2014 International Conference on Unmanned Aircraft Systems (ICUAS)
May 27-30, 2014. Orlando, FL, USA

On the Safe Navigation Problem for Unmanned Aircraft: Visual Odometry and Alignment Optimizations for UAV Positioning

Franz Andert, Nikolaus Ammann, Jan Püschel, Jörg Dittrich

Abstract— **With increasing automation of unmanned aircraft and the endeavor to fly between buildings in cities and in other occluded areas, safe navigation is essentially required but still a challenge. This paper is about the important issue of vehicle positioning in the case of satellite signal dropouts, and it presents a visual odometry method to compensate GPS positioning interruptions. The presented approach follows common triangulation principles and nonlinear optimization methods. Absolute scale is obtained by a stereo camera, although stereo is required only from time to time. In addition to the estimation of the vehicle position, the method estimates the camera alignment with respect to the vehicle, yielding a more accurate map and pose estimation. Returned vehicle poses are available in real-time with high update rates, being ready for an integration into state estimation and flight control. To demonstrate the algorithm properties, the paper incloses the evaluation of sensor data from unmanned helicopter flight tests. It shows the successful bridging of satellite positioning gaps by calculating the vehicle trajectory only by vision. Finally, the paper discusses some open issues for future work.**

I. Introduction

One of the current research challenges for vehicle automation is safe navigation, usually being solved using satellite and inertial sensors. Since satellite signal dropouts or range errors from reflections or multipath effects are very likely in urban environments or other occluded areas, navigation and thereby automatic operations of vehicles under such conditions become difficult. In order to be able to cope with all kind of scenarios for which satellite navigation means are not available (e.g. indoor) or not reliable enough (e.g. urban terrain and occluded areas), visual or vision-aided navigation principles are being developed, and their integration into real system applications has already begun. However, with regards to approval by the authorities, current small automatic aircraft systems do not yet meet airworthiness requirements.

The considered application in this paper is low-altitude outdoor exploration flights for unmanned helicopters like the testing vehicle shown in Fig. 1. Such vehicles can already operate in larger rural scenarios with a lot of free space, but they are also predestinated for operations in urban or natural canyons with many obstacles and other dangers. To develop solutions for the safe navigation problem, the paper distinguishes between various sub-challenges:

1) *Positioning:* The goal is to get vehicle pose updates from the sensor measurements as a basis for further filtering. It can also be seen as a part of the full state estimation within a navigation filter, but especially the position estimation depends on the specific sensor capabilities and data availability. Due to the variety of issues, this problem is tackled separately.
2) *Filtering and state estimation:* The goal is to provide real-time updates of the full vehicle state and its uncertainty. As already mentioned, it basically includes positioning, but it also includes the overall sensor fusion with inertial and satellite data, and real-time implementation problems.
3) *Closed-loop integration:* The final step is the combination of the state estimator and a flight controller, and their integration to a closed-loop solution. Tests can be done in a simulation environment, but final tests should be achieved with the full automatic vehicle to validate the proposed methods.

Fig. 1. Flight testbed for visual navigation. The picture shows DLR's 14kg helicopter ARTIS with a stereo camera and onboard computers [1].

Beside that, an overall solution includes a lot more problems such as the flight controller itself, collision avoidance to other vehicles or ground obstacles, or the practical part of running a complex vehicle as the engaged helicopter.

The three presented steps can be developed in the given order. This paper addresses algorithms that can be used to solve the first part (i.e. vehicle positioning), but the applicability to full state estimation and flight control is also considered throughout the paper. After describing the algorithmic approach, the paper presents UAV positioning results within temporary satellite dropouts. It also gives an outlook to the state estimator and the control integration concept which are currently under development.

Parts of the presented work are based on the project *Navigation zur Exploration von tiefffliegenden UAV in Katastrophenszenarien* (Navigation for exploration with low-flying UAVs in disaster scenarios, NExt UAV). The joint project was funded by the German Federal Ministry of Economic Affairs and Energy (BMWi) and administered by the space management of the German Aerospace Center (DLR) Bonn (support code 50 NA 1002)

F. Andert, N. Ammann, J. Dittrich are with the DLR (German Aerospace Center), Institute of Flight Systems, Unmanned Aircraft Dept., 38108 Braunschweig, Germany franz.andert, nikolaus.ammann, joerg.dittrich@dlr.de

J. Püschel is a former M.Sc. student from the Karlsruhe Institute of Technology (KIT), Institute of Photogrammetry and Remote Sensing, 76131 Karlsruhe, Germany jan.p.pueschel@gmail.com

II. Related Work

During the past ten years, many improvements have been achieved in the fields of unmanned aircraft automation and the use of cameras as movement sensors. Camera-based positioning can be a complement to satellite navigation that helps to compensate for a temporary decrease in positioning accuracy, or even an alternative solution for applications with no satellite navigation mean.

The idea of determining the aircraft position and motion with a camera goes back to remote sensing and time-consuming post-processing steps. With the technical evolution of computing resources, real-time onboard processing could be achieved, which enabled vision-aided navigation. An early approach with unmanned aircraft was is the direct integration of single tracked image features into a state estimator [2]. This algorithm is rather simple from the computer vision perspective, but it solved the hard challenge of closed-loop integration into a helicopter. A helicopter was stabilized in hovering flight and its inertial drift could be reduced in the case of GPS dropouts thanks to the use of this vision-augmented navigation. With an easy and widespread provision of flight hardware, dozens of papers have been published later on the way to solve the visual navigation problem. Kendoul's survey [3] provides an extensive list of related work.

Most newer navigation algorithms (e.g. [4], [5]) have in common that they process multiple image features and their movement characteristics, so that noise and irregularities like outliers are filtered. Since cameras measure the environment and not the motion itself, visual data processing is highly related to robotics and photogrammetry research where the challenge is is to measure the environment by means of that visual data. There are many works dealing with this category of problems, mostly named *structure from motion* (from the environment mapping perspective), *visual odometry* (from the camera movement estimation perspective), or *visual SLAM* (Simultaneous Localization And Mapping). A comprehensive overview about this topic is given in the tutorials by Fraundorfer and Scaramuzza [6], [7], which give a fairly complete and easily understandable introduction to this topic. It includes a large list of the most relevant work. With that, only specific relations to other works are given here and in the next sections describing the algorithms.

There is somehow a diffuse distinction between the two methods *odometry* and *SLAM*. Both concepts can be implemented with localization (i.e. camera resectioning) and mapping (i.e. object triangulation) methods (see e.g. [8]), based on correspondences between the current measurements (i.e. image feature points) and a *map* (i.e. object points) built out of previous measurements and, if available, known landmarks. In [6], the authors clearly distinguish between odometry and SLAM: while the term *odometry* is used for rather local map generation and incremental pose estimation, the term *SLAM* is more used for global map and trajectory generation, which includes additional consistency checks and therefore usually runs slower. In many cases, odometry is used when the user is mainly interested in real-time camera pose updates (where the map plays an accessory role, e.g. to handle satellite signal dropouts), and SLAM is used when the user is more interested in the map (with accessory poses, e.g. for remote sensing applications) or when a globally consistent mapping is required for correct pose updates (e.g. indoor navigation with loop closing). The presented odometry algorithm also includes localization and mapping, but with only local optimizations as described in [9].

All the vision-based algorithms have in common that absolute scales are not measured by a single camera. Triangulation always requires a reference scale, for example by a calibrated stereo camera, laser range (or other active) sensors, satellite positioning or known landmarks. And most of the approaches within the related work have a solution somehow. It can be argued that if the scale is determined once, a single camera might be sufficient for the rest of the flight. But in practice, scale also drifts from noise and further errors, and it should be re-estimated in certain intervals. In this paper, a stereo camera is used for scaling, even it is known that this is only a solution for rather short object distances, meaning flights at lower altitude.

III. Inertial, Visual, and Satellite Navigation

This section describes the general idea to implement a safe navigation algorithm. As already mentioned, core of this work is a positioning method, embedded into state estimation and flight control. Detailed considerations for algorithm design and implementation are the following:

- Application is outdoor navigation, which means to use common satellite and inertial navigation in the regular case. Hence, the visual odometry is designed to bridge satellite positioning gaps in the case of signal dropouts and to detect and exclude temporary errors like position jumps. It is further designed for unknown scenarios so that no a-priori information is used. The environment is assumed to be static, even though the algorithm might work with some visible dynamics that are filtered as noise or outliers.
- Global positioning consistency is assumed to be achieved by satellite navigation. This means, that local visual odometry algorithms are sufficient, without the need to optimize over the whole measuring history. Optimizations are related to local bundle adjustment [9].
- Contrary to many other approaches, the goal is not to estimate the camera position itself, but the vehicle position. The algorithm takes into account differences between camera and vehicle pose; this means that camera alignment issues are considered by design. A comparable optimization approach was presented in a previous work [10] to precisely navigate through a gate.
- The positioning output has to be capable of being integrated into a filter that runs in real-time to be used as a control input, so that the visual odometry has to run in real-time, too. This basically means to reduce the computational load, especially by limiting the number of image and map features. As proposed e.g.

in [11], localization, mapping, and also the sensor fusion filter are parallelized, so that the visual algorithms do never block a higher-ranking state estimator of the main thread. The presented data handling allows a visual odometry processing in soft real-time, i.e. it allows slight frame dropouts and calculation time jitters.

- As proposed e.g. in [12], the mapping method also takes advantages from both stereo vision, i.e. immediate 3D triangulation and absolute scale estimation, and monocular vision, i.e. more accurate multi-view triangulation from more distant viewpoints.
- The approach incorporates the uncertainty of sensor data which may not be constant over time by using probabilistic methods and weighting e.g. the reprojection error within optimization.

For an overall integration, the positioning is a part of sensor fusion, whose concept is to separate this into three separately executable modules. This means also that these processes can run on separated computers linked via network. Figure 2 gives an overview of the modules and their connections.

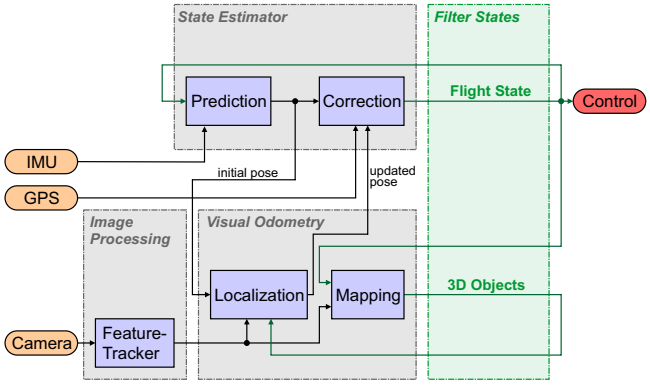

Fig. 2. The sensor fusion concept with three separate modules, states, inputs and outputs.

The first module is the *state estimator*. It is commonly using GPS and IMU data, optionally aided by a magnetometer to measure the heading angle without drift. The flight state is estimated using a Kalman Filter, whose prediction step is based on inertial strapdown calculation and where the correction step uses position and velocity information from GPS. For ease of use, this filter uses data which is pre-calculated in the receiver (RTK position and velocity if available), even though other research investigations (see [13]) take advantages from deeply coupling the satellite pseudo-ranges in order to detect further errors and to handle incomplete satellite constellations. To discriminate between the terms *state estimation* and *positioning*, the latter term refers to the provision of the correction terms to the state estimator which is used for the overall state estimation, even though the corrections include also velocity data (from GPS) or orientation angles (from visual odometry).

The second module deals with the *image processing*. It analyzes the camera images to get feature points and their relationship, which provides measurement input for the visual odometry module. There is a large variety of feature tracking or matching algorithms being capable of doing this, see the visual odometry tutorial [6]. Here, the OpenCV implementations of the Shi-Tomasi corner detector [14] and the Lucas-Kanade tracker [15] are chosen to provide image feature positions and their homologous matches over time and between left and right stereo images. An additional self-made image analysis checks whether features are suitable (e.g. to discard false matches and unsharp corners) and starts searching for new features again if their amount falls below a defined threshold. For every image, the output is a set of detected features with 2D position, uncertainty, and unique matching identifier (Fig. 3).

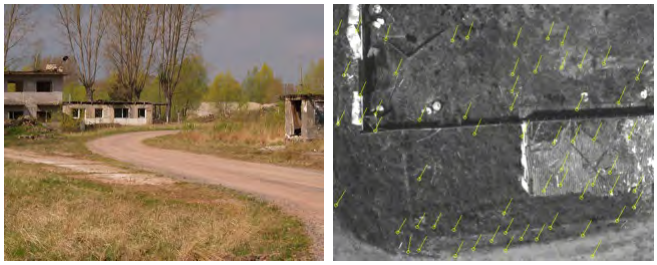

Fig. 3. Example of a flight testing area (left) and aerial image from downward-looking camera with marked features and their movements (right).

The third *visual odometry* module provides corrections to the state estimator by calculating the vehicle pose from the predicted state and the image feature measurements. This basically means to calculate the camera movement (i.e. the pose) by the characteristics of image feature movements. The algorithm follows visual odometry approaches and uses the image points from the feature tracker and the predicted vehicle pose from the state estimator. Output (from localization, see Fig. 2) is a camera pose update which is used to correct the predicted state. This is the main contribution, and the algorithm is detailed in the following section. It is then evaluated by the means of pre-recorded test image sequences. It is designed to be a rather small implementation which is independent of the other modules.

Beside the calculation modules, the block *filter states* in Fig. 2 describes the state which has to be estimated and stored internally. While a GPS/INS state estimator basically requires the vehicle state itself (including error states and biases etc.), this visual-aided estimation requires also to store the map of 3D object points that correspond with the measured image features.

IV. Pose Estimation with Visual Odometry

A. Method Overview

Visual pose correction follows common odometry concepts by using the relations between 2D image points and geodetic 3D object points. It consists of the steps

- *Localization:* The six degrees of freedom (DoF) of the vehicle pose are estimated with projecting the 3D objects. It is basically camera resectioning. As the results are an input for vehicle navigation, this should be done with high frequency, i.e. with camera image frame rate if possible.

- *Mapping:* Localization requires 3D points which are not available from the beginning. This step consists in triangulating the 3D objects from image data sets. Since no map is available at first, mapping must be done prior to localization with initial camera and vehicle poses. In many cases, not every image contains salient information, e.g. there are only slight changes between succeeding images. Hence, only a sub-set of images (key frames) is used to calculate the 3D points.
- *Optimization:* The estimated objects and vehicle poses are highly related to each other, and further optimization with local bundle adjustment is applied to the latest vehicle poses and 3D objects. This requires more calculation time than mapping and localization, so this is done parallel to the first two steps and with a further reduced update rate. If possible, the camera alignment is also optimized within this step.
- *Cleanup:* Due to calculation time limitations, the optimization will consider only recent measurements and not the whole history of objects and images. Old information is discarded to keep the number of 3D objects and the number of stored image points small and manageable (order of magnitude: 200 objects and 30 key frames with 50 to 100 features per frame).

To underline this principle, Fig. 4 illustrates the process. It shows the images taken along the flight path, the subset of images used for mapping and optimization, and the relationship to the 3D objects. The optimization with bundle adjustment takes only images that really require this parameter optimization, and needs only a subset of the visible objects.

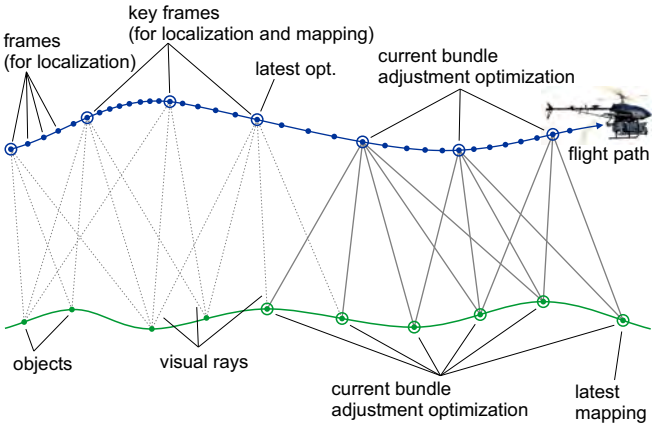

Fig. 4. Principle of the combined localization, mapping, and local bundle adjustment for optimization.

B. Image Projection and General Notations

Mathematical basis for visual odometry is the pinhole camera projection of geodetic scene points $\mathbf{q}_g : (x, y, z)^\top$ to image points $\mathbf{p} : (u, v)^\top$. The used coordinate systems are right-handed and Cartesian as illustrated in Fig. 5. Here, camera-fixed, flight carrier-fixed, and geodetic coordinates are used, being indexed to coordinates or transformation parameters. Rotations from coordinate system a to b are denoted as matrices $\mathbf{R}_{ba}$ or vectors $\mathbf{r}_{ba}$, translations as vectors (expressed in coordinate system a) $\mathbf{t}_{ba}$. For the current experiments within a limited operation area, a local geodetic system (axes: north, east, down) with origin inside is sufficient. However, this can be extended to global coordinate systems (e.g. earth-centered and earth-fixed).

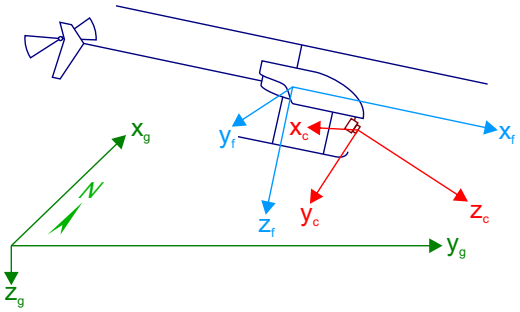

Fig. 5. Camera, flight carrier, and geodetic coordinates.

A camera with calibrated intrinsic parameters is assumed, so that the image points refer to a normalized image frame with the principal point $\mathbf{p}_0 = (0, 0)^\top$ and focal lengths $f_x = f_y = 1$. Image point output in these coordinates and by removing lens distortion is provided by the image processing module. With that, the projection of image points is notated

$$\lambda \begin{pmatrix} u \\ v \\ 1 \end{pmatrix}_c = \mathbf{R}_{cg} \begin{pmatrix} x - x_0 \\ y - y_0 \\ z - z_0 \end{pmatrix}_g \tag{1}$$

with the geodetic camera position $\mathbf{t}_{cg} = (x_0, y_0, z_0)^\top$ and rotation matrix $\mathbf{R}_{cg}$. A camera pose $[\mathbf{R}_{cg}, \mathbf{t}_{cg}]$ is composed by the camera alignment on the vehicle $[\mathbf{R}_{cf}, \mathbf{t}_{cf}]$ and the vehicle pose $[\mathbf{R}_{fg}, \mathbf{t}_{fg}]$ which is a part of the vehicle state. Eventually, the goal is to estimate this vehicle pose. It is

$$\begin{aligned} \mathbf{R}_{cg} &= \mathbf{R}_{cf} \cdot \mathbf{R}_{fg}, \\ \mathbf{t}_{cg} &= \mathbf{t}_{fg} + (\mathbf{R}_{fg}^\top \cdot \mathbf{t}_{cf}). \end{aligned} \tag{2}$$

In addition, the 3 DoF rotation vector (i.e. quaternion without real part) corresponding to a rotation matrix $\mathbf{R}_{cg}$ is notated $\mathbf{r}_{cg}$, analogous for the other indexes. To understand the algorithm description, some other terms are defined:

- A *feature* $\hat{\mathbf{p}} : [\mathbf{p}, \mathbf{\Sigma}_\mathbf{p}, i]$ stands for a single 2D image measurement. The tuple includes the image point position $\mathbf{p}$, its uncertainty covariance matrix $\mathbf{\Sigma}_\mathbf{p}$, and the identifier (ID) i. Identical identifiers denote homologous features from different images and their corresponding map objects.
- A *frame* $\mathbf{F}_t : \left[\{\hat{\mathbf{p}} | \hat{\mathbf{p}} \text{ detected in image}\}, \mathbf{r}_{fg}, \mathbf{t}_{fg}\right]_t$ describes the information extracted from one image. It comprises a set of features and the vehicle pose at timestamp t. Derived from that, also the camera pose at timestamp t and the rotation matrices belong to a frame.
- The *frameset* $\mathfrak{F} : \{\mathbf{F}\}_{1:t}$ is the stored collection of frames used for building the map. The set contains only the *key frames*, which are most relevant to build the map.
- An *object* $\hat{\mathbf{q}} : [\mathbf{q}_g, \mathbf{\Sigma}_{\mathbf{q}_g}, i]$ is a single geodetic 3D point with its uncertainty $\mathbf{\Sigma}_\mathbf{q}$. Objects are linked to features by their unique identifier i.
- The *map* $\mathbf{M} : \{\hat{\mathbf{q}}\}_{1:t}$ is the set of objects and forms a part of the filter state ($t = 1$ refers to the oldest

frame). It is usually meant that at timestamp t, the map is a calculation result based on measurements from the beginning. A-priori landmark information is not excluded by this definition, but it is not used.

C. Combined Monocular and Stereo Mapping

For every new image (or for one image of a stereo image pair), it is checked whether this frame is a key frame being relevant for updating the map. A key frame is a frame which contains at least one *key feature*, meaning:

- The feature is new, i.e. no feature with the same ID is found in all previous key frames of the frameset, or:
- An older frame, i.e. a recent key frame that contains a corresponding ID is found. If multiple corresponding frames are existent, only the newest frame in the frameset is relevant. A feature is now a key feature, if
  - the distance between the current and old frame's *camera positions* exceeds a defined threshold (typically few meters), and
  - the distance between the current and old image *feature positions* exceeds a defined threshold (typically several pixels).

For features that are not new, the minimal camera distance condition is applied due to the expected bad object triangulation from closer viewpoints. In practice, this means that not every new frame is used to update the map, and with a constant distance threshold, the proportion of key frames among all frames increases with flight speed (see Fig. 6). The second condition can be optionally used, and it is useful for flight missions at different altitudes where the flight is usually faster at higher altitudes. Since there will be larger image overlaps with decreasing feature movements, the amount of taken key frames can also be reduced. This is achieved by the second criterion mentioned above (on pixel distance). This criterion is preferable for a time-based downsampling, because it can be sometimes useful to have smaller key frame time differences (e.g. high speed at low altitude), while in the other, more relaxing case (low speed at high altitude), computational load is saved.

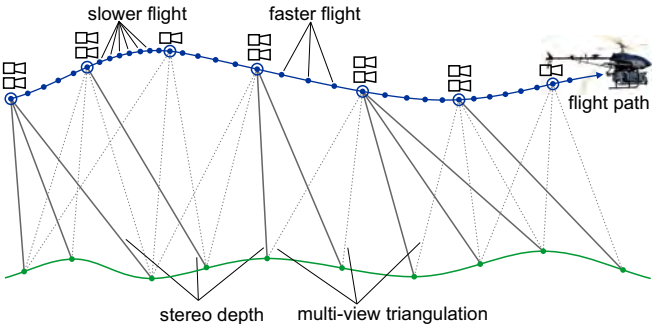

Fig. 6. Combination stereo triangulation for new key features and monocular multi-view triangulation for existing key features. Since key frames are updated by a position distance threshold, faster flights yield a higher update rate for mapping. The localization frame rate remains constant.

Usually, most of the features are key features within the same key frames, or no key feature has been identified and the frame is not used for mapping. After the image has been processed to identify key features, a new key frame is created with all the key features and added to the frameset. As depicted in Fig. 6, the basic mapping is done as follows, separately for every key feature:

1) New Key Features: The basic mapping step means object triangulation. For a new feature, this is only possible if it belongs to a valid stereo pair. Otherwise, i.e. for monocular camera use, or if no valid stereo match is available for this specific feature, no mapping can be done. However, this key feature remains in the frameset to be mapped later.

The stereo mapping step is currently implemented for standard-stereoscopic camera settings with a known baseline length b. Following the common stereo literature (e.g. [16]), the object triangulation is based on the left image feature $\mathbf{p}_l = (u, v)^\top$ and the horizontal disparity d to the right image feature $\mathbf{p}_r = (u+d, v)^\top$. Features with low horizontal disparity (pointing to far and uncertain objects) and with vertical disparity above typical noise are discarded. For a stereo feature with $\mathbf{p}_l$ and $\mathbf{p}_r$, the camera-centric point $\mathbf{q}_c = (x_c, y_c, z_c)^\top$ is

$$\begin{aligned} z_c &= b/d \quad \text{(remember: } f = 1\text{)} \\ x_c &= u \cdot z_c \\ y_c &= v \cdot z_c \end{aligned} \tag{3}$$

with uncertainty covariance matrix

$$\boldsymbol{\Sigma}_{\mathbf{q}_c} = \begin{pmatrix} z_c^2 \boldsymbol{\Sigma}_{\mathbf{p}_l,11} & 0 & 0 \\ 0 & z_c^2 \boldsymbol{\Sigma}_{\mathbf{p}_l,22} & 0 \\ 0 & 0 & z_c^4 \boldsymbol{\Sigma}_{\mathbf{p}_l,11}/b^2 \end{pmatrix}. \tag{4}$$

This point is then transferred to geodetic coordinates with

$$\mathbf{q}_g = \mathbf{R}_{cg}^\top \mathbf{q}_c + \mathbf{t}_{cg} \tag{5}$$

and

$$\boldsymbol{\Sigma}_{\mathbf{q}_g} = \mathbf{R}_{cg}^\top \boldsymbol{\Sigma}_{\mathbf{q}_c} \mathbf{R}_{cg} + \boldsymbol{\Sigma}_{\mathbf{t}_{cg}} + \|\mathbf{q}_c\|^2 \boldsymbol{\Sigma}_{\mathbf{r}_{cg}}. \tag{6}$$

In eq. 6, the covariance matrix is transformed to the principal geodetic axes. Additionally, the camera position uncertainty $\boldsymbol{\Sigma}_{\mathbf{t}_{cg}}$ (constant for one image) is added, and the camera rotation uncertainty $\boldsymbol{\Sigma}_{\mathbf{r}_{cg}}$ is added, scaled with the distance between camera and object (i.e. norm of the camera-centric point $\mathbf{q}_c$). With that, sensor and vehicle uncertainties are considered for the object point generation.

2) Existing Key Features: For key features $\hat{\mathbf{p}}$ that were already existing in previous frames of the frameset, triangulation from multiple images is performed to estimate $\hat{\mathbf{q}}_g$. Comparing to stereo imagery, the uncertainty of $\hat{\mathbf{q}}_g$ should be decreasing, since multiple images can be used and since the distances between the viewpoints are higher than the baseline from stereo (without regarding map errors due to prior localization errors). For the triangulation, a minimal and maximal number of (the most recent) corresponding key features is defined to take advantages from multi-view geometry, but not to triangulate over the whole feature history which might be inefficient. Stereo information is used by treating the left and right frame as two monocular key

frames. Here, the left and right number of features can be unequal due to missing or invalid stereo correspondences.

The estimation of $\hat{\mathbf{q}}_g$ is done with iterative linear L_2 triangulation as described in [17] and [18]. Adapted to the presented projection, $\hat{\mathbf{q}}_g$ is calculated as follows:

Let $\boldsymbol{\rho}_{ij}$ be row vectors with the i-th row of $\mathbf{R}_{cg}$ and $\mathbf{t}_j = \mathbf{t}_{cg}$ of the j-th frame. It is now

$$\begin{aligned}(u_j\boldsymbol{\rho}_{3j} - \boldsymbol{\rho}_{1j}) \cdot (\mathbf{q}_g - \mathbf{t}_j) = 0,\\ (v_j\boldsymbol{\rho}_{3j} - \boldsymbol{\rho}_{2j}) \cdot (\mathbf{q}_g - \mathbf{t}_j) = 0.\end{aligned} \tag{7}$$

For $m \geq 2$ correspondences $\hat{\mathbf{p}}_j$ from different key frames $j = 1, \ldots, m$, and beginning with the scale factors $\lambda_1 = \ldots = \lambda_m = 1$, the resolution of the linear system

$$\begin{pmatrix} \frac{1}{\lambda_1}(u_1\boldsymbol{\rho}_{31} - \boldsymbol{\rho}_{11}) \\ \frac{1}{\lambda_1}(v_1\boldsymbol{\rho}_{31} - \boldsymbol{\rho}_{21}) \\ \vdots \\ \frac{1}{\lambda_m}(u_m\boldsymbol{\rho}_{3m} - \boldsymbol{\rho}_{1m}) \\ \frac{1}{\lambda_m}(v_m\boldsymbol{\rho}_{3m} - \boldsymbol{\rho}_{2m}) \end{pmatrix} \cdot \mathbf{q}_g = \begin{pmatrix} \frac{1}{\lambda_1}(u_1\boldsymbol{\rho}_{31} - \boldsymbol{\rho}_{11})\mathbf{t}_1 \\ \frac{1}{\lambda_1}(v_1\boldsymbol{\rho}_{31} - \boldsymbol{\rho}_{21})\mathbf{t}_1 \\ \vdots \\ \frac{1}{\lambda_m}(u_m\boldsymbol{\rho}_{3m} - \boldsymbol{\rho}_{1m})\mathbf{t}_m \\ \frac{1}{\lambda_m}(v_m\boldsymbol{\rho}_{3m} - \boldsymbol{\rho}_{2m})\mathbf{t}_m \end{pmatrix} \tag{8}$$

permits to estimate $\mathbf{q}_g$, which is usually done with singular value decomposition methods. For refinement, the λ_j are updated:

$$\lambda'_j = \boldsymbol{\rho}_{3j}(\mathbf{q}_g - \mathbf{t}_j). \tag{9}$$

With the new values for the λ_j, the triangulation is repeated iteratively until $\sum_{j=1}^{m} |\lambda_j - \lambda'_j| < \epsilon$ indicates scale factor convergence.

The uncertainty $\boldsymbol{\Sigma}_{\mathbf{q}_g}$ is formed by the projection derivatives with respect to the point inputs. Let $f : \mathbf{q}_g = f(\mathbf{p}_1, \ldots, \mathbf{p}_m)$ denote the transformation function describing the triangulation from above. Its Jacobian $\mathbf{J}$ is built, and the triangulation uncertainty is now

$$\boldsymbol{\Sigma}_{\mathbf{q}_g} = \mathbf{J} \begin{pmatrix} \boldsymbol{\Sigma}_{\mathbf{p}_1} & \ldots & \mathbf{0}_{2\times 2} \\ \vdots & \ddots & \vdots \\ \mathbf{0}_{2\times 2} & \ldots & \boldsymbol{\Sigma}_{\mathbf{p}_m} \end{pmatrix} \mathbf{J}^\top + \frac{1}{m}\left(\overline{\boldsymbol{\Sigma}_{\mathbf{t}_j}} + \overline{\|\mathbf{q}_c\|^2 \boldsymbol{\Sigma}_{\mathbf{r}_j}}\right) \tag{10}$$

As in eq. 6, camera pose uncertainty is considered, here by averaging over the m key frames. With the factor $\frac{1}{m}$, the effect of decreased position noise influence with increasing m is taken into account.

D. Localization

While the map is only updated for a selection of key frames, the estimation of the vehicle pose is done for every new frame $\mathbf{F}$ from the camera sequence. This refers to camera resectioning, implemented using 2D-3D feature-to-object correspondences. This can usually be initialized with the existing object and feature values, and the predicted vehicle pose from the filter. If not available, linear algorithms (in this case, a RANSAC-augmented direct linear transform [19]) can provide initialization values.

This initial pose is optimized by nonlinear methods where the reprojection errors of the objects are minimized. The estimation is performed with a Levenberg-Marquardt (LM) algorithm. Following [20], a further optimization is done by removing correspondences with large residuals as outliers. If outliers were existent, an additional LM step is performed with the remaining sub-sets of features and objects. To account for the pixel and object uncertainty, the chosen weighted distance metric considers the covariances. From eq. 1, a reprojection $\mathbf{p}'$ is

$$\lambda \begin{pmatrix} \mathbf{p}' \\ 1 \end{pmatrix} = \mathbf{R}_{cg}\left(\mathbf{q}_g - \mathbf{t}_{cg}\right), \tag{11}$$

the covariance $\boldsymbol{\Sigma}_{\mathbf{p}'}$ is calculated by

$$\lambda^2 \begin{pmatrix} \boldsymbol{\Sigma}_{\mathbf{p}'} & \\ & 1 \end{pmatrix} = \mathbf{R}_{cg}\boldsymbol{\Sigma}_{\mathbf{q}_g}\mathbf{R}_{cg}^\top. \tag{12}$$

The reprojection error d between $\mathbf{p}'$ and the measurement $\mathbf{p}$ is now a kind of Mahalanobis distance with

$$d^2 = (\mathbf{p} - \mathbf{p}')^\top \left(\boldsymbol{\Sigma}_{\mathbf{p}} + \boldsymbol{\Sigma}_{\mathbf{p}'}\right)^{-1} (\mathbf{p} - \mathbf{p}'). \tag{13}$$

In this simple localization case, d is a function dependent on the latest vehicle pose, with constant objects and camera alignment. Using eqs. 11–13 above and eq. 2, a function $d_i : d_i = d_i(\mathbf{t}_{fg}, \mathbf{r}_{fg})$ for the i-th feature and object can be formed, parameterized by the vehicle pose. With that, the LM optimizes over

$$\sum_{\hat{\mathbf{p}}_i \in \mathbf{F}} d_i(\mathbf{t}_{fg}, \mathbf{r}_{fg}) \tag{14}$$

to estimate these six pose parameters $(\mathbf{t}_{fg}, \mathbf{r}_{fg})$. Note that this will directly return the vehicle pose and not the camera pose as done in many other works. Since only six parameters are optimized here, the localization runs fast enough for real-time use for typical computers and camera frame rates. The LM also returns a Jacobian for uncertainty calculation.

E. Nonlinear Map and Pose Optimization

After several key frames have been mapped, the map and the camera and vehicle poses are optimized. This is done in a parallel process so that the basic mapping remains available with the required frequency. According to [9], local bundle adjustment is performed over the last m key frames. It is noted that m can be higher than the maximal number of key frames used for the basic triangulation described in sec. IV-C. This subset of key frames is also denoted with indexes $j = 1, \ldots, m$ for ease of use. The adjustment means basically to optimize relevant objects *and* poses with the LM, initialized with the given object positions from triangulation or a previous optimization. The optimization has two different modes: one to enhance the vehicle poses and the other to estimate an optimal camera alignment.

1) Vehicle Poses Optimization: The map is optimized with the recent vehicle poses when required by the sensing conditions, especially when no or only erroneous absolute positions are given by GPS. With only triangulation and the localization from camera resectioning (sec. IV-D), map and trajectory may become inconsistent due to ambiguities. Such effects are reduced with this optimization.

The principle is similar to the LM optimization done for the fast localization. In this case, the reprojection error function depends on the keyframe's vehicle pose (6 parameters each) and all the objects $\mathbf{q}_g$ of the map $\mathbf{M}$ (3 parameters each), which are visible by at least one key frame. As in

eq. 14, the function d_{ij} denotes the reprojection error of the i-th feature in the j-th frame, dependent on the parameter set. The LM optimizes now over

$$\sum_{\hat{\mathbf{p}}_i \in \mathbf{F}_j} \sum_{j=1}^{m} d_{ij}\big([\mathbf{t}_{fg}, \mathbf{r}_{fg}]_j, \{\mathbf{q}_g\}\big). \tag{15}$$

To reduce the parameter set, a pose $[\mathbf{t}_{fg}, \mathbf{r}_{fg}]_j$ is not optimized if this j-th key frame position was marked as *sufficient*. This flag is set for a key frame if the filter could provide a *corrected* state based on GPS, or based on a previous LM optimization where other GPS-corrected poses were available. As a start, sufficient key frames remain in the optimization due to their significant influence on the object positions. Object and also pose covariance is achieved analogous to eq. 10 since a Jacobian of the projection function is available from the LM.

2) Camera Alignment Optimization: Usually, the camera alignment $[\mathbf{r}_{cf}, \mathbf{t}_{cf}]$ is set before the flight (hand-to-eye calibration). It is not very practical for applicational use where missions are time-critical and a calibration setup is not available in the field. Hand measurements of distances and angles between camera and IMU are a standard procedure, but this is of course inaccurate. Beside that, the measured camera alignment can be significantly and dynamically biased if the vehicle pose estimation is not true. Hence, the camera alignment should be automatically optimized during the flight.

There are ambiguities when optimizing camera and vehicle poses at the same time, so this is only done if enough (usually: all) key frames have already a pose estimate with sufficient accuracy. This case is assumed here when correct satellite data is available over the regarded key frames. The presented optimization does not account for the relative camera position, since hand-measuring reaches centimeter accuracy or better, which is not significant for the object positions. Much higher errors are due to a rotational inaccuracy. Hence, the LM optimizes over the three rotational parameters and the objects, i.e. it minimizes

$$\sum_{\hat{\mathbf{p}}_i \in \mathbf{F}_j} \sum_{j=1}^{m} d_{ij}\big(\mathbf{r}_{cf}, \{\mathbf{q}_g\}\big). \tag{16}$$

Note that this will only work if the flight path is curvy enough. Otherwise, the optimization may converge with a false rotation around the axis along the roughly linear flight path, which rotates the map as well. To overcome this, one could create special camera calibration flights before the proper mission where curves are flown in an area where correct satellite positioning is expected.

F. Cleaning up the Map and the Frameset

As mentioned in the method overview and as depicted in Fig. 4, only newer frames and objects are optimized by local bundle adjustment. This means that after the steps of localization, mapping, and optimization, a cleanup procedure is run which discards outdated frames and objects as follows:

- A timestamp is defined for each object, it marks the most recent timestamp of an update from mapping. Every time a key frame is added, all the visible object timestamps are updated to the time of the key frame. This also affects to objects visible from features which are *no* key features because they are visible at this update, too.
- The current age of an object is now the difference between current key frame timestamp and object timestamp. Logically, a currently visible object has the age of zero, and the age increases with every new key frame from where the object is not visible. If an object regains visibility later (which depends on the feature tracker whether this is possible), the age is resetted to zero.
- A maximal object age is defined. To increase the speed of map optimization, this should be a low number, e.g. from 5 to 10 frames. Old objects beyond this threshold are discarded from the map before each optimization.
- By removing old objects from the map, key frames may lose correspondences. It is now checked whether key frames do still have some. If not, these key frames are also removed from the frameset.

These steps yield a cleanup of old objects and key frames over the time, which prevents the frameset and map from becoming too large and the optimization from becoming too slow due to the growing number of inputs.

V. Experiments

After describing the positioning algorithm description, this section introduces the experimental test bed. After that, the applicability of the visual odometry method is evaluated by generating vehicle trajectories from image sequences and by comparing them to the GPS/INS flight path.

A. Test Carrier

The evaluation is based on data recorded on unmanned helicopter flights. Testing system is a helicopter of the ARTIS (Autonomous Rotorcraft Testbed for Intelligent Systems) family, operated by the DLR Institute of Flight Systems. See [1] for an overview about the system capabilities and further research activities using ARTIS. Figure 7 shows a close view of the payload hardware used for the experiments.

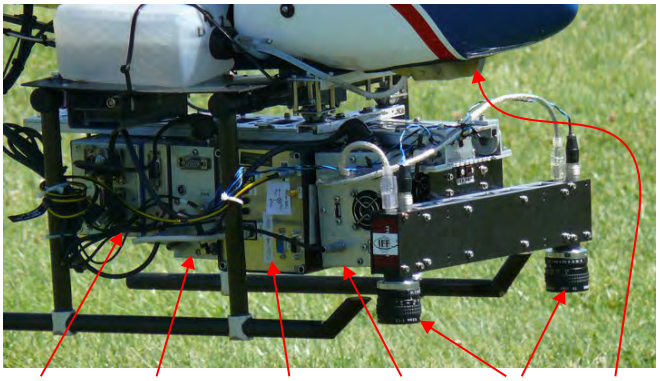

Fig. 7. Close view of the helicopter with navigation and image processing.

Some facts about the used helicopter platform are listed in Table I, the detailed configuration is given later in the flight test descriptions. For data recording, the vehicle is equipped with a navigation computer (for satellite and inertial data recording and for the state estimation module) and with an image processing computer (for image recording as well as for the feature tracking and visual odometry modules). The vehicle is operated from a ground control station and a safety pilot can take the control at any time.

Helicopter	
basis vehicle	Benda Genesis model helicopter
motor drive	1.5 kW combustion engine
rotor diameter	1.9 m
max. weight	14 kg (incl. 4 kg avionics + 2 kg image processing)
data links	R/C control, WiFi, FreeWave modem

TABLE I

INFORMATION ABOUT THE ARTIS HELICOPTER.

B. Specifying Parameters with Training Data

As presented in section IV, all the algorithms have to be tuned, and there are no general guidelines to find out the best values for their parameters. While the order of magnitude of the parameters may be offered up by hand, suitable values are identified by multiple runs of the algorithm on a training data sample, with different parameter configurations. The identified parameters are listed in Table II and will be used in the following evaluation of the test data sets.

Feature Tracking	
number of tracked features	50–80
feature distance	$\geq$25 pixels
stereo disparity	$\geq$6.0 pixels horz. / $\leq$ 2.0 pixels vert.
feature uncertainty	0.5 pixels / f (u and v uncorrelated)
Mapping Parameters	
vehicle distance between key frames	$\geq$1.0 m
feature distance between key features	$\geq$10 pixels / f
triangulation images	15–30
triangulation iterations	max. 10, $\epsilon = 10^{-3}$
LM optimization	every 10 key frames
maximal object age (cleanup)	15 key frames

TABLE II

GOOD ALGORITHM PARAMETERS.

Finding suitable parameters is often a compromise between localization and mapping accuracy, global map consistency, and of course processing time. With the listed parameters, the number of key frames in the frameset is usually between 20 and 50, with around 100 to 150 objects in the map. This number does not increase over time due to the cleanup method, and the average processing time remains constant. It was observed that feature tracking, localization, and simple mapping require each around 5 ms processing time (referred to the onboard computer), which can be processed with up to approximately 60 frames per second. Optimization, however, is calculated within 70 to 170 ms. But since this is processed with only every 10th key frame (which is usually much less than every 10th image), a parallel processing will be real-time capable if the vehicle dynamics do not require higher frame (and key frame) rates.

C. Flight Test and Evaluation

The analysis of the feature tracking and visual odometry modules is first of all done separate from the state estimator. Both modules run onboard the ARTIS image processing computer. In this paper, the algorithms are feeded with test data recorded from flight tests. In each of the two presented tests, the inputs are a stereo image sequence, and the GPS/INS state estimation. First of all, the behavior of feeding the visual positioning into the state estimation is not tested, and the mapping procedure takes the odometry positions as input. With that, this evaluation shows positioning results that are only based on visual data, initialized with the GPS/INS state as follows:

- Image and state data begins within flight, after take-off.
- The beginning of the sequence, here within the first 300 images, is used to combine visual and GPS/INS data for camera alignment optimization and to compare this the visual odometry without alignment optimization. This is the usual case where all data is available. In this initialization phase, the vehicle position is not optimized, and the plots show the GPS/INS trajectory.
- After that, a GPS dropout is simulated. Beginning with the current state, the visual odometry module now estimates the vehicle pose over time. This phase demonstrates the vision-only positioning behavior in the case of a satellite signal dropout, which is shown in the plots with the GPS/INS positioning as a reference.

Below, two flights with equal algorithm parameter values, but with different vehicle configurations are evaluated. The presented data analysis and image-based positioning is done in a post-processing step from recorded image and navigation sensor data, given are the parameters set within the specific flight.

1) First Flight: This test uses a GPS/INS developed by the TU Braunschweig [21] within the *NExt UAV* project cooperation and a custom-built stereo camera. Table III shows the hardware main specifications. Flight altitude is around 10 m to 20 m above ground, with a speed of approximately 2 m/s. The flight includes a short stop with hovering.

State Estimation	
sat. receiver	uBlox-6 + GPS+SBAS (4 Hz, LEA 6T board)
accelerometers	2x Bosch SMB 225 (2-axis MEMS), 100 Hz
gyroscopes	3x Bosch SMG 074 (1-axis MEMS), 100 Hz
nav. processing	Intel Atom, picoITX board
op. system	Linux (Ubuntu, realtime)
Image Processing System	
cameras	2x AVT Marlin F-131B, 30 cm baseline
resolution	640 $\times$ 512 pixel
frame rate	10 Hz (external trigger, $\pm$ 2 μs)
lens	8.5 mm (field of view: 55° $\times$ 45°)
image proc.	Intel Core-2, PC/104 board
op. system	Linux (openSUSE)

TABLE III

HARDWARE CONFIGURATION USED IN THE FIRST FLIGHT TEST.

Results are shown in Fig. 8, the 2D-plot shows the GPS/INS path as a reference to visual odometry. Data is

evaluated with and without camera alignment optimization during the initialization phase. Fig. 9 plots the errors to the reference.

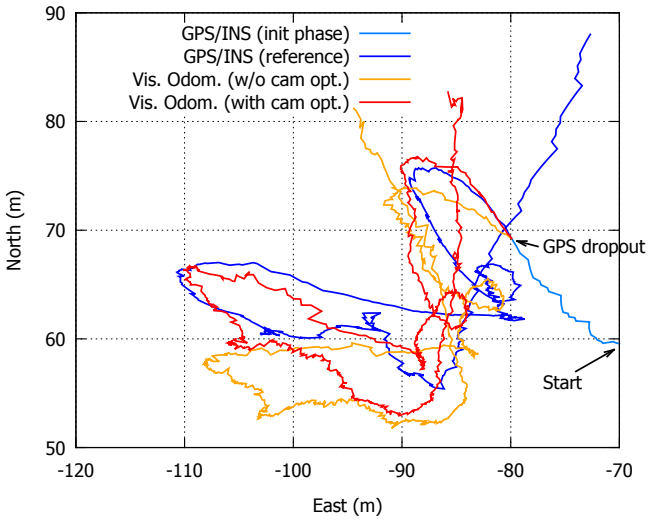

Fig. 8. Position estimation with GPS/INS and visual odometry over 70 s (700 frames). GPS dropout starts at N = 69.2 m, E = -79.9 m.

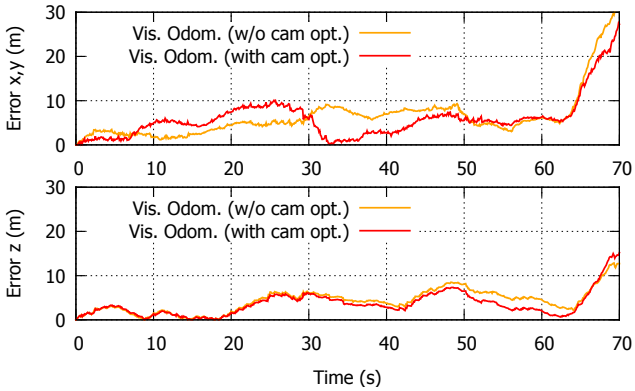

Fig. 9. Horizontal (Euclidean distance) and vertical errors of the visual solution during the GPS dropout, with respect to GPS/INS reference.

2) Second Flight: The second flight is performed with the ARTIS standard avionics hardware, conventionally used for flight control and other research investigations. Beside that, a light off-the-shelf stereo camera records the images, see Table IV.

State Estimation	
sat. receiver	NovAtel OEM4 (20 Hz, RTK GPS solution)
IMU	Inertial Science ISIS IMU, 100 Hz
magnetometer	Honeywell HMR 2300
nav. processing	Intel Pentium4, PC/104 board
op. system	QNX (realtime)
Image Processing System	
cameras	Videre Design STOC, 30 cm baseline
resolution	640 × 480 pixel
frame rate	30 Hz (internal sync)
lens	4.0 mm (field of view: 50° × 38°)
image proc.	Intel Core-2, PC/104 board
op. system	Linux (openSUSE)

TABLE IV
HARDWARE CONFIGURATION USED IN THE SECOND FLIGHT TEST.

Flight altitude and velocity are similar to the first one, without stopping the vehicle. Figs. 10 and 11 show the results in the same manner as in the first flight test.

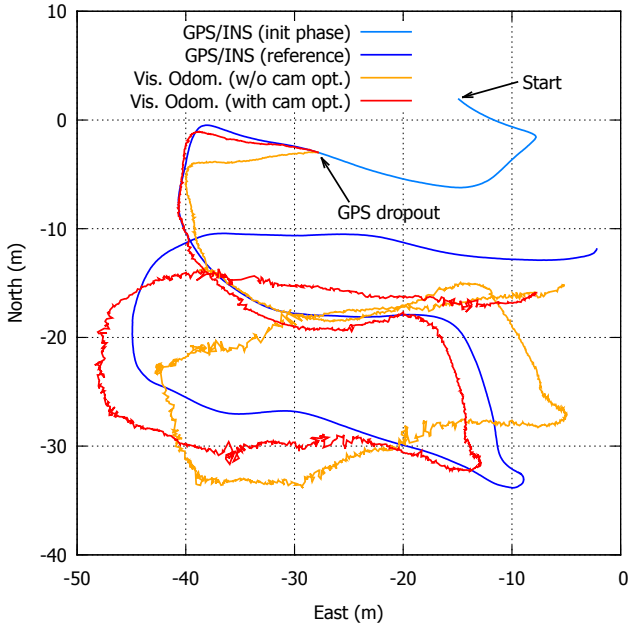

Fig. 10. Position estimation with GPS/INS and visual odometry over 55 s (1662 frames). GPS dropout starts at N = -3.0 m, E = -27.8 m.

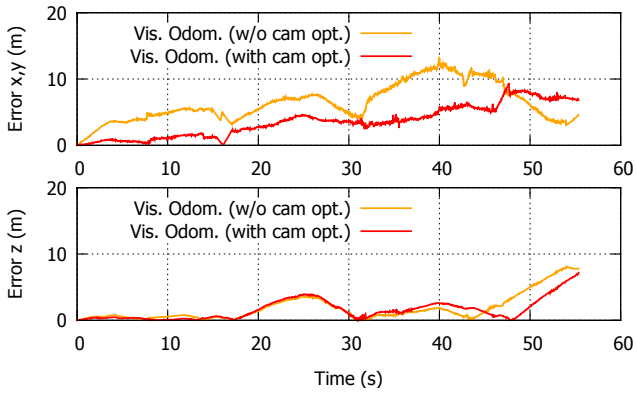

Fig. 11. Horizontal (Euclidean distance) and vertical errors of the visual solution during the GPS dropout, with respect to GPS/INS reference.

Interpretation of the Results: Initially, it can be noticed that differences of the two sensing systems for state estimation become visible by comparing the GPS/INS paths of both plots in the Figs. 8 and 10. While the first trajectory estimation is somehow choppy, the second one is much smoother and thus likely less erroneous. Presumable reason is the used low-cost hardware used in the first flight, yielding to significant INS drift errors and position jumps from GPS corrections. However, from the perspective of visual odometry evaluation, both flights are very similar.

It is shown in both cases that the visual odometry algorithm is capable of estimating the position, which is influenced by accumulating translational and rotational errors. This kind of drift is expected and comparable to other work dealing with visual navigation methods. The observed position error is about 5 m to 10 m after one minute flight time. Note that the odometry is only based on vision and not augmented by inertial sensing which will be done in the

future to further reduce the error in the case of satellite signal dropouts. It is obvious that after rotational errors, the position error to the reference will continuously increase, e.g. visible in Fig. 9 at the end of the observed flight. Within loops, this effect on increasing position errors is reduced.

Beside that, the positive effect of the alignment estimation can be demonstrated. Possible reason for an initially wrong camera alignment is not only the hand-to-eye calibration itself, but also false estimates of the vehicle's attitude angles during the flight. Especially the estimated heading (rotation around the z_g-axis) differs from the true value when a IMU is used as the only rotation sensor because of the rotational drift. Even with the help of a magnetometer as used in the second flight, the rotation estimation is significantly biased. With that, the transformation between camera and vehicle-fixed coordinate systems differs from the initial assumption, and it seems to be not constant over time. By comparing the visual odometry path calculated without camera alignment optimization with the GPS/INS path, the flight directions differ by around 20 degrees. This is processed as camera misalignment, and its correction leads to a much more suitable path. But since the camera was definitively not rotated sidewards, the observed deviation is more an indicator for an erroneous vehicle heading estimation, presumably from bad inertial strapdown and magnetometer measurements. If the camera alignment is corrected within GPS phases, the visually estimated path begins with the same direction as the GPS/INS path. In summary, visually estimated positions can be suitable corrections of the estimated vehicle state.

VI. Summary

The paper deals with the improvement of unmanned aircraft state estimation with visual sensors. Common satellite and inertial fusion comes with some drawbacks, especially due to satellite positioning errors and dropouts in the proximity of obstacles and unsuitable drift rates from small and lightweight inertial measurement units. In unknown scenarios, visual algorithms are not measuring absolute positions, but they can help to estimate relative movements in order to reduce the accumulation error from inertial strapdown. In this context, the paper presents a visual odometry algorithm which uses typical localization and mapping principles and which is designed to be integrated into the state estimator. In addition to the odometry solution that follows fairly known principles, the paper's contribution is to estimate the camera alignment relative to the vehicle pose during the flight. Camera misalignment causes false transformations between camera and vehicle and thereby especially rotated odometry trajectories. It is shown here that this error is significant if the vehicle attitude is not estimated correctly, and that such rotational errors can be reduced by estimating the camera alignment. This can be done within the flight if sufficient data is present. To solve for ambiguities, this utilizes satellite and visual positioning in flight phases where both are available. Hence, applications are first of all outdoor flights where satellite positioning is possible, and where temporary interruptions can still be compensated using rather low-cost inertial sensors. Flight test results show the capabilities of the visual algorithm. Straightforwardly, future work will tackle the the full integration into the state estimator and automatically controlled flights of an unmanned helicopter through canyons with bad satellite reception.

In terms of safer flights or future certification of small unmanned aircraft, there are more remaining challenges to be solved in future work. Optical techniques highly depend on the visible environment, which means that current systems are not able to operate within all kinds of scenarios. For example, critical issues are bad vision conditions (fog, night) or the correct handling of dynamic objects (road vehicles, crowds, water). Another aspect is the increase of robustness and the confident provision of quality information. Hence, the safe, stable and reliable navigation for automatic vehicles remains still an important challenge.

References

[1] F. Adolf, et al., "An unmanned helicopter for autonomous flights in urban terrain," in *Advances in Robotics Research*, T. Kröger and F. M. Wahl, Eds. Berlin: Springer, 2009, pp. 275–285.

[2] A. Koch, H. Wittich, and F. Thielecke, "A vision-based navigation algorithm for a VTOL UAV," in *AIAA GNC Conf.*, 2006.

[3] F. Kendoul, "Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems," *J. Field Rob.*, vol. 29, no. 2, 2012.

[4] S. Weiss, et al., "Real-time onboard visual-inertial state estimation and self-calibration of MAVs in unknown environments," in *IEEE International Conference on Robotics and Automation*, 2012.

[5] G. Chowdhary, E. N. Johnson, and D. Magree, "GPS-denied indoor and outdoor monocular vision aided navigation and control of unmanned aircraft," in *CEAS EuroGNC Conference*, 2013.

[6] D. Scaramuzza and F. Fraundorfer, "Visual odometry – part I," *IEEE Robotics & Automation Mag.*, vol. 18, no. 4, pp. 80–92, 2011.

[7] F. Fraundorfer and D. Scaramuzza, "Visual odometry – part II," *IEEE Robotics & Automation Mag.*, vol. 19, no. 2, pp. 78–90, 2012.

[8] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1052–1067, 2007.

[9] E. Mouragnon, et al., "Generic and real-time structure from motion using local bundle adjustment," *Image and Vis. Comp.*, vol. 27, 2009.

[10] F. Andert, F.-M. Adolf, L. Goormann, and J. S. Dittrich, "Autonomous vision-based helicopter flights through obstacle gates," *UAV 09 Symposium / J. of Int. and Rob. Syst.*, vol. 57, no. 1-4, pp. 259–280, 2010.

[11] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *Int. Symp. on Mixed and Augmented Reality*, 2007.

[12] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar, "Vision-based state estimation and trajectory control towards high-speed flight with a quadrotor," in *Robotics: Science and Systems*, 2013.

[13] F. Andert, et al., "A flight state estimator that combines stereo-vision, INS, and satellite pseudo-ranges," in *Advances in Aerospace Guidance, Nav. and Control*, Q. Chu et al., Ed., Springer, 2013, pp. 277–296.

[14] J. Shi and C. Tomasi, "Good features to track," in *IEEE Conference on Computer Vision and Pattern Recognition*, 1994, pp. 593–600.

[15] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *International Joint Conference on Artificial Intelligence*, 1981, pp. 674–679.

[16] L. Matthies and S. A. Shafer, "Error modeling in stereo navigation," *IEEE Journal of Rob. and Autom.*, vol. 3, no. 3, pp. 239–248, 1987.

[17] R. Hartley and P. Sturm, "Triangulation," *Computer Vision and Image Understanding*, vol. 68, pp. 146–157, 1997.

[18] F. Lu and R. Hartley, "A fast optimal algorithm for L_2 triangulation." in *Asian Conf. on Computer Vision*, 2007.

[19] Y.-H. Kwon, "Direct linear transform method. www.kwon3d.com /theory/dlt/dlt.html," 1998.

[20] N. Sünderhauf, K. Konolige, S. Lacroix, and P. Protzel, "Visual odometry using sparse bundle adjustment on an autonomous outdoor vehicle," in *Autonome Mobile Systeme*, 2005, pp. 157–163.

[21] S. Batzdorfer, et al., "Using combined IMU / stereo vision / cooperative GNSS system for positioning of UxV swarms within catastrophic urban scenarios," in *ION Pacific Meeting on Pos., Nav., and Timing*, 2013.