J Intell Robot Syst (2009) 55:299–321
DOI 10.1007/s10846-008-9304-8

Visual 3-D SLAM from UAVs

Jorge Artieda · José M. Sebastian · Pascual Campoy · Juan F. Correa · Iván F. Mondragón · Carol Martínez · Miguel Olivares

Received: 30 May 2008 / Accepted: 1 December 2008 / Published online: 15 January 2009

Abstract The aim of the paper is to present, test and discuss the implementation of Visual SLAM techniques to images taken from Unmanned Aerial Vehicles (UAVs) outdoors, in partially structured environments. Every issue of the whole process is discussed in order to obtain more accurate localization and mapping from UAVs flights. Firstly, the issues related to the visual features of objects in the scene, their distance to the UAV, and the related image acquisition system and their calibration are evaluated for improving the whole process. Other important, considered issues are related to the image processing techniques, such as interest point detection, the matching procedure and the scaling factor. The whole system has been tested using the COLIBRI mini UAV in partially structured environments. The results that have been obtained for localization, tested against the GPS information of the flights, show that Visual SLAM delivers reliable localization and mapping that makes it suitable for some outdoors applications when flying UAVs.

J. Artieda · J. M. Sebastian · P. Campoy · J. F. Correa ·
I. F. Mondragón (✉) · C. Martínez · M. Olivares
Computer Vision Group, U.P.M., C/ José Gutiérrez Abascal, 2. 28006 Madrid, Spain
e-mail: imondragon@etsii.upm.es

J. Artieda
e-mail: jatrigueros@argongra.com

J. M. Sebastian
e-mail: jose.sebastian@upm.es

P. Campoy
e-mail: pascual.campoy@upm.es

J. F. Correa
e-mail: jfcorrea@etsii.upm.es

C. Martínez
e-mail: carol.martinez@upm.es

M. Olivares
e-mail: mig_olivares@hotmail.com

Keywords Computer vision · Visual SLAM · Unmanned aerial vehicles (UAV) · 3D SLAM

1 Introduction

Vision is the richest source of information from our environment, and that is the reason why SLAM algorithms have also begun to be used with visual information. The information provided by vision systems consists on a vaste amount of data per time that requires to be processed in order to provide SLAM algorithms with useful information. Hence, image processing algorithms have to precede SLAM algorithms and they highly determine the results and successfulness of the whole visual SLAM process.

This paper tackles the whole process of visual SLAM by rotary wings UAVs in outdoor environments, from the image acquisition to the final UAV localization and environment mapping. The obtained results are shown to be a useful information source that complements, and in some applications can replace, other onboard sensors as GPS or inertial ones. The information provided by visual SLAM is related to actual objects present in the environment, constructing a map of them and localizing the UAV relatively to these objects. This information can not be provided by GPS or other onboard sensors, which can only localize the UAV without any relative localization to external objects that can be verified by actual, external data, as visual SLAM does. Visual SLAM can also be useful in cases of GPS signal drop-off. Finally Visual SLAM can also be used for sensor fusion, providing the complementary advantages of diverse and complementary sources.

Visual SLAM techniques can be first classified in Stereo and Monocular. The first includes also more than two cameras approaches. There are many successful implementations of visual slam with stereo cameras [1, 2]. Monocular approaches where started by Davison et al. [3] who used only one camera to reconstruct indoor environments. Successful results have been obtained using Visual SLAM indoors also by [4], Kim and Oh in [5] and Choi and Oh in [6].

With monocular vision the initialization of features is a difficult problem. The approach given by Davison et al. [3] used a delayed initialization algorithm. This algorithm waits until the camera position has parallax enough to determine the position of a feature and then includes it on the filter. This approach needs a depth interval in which we expect to find the features and therefore is not suitable for outdoor uses where very near features and very far features can coexist. Montiel et al. [7] proposes a solution to the aforementioned estimation problem by using the so called inverse parametrization, which is also used in our approaches and tests. This technique allows the application of the algorithm to outdoor scenes. Nonetheless, as a drawback that increases the computational cost, which is also augmented in our case due to the big amount of key points in non-structured outdoors 3D environments.

Another problem of SLAM in outdoor environments is the high number of features and the long displacement between loop-closings. The most impressive application for outdoor SLAM algorithms is [8]. Here the authors use a SLAM algorithm based on 3D laser profiles and uses a vision based algorithm to detect loop-closure after a very long loop. Other approaches like Lemaire et al. [9] uses a feature database jointly with a strong detection algorithms based on the feature topology.

There are two main filters used in the SLAM problem: Extended Kalman Filter, EKF, and Rao-Backwellized Particle Filter, RBPF. Particle filters are not widely used because it needs a very high number of particles when the state dimension is high, so the RBPF is used instead of this. Examples of implementations of Visual Slam with RBPF are [1, 2, 10]. Approaches using EKF are [3, 7].

Visual SLAM implementations mainly use point features in contrast with the implementations of 2D laser based SLAM which are based on occupancy grids. Occupancy grids are not exclusive of non visual systems as shown in [1]. Several algorithms have been used for interest point detection and visual features extraction [11, 12], as well as for their matching in consecutive images [13], or not consecutive [8]. Other approaches use other geometry entities as features, like Dailey in [10] which uses lines.

The use of visual SLAM onboard UAV is no yet very spread although there are some successful implementations like: [9, 14–16]. The system presented by Törnqvist et al. [14] uses a FastSLAM offline algorithm that is fussed with the Rao-Blackwellized particle filter in order to get 2D information that is necessary to estimate the position and the altitude of the UAV. Good results concerning the position estimation have been obtained but, on the contrary, not so good results have been achieved in relation to altitude estimation. Another work in the same field was presented by Kim and Sukkarieh [15] where vision, a Radar device and an high-quality Inertial Measurement Unit (IMU) are used for a 2D inertial-SLAM in order to get better results. Yet they are still shown in 2D SLAM, which means there is not three dimensional reconstruction of the environment. In Miniature Air Vehicle (MAV) platform, the research made by McLain et al. [16] was based on a camera positioning device that estimates the position and altitude of the MAV and the pixel location of the target in an image. Its results can localize the target in world coordinates using different techniques so as to reduce the localization error. N. Aouf et al., on the other hand, developed in [17] an airbone SLAM algorithm with Inertial Navigation System (INS) and Visual system by implementing an Extended Kalman Filter (EKF). They proposed a solution to remove the landmarks from the EKF, a methodology based on circle intersections, and gave results with virtual images taken from a downward looking virtual camera.

Therefore, we have implemented an EKF version that takes into account our specific UAV application in order to optimize the computational time, not letting it increase oversize, as detailed in Section 4. Below we present in this paper the results of applying Visual 3D SLAM techniques onboard an UAV. Those results will be compared with the flight information delivered by the GPS and IMU. These results are presented in Section 5 and they demonstrate that robust and coherent positioning and mapping are obtained,which make them suitable for being used in UAV applications where the visual information regarding its environments plays an important role, such as outdoors civilian infrastructures visual inspection, environmental events detection and tracking, and visual security applications.

2 System Description

The COLIBRI testbed [18], is based on a gas powered industrial twin helicopter with a two stroke engine 52 cc and 8 hp (Fig. 1) capable to carry up to 12 kg

Fig. 1 UPM-COLIBRI I Helicopter platform used for Visual SLAM tests

payload. The platform is equipped with a xscale-based flight computer augmented with sensors (GPS, IMU, Magnetometer, etc fused with a Kalman filter for state estimation). Additionally it includes a Pan Tilt servo controlled platform for many different cameras and sensors. On the other hand, in order to enable it to perform vision processing, it has a VIA mini-ITX 1.25 GHz onboard computer with 1 Gb RAM, a wireless interface and support for many Firewire cameras including Mono (BW), RAW Bayer, color, and stereo head for images acquisition. Additionally, it is possible to use IP cameras and analog cameras as well.

The system runs in a client-server architecture using TCP/UDP messages. Computers run Linux OS working in a multi-client wireless 802.11g ad-hoc network, allowing the integration of vision system and visual tasks with the flight control. This architecture allows embedded applications to run onboard the autonomous helicopter while it interacts with external processes through a high level switching layer. The visual control system and additional external processes are integrated with the flight control through this layer using TCP/UDP messages [19]. The layer is based on a communication API where all the messages and data types are defined. The helicopter's low-level controller is based on simple PID control loops to ensure its stability.The higher level controller uses various sensing mechanisms such as GPS and/or vision to perform tasks such as navigation, landing, visual tracking, etc. Several applications based on visual control have been achieved employing the control architecture [20, 21].

3 Interest Points Detection and Matching

The selection of interest points that can be tracked along an image sequence is a key step in the visual SLAM algorithm because it ensures the stability of the Kalman filter. Since this step needs to be a reliable process it is important to have a measure of the reliability of an extracted feature, in order to choose the most important and robust one. There are some alternatives to find robust features that can be identified

and characterized with a descriptor like the SURF [22] or SIFT [23]. For example, in [24] Danesi et al. they use SIFT for a wheeled vehicle visual servoing. Other detectors like the Harris Corner Detector [25] find corner features that are very common in semi-structured environments, like in [26], where García-García et al. used the Harris Detector with RANSAC for a robust position estimation. Based on previous work that evaluates the performance of interest point detectors for the SLAM problem [27] and [13], the most appropriate choices are the Harris detector, SIFT and SURF, but since it is a well known fact that SURF computation is faster than SIFT's and their behaviors are similar SIFT is not consider in the work.

Two alternatives are presented in this section; Harris Detector with a reliability measure (plus cross-correlation) and the SURF feature. The subsequent matching algorithm is also described for each type of feature too.

3.1 Harris Corner Detection and a Reliability Index

The Harris detector is a well known detector that is widely used in a large amount of applications. It extracts many corners very quickly based on the magnitude of the eigenvalues of the autocorrelation matrix. However, it is not enough to use this procedure to ensure the robustness of the extracted corner. The aim that is sought is to increase the probability to find it again in the next image and to match it correctly. For that purpose a quality measure has been defined and some procedures have been implemented in order to achieve the extraction of good features to track.

Due to the noise in the images caused by the sensor itself and vibration of the UAV, it is important to filter the image with a mean filter and a median filter. Then, the gradient G of the image is calculated, and only pixels with a norm of the gradient above a value are considered to be processed. Afterward the Canny edge detector [28] algorithm is used in the previously selected pixels keeping pixels laying on well define edges. This reduces the number of extracted features. After this process is completed, the corners are extracted with the standard version of the Harris detector. Next, the sub-pixel precision version of the detector is applied [29] shifting the window over three iteration. Based on the results of those two algorithms a stability measure is calculated to determine the maximum position variation e_s. Finally, the size of the detector window is increased from 5×5 to 7×7, to prove and test the stability of the position of the extracted corners, and a measure of this variation is calculated, named e_w based on a "maximum difference allowed" criteria. All those measures are integrated into a function Q (1) that returns a global value of the quality and robustness of the extracted corner using the product of the eigenvalues λ_1 and λ_2, the norm of the gradient $\|G\|$ of the interest point and the measures described above.

$$Q(\lambda_1, \lambda_2, \|G\|, e_s, e_w) = \frac{\lambda_1 \lambda_2 \|G\|}{(1 + e_s)(1 + e_w)} \tag{1}$$

The global value calculated for each point is used to reject false corners using each one of the 5×5 windows and considering only the corners with the maximum value of Q index. Also it is used to classify the extracted features into three groups. This distinction between the corners is going to drive the matching process: each group represents a level of quality of a corner. This allows one to make the assumption that good corners are going to appear in the next image, and to suppose that they are going to be found in one of the next levels in case they degrade. Figure 2 illustrates

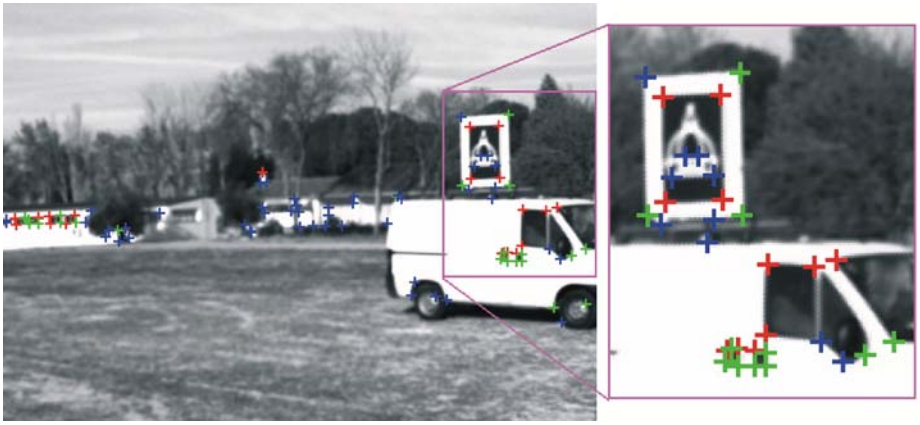

Fig. 2 Extracted corner features classified into three levels. *Red corners* are the most stable ones and belong to level 1 group. *Green corners* are classified as level 2 and *blue ones* as level 3

the three levels of classification of the corners and how the extraction method keeps features that are on structured parts of the scene none of which belongs to the landscape. Another advantage of this classification resides in the possibility to include new features into the Kalman Filter of the SLAM process only if they belong to the level 1 group.

3.2 SURF Features

Speeded Up Robust Feature algorithm extracts features from an image which can be tracked over multiple views. The algorithm also generates a descriptor for each feature that can be used to identify it. SURF features descriptor are scale and rotation invariant. Extracted features are blob like features. Ollero et al. used this method in [30] for position and motion estimation from multiple planar homographies taken from different UAVs. The kinds of extracted features are shown in Fig. 3 on a structured scene. Tests were also carried out with unstructured scenes such as the ones shown in Fig. 4.

Scale invariance is attained using different amplitude gaussian filters. The application of this filter results in an image pyramid. The level of the stack from which the feature is extracted assigns the feature to a scale. This relation provides scale invariance. The next step is to assign a repeatable orientation to the feature. The angle is calculated through the horizontal and vertical Haar wavelet responses in a circular domain around the feature. The angle calculated in this way provides a repeatable orientation to the feature. As with the scale invariance the angle invariance is attained using this relationship.

SURF descriptor is a 64 element vector. This vector is calculated in a domain oriented with the assigned angle and sized according to the scale of the feature. Descriptor is estimated using horizontal and vertical response histograms calculated in a 4 by 4 grid. There are two variants to this descriptor: the first provides a 32 element vector and the other one a 128 element vector. The algorithm uses integral images to implement the filters. This technique makes the algorithm very efficient.

Fig. 3 SURF features tested on semi-structured scenes

3.3 Corner Features Matching

Once corner features are extracted the next step is to correctly match the features of the current image with as many features of the previous image as possible. Since the corners are divided into levels, the first matching attempt is made using the level 1 corners of the current image against the 1st and 2nd levels of the previous image. Then a matrix containing the result of a similarity function is calculated for all the possible match pairs for this set of corners. The similarity function is the *normalized cross-correlation* of the context of the corners, which in this case is a 9×9 patch centered at the position of the feature in pixel resolution. However, other similarity functions can be used as, for example *sum of squared differences* or the *Earth Mover*

Fig. 4 SURF features tested on unstructured scenes

distance. The next step is to find each and every possible set of matches that maximize a cost function which can be defined as follows:

- Cost function rewards high cross-correlation.
- Cost function penalizes matching pairs whose distance differs from the average displacement of the set of matched features, based on the fact that most of them will move in solidarity.
- Cost function rewards the number of attained matches.

If I_k is the kth image, and L_q^k is a corner in that image, lets define ι as the match between a corner in the kth and *(k-1)*th image like a 2-tuple $\iota = (L_p^{k-1}, L_q^k)$, and lets define $\Omega = \iota_1, \iota_2, ..., \iota_i, ..., \iota_n$ as the set of matches between the corners of those two images. Given those definitions and the considerations described above, the cost function is:

$$J(\Omega) = \frac{n \sum_i c_i}{1 + \frac{\sum_i \sqrt{(dx_i - \bar{dx})^2 + (dy_i - \bar{dy})^2}}{\sqrt{(\bar{dx})^2 + (\bar{dy})^2}}} \tag{2}$$

where i is the ith match of Ω, c_i is the cross-correlation of ith matched features, dx_i and dy_i are the position difference of the L_q^k corner in each axis from the matched corner L_p^{k-1} in sub-pixel precision, n is the number of matched features, and $\bar{dx}$ and $\bar{dy}$ are the mean of those differences of position of the set of matches Ω.

To find all possible sets of matches, including the case there is no match for some of the current corners, a recursive algorithm is used to explore possible combinations. Yet, the amount of combinations is too large to calculate the cost function for every single possibility. The way to avoid unnecessary calculation can be found in the criteria used to formulate the cost function. A corner is consider to be matched with other if their correlation is higher than an umbral and if the difference in position is lower than a maximum displacement. These conditions reduce the number of possible sets to a more reasonable amount of possible sets to be calculated. In order to exploit the assumption that the matching will be done on consecutive images captured at a reasonable frame rate like 30 fps, empirically we have found that cross-correlation higher than 0.98, and a search radius of 100 pixels, works fine for this first step of the matching procedure. The size of this radius of search depends on the frame rate, the angular and lineal velocity of the UAV and the distance of the objects in the scene. This procedure results in the definition of a global motion parameter of the corners of the current image compared with the ones in the previous frame. Using the information of the best found match, the procedure is repeated with the unmatched corners of the 1st level and 2nd level corners of the current image. But this time candidate corners of the current image are translated $(-\bar{dx}, -\bar{dy})$ to match them with the unmatched features of the previous image in a radius of 4 pixels. Only matching pairs with a cross-correlation higher than 0.96 are considered. To find the best set of matches in this second step the cost function is $\sum c_i$.

Finally, in the third phase, the algorithm tries to match features in the previous image that were matched before. All unmatched features of the current and previous images, including 3rd level corners are matched using the same procedure of the second matching attempt, allowing matched pairs with a cross-correlation higher than 0.96 to remain only if they were matched before. Some results of the entire

Table 1 Results attained with the matching algorithm described in Section 3.3 for corner features

Image	1	2	3	4	5	6	7
Level 1 corners	15	15	15	15	15	15	15
Level 2 corners	20	20	20	20	20	20	20
Level 3 corners	52	53	55	47	43	46	46
Total corners	87	88	90	82	78	81	81
Matched on phase 1	0	11	11	10	10	13	10
Matched on phase 2	0	3	0	0	9	9	0
Matched on phase 3	0	11	5	3	9	20	6
Correct matches	0	25	16	11	24	38	14
Wrong matches	0	0	0	2	4	4	2
Tracked corners	0	0	9	8	10	16	12

stages of the matching process are summarized in Table 1, while Fig. 5 shows the attained matching graphically.

3.4 SURF Features Matching

The procedure to match SURF features is based on the descriptor associated to the extracted interest point. An interest point in the current image is compared to an interest point in the previous one by calculating the Euclidean distance between their descriptor vectors. A matching pair is detected if its distance is closer than 0.9 times the distance of the second nearest neighbor and the SSD error between the two descriptors is less than 150000. The procedure for a sequence of images begins with the extraction of all features in the first image. Thirty interest points well distributed all over the image are selected to become the initial database. Extracted SURF features in the next image are compared to the database using the Euclidean

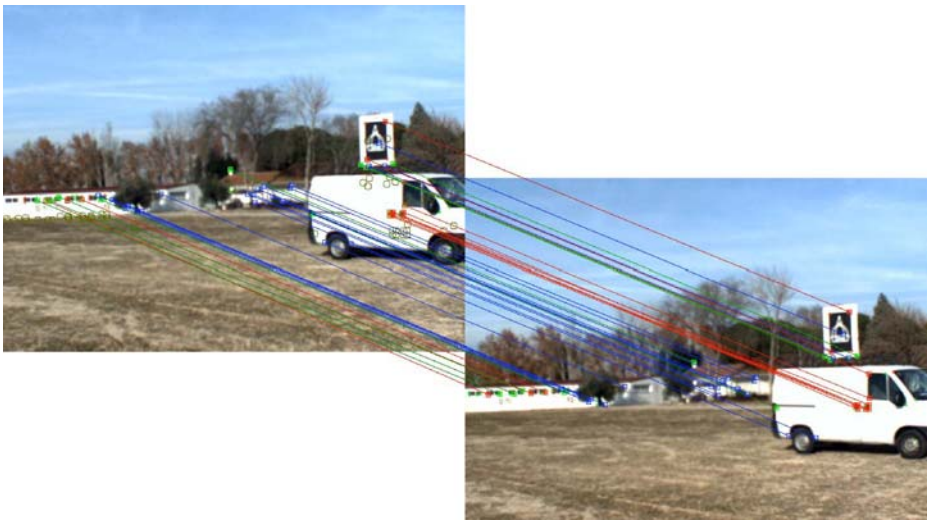

Fig. 5 Corner features are matched using the procedure described in Section 3.3. *Red lines* show matches obtained in phase 1, while *green lines* represent the matches of phase 2 and the *blue lines* depict the ones made in phase 3

Table 2 Comparison between semi-structured and unstructured scenes for SURF algorithm

Scene	Total features	Matched features	Ratio
Semi-structured	1559	884	56%
Unstructured	3590	1518	42%

distance as described above. This reduces the computational cost of matching all the possible features between frames and allows to track a constant set of features along a high number of frames. If the matching of the thirty features in the set is not possible, new features are added to this set using the same procedure employed for the first thirty. To avoid the insertion of features during short periods of no-detection of features, new features are inserted only when the number of matched features is below ten.

SURF features extraction and matching have been tested with semi-structured and unstructured scenes to use different techniques depending on the scenes and to achieve better performance in SLAM algorithms. SURF features behave similarly in both cases. Table 2 summarizes the behavior of SURF. The results of Harris detector indicate that it finds features that almost in all cases belong to structured objects of the scene. For this reason, the SURF features are used on unstructured scenes.

4 Visual SLAM

This section presents the implementation of a visual SLAM algorithm with monocular information. No prior information of the scene is needed for the proposed formulation. In this approach, no extra absolute or relative information, GPS or odometry are used. First, the formulation of the problem will be described. Then, the details of the Kalman filter are explained. Finally, the particularities of this approach are addressed.

4.1 Formulation of the Problem

The problem is formulated using state variables to describe and model the system. The state of the system is described by the vector:

$$X = [\mathbf{x}, \mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3, ...] \tag{3}$$

where x denotes the state of the camera and $\mathbf{s}_i$ represents the state of each feature. The camera state has 12 variables. The First six variables represent the position of the vehicle in iteration k and in the previous iteration. The Next six variables, vector $[p, q, r]$, represent the rotation at the iteration k and $k-1$. Rotation is expressed using Rodrigues notation. This expresses a rotation around a vector with the direction of $\omega = [p, q, r]$ of an angle $\theta = \sqrt{p^2 + q^2 + r^2}$. The rotation matrix is calculated from this representation using

$$e^{\tilde{\omega}\theta} = I + \tilde{\omega} sin(\theta) + \tilde{\omega}^2(1 - cos(\theta)) \tag{4}$$

where I is the 3×3 identity matrix and $\tilde{\omega}$ denotes the antisymmetric matrix with entries

$$\tilde{\omega} = \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix} \tag{5}$$

Therefore the state of the camera, not including the features, is composed by the following 12 variables,

$$\mathbf{x} = [x_k, x_{k-1}, y_k, y_{k-1}, z_k, z_{k-1}, p_k, p_{k-1}, q_k, q_{k-1}, r_k, r_{k-1}] \tag{6}$$

Other implementations of monocular SLAM uses quaternion to express the rotation [7]. The use of Rodrigues notation, instead of quaternion, allows to reduce the dimension of the problem using only three variables to represent the rotation.

Rodrigues representation avoids the singularities of other three-parameter representations but has a discontinuity at rotations of 180 degrees. This parametrization is chosen instead of quaternions since quaternions force the introduction of a unit norm restriction. This restriction is difficult to handle in the context of a conventional EKF. It can even lead to singularities in the Kalman filter matrices [31] although noise and system imperfections help to avoid this situation.

Using a discrete system storing the states at instant k and k-1 instead of considering a state composed of position and velocities at instant k helps the introduction of angular representations that are not linear with angular velocities. It also allows the introduction of movement models without many changes in the algorithm structure. Both formulations are equivalent mathematically.

Each feature is represented as a vector $[\mathbf{s}_i]$ of dimension 6 using the inverse depth parametrization proposed by Javier Civera in [7]. This parametrization uses six parameters to define the position of a feature in a 3Dimensional space. Each feature is defined by the position of a point, the direction of a line based on the point and the inverse distance form the point to the feature along the line. This parametrization is shown in Fig. 6. This reference system allows the initialization of the features without any prior knowledge about the scene. This is important in exterior scenes where features with very different depths can coexist.

$$s_i = [x_0, y_0, z_0, \theta, \phi, \rho] \tag{7}$$

This parametrization is converted to 3D world coordinates using

$$m(\theta, \phi) = \begin{bmatrix} cos(\theta)sin(\phi) \\ -sin(\theta) \\ cos(\theta)cos(\phi) \end{bmatrix}$$

$$[x_w, y_w, z_w] = [x_o, y_o, z_o] + \frac{1}{\rho} \cdot m(\theta, \phi) \tag{8}$$

4.2 Prediction and Correction Stages

The algorithm's main loop has two stages: prediction and correction. In the prediction stage, uncertainty is propagated using the movement model. The correction stage uses real measurements and predicted measurements to compute a correction

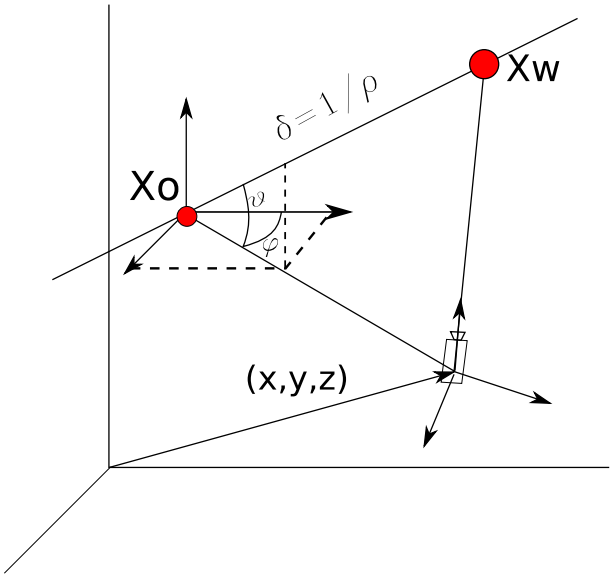

Fig. 6 Inverse depth parametrization. The position x_w of a feature s_i is given by the position of a point x_o, the direction of a line, θ, ϕ, and the inverse of the distance from the point x_o to the feature x_w. The state vector is completed by the position of the camera and its rotation

to the prediction stage. Both stages need a precise description of the stochastic variables involved in the system.

There are mainly two approaches to implement this filter: extended Kalman filter and particle filter (FastSLAM). Both filters use the same formulation of the problem but have different approaches to the solution. The advantages of the Kalman filter are the direct estimation of the covariance matrix and the fact that it is a closed mathematical solution. Its disadvantages are the increasing computational requirements with the number of features, the need of linearization of the model and the assumption of gaussian noise. On the other hand, particle filters can deal with non-linear, non-gaussian models but the solution they provide depends on an initial random set of particles which can differ in each execution.

Given the previous facts, the Kalman filter has thus been chosen since its results can be traced back and experiments are repeatable. The Extended Kalman filter allows the use of non-linear models through equation linearization.

The prediction stage is formulated using linear equations

$$\begin{aligned} \hat{X}_{k+1} &= A \cdot X_k + B \cdot U_k \\ \hat{P}_{k+1} &= A \cdot P_k \cdot A^T + Q \end{aligned} \tag{9}$$

where A is the transition matrix, B is the control matrix and Q is the model covariance. Camera movement is modeled using a constant velocity model. Accelerations are included in a random noise component. For a variable n which represents any of the position components (x, y, z) or the rotation components (p, q, r) we have:

$$n_{k+1} = n_k + v_k \cdot \Delta t \tag{10}$$

Where v_k is the derivative of n or speed. We can estimate v_k as the differences in position,

$$n_{k+1} = n_k + \left(\frac{n_k - n_{k-1}}{\Delta t}\right)\Delta t = 2n_k - x_{n-1} \tag{11}$$

Feature movement is considered constant and therefore is modeled by an identity matrix. Now full state model can be constructed

$$\begin{bmatrix} x_{k+1} \\ x_k \\ y_{k+1} \\ y_k \\ z_{k+1} \\ z_k \\ r_{k+1} \\ r_k \\ p_{k+1} \\ p_k \\ q_{k+1} \\ q_k \\ s_{1,k+1} \\ \cdots \end{bmatrix} = \begin{bmatrix} 2 & -1 & & & & & & & & & & & & \\ 1 & 0 & & & & & & & & & & & & \\ & & 2 & -1 & & & & & & & & & & \\ & & 1 & 0 & & & & & & & & & & \\ & & & & 2 & -1 & & & & & & & & \\ & & & & 1 & 0 & & & & & & & & \\ & & & & & & 2 & -1 & & & & & & \\ & & & & & & 1 & 0 & & & & & & \\ & & & & & & & & 2 & -1 & & & & \\ & & & & & & & & 1 & 0 & & & & \\ & & & & & & & & & & 2 & -1 & & \\ & & & & & & & & & & 1 & 0 & & \\ & & & & & & & & & & & & \mathbf{I} & \\ & & & & & & & & & & & & & \ddots \end{bmatrix} \begin{bmatrix} x_k \\ x_{k-1} \\ y_k \\ y_{k-1} \\ z_k \\ z_{k-1} \\ r_k \\ r_{k-1} \\ p_k \\ p_{k-1} \\ q_k \\ q_{k-1} \\ s_{1,k} \\ \cdots \end{bmatrix} \tag{12}$$

The correction stage uses a non-linear measurement model. This model is the pin-hole camera model. The formulation of the Extended Kalman Filter in this scenario is

$$\begin{aligned} K_k &= \hat{P}_k \cdot J^T (J \cdot P \cdot J^T + R)^{-1} \\ X_k &= \hat{X}_k + K_k \cdot (Z_k - H(\hat{X}_k)) \\ P_k &= \hat{P}_k - K_k \cdot J \cdot \hat{P}_k \end{aligned} \tag{13}$$

Where Z_k is the measurement vector, $H(X)$ is the non-linear camera model, J is the jacobian of the camera model and K_k is the Kalman gain.

The movement of the system is modeled as a solid with constant motion. Acceleration is considered a perturbation to the movement. A pin-hole camera model is used as a measurement model.

$$\begin{bmatrix} nu \\ nv \\ n \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot [R|T] \cdot \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \tag{14}$$

where u and v are the projected feature central coordinates. Distortion is considered using a four parameters model (k1, k2, k3, k4)

$$\begin{aligned} r^2 &= u^2 + v^2 \\ C_{dist} &= 1 + k_0 r^2 + k_1 r^4 \\ x_d &= u \cdot C_{dist} + k_2(2u \cdot v) + k_3(r^2 + 2u^2) \\ y_d &= v \cdot C_{dist} + k_2(r^2 + 2v^2) + k_3(2u \cdot v) \end{aligned} \tag{15}$$

The state error covariance matrix is initialized in a two part process. First, elements related to the position and orientation of the camera, **x**, are initialized as zero or as a diagonal matrix with very small values. This represents that the position is known, at the first instant, with very low uncertainty. The initialization of the values related to the features, s_i, must be done for each feature seen for the first time. This initialization is done using the results from [7]:

$$\mathbf{P}_{k|k}^{new} = J \begin{bmatrix} \mathbf{P}_{k|k} & & \\ & \mathbf{R}_i & \\ & & \sigma_\rho^2 \end{bmatrix} J^T \tag{16}$$

Where

$$\mathbf{J} = \begin{bmatrix} I & & 0 & 0 \\ \dfrac{\partial \mathbf{s}}{\partial \mathbf{xyz}} \; \dfrac{\partial \mathbf{s}}{\partial \mathbf{pqr}} & 0\,0\cdots & \dfrac{\partial \mathbf{s}}{\partial x_d,\, y_d} & \dfrac{\partial \mathbf{s}}{\partial \rho_0} \end{bmatrix} \tag{17}$$

$$\frac{\partial \mathbf{s}}{\partial \mathbf{xyz}} = \begin{bmatrix} 1\,0\,0 \\ 0\,1\,0 \\ 0\,0\,1 \\ 0\,0\,0 \\ 0\,0\,0 \\ 0\,0\,0 \end{bmatrix}; \frac{\partial \mathbf{s}}{\partial \mathbf{pqr}} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \frac{\partial \theta}{\partial p} & \frac{\partial \theta}{\partial q} & \frac{\partial \theta}{\partial r} \\ \frac{\partial \phi}{\partial p} & \frac{\partial \phi}{\partial q} & \frac{\partial \phi}{\partial r} \\ 0 & 0 & 0 \end{bmatrix}; \frac{\partial \mathbf{s}}{\partial x_d,\, y_d} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{\partial \theta}{\partial x_d} & \frac{\partial \theta}{\partial y_d} \\ \frac{\partial \phi}{\partial x_d} & \frac{\partial \phi}{\partial y_d} \\ 0 & 0 \end{bmatrix}; \frac{\partial \mathbf{s}}{\partial \rho_0} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{18}$$

Robust feature tracking and detection is a key element in the system. In order to improve the robustness of the feature matching process a Mahalanobis test is used. The filter is implemented using Mahalanobis distance between the predicted feature measurement and the real measurement. Mahalanobis distance weighs Euclidean

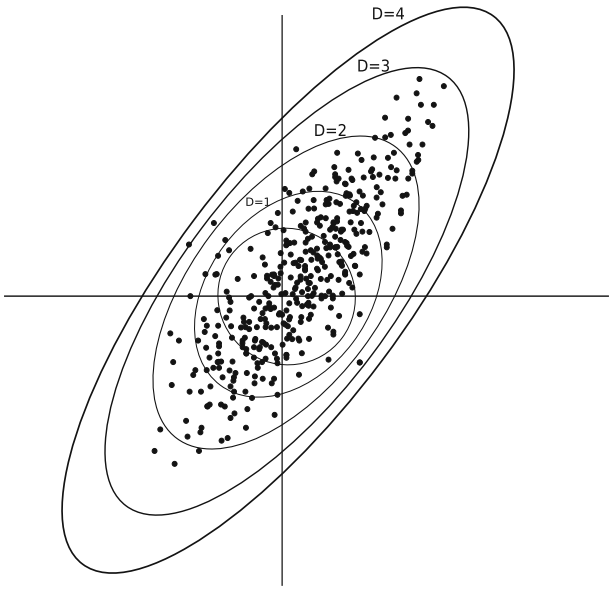

Fig. 7 Mahalanobis distance representation

distance with the covariance matrix. Figure 7 shows a representation of Mahalanobis distance. This distance is the input to a χ^2 test which rejects false matches.

$$(Z - J \cdot X)^t \cdot C^{-1}(Z - J \cdot X) > \chi_n^2 \tag{19}$$

where

$$C = H \cdot P \cdot H^T + R \tag{20}$$

The scale of the reconstruction is an unobservable system state. This problem is covered in [32] by Javier Civera. The use of inverse depth parametrization avoids the use of initialization features of a known 3D position. This allows the use of the algorithm in any video sequence. Without these initialization features, the problem becomes dimensionless. The scale of the system can be recovered using the distance between two points or the position of the camera and one point. Computational cost is dependant on the number of features in the scene, and so the increasing scene complexity affects processing time in a negative way. Robust feature selection and matching is very important to the stability of the filter to achieve a correct mapping.

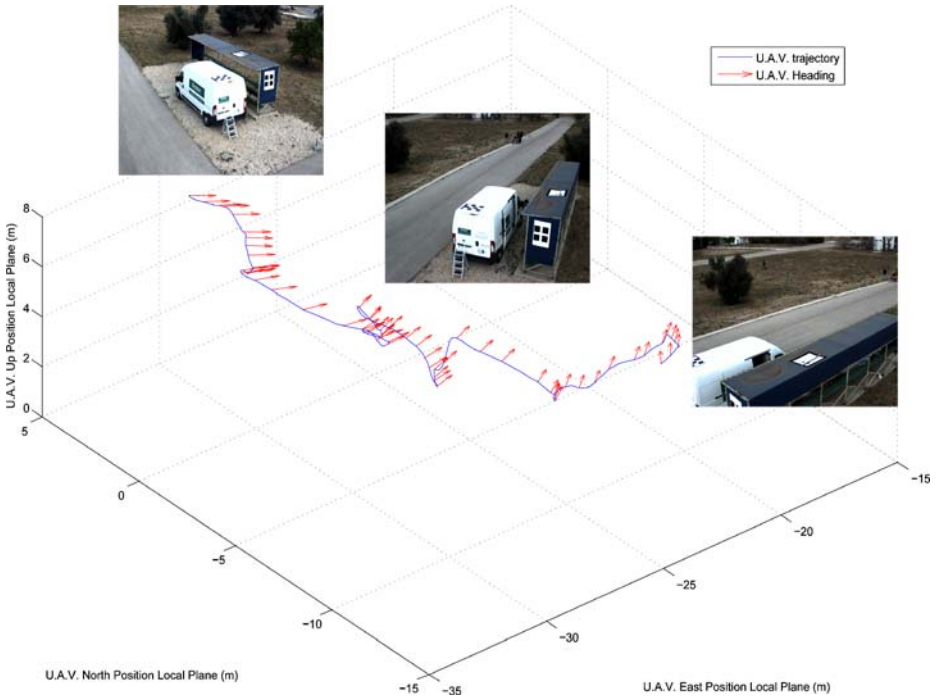

Fig. 8 3D flight trajectory and camera position reconstruction, obtained using the flightlog data. The *blue line* depicts the translational movement and the *red arrows* represent the heading direction of the camera (pitch and yaw angles). Superimposed Images show the different perspectives obtained during the flight sequence around the semi-structured scene

5 Results

Several tests have been made using the Colibri I testbed. In this test, a series of trajectories around a 3D scene were performed flying in autonomous mode navigation based on way points and desired heading values. The scene is composed of many objects, including a grandstand, a van and many other elements, and also of a series of marks feasible for features and corners detection. For each flight test a 30 f.p.s. image sequence of the scene was obtained, associating the U.A.V. attitude

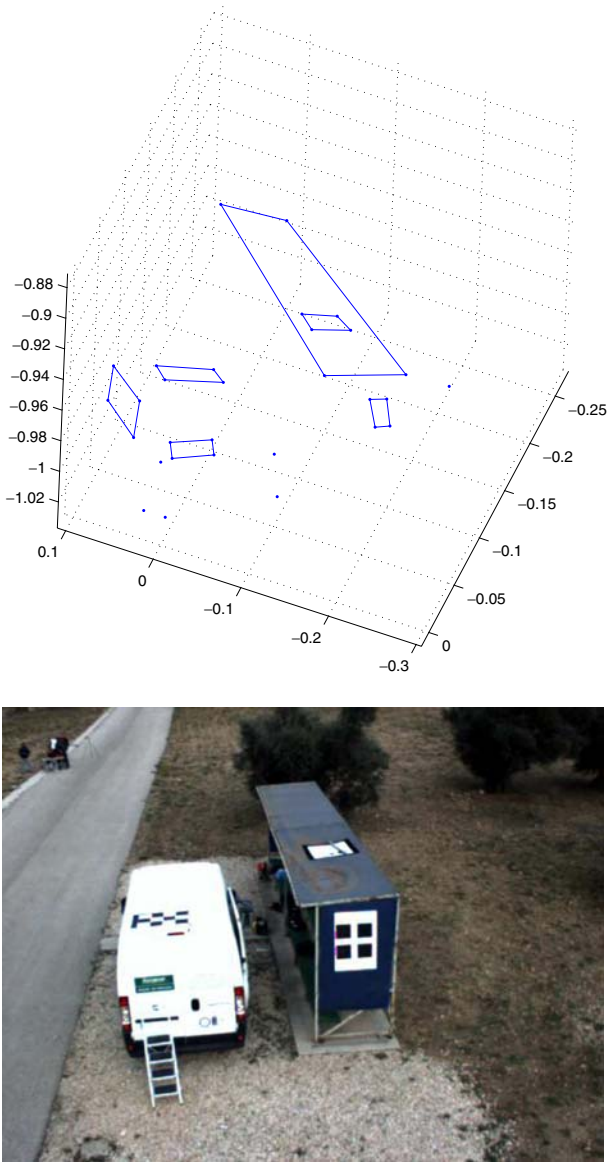

Fig. 9 Semi-structured scene reconstruction. The *upper figure* shows reconstructed points from the scene shown in the *lower figure*. Points are linked manually with lines to ease the interpretation of the figure. All the reconstruction is done dimensionless to show the original results. To recover the scale at least two points must have known coordinates

information for each one. That includes the GPS position, IMU data (Heading, body frame angles and displacement velocities) and the helicopter position estimated by controller Kalman Filter, on the local plane with reference to the takeoff point.

Using the flightlog it is possible to reconstruct the 3D trajectory of the vehicle and the camera and/or helicopter pointing direction. Figure 8 shows a reconstruction of one flight around the test scene.

Tests have been made with semi-structured scenes and un-structured scenes. Also, very different distances to the features have been used. The implementation of inverse depth parameterized features and of dimensionless reconstruction allows the use of the algorithm in relation to different kind of scenarios.

5.1 Semi-structured Scene

Results for tests using a tracking algorithm for structured elements are shown on Fig. 9. Reconstructed features are shown as crosses. In the figure some references planes were added by hand in order to help with the interpretation. Figure 9 shows an image from the sequence used in this test.

Results show that the reconstruction has a coherent structure but that the scale of the reconstruction is function of the initialization values. The scale can be recovered using the distance between two points or the positions of one point and the camera.

The uncertainty of the features is reduced if observations of better known features are used. Figure 10 shows the variance of the features. Uncertainty is represented as a point cloud around the reconstructed position. Ellipsoids are not an appropriate form of representation due to the inverse depth parametrization. The figure shows

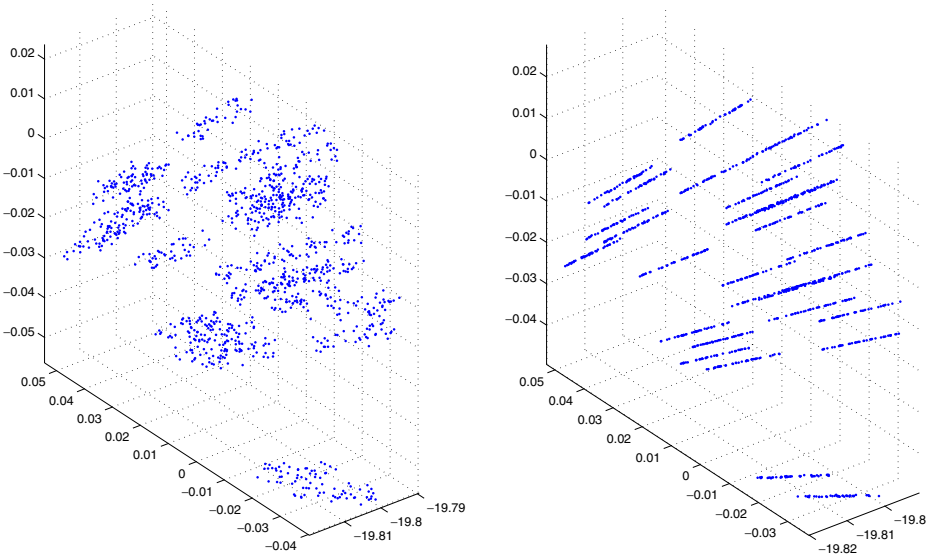

Fig. 10 Covariance matrix evolution. In this figure, uncertainty is represented as a point cloud. *The figure on the right* shows the reduction of uncertainty after a few observations. The uncertainty in depth direction is still hight due to low parallax in this short movement

how uncertainty is reduced in sequential observation. It can also be seen how depth uncertainty is much greater than uncertainty of other directions.

Uncertainty point cloud is represented in Fig. 11 as a group of small points. Numbers represent the detected features. The sequence shows the evolution of the features position and their uncertainty.

Finally the camera movement relative to the first image is compared with the real flight trajectory. For this the (x, y, z) axis on the camera plane are rotated to be coincident with the world reference plane used by the UAV. The Heading or Yaw angle (ψ) and the Pitch angle (θ) of the helicopter in the first image of SLAM sequence, define the rotational matrix used to align the camera and UAV frames. The Rotation Matrix is defined by:

$$\mathbf{R}(\psi,\theta) = \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ \cos(\theta)\sin(\psi) & \cos(\theta)\cos(\psi) & -sin(\theta) \\ \sin(\theta)\sin(\psi) & \sin(\theta)\cos(\psi) & \cos(\theta) \end{bmatrix} \tag{21}$$

The displacement values obtained using SLAM, are rotated and then scaled to be compared with the real UAV trajectory. Figure 12 shows the UAV and SLAM trajectories and the medium square error (MSE) between real flight and SLAM displacement, for each axe. In X and Y axes, the trajectory adjusts better to the real flight as soon as the features reduce theirs uncertainty, as soon as more images are

Fig. 11 Covariance evolution. The uncertainty represented as a cloud of small points (*red*) decreases with sequential observations. *Numbers* show the predicted and observed features position

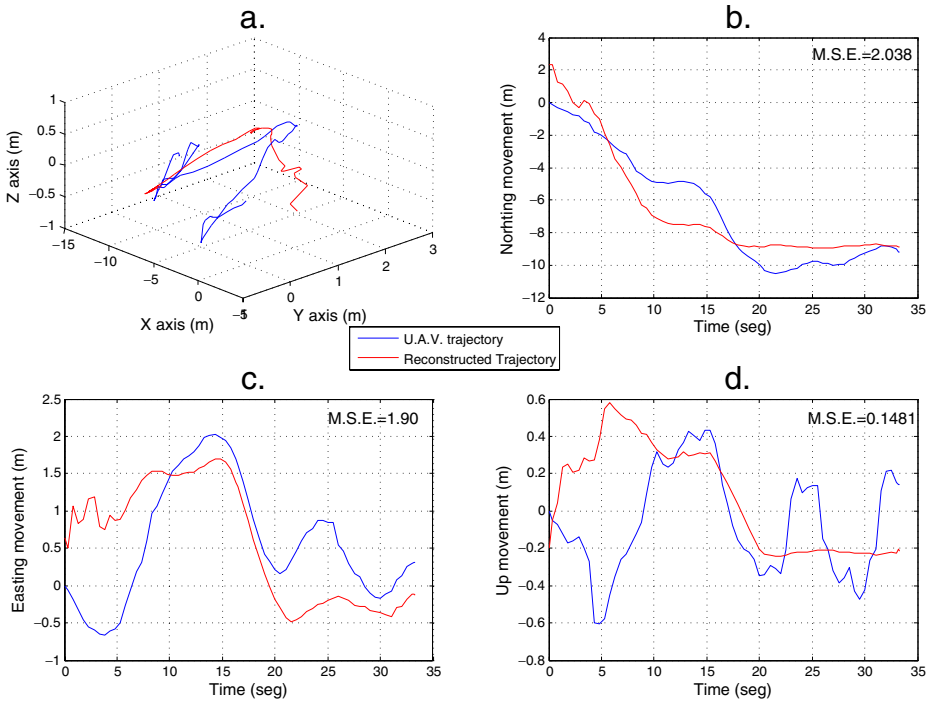

Fig. 12 SLAM reconstructed trajectory vs. UAV trajectory. **a** 3D flight. **b** North axis in meters, **c** East axis in meters, **c** Altitude in meters. The reconstructed trajectory on X and Y axes, adjusts better to the real flight as soon as more images are processed and the uncertainty of the features is reduced. Altitude measurement has a precision of ± 2 m causing that Z axis results can't be compared. The initial altitude of the test is 6.88 m

Fig. 13 Scene and tracked features during a non structured visual flight. Video sequence was taken by a manned helicopter traveling along a rectilinear trajectory for several hundred meters. The scene contains mainly non manmade features. Results are satisfactory although vibrations and image quality made the feature matching a difficult task

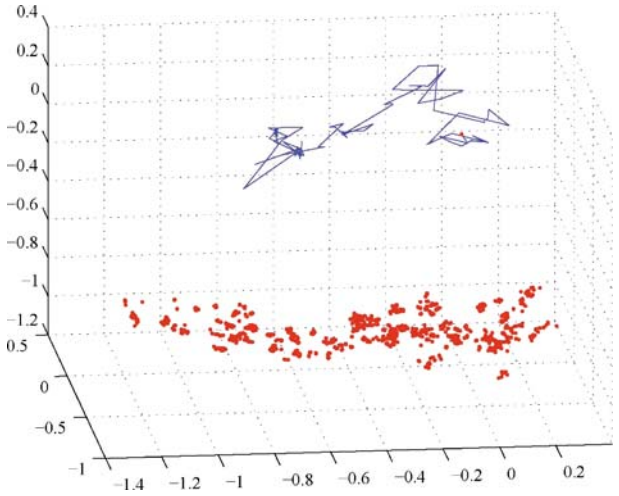

Fig. 14 Feature and track reconstruction. The figure shows the reconstruction of the helicopter position by visual SLAM, shown as a *blue line* on top of the figure. On the *lower parts*, in *red*, are the reconstructions of the observed features. The main direction of the movement is coherent with the movement along a straight line done by the vehicle. Below, the features are reconstructed over a horizontal surface. In this test the reconstructed trajectory shows big amplitude movements due to the vibrations, which make fast image changes not well modeled by the system. All the reconstruction is done dimensionless to show the original results. To recover the scale at least two points must have known coordinates

processed. However, in Z axis it doesn't look to have a good adjustment compared with the ground-truth but it has to be noticed that altitude values measured using the GPS on the UAV have precision of ± 2 m and that changes from initial altitude of 6.8 m are not significant.

5.2 Unstructured Scene

Another test was made using images from a manned helicopter. This scene has fewer structured elements and has been recorded from a greater distance. This Fig. 13 shows a frame of the image sequence. The results of the reconstruction of the features and the track of the camera are shown in Fig. 14.

This test has two results. The first one is the successful application in unstructured environments, which is shown by the great number of features tracked. The second result is the performance of the algorithm in a scene with features that are very far from each other. All the reconstruction was made using the same parameters as in the previously described test. Inverse depth parametrization and the dimensionless formulation allow the application of the algorithm in outdoor scenes without prior knowledge of the scene and without specific adjustments.

6 Conclusions

This paper shows that it is possible to obtain robust and coherent results using Visual SLAM for 3D mapping and positioning in vague structured outdoor scenes from

a mini UAV. In order to obtain these results, several stages of the whole process need to be solved, starting with image acquisition, going on with image processing, interest point detection, features extraction and matching, and finishing with the SLAM algorithm itself, EKF prediction and correction matrix model estimation, state definition and distance parametrization.

The quality, resolution and frame rate of the images should be enough to detect interest points that have to be tracked in several consecutive frames. The best results in this paper have been obtained using a RAW Bayer non-interlaced camera, 640×480 pixels at 30 frames per second (FPS), with a 6 mm. optic, while the coherently mapped environment has been in the range of 5 to 50 m. The video sequence is proceeded off-line at an average of 12 FPS.

Interest points' detection and features to be tracked have been found based on two different approaches, Harris corner detection and SURF invariant feature extraction. The approach based on Harris is very quick and selective, therefore very convenient for this computational intensive application, but it needs to be improved with an exhaustive and robust corner descriptor, as the one proposed in Section 3.1, that enables robust matching and tracking of the detected points over time. Harris based detectors have shown to be very efficient for scenes with significant structure objects, such as houses, vans, cars and, generally human made structures. Scenes with significant structures have strong and stable contours that give reliable edges to fix Harris points.

The SURF based feature extractors are on the contrary, more efficient when the scene is basically made up of non-structured objects, which is the case of natural environments, among others. In those cases, the SURF based algorithms have the advantage that they calculate a vast amount of features in the image, many of which vanish in following images, but a significant amount of them still remain in the following ones. That is the key point to match and track them for an efficient SLAM. SURF based algorithm also provide in those cases, an exhaustive enough, scale invariant feature descriptor that accomplishes the matching requirements for its tracking.

The use of an extended descriptor for Harris based corner detection and the scale invariant SURF features enable the sorting of the interest points into different clusters (three chosen clusters in this paper) dependant on their relevance. That allows the search of matching pairs in different stages according to the points relevance and the number of matched points necessary for the SLAM algorithm (ten in the presented results), according to the procedure described in Section 3.3, that reduces the computational effort. The criterium for matching two interest points in consecutive images evaluates both, the features correlation and the deviation of the distance from the evaluated pair to the average distance between of other matched pairs.

SLAM algorithm has been implemented using only visual information without considering any odometric or GPS information (which have been used afterwards to compare and evaluate the obtained results). The state of the system comprises a 12 variable array (position, orientation and their rates), where the inverse depth parametrization has been used in order to avoid the initialization of the distances to the detected visual features, that otherwise becomes a drawback when using SLAM outdoors in unknown environments. The rest of the state array is made up of the tracked features, with ten being the minimum allowed number. The prediction stage in EKF has been modeled considering constant velocity for both,

the position-orientation coordinates and the feature movements in the image plane. The correlation stage in the EKF uses a non-linear camera model that includes a pin-hole distortion model for the sake of more accurate results. Within the implemented SLAM algorithm, the Mahalanobis distance is used to disregard far away matched pairs that can otherwise distort the results.

The whole described procedure has been tested in several 3D semi-structured environments from a camera situated onboard an unmanned operated mini-UAV. The previous results show that the detected features covariance matrix decreases over time and that the structure made up by joining these detected features is coherent with the objects in the scene, with the absolute distance being a free parameter that has to be solved out by knowing the real distance between two known 3D points in the scene.

The performed flights were not closed loops, so that the UAV didn't come back to previous positions. Therefore the position-orientation correlation is always increasing in performed flights, even though the 3D position calculated by the SLAM has been compared with the GPS position and it is made clear that the horizontal positioning of the UAV is performed quite well by the SLAM in our experiments, where the flights had a dominant horizontal movement. The obtained MSE of the differences between the SLAM and the GPS horizontal coordinates decreases over time and has an approximate average value of $2m^2$ in our experiments. The altitude estimation doesn't show such a good correlation, due to the limited range of this movement during the flights, and for the same reason it has a lower MSE that is of around $0.14m^2$.

Acknowledgements The work reported in this paper is the consecution of several research stages at the Computer Vision Group—Universidad Politécnica de Madrid. The authors would like to thank Jorge Leon for supporting with the flight trials and I.A. Institute—CSIC for collaborating in the flights consecution. This work has been sponsored by the Spanish Science and Technology Ministry under the grant CICYT DPI2007-66156 by Comunidad Autónoma de Madrid under the grant SLAM visual 3D.

References

1. Se, S., Barfoot, T., Jasiobedzki, P.: Visual motion estimation and terrain modeling for planetary rovers. In: Proceedings of ISAIRAS (1995)
2. Sim, R., Elinas, P., Griffin, M., Little, J.J.: Vision-based SLAM using the Rao-Blackwellised particle filter. In: IJCAI Workshop on Reasoning with Uncertainty in Robotics (RUR) (2005)
3. Davison, A.J., Reid, I., Molton, N., Stasse, O.: MonoSLAM: real-time single camera SLAM. IEEE Trans. Pattern Anal. Mach. Intell. **29**(6), 1052–1067 (2007)
4. Munguia, R., Grau, A.: Monocular slam for visual odometry. In: IEEE International Symposium on Intelligent Signal Processing, 2007. WISP 2007, pp. 1–6. IEEE, Piscataway (2007)
5. Kim, S., Oh, S.-Y.: Slam in indoor environments using omni-directional vertical and horizontal line features. J. Intell. Robot. Syst. **51**(1), 31–43 (2008)
6. Choi, Y.-H., Oh, S.-Y.: Grid-based visual slam in complex environments. J. Intell. Robot. Syst. **50**(3), 241–255 (2007)
7. Montiel, J.M.M., Civera, J., Davison, A.J.: Unified inverse depth parametrization for monocular slam. In: Robotics: Science and Systems (2006)
8. Ho, K.L., Newman, P.: Detecting loop closure with scene sequences. Int. J. Comput. Vis. **74**(3), 261–286 (2007)
9. Lemaire, T., Berger, C., Jung, I., Lacroix, S.: Vision-based SLAM: stereo and monocular approaches. Int. J. Comput. Vis. **74**(3), 343–364 (2007)

10. Dailey, M., Parnichkun, M.: Simultaneous localization and mapping with stereo vision. In: Proceedings of the IEEE International Conference on Automation, Robotics, and Computer Vision (ICARCV) (2006)
11. Klippenstein, J., Zhang, H.: Quantitative evaluation of feature extractors for visual slam. In: Fourth Canadian Conference on Computer and Robot Vision, 2007. CRV '07., pp. 157–164 (2007)
12. Lee, Y.-J., Song, J.-B.: Autonomous selection, registration, and recognition of objects for visual slam in indoor environments. In: Fourth Canadian Conference on Computer and Robot Vision, 2007. CRV '07., pp. 668–673 (2007)
13. Mikolajczyk, M., Smid, C.: A performance evaluation of local descriptors. IEEE Trans. Pattern Anal. Mach. Intell. **27**(10), 1615–1630 (2005)
14. Törnqvist, D., Conte, G., Kärlsson, R., Schon, T.B., Gustafsson, F.: Utilizing model structure for efficient simultaneous localization and mapping for a uav application. In: Proceeding of the IEEE Aerospace Conference (2008)
15. Kim, J., Sukkarieh, S.: Real-time implementation of airborne inertial-slam. Robot. Auton. Syst. **55**(1), 62–71 (2007)
16. McLain, T.W., Beard, R.W., Barber, D.B., Redding, J.D., Taylor, C.N.: Vision-based target geolocation using a fixed-wing miniature air vehicle. J. Intell. Robot. Syst. **47**(4), 361–382 (2006)
17. Tsourdos, A., Aouf, N., Sazdovski, V., White, B.: Low altitude airbone slam with ins aided vision system. In: AIAA Guidance, Navigation and Control Conference and Exhibit, Hilton Head, South Carolina, AIAA (2007)
18. Mejías, L., Mondragón, I., Correa, J.F., Campoy, P.: Colibri: vision-guided helicopter for surveillance and visual inspection. In: Video Proceedings of IEEE International Conference on Robotics and Automation, Rome, April 2007
19. Mejias, L.: Control visual de un vehiculo aereo autonomo usando detección y seguimiento de características en espacios exteriores. Ph.D. thesis, Escuela Técnica Superior de Ingenieros Industriales. Universidad Politécnica de Madrid, Madrid, December 2006
20. Mejias, L., Saripalli, S., Campoy, P., Sukhatme, G.: Visual servoing of an autonomous helicopter in urban areas using feature tracking. J. Field Robot. **23**(3–4), 185–199 (2006)
21. Mejias, L., Campoy, P., Mondragon, I., Doherty, P.: Stereo visual system for autonomous air vehicle navigation. In: 6th IFAC Symposium on Intelligent Autonomous Vehicles (IAV 07), Toulouse, September 2007
22. Bay, H., Tuytelaars, T., Van Gool, L.: SURF: speeded up robust features. In: Proceedings of the Ninth European Conference on Computer Vision, May (2006)
23. Lowe, D.G.: Distintive image features from scale-invariant keypoints. Int. J. Comput. Vis. **60**(2), 91–110 (2004)
24. Fontanelli, D., Danesi, A., Bicchi, A.: Visual servoing on image maps. In: Springer Tracts in Advanced Robotics. Experimental Robotics, vol. 39. Springer, New York (2008)
25. Harris, C.G., Stephens, M.: A combined corner and edge detection. In: Proceedings of the 4th Alvey Vision Conference, pp. 147–151 (1988)
26. Parra, I., Fernández, D., Naranjo, J.E., García-García, R., Sotelo, M.A., Gavilán, M.: 3d visual odometry for road vehicles. J. Intell. Robot. Syst. **51**(1), 113–134 (2008)
27. Mozos, O.M., Gil, A., Ballesta, M., Reinoso, O.: In: Lecture Notes in Computer Science, Current Topics in Artificial Intelligence, Chapter Interest Point Detectors for Visual SLAM, vol. 4788, pp. 170–179. Springer, Berlin (2008)
28. Canny, J.: A computational approach to edge detection. IEEE Trans. Pattern Anal. Mach. Intell. **8**(6), 679–698 (1986)
29. OpenCV: Open Source Computer Vision Library OpenCV. http://www.intel.com/research/mrl/research/opencv/ (2001)
30. Wiklund, J., Caballero, F., Moe, A., De Dios, J.R.M., Forssen, P.-E., Nordberg, K., Ollero, A., Merino, L.: Vision-based multi-uav position estimation. In: Robotics And Automation Magazine, vol. 13, September 2006
31. Carmi, A., Oshman, Y.: On the covariance singularity of quaternion estimators. In: AIAA Guidance, Navigation and Control Conference, Hilton Head, South Carolina (Paper No. AIAA-2007-6814), 20–23 August 2007
32. Civera, J., Davison, A.J., Montiel, J.M.M.: Dimensionless monocular slam. In: IbPRIA, pp. 412–419 (2007)