. and Rotstein,H., *Real-Time Mosaic-Aided Aircraft Navigation: II. Sensor Fusion*, AIAA GN&C Conference, USA, 2009.

[2]Indelman, V., Gurfil, P., Rivlin, E. and Rotstein,H., *Navigation Performance Enhancement Using Rotation and Translation Measurements from Online Mosaicking*, AIAA GN&C Conference, SC, USA, 2007.

[3]Indelman, V., Gurfil, P., Rivlin, E. and Rotstein,H., *Navigation Aiding Using On-Line Mosaicking*, IEEE/ION PLANS, California, USA, 2008.

[4]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.

[5]Eustice, R., Pizarro, O. and Singh, H., "Visually Augmented Navigation in an Unstructured Envi-

ronment Using a Delayed State History," *IEEE International Conference on Robotics & Automation*, April 2004, pp. 25–32.

[6]Hartley, R. and Zisserman, A., *Multiple View Geometry*, Cambridge University Press, 2000.

[7]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, January 2005, pp. 221–228.

[8]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, August 1982, pp. 525–534.

[9]Gracias, N. and Santos-Victor, J., "Underwater Video Mosaics as Visual Navigation Maps," *Computer Vision and Image Understanding*, Vol. 79, 2000, pp. 66–91.

[10]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*, Orlando, Florida, May 2006, pp. 2860–2865.

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

[12]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*, May 2002, pp. 2793–2798.

[13]Peleg, S. and Herman, J., "Panoramic mosaics by manifold projection," *In CVPR*, 1997, pp. 338–343.

[14]Szeliski, R., *Image alignment and stitching: A tutorial*, Tech. Rep. MSR-TR-2004-92, Microsoft Research, 2005.

[15]Fleischer, S., Wang, H., and Rock, S., "Video Mosaicking Along Arbitrary Vehicle Paths," *Proceedings of the Symposium on Vehicle Technology*, 1996, pp. 293–299.

[16]Gracias, N., Costeira, J., and J.Santos-Victor, *Linear Global mosaic For Underwater Surveying*, IAV2004, 2004.

[17]Kanazawa, Y. and Kanatani, K., "Image Mosaicing by Stratified Matching," *Image and Vision Computing*, Vol. 22, 2004, pp. 93–103.

[18]Peleg, S., Rousso, B., Rav-Acha, A., and Zomet, A., "Mosaicing on Adaptive Manifolds," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 22, No. 10, 2000, pp. 1144–1154.

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

[20]Zelnik-Manor, L. and Irani, M., "Multiview Constraints on Homographies," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 24, No. 2, 2002, pp. 214–223.

[21]Richmond, K. and Rock, S., "An Operational Real-Time Large-Scale Visual Mosaicking and Navigation System," *OCEANS*, September 2006, pp. 1–6.

[22]Ferrer, J., Elibol, A., Delaunoy, O., Gracias, N. and Garcia, R., "Large-Area Photo-Mosaics Using Global Alignment and Navigation Data," *OCEANS*, September 2007, pp. 1–9.

[23]Irani, M., Anandan, P., and Hsu, S., "Mosaic Based Representations of Video Sequences and Their Applications," *Proc. of IEEE ICCV*, 1995, pp. 605–611.

[24]Zhang, P., Milios, E. E., and Gu, J., *Graph-based Automatic Consistent Image Mosaicking*, IEEE International Conference on Robotics and Biomimetics, Shenyang, China, Paper no. 332, 2004.

[25]Zhu, Z., Hanson, A., and Riseman, E., "Generalized Parallel-Perspective Stereo Mosaics from Airborne Video," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 26, No. 2, 2004, pp. 226–237.

[26]Fischler, M. and Bolles, R., "Random Sample Consensus: a Paradigm for Model Fitting with Application to Image Analysis and Automated Cartography," *Commun. Assoc. Comp. Mach.*, Vol. 24, 1981, pp. 381–395.

[27]Rousseeuw, P., "Least Median of Squares Regression," *Journal of the American Statistical Association*, Vol. 79, 1984, pp. 871–880.

[28]Kang, S., Szeliski, R., and Uyttendaele, M., *Seamless Stitching using Multi-Perspective Plane Sweep*, Tech. Rep. MSR-TR-2004-48, Microsoft Research, 2004.

[29]Garcia, R., *A proposal to estimate the motion of an underwater vehicle through visual mosaicking*, Phd thesis. University of Girona, Spain, 2002.

[30]Garcia, R., Nicosevici, T., Ridao, P., and Ribas, D., "Towards a Real-Time Vision-Based Navigation System for a Small-Class UUV," *Conference on Intelligent Robots and Systems*, Vol. 1, 2003, pp. 818–823.

[31]Garcia, R., Puig, J., Ridao, P., and Cufi, X., "Augmented State Kalman Filtering for AUV Navigation," *IEEE Proceedings on International Conference Robotics and Automation*, Vol. 4, 2002, pp. 4010– 4015.

[32]Davison, A.J., Reid, I.D. and Molton, N.D., "MonoSLAM: Real-Time Single Camera SLAM," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 29, No. 6, 2007.

[33]Bryson, M. and Sukkarieth, S., "Active Airborne Localization and Exploration in Unknown Environments using Inertial SLAM," *IEEE Aerospace Conference*, 2006.

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

[35]Kim, J. and Sukkarieth, S., "6DoF SLAM aided GNSS/INS Navigation in GNSS Denied and Unknown Environments," *Journal of Global Positioning Systems*, Vol. 4, No. 1-2, pp. 120–128, 2005.

[36]Lowe, D., "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, Vol. 60, No. 2, November 2004, pp. 91–110.

[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, January 2002, pp. 79–99.

[38]http://earth.google.com/index.html.

[39]Vedaldi, A., "An open implementation of the SIFT detector and descriptor," *UCLA CSD Technical Report 070012*, 2007.