Distributed Vision-Aided Cooperative Localization and Navigation based on Three-View Geometry

Vadim Indelman, Pini Gurfil, Ehud Rivlin and Hector Rotstein

Abstract—This paper presents a new method for distributed vision-aided cooperative localization and navigation for multiple autonomous platforms based on constraints stemming from the three-view geometry of a general scene. Each platform is assumed to be equipped with a standard inertial navigation system and an on-board, possibly gimbaled, camera. The platforms are also assumed to be capable of intercommunicating. No other sensors, or any a priori information is required. In contrast to the traditional approach for cooperative localization that is based on relative pose measurements, the proposed method formulates a measurement whenever the same scene is observed by different platforms. Each such measurement is constituted upon three images, which are not necessarily captured at the same time. The captured images, attached with some navigation parameters, are stored in repositories by each, or some, of the platforms in the group. A graph-based approach is applied for calculating the correlation terms between the navigation parameters associated to images participating in the same measurement. The proposed method is examined using a statistical simulation in a leader-follower scenario, and is demonstrated in an experiment that involved two vehicles in a holding pattern scenario.

1. Introduction

Cooperative localization and navigation has been an active research field for over a decade. Vast research efforts have been devoted to develop methods allowing a group of platforms to autonomously perform different missions. These missions include cooperative mapping and localization [1], [2], [3], formation flying [4], military tasks, such as cooperative tracking [5], autonomous multi-vehicle transport [6], and other applications. Precise navigation is a key requirement for carrying out any autonomous mission by a group of cooperative platforms.

Assuming the GPS is available, each of the platforms is capable of computing the navigation solution on its own, usually by fusing the GPS data with an inertial navigation system (INS). However, whenever the GPS signal is absent, such as when operating indoor, underwater, in space, in urban environments, or on other planets, alternative methods should be devised for updating the navigation system, due to the evolving dead reckoning errors.

While various methods exist for navigation-aiding of a single platform, collaboration among several, possibly heterogeneous, platforms, each equipped with its own set of sensors, is expected to improve performance even further [7]. Indeed, different methods have been developed for effectively localizing a group of platforms, with respect to some static coordinate system, or with respect to themselves. Typically, these methods assume that each platform is capable of measuring the relative range and bearing to other platforms located close enough.

One of the pioneering works on cooperative localization is [8], in which it was proposed to restrain the development of navigation errors by using some of the robots, as static landmarks, for updating the other robots in the group, and then switching between the roles after a certain amount of time. While the method was further improved by others, all of the derived methods share the same drawback of having to stop the motion of some of the robots for updating the others, which is, for example, not possible for fixed-wing aerial platforms.

Another important work is by Roumeliotis and Bekey [7], where a centralized approach for sensors fusion was applied based on the available relative pose measurements between the platforms in the group. This architecture was then de-centralized and distributed among the platforms. In [9] an extension was proposed to handle more general relative observation models. The setup of having a group of platforms capable of measuring relative poses to adjacent platforms have been studied also in other works, including [6], [10], [11], [12] and [13].

A different body of works [14], [15], [16] suggests to maintain in each platform an estimation of parameters for all the platforms in the group. For example, in [14] and [15] each platform estimates the pose of *every* other platform relative to itself, while in [16] each platform estimates the navigation state (position, velocity and attitude) of all the other platforms by exchanging IMU information and relative pose measurements.

V. Indelman and P. Gurfil are with the Faculty of Aerospace Engineering, Technion - Israel Institute of Technology, Haifa 32000, Israel (e-mail: ivadim@tx.technion.ac.il, pgurfil@technion.ac.il).

E. Rivlin is with the Department of Computer Science, Technion - Israel Institute of Technology, Haifa 32000, Israel (e-mail: ehudr@cs.technion.ac.il).

H. Rotstein is with RAFAEL - Advanced Defense Systems Limited, Israel (e-mail: hector@rafael.co.il).

This work proposes to utilize vision-based navigation-aiding techniques for cooperative navigation. Each platform is assumed to be equipped with a standard INS and a camera. No further sensors or a priori information is assumed. Several works [17], [18] on cooperative navigation have been published relying on this setup. In [17], a leader-follower formation of mobile ground vehicles is considered, in which each vehicle is equipped with a camera that provides bearing to the other vehicles. It is shown that it is possible to estimate the relative pose between the followers and the leader, except for configurations that are not observable. The authors of [18] propose to estimate the relative location of two moving ground robots by fusing camera bearing measurements with the robots' odometry.

The method proposed herein relies on a recently-developed technique [19], [20] for vision-aided navigation of a single platform based on three-view geometry of a general scene. Cooperative navigation is a natural extension of this technique. As opposed to all the methods for cooperative localization and navigation mentioned above, the platform camera is *not* required to be aimed towards other platforms for obtaining measurements of relative parameters (bearing, range) between the platforms. Instead, a measurement is formulated whenever the same scene is observed by different platforms.

A similar concept has been already proposed in [3] and [21], considering measurements that combine pairs of platforms. In [21], a homography matrix is estimated whenever two airborne platforms capture images of the same scene, which is assumed to be planar. Next, the relative motion between the two platforms is extracted from the homography matrix, and assuming the position of the first platform, and its height above ground level are known, the estimated relative motion is then used for updating the position of the second platform. Kim et. al. [3] perform nonlinear optimization, involving the pose history of all the platforms in the group, each time a new measurement arrives, considering relative pose and two-view measurements between pairs of platforms.

In this work, three views of a common general scene are required for each measurement. These views are acquired by at least two different platforms, i. e. either each view is captured by a different platform, or two of the three views are captured by the same platform. The scenario in which all the views were captured by the same platform is handled in [19], [20]. The constraints stemming from a general scene, observed in three different views [19], [20], allow to reduce the developed navigation errors of the updated platform without any further a priori information or any other sensors (such as height above ground level, range sensor, etc.).

Another key aspect of the proposed method, is that the three images of the same region are *not* necessarily captured at the same time. All, or some, of the platforms maintain a local repository of captured images that are associated with some navigation parameters [19]. These repositories are accessed on demand to check if a region, currently observed by one of the platforms, denoted as the querying platform, has been observed in the past by other platforms in the group. Images containing the same region are transmitted, with the attached navigation data, to the querying platform. The received information from other platforms, in addition to the navigation and imagery data of the querying platform, allow updating the querying platform navigation system.

The navigation information participating in the update process may be statistically dependent. Therefore, in order to obtain a consistent navigation solution, the involved correlation terms should be either maintained, or calculated upon demand. However, maintaining the correlation terms is impractical since the time instances of the images participating in the measurements are a priori unknown, in addition to the a priori unknown identity of the platforms that captured these images. Thus, one has either to neglect the involved correlation terms, or to calculate these upon request. This paper follows the latter approach, adjusting a graph-based technique for calculating correlation for general multi-platform measurement models [20] to the three-view measurement model used herein.

2. Method Overview

Fig. 1 shows the overall concept of the proposed method for multi-platform vision-aided navigation. The proposed method assumes a group of cooperative platforms capable of communicating between themselves. Each platform is equipped with a standard inertial navigation system (INS) and an onboard camera, which may be gimbaled. Some, or all, of the platforms maintain a local repository comprised of images captured along the mission. These images are attached with navigation data when they are captured. The INS is comprised of an inertial measurement unit (IMU) whose measurements are integrated into a navigation solution.

In a typical scenario, a platform captures an image and broadcasts it, along with its current navigation solution, to other platforms in the group, inquiring if they have previously captured images containing the same region. Upon receiving such a query, each platform performs a check in its repository looking for appropriate images. Among these images, only images with a smaller navigation uncertainty compared to the uncertainty in the navigation data of the query image, are transmitted back. Platforms that do not maintain a repository, perform the check only on the currently-captured image.

The process of the querying platform is schematically described in Fig. 1. After receiving the images and the attached navigation data from other platforms in the group, two best images are chosen and, together with the querying image, are used for formulating the three-view constraints (Section 3). These constraints are then reformulated into a measurement and are used for updating the navigation system of the querying platform, as described in Section 4. Since the navigation data attached to the chosen three images may be correlated, a graph-based technique is applied for calculating the required cross-covariance terms for the fusion process. This technique is discussed in Section 5. The overall protocol for information sharing among the platforms in the group is discussed in Section 6.

Throughout this paper, the following coordinate systems are used:

• 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 a Cartesian right hand system.
• C - Camera-fixed reference frame. Its origin is set at the camera center-of-projection. Z_C points toward the FOV center, X_C points toward the right half of the FOV when viewed from the camera center-of-projection, and Y_C completes the setup to yield a Cartesian right hand system.

3. Three-View Geometry Constraints

Assume some general scene is observed in three different views, captured by different platforms. Fig. 2 depicts such a scenario, in which a static landmark $\mathbf{p}$ is observed in three images I_1, I_2 and I_3. The image I_3 is the currently-captured image of the third platform, while I_1 and I_2 are two images captured by the first two platforms. These two images may be the currently-captured images of these platforms, but they could also be captured in the past and stored in the repository of each platform, as is indeed illustrated in the figure. Alternatively, I_1 and I_2 could also be captured by the same platform.

Denote by $\mathbf{T}_{ij}$ the translation vector from the ith to the jth view, with $i, j \in \{1, 2, 3\}$ and $i \neq j$. Let also $\mathbf{q}_i$ and λ_i be a line of sight (LOS) vector and a scale parameter, respectively, to the landmark $\mathbf{p}$ at time t_i, such that $\|\lambda_i \mathbf{q}_i\|$ is the range to this landmark. In particular, if $\mathbf{q}_i$ is a unit LOS vector, then λ_i is the range to $\mathbf{p}$. The constraints stemming from three views observing the same landmark can be formulated as follows [19], [20]:

$$\mathbf{q}_1^T(\mathbf{T}_{12} \times \mathbf{q}_2) = 0 \tag{1a}$$
$$\mathbf{q}_2^T(\mathbf{T}_{23} \times \mathbf{q}_3) = 0 \tag{1b}$$
$$(\mathbf{q}_2 \times \mathbf{q}_1)^T(\mathbf{q}_3 \times \mathbf{T}_{23}) = (\mathbf{q}_1 \times \mathbf{T}_{12})^T(\mathbf{q}_3 \times \mathbf{q}_2) \tag{1c}$$

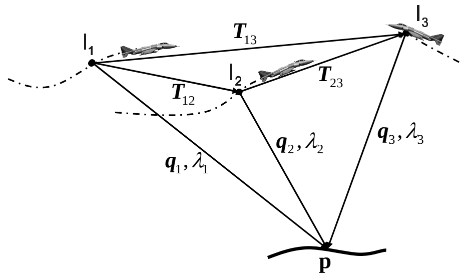


Figure 2. Three-view geometry: a static landmark $\mathbf{p}$ observed by three different platforms. Images I_1 and I_2 were captured by the first two platforms in the past, while image I_3, is the currently-acquired image by the third platform.

All the parameters in Eqs. (1) should be expressed in the same coordinate system using the appropriate rotation matrices taken from navigation systems of the involved platforms. It is assumed that this coordinate system is the LLLN system of the platform that captured the second image at t_2.

Recall that Eqs. (1a) and (1b) are the well-known epipolar constraints, forcing the translation vectors to be coplanar with the LOS vectors. Thus, given multiple matching features, the translation vectors $\mathbf{T}_{12}$ and $\mathbf{T}_{23}$ can be determined only up to scale. The two scale parameters are different in the general case. Eq. (1c) relates between these two scale parameters, thereby allowing to calculate one of the translation vectors given the other [19].

In practice, the three views will have more than a single landmark in common. Therefore, three different sets of matching LOS vectors are defined: A set of matching pairs of features between the first and second view, another set of matching pairs of features between the second and third view, and a set of matching triplets between all the three views. All the LOS vectors are expressed in an LLLN system, as mentioned above. These sets are denoted by $\{\mathbf{q}_{1_i}, \mathbf{q}_{2_i}\}_{i=1}^{N_{12}}, \{\mathbf{q}_{2_i}, \mathbf{q}_{3_i}\}_{i=1}^{N_{23}}$ and $\{\mathbf{q}_{1_i}, \mathbf{q}_{2_i}, \mathbf{q}_{3_i}\}_{i=1}^{N_{123}}$, respectively, where N_{12}, N_{23} and N_{123} are the number of matching features in each set, and $\mathbf{q}_{j_i}$ is the ith LOS vector in the jth view, $j \in (1, 2, 3)$.

Taking into consideration all the matching pairs and triplets, the constraints (1) turn into [19], [20]:

$$\begin{bmatrix} U \\ F \\ 0 \end{bmatrix}_{N\times 3} \mathbf{T}_{23} = \begin{bmatrix} W \\ 0 \\ G \end{bmatrix}_{N\times 3} \mathbf{T}_{12} \tag{2}$$

where $N \doteq N_{12} + N_{23} + N_{123}$ and

$$U = \begin{bmatrix} \mathbf{u}_1 & \dots & \mathbf{u}_{N_{123}} \end{bmatrix}^T \quad W = \begin{bmatrix} \mathbf{w}_1 & \dots & \mathbf{w}_{N_{123}} \end{bmatrix}^T$$
$$F = \begin{bmatrix} \mathbf{f}_1 & \dots & \mathbf{f}_{N_{23}} \end{bmatrix}^T \quad G = \begin{bmatrix} \mathbf{g}_1 & \dots & \mathbf{g}_{N_{12}} \end{bmatrix}^T$$

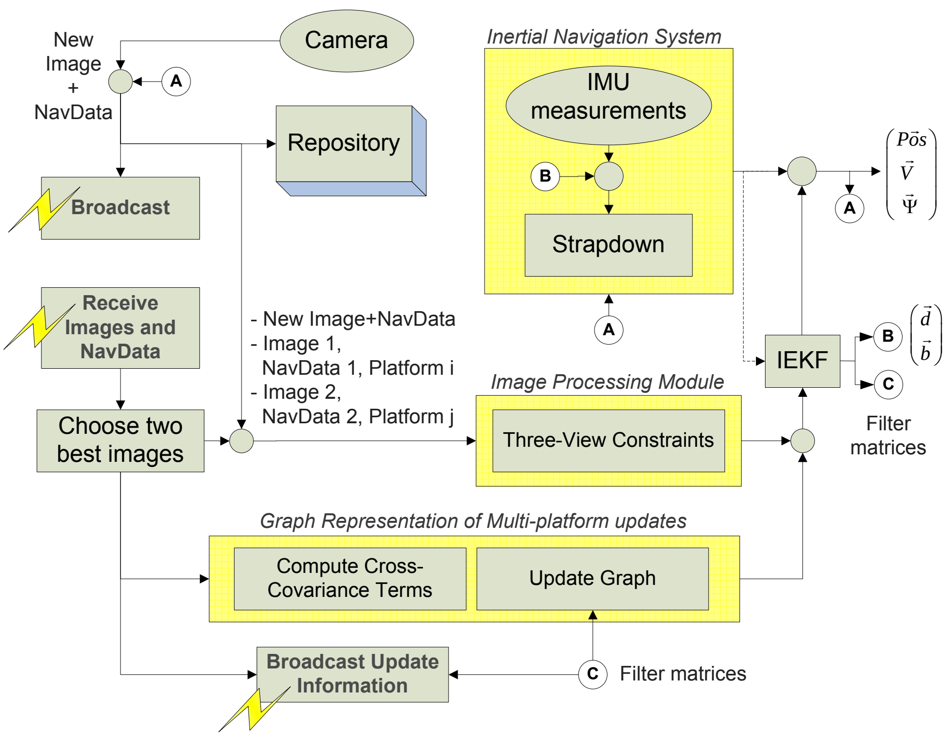

Camera
New Image + NavData
A
Repository
Broadcast
Receive Images and NavData
Choose two best images
- New Image+NavData
- Image 1, NavData 1, Platform i
- Image 2, NavData 2, Platform j
Inertial Navigation System
IMU measurements
B
Strapdown
A
$\vec{Pos}$, $\vec{V}$, $\vec{\Psi}$
IEKF
B
$\vec{d}$, $\vec{b}$
C
Filter matrices
Image Processing Module
Three-View Constraints
Graph Representation of Multi-platform updates
Compute Cross-Covariance Terms
Update Graph
Broadcast Update Information
C
Filter matrices

Figure 1. Multi-platform navigation aiding - querying platform scheme

while the vectors $\mathbf{f}, \mathbf{g}, \mathbf{u}, \mathbf{w} \in \mathbb{R}^{3\times 1}$ are defined as

$$\begin{aligned}
\mathbf{f}^T &\doteq (\mathbf{q}_2 \times \mathbf{q}_3)^T \\
\mathbf{g}^T &\doteq (\mathbf{q}_1 \times \mathbf{q}_2)^T \\
\mathbf{u}^T &\doteq (\mathbf{q}_1 \times \mathbf{q}_2)^T [\mathbf{q}_3]_\times = \mathbf{g}^T [\mathbf{q}_3]_\times \\
\mathbf{w}^T &\doteq (\mathbf{q}_2 \times \mathbf{q}_3)^T [\mathbf{q}_1]_\times = \mathbf{f}^T [\mathbf{q}_1]_\times
\end{aligned}$$

4. Three-View Based Navigation Update

Define the state vector of each platform to be its own navigation errors and IMU error parameterization:

$$\mathbf{X} = \begin{bmatrix} \Delta\mathbf{P}^T & \Delta\mathbf{V}^T & \Delta\mathbf{\Psi}^T & \mathbf{d}^T & \mathbf{b}^T \end{bmatrix}^T$$

where $\Delta\mathbf{P} \in \mathbb{R}^3, \Delta\mathbf{V} \in \mathbb{R}^3, \Delta\mathbf{\Psi} = (\Delta\phi, \Delta\theta, \Delta\psi)^T \in [0, 2\pi] \times [0, \pi] \times [0, 2\pi]$ are the position, velocity and attitude errors, respectively, and $(\mathbf{d}, \mathbf{b})$ is the parameterization of errors in the inertial sensor measurements: $\mathbf{d} \in \mathbb{R}^3$ is the gyro drift, and $\mathbf{b} \in R^3$ 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.

The corresponding transition matrix $\Phi^i_{t_a \to t_b}$ satisfying

$$\mathbf{X}_i(t_b) = \Phi^i_{t_a \to t_b} \mathbf{X}_i(t_a) + \boldsymbol{\omega}^i_{t_a:t_b} \tag{3}$$

for the ith platform is given in [22]. Here $\boldsymbol{\omega}_{t_a:t_b}$ is a discrete process noise.

When real navigation and imagery data is considered, the constrains in Eq. (2) will not be satisfied. Thus, a residual measurement is defined as

$$\mathbf{z} \doteq \begin{bmatrix} U \\ F \\ 0 \end{bmatrix}_{N\times 3} \mathbf{T}_{23} - \begin{bmatrix} W \\ 0 \\ G \end{bmatrix}_{N\times 3} \mathbf{T}_{12} \doteq \mathcal{A}\mathbf{T}_{\in\ni} - \mathcal{B}\mathbf{T}_{\infty\in}$$

Noting that $\mathbf{T}_{12} = \mathbf{Pos}_2 - \mathbf{Pos}_1$, $\mathbf{T}_{23} = \mathbf{Pos}_3 - \mathbf{Pos}_2$, and the matrices F, G, U, W are functions of the LOS vectors, the residual measurement $\mathbf{z}$ is a nonlinear function of the following parameters[3]:

$$\mathbf{z} = \mathbf{h}\left(\mathbf{Pos}_3, \mathbf{\Psi}_3, \mathbf{Pos}_2, \mathbf{\Psi}_2, \mathbf{Pos}_1, \mathbf{\Psi}_1, \left\{\mathbf{q}^{C_1}_{1_i}, \mathbf{q}^{C_2}_{2_i}, \mathbf{q}^{C_3}_{3_i}\right\}\right) \tag{4}$$

Linearizing Eq. (4) we obtain

$$\mathbf{z} \approx H_3\mathbf{X}_3 + H_2\mathbf{X}_2 + H_1\mathbf{X}_1 + D\mathbf{v} \tag{5}$$

where the Jacobian matrices H_3, H_2, H_1 and D in the above equation are given in [20], and $\mathbf{X}_i$ is the state vector of the appropriate platform at the capture-time of the ith image. This state vector models the errors in the navigation data attached to this image. The term $D\mathbf{v}$ represents contribution of image noise from all the three images and of linearization errors to the measurement $\mathbf{z}$.

[3] In Eq. (4), the notation $\left\{\mathbf{q}^{C_1}_{1_i}, \mathbf{q}^{C_2}_{2_i}, \mathbf{q}^{C_3}_{3_i}\right\}$ refers to the fact that LOS vectors from all the three images are used for calculating the residual measurement $\mathbf{z}$. Note that each of the matrices F, G, U, W is a function of a different set of matching points [20].

As can be seen, the residual measurement is a function of all the three state vectors, which in the general case may be correlative. Thus, all the involved platforms in the measurement can be updated.

This is indeed an approach used in some works (e. g. [7]), in which the measurement is a function of the navigation parameters from *current* time of several platforms. Since navigation parameters only from the current time participate in the measurement, it is possible to maintain the cross-covariance terms between the platforms in the group. Assuming M platforms in the group, and an $m \times m$ covariance matrix P_i for each platform i, the total covariance matrix of the group, containing also all the cross-covariance terms among platforms in the group is an $Mm \times Mm$ matrix

$$\mathcal{P}_{\text{Total}} = \begin{bmatrix} P_1 & P_{12} & \cdots & P_{1M} \\ P_{21} & P_2 & \cdots & P_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ P_{M1} & P_{M2} & \cdots & P_M \end{bmatrix}$$

where $P_i = E[\tilde{\mathbf{X}}_i \tilde{\mathbf{X}}_i^T]$ and $P_{ij} = E[\tilde{\mathbf{X}}_i \tilde{\mathbf{X}}_j^T]$. The matrix $\mathcal{P}_{\text{Total}}$ may be efficiently calculated in a distributed framework [7].

Yet, the measurement model in the method proposed herein, Eq. (5), involves data from different platforms and from different, and *a priori unknown* time instances. Maintaining a total covariance matrix $\mathcal{P}_{\text{Total}}$ containing the covariance for every platform and cross-covariance terms between each pair of platforms in the group, for the *whole* mission duration is not a practical solution. Thus, an alternative technique should be used. The proposed technique, described in Section 5, represents the platforms update events in a directed acyclic graph, locally maintained by every platform in the group. Given a measurement, the relevant cross-covariance terms are computed based on this graph representation, going back and forth on the time domain according to the time instances involved in the measurement. This technique is developed in [20] for a general multi-platform measurement model, and is adopted in this work for three-view measurements.

The graph needs to be acyclic, since otherwise, a measurement might trigger recursive updates in past measurements. In a general scenario involving three-view measurements between different platforms and different time instances, the graph is guaranteed to be acyclic if only platforms that contributed their *current* (and not past) image and navigation data are updated. For simplicity, in this paper we consider only one such platform, which is the querying platform, i. e. the platform that broadcasted the query image to the rest of the platforms in the group. Moreover, without loss of generality, it is assumed that the querying platform captures the *third* image, as illustrated in Fig. 2.

An implicit extended Kalman filter (IEKF) is applied, whereby the Kalman gain matrix is computed as [20] $K = P_{\mathbf{X}_3\mathbf{z}} P_{\mathbf{z}}^{-1}$, where

$$P_{\mathbf{X}_3\mathbf{z}} = P_3^- H_3^T + P_{32}^- H_2^T + P_{31}^- H_1^T \tag{6}$$

$$\begin{aligned} P_{\mathbf{z}} &= H_3 P_3^- H_3^T + \\ &+ \begin{bmatrix} H_2 & H_1 \end{bmatrix} \begin{bmatrix} P_2 & P_{21} \\ P_{21}^T & P_1 \end{bmatrix} \begin{bmatrix} H_2 & H_1 \end{bmatrix}^T + DRD^T \end{aligned} \tag{7}$$

As can be seen from Eqs. (6) and (7), the cross-covariance terms P_{21}, P_{31} and P_{32} indeed participate in the update process, and therefore need to be calculated.

The update equations are the IEKF standard equations. In particular, the a posteriori estimation error of the querying platform is given by:

$$\begin{aligned} \tilde{\mathbf{X}}_3^+ &= [I - K_3 H_3]\tilde{\mathbf{X}}_3^- - K_3 H_2 \tilde{\mathbf{X}}_2^- - \\ &- K_3 H_1 \tilde{\mathbf{X}}_1^- - K_3 D\mathbf{v} \end{aligned} \tag{8}$$

where $\tilde{\mathbf{X}}$ denotes the estimation error of $\mathbf{X}$.

It is worth mentioning that there are specific scenarios, in which *all* the participating platforms in the measurement may be updated, since it is guaranteed that the graph will always be acyclic. In these scenarios, the filter formulation changes as described next. An example of such a scenario is given in Section 7.

All the Involved Platforms are Updated

The following augmented state vector is defined:

$$\mathcal{X} \doteq \begin{bmatrix} \mathbf{X}_3^T & \mathbf{X}_2^T & \mathbf{X}_1^T \end{bmatrix}^T$$

with an augmented covariance matrix $\mathcal{P} \doteq \mathcal{E}[\mathcal{X}\mathcal{X}^{\mathcal{T}}]$. The a posteriori estimation errors of the state vectors in $\mathcal{X}$ are:

$$\begin{aligned} \tilde{\mathbf{X}}_3^+ &= \left[I - \breve{K}_3 H_3\right]\tilde{\mathbf{X}}_3^- - \breve{K}_3 H_2 \tilde{\mathbf{X}}_2^- - \breve{K}_3 H_1 \tilde{\mathbf{X}}_1^- - \breve{K}_3 D\mathbf{v} \\ \tilde{\mathbf{X}}_2^+ &= \left[I - \breve{K}_2 H_2\right]\tilde{\mathbf{X}}_2^- - \breve{K}_2 H_3 \tilde{\mathbf{X}}_3^- - \breve{K}_2 H_1 \tilde{\mathbf{X}}_1^- - \breve{K}_2 D\mathbf{v} \\ \tilde{\mathbf{X}}_1^+ &= \left[I - \breve{K}_1 H_1\right]\tilde{\mathbf{X}}_1^- - \breve{K}_1 H_3 \tilde{\mathbf{X}}_3^- - \breve{K}_1 H_2 \tilde{\mathbf{X}}_2^- - \breve{K}_1 D\mathbf{v} \end{aligned}$$

while the augmented covariance matrix is updated according to

$$\mathcal{P}^+ = [\mathcal{I} - \mathcal{K}\mathcal{H}]\mathcal{P}^- [\mathcal{I} - \mathcal{K}\mathcal{H}]^{\mathcal{T}} + [\mathcal{K}\mathcal{D}]\mathcal{R}[\mathcal{K}\mathcal{D}]^{\mathcal{T}}$$

where $\mathcal{H} \doteq \begin{bmatrix} H_3 & H_2 & H_1 \end{bmatrix}$ and $\mathcal{K} \doteq \begin{bmatrix} \breve{K}_3^T & \breve{K}_2^T & \breve{K}_1^T \end{bmatrix}^{\mathcal{T}}$. The augmented Kalman gain matrix $\mathcal{K}$ is computed as $\mathcal{K} = \mathcal{P}^- \mathcal{H}^{\mathcal{T}} \left(\mathcal{H}\mathcal{P}^- \mathcal{H}^{\mathcal{T}} + \mathcal{R}\right)^{-\infty}$.

5. Cross-Covariance Calculation

In this section we discuss how the cross-covariance terms can be calculated upon demand. Recall that it is unknown a priori what platforms and which time instances will participate in each three-view measurement. First, the development of expressions for the cross-covariance terms is presented in a simple scenario. Next, the concept of a graph-based technique for automatic calculation of these terms in general scenarios is discussed.

Simple Scenario

Consider the scenario shown in Fig. 3. Platform III is the querying platform, and therefore only this platform is updated. Two three-view updates are performed. In each of these updates, the first two images are transmitted by platform I. For example, the first measurement is formulated using images and navigation data denoted by a_1, a_2 and a_3, where a_1, a_2 are obtained from platform I.

It is assumed that the platforms do not apply any updates from other sources. As shall be seen, these restrictions are not necessary, and are undertaken here for clarifying the basics of the approach.

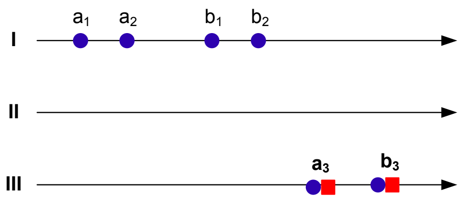


Figure 3. Measurement schedule example. Platform III is updated based on images transmitted by platform I. The filled circle marks denote images participating in the measurement, square marks indicate update events.

The cross-covariance terms are computed in the following recursive way. Assume the first measurement, comprised of a_1, a_2 and a_3 was carried out, and that the a priori and a posteriori covariance and cross-covariance terms are known. Now, it is required to calculate the cross-covariance terms $E[(\tilde{\mathbf{X}}^-_{b_3})(\tilde{\mathbf{X}}^-_{b_2})^T]$, $E[(\tilde{\mathbf{X}}^-_{b_3})(\tilde{\mathbf{X}}^-_{b_1})^T]$ and $E[(\tilde{\mathbf{X}}^-_{b_2})(\tilde{\mathbf{X}}^-_{b_1})^T]$ for performing the second three-view update.

The following equations may be written regarding the state propagation:

$$\begin{aligned} \tilde{\mathbf{X}}^-_{b_3} &= \Phi_{a_3\to b_3}\tilde{\mathbf{X}}^+_{a_3} + \boldsymbol{\omega}_{a_3:b_3} \\ \tilde{\mathbf{X}}^-_{b_i} &= \Phi_{a_2\to b_i}\tilde{\mathbf{X}}^-_{a_2} + \boldsymbol{\omega}_{a_2:b_i} \ , \ \ i = 1,2 \end{aligned}$$

where $\boldsymbol{\omega}_{i:j}$ is the equivalent process noise between the time instances t_i and t_j of the appropriate platform[4]. The cross-covariance terms $E[(\tilde{\mathbf{X}}^-_{b_3})(\tilde{\mathbf{X}}^-_{b_i})^T]$ with $i = 1,2$, may be calculated as:

$$E\left[\left(\Phi_{a_3\to b_3}\tilde{\mathbf{X}}^+_{a_3} + \boldsymbol{\omega}_{a_3:b_3}\right)\left(\Phi_{a_2\to b_i}\tilde{\mathbf{X}}^-_{a_2} + \boldsymbol{\omega}_{a_2:b_i}\right)^T\right] \tag{9}$$

The a posteriori estimation error $\tilde{\mathbf{X}}^+_{a_3}$ is given, according to Eq. (8), by:

$$\begin{aligned} \tilde{\mathbf{X}}^+_{a_3} &= (I - K_{a_3}H_{a_3})\tilde{\mathbf{X}}^-_{a_3} - K_{a_3}H_{a_2}\tilde{\mathbf{X}}^-_{a_2} - \\ &- K_{a_3}H_{a_1}\tilde{\mathbf{X}}^-_{a_1} - K_{a_3}\mathcal{D}_{\dashv}\mathbf{v}_{\dashv} \end{aligned} \tag{10}$$

Since $\boldsymbol{\omega}_{a_2:b_2}$ is statistically independent of $\tilde{\mathbf{X}}^-_{a_3}, \tilde{\mathbf{X}}^-_{a_2}, \tilde{\mathbf{X}}^-_{a_1}$, and since $\boldsymbol{\omega}_{a_3:b_3}$ is statistically independent of $\tilde{\mathbf{X}}^-_{a_2}$ and $\boldsymbol{\omega}_{a_2:b_2}$ (cf. Fig. 3)

$$\begin{aligned} &E\left[\tilde{\mathbf{X}}^+_{a_3}\boldsymbol{\omega}^T_{a_2:b_2}\right] = 0 \\ &E\left[\boldsymbol{\omega}_{a_3:b_3}\left(\Phi_{a_2\to b_2}\tilde{\mathbf{X}}^-_{a_2} + \boldsymbol{\omega}_{a_2:b_2}\right)^T\right] = 0 \end{aligned}$$

Denoting $P_{ab} \doteq E[(\tilde{\mathbf{X}}_a)(\tilde{\mathbf{X}}_b)^T]$ and incorporating the above into Eq. (9) yields

$$\begin{aligned} P^-_{b_3b_2} &= \Phi_{a_3\to b_3}\left\{(I - K_{a_3}H_{a_3})P^-_{a_3a_2} - \right. \\ &\left. -K_{a_3}H_{a_2}P^-_{a_2a_2} - K_{a_3}H_{a_1}P^-_{a_1a_2}\right\}\Phi^T_{a_2\to b_2} \end{aligned}$$

where the expectation terms involving a_3, a_2, a_1 are known (from previous update). In a similar manner we get

$$\begin{aligned} P^-_{b_3b_1} &= \Phi_{a_3\to b_3}\left\{(I - K_{a_3}H_{a_3})P^-_{a_3a_2} - \right. \\ &\left. -K_{a_3}H_{a_2}P^-_{a_2a_2} - K_{a_3}H_{a_1}P^-_{a_1a_2}\right\}\Phi^T_{a_2\to b_1} \end{aligned} \tag{11}$$

while $P^-_{b_2b_1} \doteq E[(\tilde{\mathbf{X}}^-_{b_2})(\tilde{\mathbf{X}}^-_{b_1})^T]$ is given by

$$P^-_{b_2b_1} = \Phi_{b_1\to b_2}P^-_{b_1b_1}$$

Graph-Based Cross-Covariance Calculation

The required cross-covariance terms may be calculated using a graph representing the history of the applied three-view measurements. Such a method is presented in [20] for a general multi-platform measurement model and it can be easily adjusted for handling the three-view measurement model discussed in this paper. Thus, this section describes only the concept of the method.

A directed acyclic graph (DAG) $G = (V, E)$ is locally maintained by every platform in the group, where V is the set of nodes and E is the set of directed arcs connecting between the nodes in V. Two type of nodes exist in this graph: nodes that represent images and the attached navigation data that participated in some multi-platform update, and update-event nodes. The nodes are connected by directed weighted arcs. The weight of each arc reflects the information flow between the two connected nodes. Each node in G, can be connected to

[4]From now on, explicit notations of platform identities and time instances are omitted for conciseness, since these may be concluded by context.

another node by a transition relation, and in addition, it may be involved in a three-view measurement, in which case it would be also connected to an update-event node by an update relation.

The transition relation is given by Eq. (3), relating between the state vectors of the ith platform at two different time instances t_a and t_b. G will contain two nodes, representing these two time instances, only if each of them participates in some three-view measurement. In this case, these two nodes will be connected by an arc, weighted by the transition matrix $\Phi^i_{t_a \rightarrow t_b}$. The noise process covariance matrix $Q^i_{t_a:t_b} \doteq E[\boldsymbol{\omega}^i_{t_a:t_b}(\boldsymbol{\omega}^i_{t_a:t_b})^T]$ is associated to this arc as well.

The update relation is given by Eq. (8):

$$\tilde{\mathbf{X}}_3^+ = [I - K_3 H_3]\,\tilde{\mathbf{X}}_3^- - K_3 H_2 \tilde{\mathbf{X}}_2^- - K_3 H_1 \tilde{\mathbf{X}}_1^- - K_3 D \mathbf{v}$$

Thus, G will contain 4 nodes representing the above equation. Let the nodes β_i represent the a priori estimation errors $\tilde{\mathbf{X}}_i^-$, with $i = 1, 2, 3$, and the node α represent the a posteriori estimation error $\tilde{\mathbf{X}}_3^+$. Then, the arc weights connecting the nodes β_1, β_2 and β_3 with the node α are $-K_3H_1$, $-K_3H_2$ and $I - K_3H_3$, respectively. Each such arc is also associated with the relevant measurement noise covariance matrix [20].

It is assumed that the a priori and a posteriori covariance and cross-covariance terms between the nodes participated in the *same* multi-platform update, that has been already carried out, are known (e. g. this information may be stored in the nodes themselves).

Consider, for example, the simple scenario discussed in the previous section (cf. Fig. 3). The equivalent graph is given in Fig. 4(a). As seen, two update events are carried out, both on platform III. At each update, the first two images of the three are transmitted by platform I, while the third image is the currently-captured image by the querying platform III. Platform II has not transmitted any images and therefore has no nodes in the graph. The transmission action is denoted by a dashed arc in the graph. Nodes of the first type are denoted as circle nodes, while the second-type nodes are designated by a square notation. The arc weights are not explicitly specified in the graph (for clarity of representation). For example, the weight of an arc connecting between the nodes a_1^- and a_2^- is the transition matrix $\phi_{a_1 \rightarrow a_2}$, since no measurement updates were performed between these two time instances. On the other hand, the arcs connecting a_1^-, a_2^- and a_3^- to a_3^+ are weighted, according to Eq. (10), as $-K_{a_3}H_{a_1}$, $-K_{a_3}H_{a_2}$ and $I - K_{a_3}H_{a_3}$, respectively. In addition, each arc is also associated with the appropriate noise covariance, as mentioned above.

The construction process of the graph, as well as the communication protocol between the platforms in the group, is delayed until Section 6.

The graph representation suggests a convenient approach for computing the correlation terms. Assume we need to calculate the cross-covariance between some two nodes c and d in the graph, representing $\tilde{\mathbf{X}}_c$ and $\tilde{\mathbf{X}}_d$, respectively. The first step is to construct two inverse-trees, containing all the possible routes in the graph G to each of the nodes c and d. This is performed as follows. The first tree, T_c, is initialized with the node c. Each next level is comprised of the parents of nodes that reside in the previous level, as determined from the graph G. Thus, for example, the second level of T_c contains all the nodes in G that are directly connected to c. The same process is executed for constructing a tree T_d for the node d. Fig. 4(b) shows an example of two trees with $c \doteq b_3^-$ and $d \doteq b_1^-$, constructed based on the graph, given in Fig. 4(a), for calculating the cross-covariance $P^-_{b_3b_1}$. This term and the terms $P^-_{b_3b_2}, P^-_{b_2b_1}$ are required for carrying out the measurement update b_3^+.

Note that each node in T_c and T_d has only one child but may have one or three parents. In the latter case, the node represents an update event.

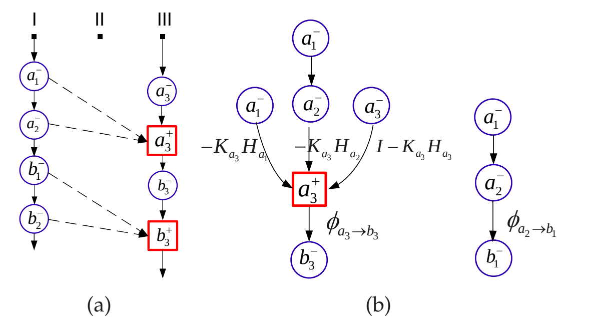


Figure 4. (a) Graph representation for the scenario shown in Fig. 3. (b) The two inverse-trees $T_{b_3^-}$ and $T_{b_1^-}$ required for calculating $P^-_{b_3b_1}$.

The concept of the proposed graph-based approach for calculating the cross-covariance terms is as follows. We start with the two nodes c and d, which are the first-level nodes in the trees T_c and T_d, respectively. Since the term $E[\tilde{\mathbf{X}}_c \tilde{\mathbf{X}}_d^T]$ is unknown, we proceed to the parents of these nodes. As noted above, two types of relations exist for a general graph topology. At this point it is assumed that both of the nodes c and d are related to the nodes in the next level by a transition relation, and therefore have only one parent. This assumption is made only for clarity of explanation[5]. Denote the parents of c and d as c_2 and d_2, respectively. The nodes c_2 and d_2 constitute the second level in the trees T_c and T_d, respectively. For example, c_2 and c are connected via $\tilde{\mathbf{X}}_c = \Phi_{c_2 \rightarrow c}\tilde{\mathbf{X}}_{c_2} + \boldsymbol{\omega}_{c_2 \rightarrow c}$.

[5]In practice, c and d will usually represent images that are going to participate in a three-view update event, and therefore c and d will indeed have only one parent each.

The convention used here is that if some node a_i has several parents, the jth parent is denoted as a_{i+1}^j. Also, $a \equiv a_1$.

Now, the required cross-covariance term $P_{cd} \doteq E[\tilde{\mathbf{X}}_c \tilde{\mathbf{X}}_d^T]$ may be written in several forms:

$$E\left[\tilde{\mathbf{X}}_c \left(\Phi_{d_2 \to d} \tilde{\mathbf{X}}_{d_2} + \boldsymbol{\omega}_{d_2:d}\right)^T\right]$$
$$E\left[\left(\Phi_{c_2 \to c} \tilde{\mathbf{X}}_{c_2} + \boldsymbol{\omega}_{c_2:c}\right) \tilde{\mathbf{X}}_d^T\right]$$
$$E\left[\left(\Phi_{c_2 \to c} \tilde{\mathbf{X}}_{c_2} + \boldsymbol{\omega}_{c_2:c}\right)\left(\Phi_{d_2 \to d} \tilde{\mathbf{X}}_{d_2} + \boldsymbol{\omega}_{d_2:d}\right)^T\right]$$

Since the expression from the previous level, i. e. the first level, was already checked, it is now required to check whether any of the expressions involving nodes from the current level are known. In other words, the question is whether any of the pairs P_{cd_2}, P_{c_2d} and $P_{c_2d_2}$ are known. In addition, it is also required to know the correlation between the noise terms and the state vectors.

Assuming none of the pairs is known, we proceed to the next level, the third level. *Each* node in the second level may have either transition or update relation, given by Eqs. (3) and (8), respectively, with the nodes in the third level. In this case, since the second level contains only a single node in each tree (c_2 and d_2), there are four possible cases: transition relation for c_2 and update relation for d_2; update relation for c_2 and transition relation for d_2; update relations for c_2 and d_2; transition relations for c_2 and d_2. At this point, we choose to analyze the first case. Other cases are treated in a similar manner.

Thus, c_2 has a single parent, denoted by c_3, while d_2 has three parents denoted by d_3^1, d_3^2 and d_3^3 (in this case d_2 represents an update event, while d_3^1, d_3^2, d_3^3 represent the three participating images). The relations for c_2 and d_2 can be written as

$$\begin{aligned}
\tilde{\mathbf{X}}_{c_2} &= \Phi_{c_3 \to c_2} \tilde{\mathbf{X}}_{c_3} + \boldsymbol{\omega}_{c_3:c_2} \\
\tilde{\mathbf{X}}_{d_2} &= A_{d_3^3} \tilde{\mathbf{X}}_{d_3^3} + A_{d_3^2} \tilde{\mathbf{X}}_{d_3^2} + A_{d_3^1} \tilde{\mathbf{X}}_{d_3^1} + A_{d_3^{123}} \mathbf{v}_{d_3^{123}}
\end{aligned}$$

where $A_{d_3^3} \doteq (I - K_{d_3^3} H_{d_3^3})$, $A_{d_3^2} \doteq -K_{d_3^3} H_{d_3^2}$, $A_{d_3^1} \doteq -K_{d_3^3} H_{d_3^1}$ and $A_{d_3^{123}} \doteq -K_{d_3^3} D_{d_3^{123}}$.

Having reached a new level, the third level, new expressions for the required term $E[\tilde{\mathbf{X}}_c \tilde{\mathbf{X}}_d^T]$ may be written utilizing nodes from *this* level and *lower* levels. Note that all the expressions from the previous (second) level were already analyzed.

Consider that some term, for example, $E[\tilde{\mathbf{X}}_{c_3} \tilde{\mathbf{X}}_{d_3^3}^T]$, is known, which means that the nodes c_3 and d_3^3 in T_c and T_d, respectively, either represent images that participated in the same three-view update in the past, or that these two nodes are identical ($c_3 \equiv d_3^3$). In any case, the known term $E[\tilde{\mathbf{X}}_{c_3} \tilde{\mathbf{X}}_{d_3^3}^T]$, accordingly weighted, as described in [20], is part of $E[\tilde{\mathbf{X}}_c \tilde{\mathbf{X}}_d^T]$.

Having a known term also means that there is no need to proceed to nodes of higher levels which are related to this term. In the case of a known $E[\tilde{\mathbf{X}}_{c_3} \tilde{\mathbf{X}}_{d_3^3}^T]$, we would not proceed to the parents of c_3 and d_3^3, unless this is required by the *unknown* terms in the current level. In this example, if the unknown terms are $E[\tilde{\mathbf{X}}_{c_3} \tilde{\mathbf{X}}_{d_3^2}^T]$ and $E[\tilde{\mathbf{X}}_{c_3} \tilde{\mathbf{X}}_{d_3^1}^T]$, then we would proceed to the parents of c_3 in T_c, and of d_3^1 and d_3^2 in T_d, but *not* to the parents of d_3^3 in T_d.

The procedure proceeds to higher levels until either all the terms required for calculating the cross-covariance $E[\tilde{\mathbf{X}}_c \tilde{\mathbf{X}}_d^T]$ are known, or reaching the top level in both trees. In the latter case, the unknown terms of cross-covariance are actually zero.

The process noise terms are assumed to be statistically independent of each other, $E[\boldsymbol{\omega}_{i_1:i_2} \boldsymbol{\omega}_{j_1:j_2}^T] = 0$, if $\boldsymbol{\omega}_{i_1:i_2}$ and $\boldsymbol{\omega}_{j_1:j_2}$ belong to different platforms, or in the case the two noise terms belong to the same platform but $\{i_1 : i_2\} \cap \{j_1 : j_2\} = \{\}$. The measurement noise is assumed to be statistically independent with the process noise. On the other hand, the process and measurement noise terms are *not* necessarily statistically independent with the involved state vectors. Their contribution to the required cross-covariance $E[\tilde{\mathbf{X}}_c \tilde{\mathbf{X}}_d^T]$ is analyzed in [20].

Computational Complexity

The computational complexity changes from one scenario to another. However, it can be shown [20] that the worst case is bounded by $O\left(n^2 \log n\right)$, where n is the number of multi-platform updates represented in G. Moreover, the actual computational complexity can be significantly reduced using efficient implementation methods [20].

It is worth noting that in practice, many scenarios exist in which the worst-case computational complexity is significantly less. One example is the scenario considered in Figs. 3 and 4, which requires processing only 3 levels in each tree.

6. Overall Distributed Scheme

Assume a scenario of M cooperative platforms. Each, or some, of these platforms maintain a repository of captured images attached with navigation data. All the platforms maintain a local copy of the graph, that is updated upon every multi-platform update event. This graph contains M threads, one thread for each platform in the group. The graph is initialized to M empty threads. The formulation of a single multi-platform update event is as follows.

The querying platform broadcasts its currently-captured image and its navigation solution to the rest of the platforms. A platform that receives this query, performs a check in its repository whether it has previously captured images of the same region. Platforms that do not maintain such a repository perform this check over the currently captured image only. Different procedures for performing this query may be devised. One possible alternative is to check only those images in the repository, that have a reasonable navigation data attached, e. g. images that were captured from a vicinity of the transmitted position of the querying platform.

Among the chosen images, only images that have a smaller uncertainty in their attached navigation data, compared to the uncertainly in the transmitted navigation data of the querying platform, are transmitted back to the querying platform. More specifically, denote by P_Q the covariance matrix of the querying platform, and P the covariance matrix attached to one of the chosen images from a repository of some other platform in the group. Then, in our current implementation, this image is transmitted back to the querying platform only if its position uncertainty is smaller than the position uncertainty of the querying platform, i. e.:

$$(P)_{ii} < \alpha(P_Q)_{ii} \quad , \quad i = 1, 2, 3 \tag{12}$$

where $(A)_{ij}$ is the member from the ith row and jth column of some matrix A, and α is a constant satisfying $0 < \alpha \leq 1$. Naturally, other criteria may be applied as well.

The chosen images, satisfying the above condition are transmitted to the querying platform, along with their attached navigation data. In addition, a transition matrix between the transmitted images, should more then one image is transmitted by the same platform, is sent. In case the replying platform has already participated in at least one multi-platform update of any platform in the group, its thread in the graph will contain at least one node. Therefore, transition matrices bridging the navigation data attached to the images being transmitted in the current multi-platform update to the closest nodes in this thread are also sent.

As an example, consider the scenario shown in Fig. 5. Fig. 6 presents the construction details of the graph for this scenario, for each of the executed three-view measurement updates. Assume the first update, a_3^+, was executed, and focus on the second update, b_3^+. As shown in Fig. 6(b), platform I transmits two images and navigation data, denoted by the nodes b_1^- and b_2^- in the graph. However, in addition to the transmitted transition matrix and process noise covariance matrix between these two nodes, $\phi_{b_1 \to b_2}$ and $Q_{b_1 \to b_2}$, the transition matrix and noise covariance matrix between the nodes b_2 and a_3, $\phi_{b_2 \to a_3}$ and $Q_{b_2 \to a_3}$, are transmitted as well.

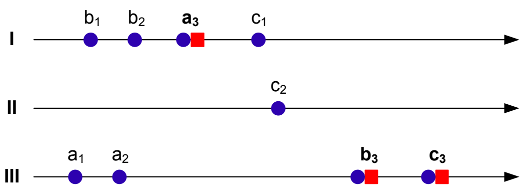


Figure 5. Three-platform scenario

Upon receiving the transmitted images and the navigation data, two best images are selected[6], the cross-covariance terms are calculated based on the local graph, as discussed in Section 5, followed by computation of all the relevant filter matrices: $H_3, H_2, H_1, \mathcal{A}, \mathcal{B}, \mathcal{D}$.

Next, the update of the querying platform is carried out based on Section 4. Now, it is only required to update the local graphs of all the platforms in the group by the performed update event. The querying platform broadcasts the following information: *a)* identity of the involved platforms in the current update; *b)* time instances (or some other identifiers) of the involved images; required transition matrices of the involved images; *c)* a priori and a posteriori covariance and cross-covariance matrices; *d)* filter matrices K_3, H_3, H_2 and H_1. Then, each platform updates its own graph representation.

The described-above process is summarized in Algorithms 1 and 2. Algorithm 1 contains a protocol of actions carried out by the querying platform, while Algorithm 2 provides the protocol of actions for the rest of the platforms in the group.

Handling Platforms Joining or Leaving the Group

Whenever a platform joins an existing group of cooperative platforms, it must obtain the graph describing the history of multi-platform updates among the platforms in the group. This graph may be transmitted to the joining platform by one one of the platforms in the group. Departure of a platform from the group does not require any specific action.

An interesting scenario is one in which there are *several* groups of cooperative platforms, and a platform has to migrate from one group to another. Refer the former and the latter groups as the source and destination groups. For example, this might be the case when each cooperative group operates in a distinct location and there is a need to move a platform within these groups. In these scenarios the migrating platform has already a local graph representing the multi-platform events of the

[6]The selection is according to some criteria, e. g., Eq. (12). Alternatively, the proposed approach may be also applied on more than three images.

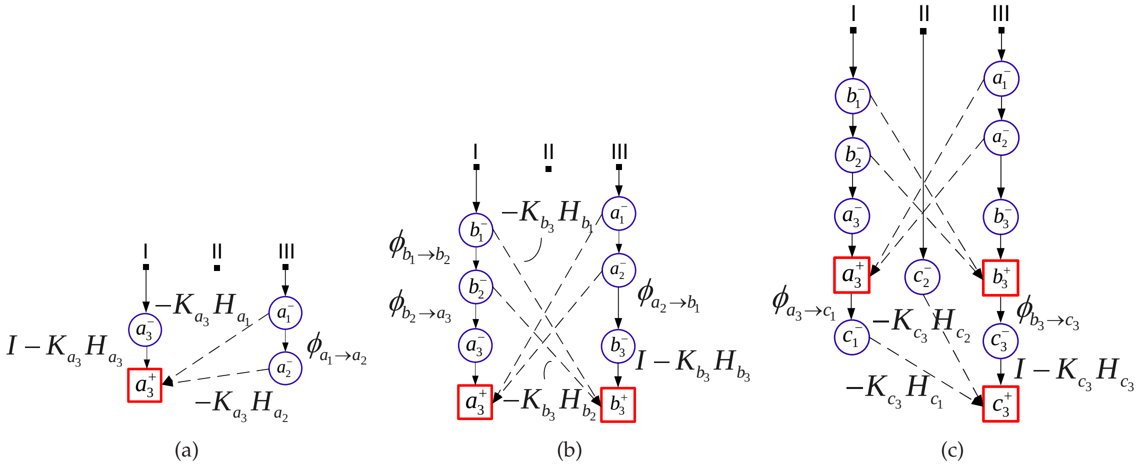


Figure 6. Graph update process: **a)** update event a_3^+; **b)** update event b_3^+; **c)** update event c_3^+.

Algorithm 1 Querying Platform Protocol

```
1:  Notations: $Q$ - Querying platform; $A, B$ - two other platforms.
2:  Broadcast current image $I^Q$ and current navigation data.
3:  Receive a set of images and associated navigation data from other platforms. See steps 2-11 in Algorithm 2.
4:  Choose two best images $I^A, I^B$ transmitted by platforms $A$ and $B$, respectively.
5:  **First graph update:**
    • Add a new node for each image in the appropriate thread ($A, B$ and $Q$).
    • Denote these three new nodes in threads $A, B$ and $Q$ as $\beta_1, \beta_2$ and $\beta_3$, respectively.
    • Connect each such node to previous and next nodes (if exist) in its thread by directed arcs associated with the
      transition matrices and with the process noise covariance matrices.
6:  Calculate cross-covariance terms based on the local graph.
7:  Calculate the measurement **z** and the filter matrices $K_3, H_3, H_2, H_1, D$ based on the three images $I^A, I^B, I^Q$ and the
    attached navigation data.
8:  Perform navigation update on platform $Q$.
9:  **Final graph update:**
    • Add an update-event node, denoted by $\alpha$, in the thread $Q$.
    • Connect the nodes $\beta_1, \beta_2$ and $\beta_3$ to the update-event node $\alpha$ by directed arcs weighted as $-K_3H_1, -K_3H_2$ and
      $I - K_3H_3$, respectively. Associate also measurement noise covariance matrix to each arc.
    • Store a priori and a posteriori covariance and cross-covariance terms (e. g. in the nodes $\beta_1, \beta_2, \beta_3$ and $\alpha$).
10: Broadcast update event information.
```

Algorithm 2 Replying Platform Protocol

```
1:  Notations: $Q$ - Querying platform; $A$ - current platform.
2:  **if** a query image and its navigation data are received **then**
3:  Search repository for images containing the same scene.
4:  Choose images that satisfy the navigation uncertainty criteria (12).
5:  For each chosen image, captured at some time instant $k$, look among all the nodes in thread $A$ in the local graph,
    for two nodes with time $l$ and $m$ that are closest to $k$, such that $l < k < m$.
6:  Calculate transition matrices $\phi_{l\rightarrow k}$ and $\phi_{k\rightarrow m}$ and noise covariance matrices $Q_{l:k}$ and $Q_{k:m}$.
7:  **if** more than one image was chosen in step 4 **then**
8:  Calculate transition matrices and noise covariance matrices between the adjacent chosen images.
9:  **end if**
10: Transmit the chosen images, their navigation data and the calculated transition and noise covariance matrices
    to the querying platform $Q$.
11: **end if**
12: **if** update message is received from $Q$ **then**
13: Update local graph following steps 5 and 9 in Algorithm 1.
14: **end if**
```

source group, while the destination group has its own graph.

These two graphs have no common threads only when each platform is assigned only to one group, and, in addition, no migration between the groups have occurred in the past. In any case, upon receiving the graph of the destination group, the joining platform may fuse the two graphs and broadcast the updated graph to all the platforms in the destination group.

Efficient Calculation of Transition and Process Noise Covariance Matrices

The problem this section refers to is of calculating the transition matrix and the process noise covariance matrix between some two time instances which are unknown a priori. These matrices participate in calculation of the cross-covariance terms, as explained in Section 5. We first handle calculation of transition matrices. Recall that each image stored in the platform repository is associated with navigation parameters taken when the image was captured. In particular, the transition matrix from the previously-stored image time instant to current image that is about to be added to the repository is calculated.

A naive approach for calculating the transition matrix $\phi_{i\to j}$ between some image i to some other image j in the repository would be based on

$$\phi_{i\to j} = \phi_{j-1\to j} \cdot \ldots \cdot \phi_{i\to i+1} \tag{13}$$

However, a much more time-efficient alternative is to calculate $\phi_{i\to j}$ using transition matrices bridging between several images time instances. For example, if we had available the matrix $\phi_{i\to j-1}$, the computation of $\phi_{i\to j}$ would require multiplication of only two matrices: $\phi_{i\to j} = \phi_{j-1\to j} \cdot \phi_{i\to j-1}$. This concept may be obtained by maintaining a skip list [23] type database. The lowest level is comprised of the stored images and its associated navigation data, including the transition matrices between adjacent stored images. This level is a possible implementation of the repository maintained by all/some platforms. Each next level is constructed by skipping several nodes in the lower level, and assigning the appropriate transition matrix, transferring from previous node to next node in the same level. No other data is stored outside the first level nodes.

An example of this concept is given in Fig. 7, in which every two nodes in some level contribute a node in the next level. Thus, for instance, calculation of $\phi_{2\to 5}$ may be performed by searching for the appropriate route in the skip list formation, which will yield $\phi_{2\to 5} = \phi_{3\to 5}\phi_{2\to 3}$, instead of carrying out the three matrix multiplications $\phi_{2\to 5} = \phi_{4\to 5}\phi_{3\to 4}\phi_{2\to 3}$.

The process noise covariance matrix between any two time instances may be efficiently calculated following a similar approach. For example, $Q_{i:j}$, for general i and j, $i < j$ is given by

$$\begin{aligned} Q_{i:j} &= Q_{j-1:j} + \Phi_{j-1\to j} Q_{j-2:j-1} \Phi^T_{j-1\to j} + \\ &+ \cdots + \Phi_{i+1\to j} Q_{i:i+1} \Phi^T_{i+1\to j} \end{aligned}$$

However, if each node in the skip list database contains the noise covariance matrix between the previous node in the same level, $Q_{i:j}$ may be also calculated, for instance, as $Q_{i:j} = Q_{j-1:j} + \Phi_{j-1\to j} Q_{i:j-1} \Phi^T_{j-1\to j}$.

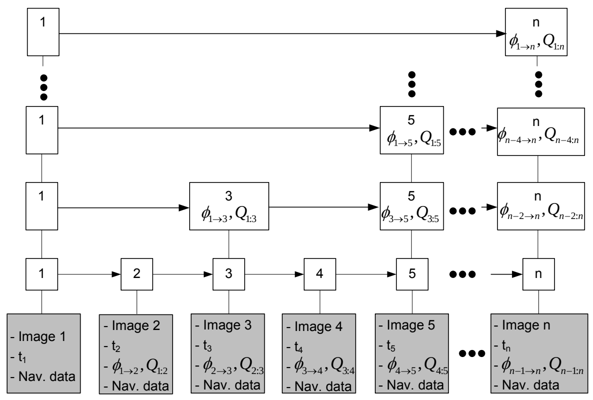


Figure 7. Skip list repository database example.

Incorporating Other Measurements

The proposed graph-based technique for calculating cross-covariance terms may be also applied when, in additional to the multi-platform three-view updates, other measurements should be incorporated as well. For instance, each platform can apply epipolar-geometry constraints based on images captured by its *own* camera (e. g. [24]). Moreover, some of the platforms may be equipped with additional sensors, or additional information might be available (e. g. DTM).

For simplicity, we assume at this point a standard measurement model for these additional measurement types, i. e. $\mathbf{z} = H\mathbf{X} + \mathbf{v}$. These measurement updates will be termed in this section as basic measurement updates. Next, it is shown how the basic measurements may be incorporated with the proposed approach for cooperative localization and navigation.

Since a standard measurement model was assumed, the a posteriori estimation error is given by

$$\tilde{\mathbf{X}}^+ = (I - KH)\tilde{\mathbf{X}}^- - K\mathbf{v} \tag{14}$$

Going back to the three-view measurement model, consider the simple scenario shown in Fig. 3. Assume a single basic update was performed between the first update event, at a_3, and the second update event, at b_3. Denote by γ the time instant of this additional update,

$\gamma \in (a_3, b_3)$. $\tilde{\mathbf{X}}_{b_3}^{-}$ is no longer inertially propagated from $\tilde{\mathbf{X}}_{a_3}^{+}$, but instead may be expressed as

$$\tilde{\mathbf{X}}_{b_3}^{-} = \phi_{\gamma\to b_3}\tilde{\mathbf{X}}_{\gamma}^{+} + \boldsymbol{\omega}_{\gamma:b_3} \tag{15}$$

Based on Eq. (14), $\tilde{\mathbf{X}}_{b_3}^{-}$ may be expressed as

$$\phi_{\gamma\to b_3}\left[(I - K_\gamma H_\gamma)\left(\phi_{a_3\to\gamma}\tilde{\mathbf{X}}_{a_3}^{+} + \boldsymbol{\omega}_{a_3:\gamma}\right) - K_\gamma \mathbf{v}\right] + \boldsymbol{\omega}_{\gamma:b_3}$$

or, alternatively:

$$\tilde{\mathbf{X}}_{b_3}^{-} = \phi_{a_3\to b_3}^{*}\tilde{\mathbf{X}}_{a_3}^{+} + \boldsymbol{\omega}_{a_3:b_3}^{*} \tag{16}$$

where $\phi_{a_3\to b_3}^{*} \doteq \phi_{\gamma\to b_3}(I - K_\gamma H_\gamma)\phi_{a_3\to\gamma}$ is the equivalent transition matrix and $\boldsymbol{\omega}_{a_3:b_3}^{*} \doteq \phi_{\gamma\to b_3}(I - K_\gamma H_\gamma)\boldsymbol{\omega}_{a_3:\gamma} - \phi_{\gamma\to b_3}K_\gamma\mathbf{v} + \boldsymbol{\omega}_{\gamma:b_3}$ is the equivalent noise term with noise covariance $Q_{a_3:b_3}^{*}$ given by

$$\phi_{\gamma\to b_3}(I - K_\gamma H_\gamma)Q_{a_3:\gamma}\left[\phi_{\gamma\to b_3}(I - K_\gamma H_\gamma)\right]^T + \\ +\phi_{\gamma\to b_3}K_\gamma R K_\gamma^T \phi_{\gamma\to b_3}^T + Q_{\gamma:b_3}$$

where $R \doteq E[\mathbf{v}\mathbf{v}^T]$.

Thus, for example, $P_{b_3b_1}^{-}$ is given by (cf. Eq. (11)):

$$P_{b_3b_1}^{-} = \Phi_{a_3\to b_3}^{*}\left\{(I - K_{a_3}H_{a_3})P_{a_3a_2}^{-} - \right. \\ \left. -K_{a_3}H_{a_2}P_{a_2a_2}^{-} - K_{a_3}H_{a_1}P_{a_1a_2}^{-}\right\}\Phi_{a_2\to b_1}^{T}$$

In the general case, there might be a number of basic updates in each of the platforms. However, these updates are treated in a similar manner, by calculating the equivalent transition matrix Φ^* and noise covariance matrix Q^* between the time instances that participate in the three-view measurement and updating accordingly the repository database (cf. Section 6).

7. Simulation and Experimental Results

In this section the proposed approach for vision-aided cooperative navigation is studied in two different scenarios. First, a formation flying scenario is considered, involving two platforms, a leader and a follower. Statistical results, based on simulated navigation data and synthetic imagery are presented. Next, a holding pattern scenario is demonstrated in an experiment using real imagery and navigation data.

Implementation Details

Navigation Simulation—The navigation simulation for *each* of the two platforms consists of the following steps [24]: (a) Trajectory generation; (b) velocity and angular velocity increments extraction from the created trajectory; (c) inertial measurement unit (IMU) error definition and contamination of pure increments by noise; and (d) strapdown calculations. The strapdown mechanism provides, at each time step, the calculated position, velocity and attitude of the platform. Each platform is handled independently based on its own trajectory. Once a platform obtains three images with a common overlapping area, the developed algorithm is executed: cross-covariance terms are computed, followed by estimation of the state vector. The estimated state vector is then used for updating the navigation solution and the IMU measurements (cf. Fig. 1). Next, the update information is stored and delivered to the second platform in the group.

Image Processing Module—Given three images with a common overlapping area, the image processing phase includes [20] features extraction from each image using the SIFT algorithm [25] and computation of sets of matching pairs between the first two images, $\left\{\mathbf{x}_1^i, \mathbf{x}_2^i\right\}_{i=1}^{N_{12}}$, and between the last two images, $\left\{\mathbf{x}_2^i, \mathbf{x}_3^i\right\}_{i=1}^{N_{23}}$, where $\mathbf{x}^i = (x^i, y^i)^T$ are the image coordinates of the ith feature. This computation proceeds as follows. First, the features are matched based on their descriptor vectors (that were computed as part of the SIFT algorithm), yielding the sets $\left\{\mathbf{x}_1^i, \mathbf{x}_2^i\right\}_{i=1}^{\tilde{N}_{12}}, \left\{\mathbf{x}_2^i, \mathbf{x}_3^i\right\}_{i=1}^{\tilde{N}_{23}}$. Since this step occasionally produces false matches (outliers), the RANSAC algorithm [26] is applied over the fundamental matrix [27] model in order to reject the existing false matches, thus obtaining the refined sets $\left\{\mathbf{x}_1^i, \mathbf{x}_2^i\right\}_{i=1}^{N_{12}}$ and $\left\{\mathbf{x}_2^i, \mathbf{x}_3^i\right\}_{i=1}^{N_{23}}$. The fundamental matrices are not used in further computations.

The next step is to use these two sets for calculating matching triplet features, i. e. matching features in the three given images. This step is performed by matching all $\mathbf{x}_1 \in \left\{\mathbf{x}_1^i, \mathbf{x}_2^i\right\}_{i=1}^{N_{12}}$ with all $\mathbf{x}_3 \in \left\{\mathbf{x}_2^i, \mathbf{x}_3^i\right\}_{i=1}^{N_{23}}$, yielding a set of matching triplets $\left\{\mathbf{x}_1^i, \mathbf{x}_2^i, \mathbf{x}_3^i\right\}_{i=1}^{N_{123}}$. The matching process includes the same steps as described above.

When using synthetic imagery data, a set of points in the real-world are randomly drawn. Then, taking into account the camera motion, known from the true platform trajectory, and assuming specific camera calibration parameters, the image coordinates of the observed real-world points are calculated using a pinhole projection [27] at the appropriate time instances. See, for example, Ref. [28] for further details. Consequently, a list of features for each time instant of the three time instances, which are manually specified, is obtained: $\left\{\mathbf{x}_1^i\right\}$, $\left\{\mathbf{x}_2^i\right\}$ and $\left\{\mathbf{x}_3^i\right\}$. The mapping between these three sets is known, since these sets were calculated using the pinhole projection based on the same real-world points. Thus, in order to find the matching sets $\left\{\mathbf{x}_1^i, \mathbf{x}_2^i, \mathbf{x}_3^i\right\}_{i=1}^{N_{123}}$, $\left\{\mathbf{x}_1^i, \mathbf{x}_2^i\right\}_{i=1}^{N_{12}}$ and $\left\{\mathbf{x}_2^i, \mathbf{x}_3^i\right\}_{i=1}^{N_{23}}$ it is only required to check which features are within the camera field of view at all the three time instances.

Finally, the calculated sets of matching features are transformed into sets of matching LOS vectors. A LOS vector, expressed in the camera system for some feature $\mathbf{x} = (x, y)^T$, is calculated as $\mathbf{q}^C = (x, y, f)^T$, where f is the camera focal length. As a result, three matching LOS sets are obtained: $\left\{\mathbf{q}_{1_i}^{C_1}, \mathbf{q}_{2_i}^{C_2}, \mathbf{q}_{3_i}^{C_3}\right\}_{i=1}^{N_{123}}$, $\left\{\mathbf{q}_{1_i}^{C_1}, \mathbf{q}_{2_i}^{C_2}\right\}_{i=1}^{N_{12}}$ and $\left\{\mathbf{q}_{2_i}^{C_2}, \mathbf{q}_{3_i}^{C_3}\right\}_{i=1}^{N_{23}}$. When handling real imagery, the camera focal length, as well as other camera parameters, are found during the camera calibration process. In addition, a radial distortion correction [27] was applied to camera-captured images, or alternatively, to the extracted feature coordinates.

Formation Flying Scenario - Statistical Results

In this section the proposed method for vision-aided cooperative navigation is applied on a formation flying scenario, comprised of a leader platform and a single follower platform. Each platform is equipped with a camera and an IMU. In this scenario, the leader's IMU is of a better quality than the follower's IMU. It is also assumed that the leader's initial navigation errors are small compared to those of the follower. Table 1 summarizes the assumed initial navigation errors and IMU errors for the two platforms.

Both platforms perform the same trajectory, which is a straight and level north-headed flight at a 100 m/s velocity. The mean height above ground level is 2000 m. The distance between the leader and follower platforms is 2000 meters (the follower is behind the leader), i. e. 2 seconds delay. The synthetic imagery data was obtained by assuming a $20^0 \times 30^0$ camera field of view, focal length of 1570 pixels, and image noise of 0.5 pixel. The ground landmarks were randomly drawn with a height variation of ±200 meters relative to the mean ground level.

The follower was updated using the proposed method every 10 seconds, applying the same measurement schedule as in Fig. 3 (platform I in the figure is the leader, platform III is the follower). The first update was carried out after 27 seconds of inertial flight, while the leader platform performed an inertial flight the whole time duration. The true translation motion between any three views participating in the same measurement is $T_{12} = 200$ meters and $T_{23} = 400$ meters, in north direction.

In each update, two of the three images[7], that participate in the measurement, were taken from the leader. Since the two platforms perform the same trajectory, with a 2 seconds time delay, these two images have been acquired by the leader 2 seconds before the measurement.

[7] Since in this section a synthetic imagery data is used, the term "image" refers to a synthetic data, e. g. features coordinates.

Therefore they were stored in the leader's repository and retrieved upon request. The cross-covariance terms were calculated in each update according to Eq. (11).

The Monte-Carlo results (1000 runs) for the follower platform are presented in Fig. 8, in terms of the mean navigation error (μ), standard deviation (σ) and square root covariance of the filter. In addition, the results are compared to inertial navigation of the follower. As seen, the position and velocity errors (Figs. 8(a) and 8(b)) are significantly reduced, compared to the inertial scenario, in *all* axes. The bias state is estimated also in *all* axes, while the drift state is only partially estimated. The updates yielded a mild reduction in Euler angle errors as well.

A comparison of the follower navigation errors to the leader navigation errors, given in Fig. 9, reveals further insight. Since leader images and navigation data with a 2 second delay were used for updating the follower, the comparison should be made between the follower navigation errors and the leader navigation errors 2 seconds back in time.

The position errors (and velocity errors to a less extent) of the follower are *lower* than those of the leader (Fig. 9(a)), despite the fact that the leader has a considerably better navigation system, and the follower is updated solely based on the leader's navigation data. The reason for this phenomenon is that the measurement $\mathbf{z}$, given in Eq. (4), is a function of both the follower's and the leader's navigation parameters, while only the follower is actually updated (cf. Section 4). Carrying out the updates on both platforms, using the filter formulation discussed in Section 4, will yield an improvement also in the leader's navigation errors [7]. Assuming the measurement schedule given in Fig. 3, it is guaranteed that the graph will remain acyclic even if both of the platforms are updated each measurement.

It is also worth mentioning that should the leader perform self-updates based on the available sensors and information (e. g. epipolar constraints, GPS, DTM), improved navigation errors will be obtained not only in the leader but also in the follower navigation system.

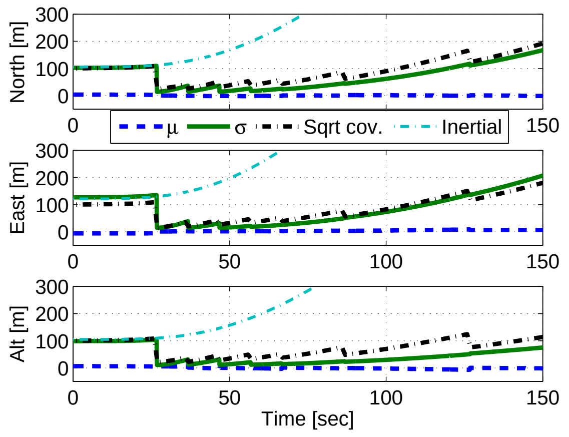
The importance of incorporating the cross-covariance terms in the update process is clearly evident when comparing the results of Fig. 8 with Fig. 10, that presents Monte-Carlo results when the cross-covariance terms are neglected. As seen in Fig. 10, the position and velocity errors are biased, mainly along the flight heading.

Holding Pattern Scenario - Experiment Results

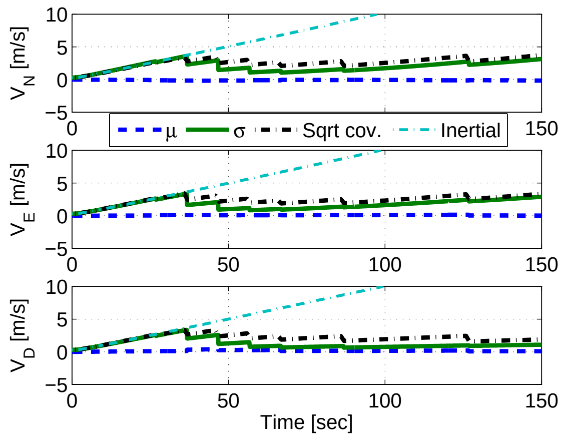
In this section the proposed method is demonstrated in an experiment. The experiment setup consists of a single ground vehicle, attached with a 207MW Axis network

Table 1. Initial Navigation Errors and IMU Errors in Formation Flying Scenario

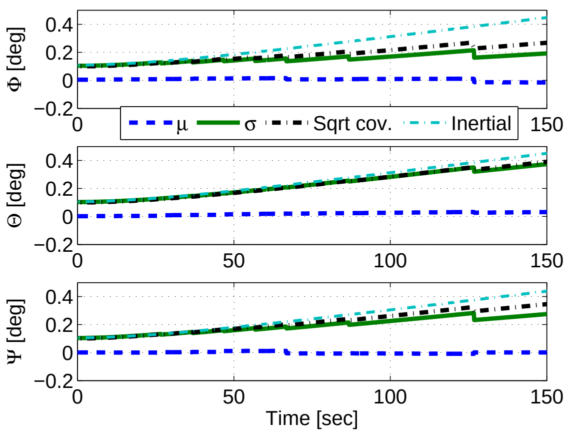
Parameter	Description	Leader	Follower	Units
$\Delta\mathbf{P}$	Initial position error (1σ)	$(10, 10, 10)^T$	$(100, 100, 100)^T$	m
$\Delta\mathbf{V}$	Initial velocity error (1σ)	$(0.1, 0.1, 0.1)^T$	$(0.3, 0.3, 0.3)^T$	m/s
$\Delta\mathbf{\Psi}$	Initial attitude error (1σ)	$(0.1, 0.1, 0.1)^T$	$(0.1, 0.1, 0.1)^T$	deg
$\mathbf{d}$	IMU drift (1σ)	$(1, 1, 1)^T$	$(10, 10, 10)^T$	deg/hr
$\mathbf{b}$	IMU bias (1σ)	$(1, 1, 1)^T$	$(10, 10, 10)^T$	mg



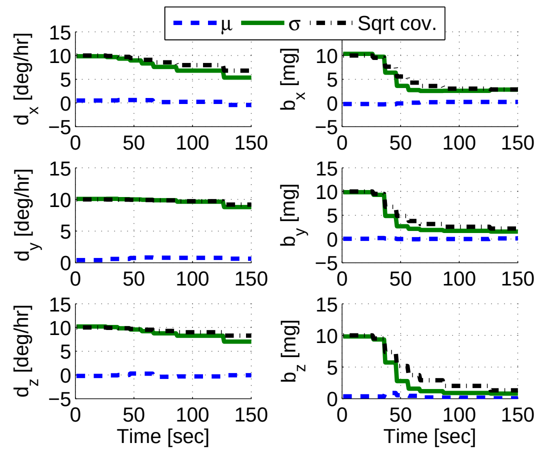
(a)Position errors.



(b)Velocity errors.



(c)Euler angles errors.



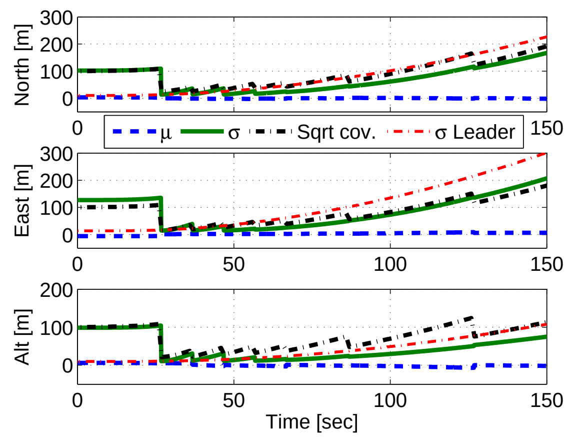
(d)Drift and bias estimation errors.

Figure 8. Formation flying scenario - Monte Carlo (1000 runs) results; Follower navigation errors compared to inertial navigation: Reduced position and velocity errors in *all* axes. Bias estimation to the leader platform's bias levels (see also Fig. 9).

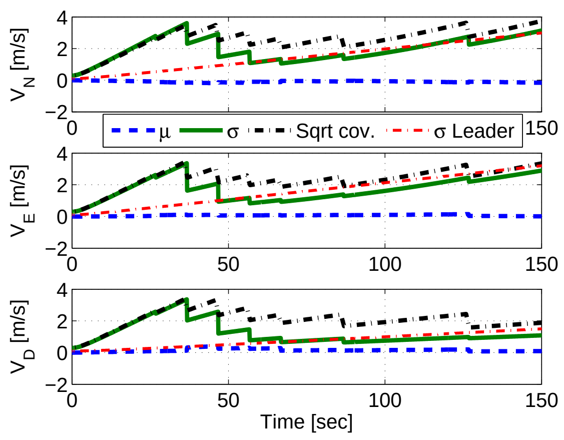
camera[8] and MTi-G Xsens IMU/INS[9]. The vehicle was manually commanded using a joystick, while the camera captured images perpendicular to the motion heading. As in [19], the IMU and imagery data was recored for post-processing at 100 Hz and 15 Hz, respectively. These two sources of data were synchronized [19].

The vehicle performed two different trajectories. The IMU and the camera were turned off between these two trajectories, thereby legitimating to treat each trajectory as if it was performed by a different vehicle, equipped with a similar hardware (IMU and camera), as opposed to Section 7, where one of the vehicles was assumed to be equipped with a better navigation system. Thus, we have two ground vehicles, each performing its own trajectory and recording its own IMU and imagery data.

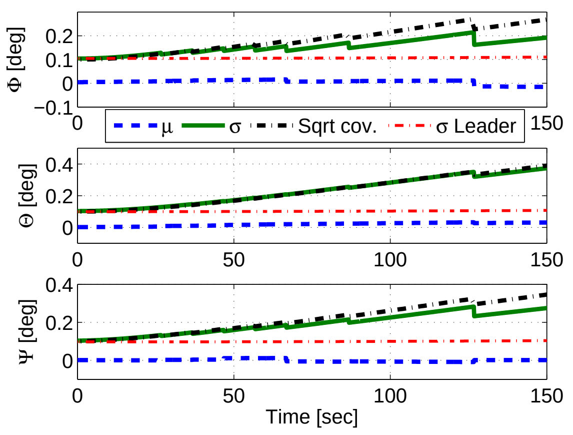
[8]http://www.axis.com/products/cam_207mw/index.htm.
[9]http://www.xsens.com/en/general/mti-g.



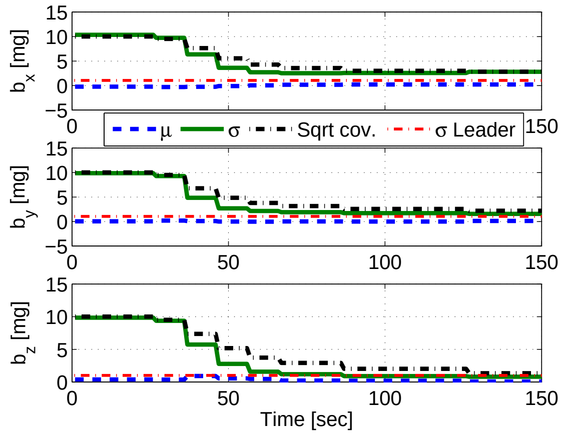
(a)Position errors.



(b)Velocity errors.

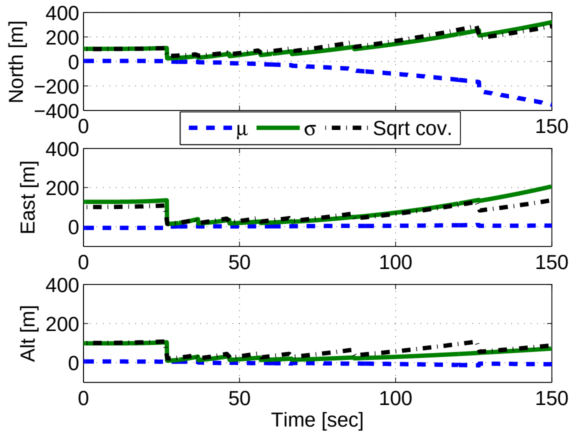


(c)Euler angles errors.

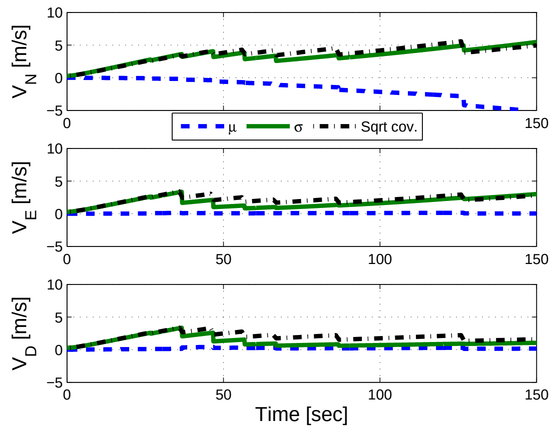


(d)Bias estimation errors.

Figure 9. Formation flying scenario - Monte Carlo (1000 runs) results; Follower navigation errors compared to navigation errors of the leader: Position and velocity errors are reduced below the leader level of errors. Bias estimation to the leader's bias levels (1 mg). Euler angles are also reduced, however do not reach the leader's levels due to poor estimation of the drift state (cf. Fig. 8(d)).



(a)Position errors.



(b)Velocity errors.

Figure 10. Formation flying scenario - Monte Carlo (1000 runs) results; Follower navigation errors when cross-covariance terms are neglected: Biased estimation along the motion heading.

The only available ground-truth data is the manually measured trajectories, since the experiment was carried out indoors and GPS was therefore unavailable [19]. The two trajectories represent a holding pattern scenario. Each platform performs the same basic trajectory: vehicle I performs this basic trajectory twice, while vehicle II performs the basic trajectory once, starting from a different point along the trajectory, and reaching the starting point of vehicle I after about 26 seconds. The reference trajectories of vehicle I and II are shown in Fig. 11. The diamond and square marks denote the manual measurements of the vehicles position. Each two adjacent marks of the same platform are connected using a linear interpolation.

The proposed method for multi-platform (MP) three-view based updates was applied several times in the experiment. In addition, the method was executed in a self-update mode, in which all the images are captured by the same vehicle [19]. The cross-covariance terms in this case were computed exactly as in the case of multi-platform updates. A schematic sketch of the measurements schedule is given in Fig. 12. Table 2 provides further information, including the time instances of each participating triplet of images in the applied measurements.

As seen, vehicle I is updated twice using data obtained from vehicle II (measurements **c** and **e**), and four times based on its own images (measurements **f**, **g**, **h** and **i**). Vehicle II is updated three times utilizing the information received from vehicle I (measurements **a**, **b** and **d**). The vehicles performed inertial navigation elsewhere, by processing the recorded IMU data.

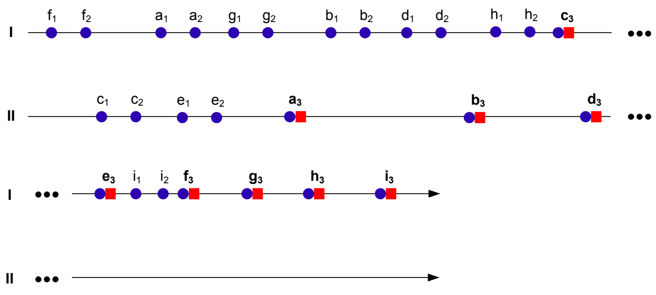


Figure 12. Schematic sketch of the measurement schedule in the experiment. Further information regarding each measurement is given in Table 2.

The images participating in each three-view update were manually identified and chosen. Fig. 13 shows, for example, the three images of measurement **a**: images 13(a) and 13(b) were captured by vehicle I, while image 13(c) was captured by vehicle II. Features that were found common to all the three images (triplets) are also shown in the figure. Note that two objects (a bottle, and a bag) that appear in images 13(a) and 13(b) are missing in image 13(c). These two objects were removed between the two trajectories. Therefore, as seen in Fig. 13, these two objects are not represented by matched triplets of features (but can be represented by matched pairs of features between the first two views). Additional details regarding the image processing phase in the experiment can be found in [20].

The experiment results are given in Fig. 14: Figs. 14(a) and 14(b) show the position errors for vehicle I and II, while Figs. 14(c) and 14(d) show the velocity errors. Each figure consists of three curves: navigation error, square root covariance of the filter, and navigation error in an inertial scenario (given for reference). The measurement type (MP-update or self-update) is also denoted in the appropriate locations.

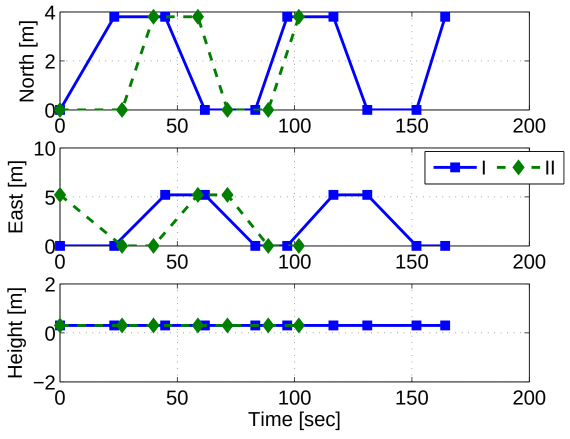
The position error was calculated by subtracting the navigation solution from the true trajectories (cf. Fig. 11). In a similar manner, the velocity error was computed by subtracting the navigation solution from the true velocity profiles. However, since velocity was not measured in the experiment, it was only possible to obtain an approximation of it. The approximated velocity was calculated assuming that the vehicles moved with a constant velocity in each phase[10].

As seen from Fig. 14(a), the position error of vehicle I was nearly nullified in all axes as the result of the first update, which was of MP type. The next update (also MP) caused to another reduction in the north position error. After completing a loop in the trajectory, it became possible to apply the three-view updates in a Self-update mode for vehicle I, i. e. all the three images were captured by vehicle I. In the overall, due to the applied 6 three-view updates, the position error of vehicle I has been confined to around 50 meters in north and east directions, and 10 meters in altitude. As a comparison, the position error of vehicle I in an inertial scenario reaches, after 150 seconds of operation, 900, 200 and 50 meters in north, east and down directions, respectively. The position error of vehicle II (cf. Fig. 14(b)) has been also dramatically reduced as the result of the three-view multi-platform updates. For example, after the third update ($t \approx 60$ seconds), the position error was nearly nullified in north direction and reduced from 50 to 20 meters in east direction. One can observe that the velocity errors are also considerably reduced in all axes (cf. Figs. 14(c) and 14(d)).

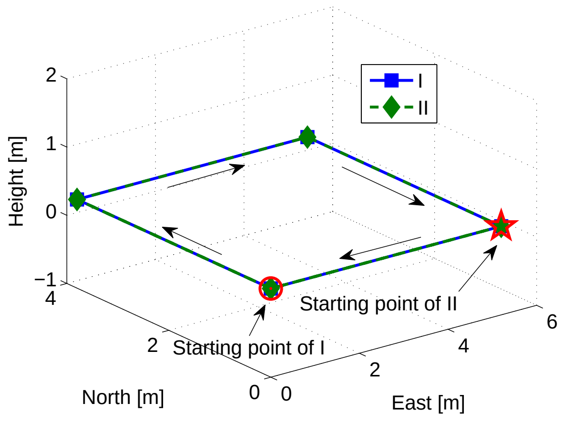
8. Conclusions

This paper presented a new method for distributed vision-aided cooperative navigation based on three-view geometry constraints. Each platform was assumed to be equipped with an INS and a camera. The plat-

[10]The phase duration and the translation that each vehicle has undergone in each phase are known from analyzing the IMU measurements and from the true trajectories.



(a)

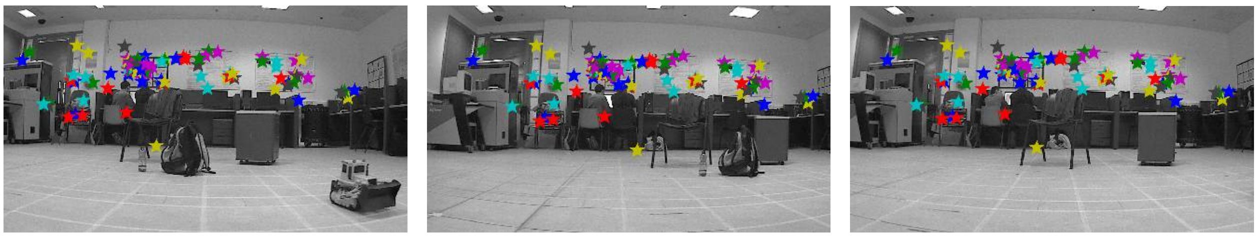


(b)

Figure 11. Trajectories of vehicles I and II in the experiment. Diamond and square marks indicate manually-measured vehicle locations. Circle and star marks in (b) denote the starting point of each platform.

Table 2. Measurement details in the experiment.

Measurement notation	Type	Querying vehicle	t_3 [sec]	Replying vehicle	t_1, t_2 [sec]
a	MP update	II	32.6	I	8.4, 14.2
b	MP update	II	53.2	I	35.9, 39.1
c	MP update	I	60.0	II	2.3, 5.6
d	MP update	II	60.6	I	47.9, 49.2
e	MP update	I	66.8	II	10.3, 12.1
f	Self update	I	81.1	I	0.3, 1.3
g	Self update	I	97.0	I	22.8, 24.3
h	Self update	I	124.7	I	54.3, 55.6
i	Self update	I	142.0	I	70.8, 72.1

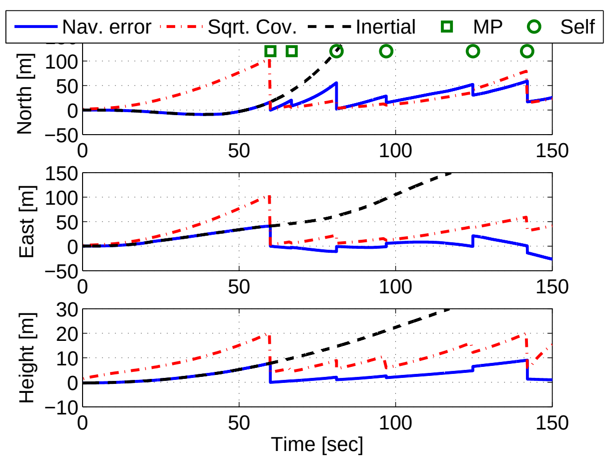


(a) (b) (c)

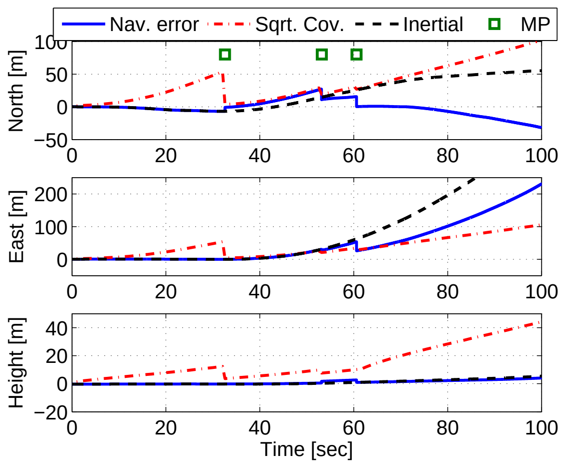
Figure 13. Images participating in measurement **a** and matched triplets of features. Images (a) and (b) were captured by vehicle I; Image (c) was captued by vehicle II. The images (a),(b) and (c) are represented in Fig. 12 as a_1, a_2 and a_3.

forms were also assumed to be capable of communicating between themselves. While traditional approaches for cooperative localization and navigation utilize relative pose measurements, in the proposed method a measurement is formulated whenever the same general scene is observed by different platforms.

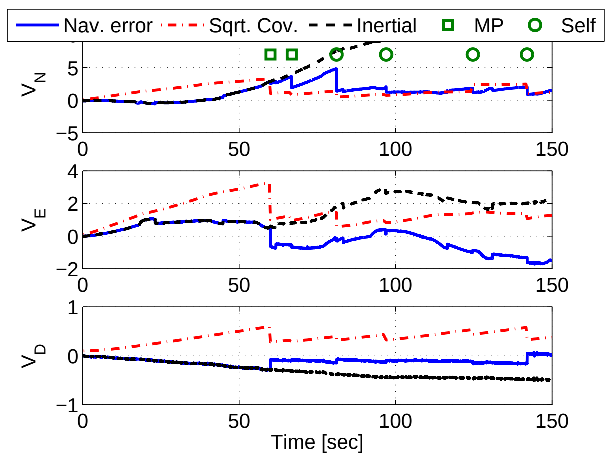
Three images of a common region are required for each measurement. These images are not necessarily captured at the same time. All, or some, of the platforms maintain a local repository of captured images, that are associated with some navigation parameters. In a typical scenario, a platform captures an image and broadcasts it, along with its current navigation solution, to



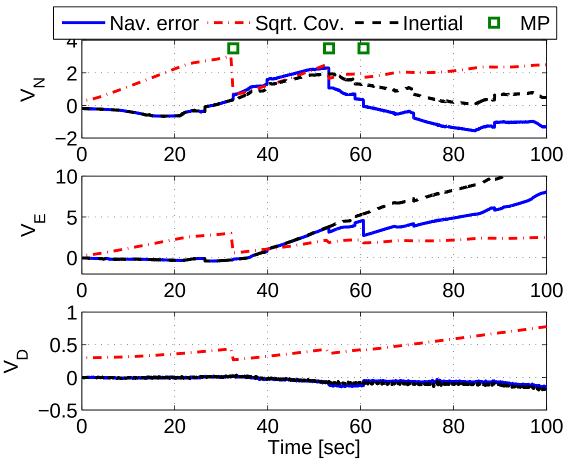
(a)Position errors - vehicle I



(b)Position errors - vehicle II



(c)Velocity errors - vehicle I



(d)Velocity errors - vehicle II

Figure 14. Position and velocity errors of vehicles I and II in the experiment.

other platforms in the group, inquiring if they have previously captured images containing the same region. Upon receiving such a query, each platform performs a check in its repository looking for appropriate images. Among these images, only images with a smaller navigation uncertainty compared to the uncertainty in the navigation data of the query image, are transmitted back.

The currently-captured image and two of the transmitted images, along with the attached navigation parameters allow to update the navigation system of the broadcasting platform. The navigation parameters associated with the three images participating in the same measurement may be correlated. Since the identity of the platforms that captured these images and the capture time of the images is unknown a priori, a graph-based technique was applied for on-demand calculation of the required correlation terms. The graph is locally maintained by each platform in the group. After carrying out a multi-platform update, the relevant update information is broadcasted to the platforms in the group, which then independently update their own copy of the graph.

The proposed method was studied in a simulated environment and in an experiment. Statistical results are presented, based on simulated navigation and synthetic imagery data, for a leader-follower scenario, in which the leader is equipped with a higher quality INS compared to the follower INS. The developed method allowed to reduce the rapidly-developing navigation errors of the follower to the level of errors of the leader. A holding pattern scenario is demonstrated in an experiment, involving two ground vehicles, equipped with identical inertial measurement units and cameras. Significant reduction in the navigation errors of both of the vehicles was obtained as a result of activating the proposed method.

References

[1] Madhavan, R., Fregene, K. and Parker, L. E.,"Distributed Cooperative Outdoor Multirobot Localization and Mapping," *Autonomous Robots*,

Vol. 17, 2004, pp. 23–29.

[2] Nettletona, E., Thrun, S., Durrant-Whyte, H. and Sukkarieh, S.,"Decentralised SLAM with Low-Bandwidth Communication for Teams of Vehicles," *Proceedings of the International Conference on Field and Service Robotics*, Lake Yamanaka, Japan, 2003.

[3] Kim, B., Kaess, M., Fletcher, L., Leonard, J., Bachrach, A., Roy, N. and Teller, S.,"Multiple Relative Pose Graphs for Robust Cooperative Mapping," *Proceedings of the IEEE International Conference on Robotics and Automation*, Anchorage, Alaska, May 2010.

[4] Lazaro, M. T. and Castellanos, J. A.,"Localization of Probabilistic Robot Formations in SLAM," *Proceedings of the IEEE International Conference on Robotics and Automation*, Anchorage, Alaska, May 2010.

[5] Shaferman, V. and Shima, T.,"Unmanned Aerial Vehicles Cooperative Tracking of Moving Ground Target in Urban Environments," *Journal of Guidance, Control and Dynamics*, Vol. 31, No. 5, 2008, pp. 1360–1371.

[6] Smaili, C., Najjar, M. E. E. and Charpillet, F.,"Multi-sensor Fusion Method Using Bayesian Network for Precise Multi-vehicle Localization," *Proceedings of the IEEE International Conference on Intelligent Transportation Systems*, Beijing, China, 2008, pp. 906–911.

[7] Roumeliotis, S. I. and Bekey, G. A.,"Distributed Multirobot Localization," *IEEE Transactions on Robotics and Automation*, Vol. 18, No. 5, 2002, pp. 781–795.

[8] Kurazume, R., Nagata, S. and Hirose, S.,"Cooperative Positioning with Multiple Robots," *Proceedings of the IEEE International Conference on Robotics and Automation*, San Diego, CA, May 1994, pp. 1250–1257.

[9] Martinelli, A., Pont, F. and Siegwart, R.,"Multi-Robot Localization Using Relative Observations," *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, Barcelona, Spain, 2005, pp. 2797–2802.

[10] Caglioti, V., Citterio, A. and Fossati, A.,"Cooperative, Distributed Localization in Multi-robot Systems: a Minimum-entropy Approach," *IEEE Workshop on Distributed Intelligent Systems*, 2006, pp. 25–30.

[11] Knuth, J. and Barooah, P.,"Distributed collaborative localization of multiple vehicles from relative pose measurements," *Forty-Seventh Annual Allerton Conference*, Illinois, USA, 2009, pp. 314–321.

[12] Nerurkar, E. D., Roumeliotis, S. I. and Martinelli, A.,"Distributed Maximum A Posteriori Estimation for Multi-robot Cooperative Localization," *Proceedings of the IEEE International Conference on Robotics and Automation*, Kobe, Japan, 2009, pp. 1402–1409.

[13] Nerurkar, E. D. and Roumeliotis, S. I.,"Multi-Centralized Cooperative Localization under Asynchronous Communication," *Department of Computer Science and Engineering, University of Minnesota*, Technical Report, March 2010.

[14] Howard, A., Mataric, M. J. and Sukhatme, G. S.,"Putting the 'I' in 'Team' - an Ego-Centric Approach to Cooperative Localization," *Proceedings of the IEEE International Conference on Robotics and Automation*, Taipei, Taiwan, 2003, pp. 868–874.

[15] Karam, N., Chausse, F., Aufrere, R. and Chapuis, R.,"Localization of a Group of Communicating Vehicles by State Exchange," *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, Beijing, China, 2006, pp. 519–524.

[16] Sharma, R. and Taylor, C.,"Cooperative Navigation of MAVs In GPS Denied Areas," *Proceedings of the IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, Seoul, Korea, 2008, pp. 481–486.

[17] Mariottini, G. L., Pappas, G., Prattichizzo, D. and Daniilidis, K.,"Vision-based Localization of Leader-Follower Formations," *Proceedings of the IEEE International Conference on Decision and Control*, Seville, Spain, 2005, pp. 635–640.

[18] Montesano, L., Gaspar, J., Santos-Victor, J. and Montano, L.,"Fusing vision-based bearing measurements and motion to localize pairs of robots," *ICRA Workshop on Cooperative Robotics,*, Barcelona, Spain, 2005.

[19] Indelman, V., Gurfil, P., Rivlin, E. and Rotstein, H., "Mosaic Aided Navigation: Tools, Methods and Results," *IEEE/ION PLANS*, CA, USA, May 2010, pp. 1212–1225.

[20] Indelman, V., "Navigation Performance Enhancement Using Online Mosaicking," *PhD thesis*, in preparation, Technion, Israel.

[21] Merino, L., Wiklund, J., Caballero, F., Moe, A., Ramiro, J., Forssen, E., Nordberg, K. and Ollero, A.,"Vision-Based Multi-UAV Position Estimation," *IEEE Robotics and Automation Magazine*, September 2006, pp. 53–62.

[22] Indelman, V., Gurfil, P., Rivlin, E. and Rotstein,H., "Real-Time Mosaic-Aided Aircraft Navigation: II. Sensor Fusion," *Proceedings of the AIAA Guidance, Navigation and Control Conference*, Chicago, IL, USA, 2009.

[23] Pugh, W., "Skip Lists: A Probabilistic Alternative to Balanced Trees," *Communications of the ACM*, Vol. 33, No. 6, 1990, pp. 668–676.

[24] Indelman, V., Gurfil, P., Rivlin, E. and Rotstein,H., "Navigation Aiding Based on Coupled Online Mosaicking and Camera Scanning," *Journal of Guidance, Control and Dynamics*, accepted.

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

[26] 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.

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

[28] 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.