Visual SLAM for Flying Vehicles

Bastian Steder Giorgio Grisetti Cyrill Stachniss Wolfram Burgard

***Abstract*— The ability to learn a map of the environment is important for numerous types of robotic vehicles. In this paper, we address the problem of learning a visual map of the ground using flying vehicles. We assume that the vehicles are equipped with one or two cheap down-looking cameras in combination with an attitude sensor. Our approach is able to construct a visual map that can later on be used for navigation. Key advantages of our approach are that it is comparably easy to implement, that it can robustly deal with noisy camera images, and that it can operate either with a monocular camera or a stereo camera system. Our technique uses visual features and estimates the correspondences between features using a variant of the PROSAC algorithm. This allows our approach to extract spatial constraints between camera poses which can then be used to address the SLAM problem by applying graph methods. Furthermore, we address the problem of efficiently identifying loop closures. We performed several experiments with flying vehicles which demonstrate that our method is able to construct maps of large outdoor and indoor environments.**

***Index Terms*— SLAM, vision, flying vehicles, attitude sensor**

I. Introduction

The problem of learning maps with mobile robots is a large and active research field in the robotic community. Traditional solutions to the simultaneous localization and mapping (SLAM) problem focus on learning 2D maps of large-scale environments [20]. Also different systems for building 3D maps have been proposed [7, 15, 22]. However, most of these approaches rely on bulky sensors having a high range and accuracy (e.g., SICK laser range finders) which cannot be used on robots such as small flying vehicles. As a result, several researchers focused on utilizing vision sensors instead of laser range finders. Cameras are an attractive alternative due to their limited weight and low power consumption. Existing approaches that address the vision-based SLAM problem mainly focus on scenarios in which a robot repeatedly observes a set of features [6, 14] and they have been shown to learn accurate feature maps.

This paper presents a system that allows aerial vehicles to acquire visual maps of large environments using an attitude sensor and low quality cameras pointing downwards. Such a setup can be found on different air vehicles such as blimps or helicopters. Our system deals with cameras that provide comparably low quality images which are also affected by significant motion blur. Furthermore, it can operate in two different configurations: with a stereo as well as with a monocular camera. If a stereo setup is available, our approach is able to learn visual elevation maps of the ground. If, however, only one camera is carried by the vehicle, our system can be applied by making a flat ground assumption providing a visual map without elevation information. To simplify the problem, we used an attitude (roll and pitch) sensor. In our system, we used an XSens MTi IMU, which has an error below 0.5 degrees. The advantages of our approach is that it is easy to implement, provides robust pose and map estimates, and that is suitable for small flying vehicles. Figure 1 depicts our blimp and helicopter used to evaluate this work as well as an example camera image obtained with our light-weight camera.

II. Related Work

Building maps with robots equipped with perspective cameras has received increasing attention in the last decade. Davison *et al.* [6] proposed a single camera SLAM algorithm based on a Kalman filter. The features are initialized by using a particle filter which estimates their depth. Montiel *et al.* [14] extended this framework by proposing an inverse depth parameterization of the landmarks. Since this parameterization can be better approximated by a Gaussian, the particle filter can be avoided in the initialization of the features. Subsequently, Clemente *et. al* [5] integrated this technique in a hierarchical SLAM framework which has been reported to successfully build large scale maps with comparably poor sensors.

All authors are members of the University of Freiburg, Department of Computer Science, D-79110 Freiburg, Germany

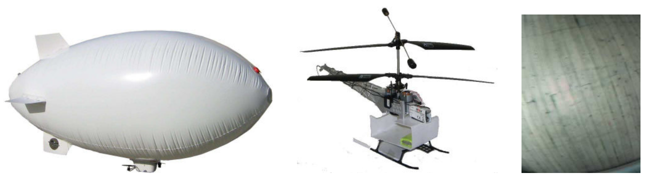
Fig. 1. Two aerial vehicles used to evaluate our mapping approach as well as an example image recorded from an on-board camera.

Chekhlov *et al.* [3] proposed an online visual SLAM framework which uses a SIFT-like feature descriptors and track the 3D motion of a single camera by using an unscented Kalman filter. The computation of the features is speeded up by utilizing the estimated camera position to guess the scale. Jensfelt *et al.* [10] proposed an effective way for online mapping applications by combining a SIFT feature extractor and an interest points tracker. While the feature extraction can be performed at low frequency, the movement of the robot is constantly estimated by tracking the interest points at high frequency.

Other approaches utilize a combination of inertial sensors and cameras. For example, Eustice *et. al* [7] rely on a combination of highly accurate gyroscopes, magnetometers, and pressure sensors to obtain a good estimate for the orientation and altitude of an underwater vehicle. Based on these estimates, they construct an accurate global map using an information filter based on high resolution stereo images. Piniés *et al.* [17] implemented a SLAM system for hand-held monocular cameras and employ an IMU to improve the estimated trajectories. Andreasson *et al.* [1] presented a technique that is based on a local similarity measure for images. They store reference images at different locations and use these references as a map. In this way, their approach is reported to scale well with the size of the environment.

Recently, Konolige and Agrawal [12] presented a technique inspired by scan-matching with laser range finders. These poses of the camera are connected by synthetic measurements obtained from incremental bundle adjustment performed on the images acquired at these poses, and an optimization procedure is used to find the configuration of camera poses which is maximally consistent with the measurement. Our approach uses a similar SLAM formulation but it computes the synthetic measurements between poses based on an efficient pairwise frame alignment technique.

Jung *et al.* [11] proposed a technique which is close to our approach. They use a high resolution stereo camera for building elevation maps with a blimp. The map consists of 3D landmarks extracted from interest points in the stereo image obtained by a Harris corner detector and the map is estimated using an Extended Kalman filter. Due to the wide field of view and the high quality of the images,

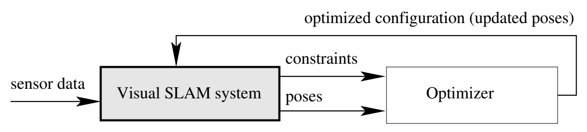

Fig. 2. System overview. This paper describes the visual SLAM system represented by the grayish box.

the non-linearities in the process were adequately solved by the EKF. In contrast to this, our approach is able to deal with low-resolution and low-quality images. It is suitable for mapping indoor and outdoor environments and for operating on small-size flying vehicles. We furthermore apply a more efficient error minimization approach [8] than the Kalman filter which is an extension of the work of Olson *et al.* [16].

Our previous work [19] focused more on the optimization approach and neglected the uncertainty of the vehicle when seeking for loop-closures. In the approach presented in this paper, we consider this uncertainty and provide a significantly improved experimental evaluation using real air vehicles.

III. Graph-based SLAM

In this paper, we address the SLAM problem by using its graph-based formulation. In this framework, the poses of the robot are described by the nodes of a graph. Edges between these nodes represent spatial constrains between them. They are typically constructed from observations or from odometry. Under this formulation, a solution to the SLAM problem is a configuration of the nodes which minimizes the error introduced by the constraints.

We apply an online variant of the 3D optimization technique recently presented by Grisetti *et al.* [8] to compute the maximum likely configuration of the nodes. Online performance is achieved by optimizing only the portions of the graph that require updates after introducing new constraints. Additional speedups result from reusing previously computed solutions to obtain the current one, as explained in [9]. Our system can be used as a black box to which one provides an initial guess of the position of the nodes as well as the edges and it computes the new configuration of the network (see Figure 2). The computed solution minimizes the error introduced by contradicting constraints.

In our approach, each node x_i models a 6DoF camera pose. The spatial constraints between two poses are computed from the camera images and the attitude measurements. An edge between two nodes i and j is represented by the tuple $\langle \delta_{ji}, \Omega_{ji} \rangle$, where δ_{ji} and Ω_{ji} are the mean and the information matrix of the measurement. Let $e_{ji}(\mathbf{x})$ be the error introduced by the constraint $\langle j, i \rangle$. Assuming the independence of the constraints, a solution to the SLAM problem is given by

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}} \sum_{\langle j,i \rangle} e_{ji}(\mathbf{x})^T \Omega_{ji} e_{ji}(\mathbf{x}). \quad (1)$$

Our approach relies on visual features extracted from the images obtained from two down-looking cameras. We use SURF features [2] which are invariant with respect to rotation and scale. Each feature is represented by a descriptor vector and the position, orientation, and scale in the image. By matching features between different images, one can estimate the relative motion of the camera and thus construct the graph which serves as input to the optimizer. In addition to that, the attitude sensor provides the roll and pitch angle of the camera. In our experiments, we found that the roll and the pitch measurements are comparably accurate even for low-cost sensors and can be directly integrated into the estimate. This reduces the dimensionality of each pose that needs to be estimated from $\mathbb{R}^6$ to $\mathbb{R}^4$.

In this context, the main challenge is to compute the constraints between the nodes (here camera poses) based on the data from the camera and the attitude sensor. Given these constraints, the optimizer processes the incrementally constructed graph to obtain estimates of the most likely configuration on-the-fly.

IV. Spatial Relation between Camera Poses

The input to the optimization approach mentioned in the previous section is a set of poses and constraints between them. In this section, we describe how to determine such constraints.

As a map, we directly use the graph structure of the optimizer. Thus, each camera pose corresponds to one node. Additionally, we store for each node the observed features as well as their 3D positions relative to the node. The constraints between nodes are computed from the features associated with the nodes. In general, at least three pairs of correspondences between image points and their 3D positions in the map are necessary to compute the camera position and orientation [18]. However, in our setting we need only two such pairs since the attitude of the camera is known from the IMU.

In practice, we can distinguish two different situations when extracting constraints: *visual odometry* and *place revisiting*. Odometry describes the relative motion between subsequent poses. To obtain an odometry estimate, we match the features in the current image to the ones stored in the previous n nodes. This situation is easier than place revisiting because the set of potential features correspondences is relatively small. In case of place revisiting, we compare the current features with all the features acquired from robot poses which lie within the 3σ confidence interval given by the pose uncertainty. This interval is computed with the approach of Tipaldi *et. al* [21] and applies covariance intersection on a spanning tree to obtain conservative estimates of the covariances. Since the number of features found during place revisiting can be quite high, we introduce a further approximation in the search procedure. First, we use only a small number of features from the current image when looking for potential correspondences. These features are the one which were better matched when computing visual odometry (which have the lowest descriptor distance). Second, we apply a *k*d-tree to efficiently query for similar features and we use the best-bins-first technique proposed by Lowe [13].

Every time a new image is acquired, we compute the current pose of the camera based on both visual odometry and place revisiting and augment the graph accordingly. The optimization of the graph is performed only if the computed poses are contradictory.

In the remainder of this section, we first describe how to compute a camera pose given a pair of known correspondences, and subsequently we describe our PROSAC-like procedure for determining the best transformation given a set of correspondences between the features in the current image and another set of features computed either via visual odometry or place revisiting.

A. Computing a Transformation from Feature Correspondences

In this section, we explain how to compute the transformation of the camera if we know the 3D position of two features f_1 and f_2 in the map and their projections i_1 and i_2 on the current image. Assuming known camera calibration parameters, we can compute the projections of the points on the normalized image plane. By using the attitude measurements from the IMU, we can compute the positions of these points as they would have been captured from a perfectly downwards facing camera. Let these transformed positions be i'_1, i'_2.

Subsequently, we compute the altitude of the camera according to the procedure illustrated in Figure 3, by exploiting the similarity of triangles. Once the altitude is known, we can compute the yaw of the camera by projecting the map features f_1 and f_2 into the same plane

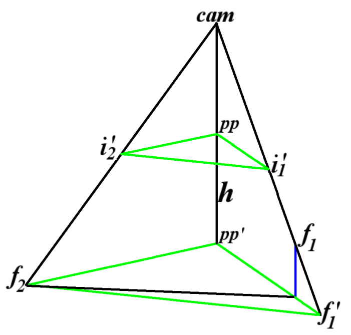

Fig. 3. This figure illustrates how to compute the height of the camera, given two corresponding features, under known attitude. cam is the camera position, i'_1, i'_2 are the projections of the features f_1 and f_2 on the normalized image plane, already rotated according to the attitude measured by the IMU. pp is the principle point of the camera (vertically downwards from the camera) on the projection plane. pp' and f'_1 are the projections of pp and f_1 at the altitude of f_2. h is the altitude difference between the camera and f_2, and it can be determined by exploiting the similarity of the triangles $\{i'_1, i'_2, pp\}$ and $\{f'_1, f_2, pp'\}$.

as i'_1, i'_2 and then the yaw is the angle between the two resulting lines on this plane.

Finally, we determine x and y as the difference between the positions of the map features and the projections of the corresponding image points, by reprojecting the image features into the map according to the known altitude and yaw angle.

B. Computing the Best Camera Transformation Based on a set of Feature Correspondences

In the previous section, we described how to compute the camera pose given only two correspondences. However, both visual odometry and place revisiting return a set of correspondences. In the following, we describe our procedure to efficiently select from the input set the pair of correspondences for computing the most likely camera transformation.

We first order these correspondences according to the Euclidean distance of their descriptor vectors. Let this ordered set be $C = \{c_1, ..., c_n\}$. Then we select pairs of correspondences in the order defined by the following predicate:

$$\begin{aligned}\langle c_{a_1}, c_{b_1}\rangle < \langle c_{a_2}, c_{b_2}\rangle \quad \Leftrightarrow \quad & (b_1 < b_2 \vee (b_1 = b_2 \wedge a_1 < a_2)) \\ & \wedge a_1 < b_1 \wedge a_2 < b_2. \end{aligned} \tag{2}$$

In this way, the best correspondences (according to the descriptor distance) are used first but the search procedure will not get stuck for a long time in case of false matches with low descriptor distances. This is illustrated in the following example: assume that the first correspondence c_1 is a false match. Our selection strategy generates the sequence $\langle c_1, c_2\rangle$, $\langle c_1, c_3\rangle$, $\langle c_2, c_3\rangle$, $\langle c_1, c_4\rangle$, $\langle c_2, c_4\rangle$, $\langle c_3, c_4\rangle$ ..... A pair without the false match $\langle c_2, c_3\rangle$ will be selected in the third step. A more naive selection strategy will try first all pairs of correspondences $\langle c_1, c_x\rangle$ with c_1 in the first position, and results in a less efficient search.

Only pairs that involve different features are used. The corresponding transformation T_{c_a,c_b} is then determined for the current pair (see section IV-A). This transformation is then evaluated based on the other features in both sets using a score function, which is presented in the next subsection. The process can be stopped, when a transformation with a satisfying score is found or when a timeout is reached. The solution with the highest score is returned as the current assumption for the transformation.

C. Evaluating a Camera Transformation

In the previous sections, we explained how to compute a camera transformation based on two pairs of corresponding features, and how to select those pairs from two input sets of features. By using different pairs, we can compute a set of candidate transformations. In this section, we explain how to evaluate them and how to choose the best one among them.

To select the best transformation, we rank them according to a score function. The score is computed by projecting the features in the map into the current camera image and by then comparing the distance between the feature positions in the image. The score is given by

$$\text{score}(T_{c_a,c_b}) \quad = \sum_{\{i \mid i \notin \{a,b\}\}} v(c_i). \tag{3}$$

In this equation, the function $v(c_i)$ is defined as the weighted sum of the relative displacement of the corresponding features in the current image and the Euclidean distance of their feature descriptors:

$$v(c_i) = 1 - \left[\alpha \frac{d^{\text{desc}}(c_i)}{d^{\text{desc}}_{\max}} + (1-\alpha)\frac{d^{\text{img}}(c_i)}{d^{\text{img}}_{\max}}\right] \tag{4}$$

In the sum of Eq. (3), we consider only those feature correspondences c_i whose distances $d^{\text{img}}(c_i)$ in the image and distances $d^{\text{desc}}(c_i)$ in the descriptor space are smaller than the thresholds $d^{\text{img}}_{\max}$ and $d^{\text{desc}}_{\max}$ introduced in Eq. (4). This prevents single outliers from leading to overly bad scores.

More in detail, $d^{\text{img}}_{\max}$ is the maximum distance in pixels between the original and the re-projected feature. In our experiments this value was set to 2 pixels for images of 320×240 pixels. The higher the motion blur in the image the larger this value should be set. The minimum value depends on the accuracy of the feature extractor. Increasing this threshold also allows the matching procedure to return less accurate solutions for the position estimation. The blending factor α mixes the contribution of the descriptor distance and the re-projection error. The more distinct the features, are the higher alpha can be chosen. In all our experiments, we set $\alpha = 0.5$. The value $d^{\text{desc}}_{\max}$ has been manually tuned. When using 64-dimensional SURF descriptors we had good results by setting this threshold to values around 0.3. The lower the quality of the image, the higher $d^{\text{desc}}_{\max}$ should be chosen.

Note that the technique to identify the correspondences between images is similar to the PROSAC [4] algorithm which is a variant of RANSAC. PROSAC takes into account a quality measure of the correspondences while sampling, conversely RANSAC draws the samples uniformly. We use the distance between feature descriptors as a quality measure. In our variant of PROSAC, the correspondences are selected deterministically. Since we only need two correspondences to compute the camera transformation, the chances that the algorithm gets stuck due to wrong correspondences are very small.

After identifying the transformation between the current pose of the camera and a node in the map, we can directly add a constraint to the graph. In the subsequent iteration of the optimizer, the constraint is thus taken into account when computing the updated positions of the nodes.

V. Experiments

In this section, we present the experiments carried out to evaluate our approach. We used only real world data which we partially recorded with a sensor platform carried in the hand of a person as well as with a real blimp and a helicopter (see Figure 1). In all experiments our system was running at 5 to 15 hertz on a 2.4 GHz Dual-core. Videos of the experiments can be downloaded at `http://www.informatik.uni-freiburg.de/~steder/homepage/videos`.

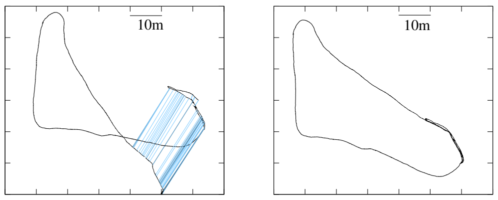

Fig. 4. The left image shows the path of the camera in black and the matching constraints in gray. The right image shows the corrected trajectory after applying the optimization technique.

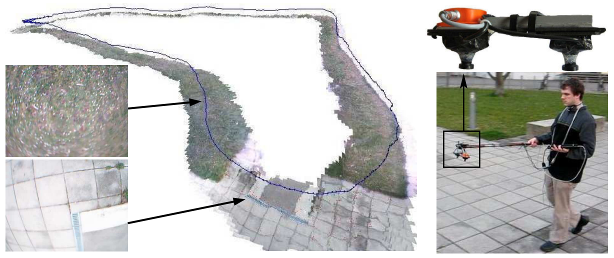

Fig. 5. The left image shows a perspective view of the map of the outdoor experiment together with two camera images recorded at the corresponding locations. The right one shows a person with the sensor platform mounted on a rod to simulate a freely floating vehicle.

A. *Outdoor Environments*

In the first experiment, we measured the performance of our algorithm using data recorded in outdoor environments. Since even calm winds outside buildings prevent us from making outdoor experiments with our blimp or our small size helicopter, we mounted a sensor platform on the tip of a rod and carried this by hand to simulate a freely floating vehicle. This sensor platform is equipped with two standard Web cams (Logitech Communicate STX). The person carried the platform along a long path around a building over different types of ground like grass and pavement. The trajectory has a length of about 190 m. The final graph contains approximately 1400 nodes and 1600 constraints. The trajectory resulting from the visual odometry is illustrated in the left image of Figure 4. Our system autonomously extracted data association hypotheses and constructed the graph. These matching constraints are colored light blue/gray in the same image. After applying our optimization technique, we obtained a map in which the loop has been closed successfully. The corrected trajectory is shown in the right image of Figure 4. A perspective view, which also shows the elevations, is depicted in Figure 5.

This experiment illustrates that our approach is able to build maps of comparably large environments and that it is able to find the correct correspondences between observations. Note that this result has been achieved without any odometry information and despite the fact that the cameras are of low quality and that the images are blurry due to the motion and mostly show grass and concrete.

B. *Statistical Experiments*

The second experiment evaluates the performance of our approach quantitatively in an indoor environment. The data was acquired with the same sensor setup as in the previous experiment. We moved in the corridor of our building which has a wooden floor. For a statistical evaluation of the accuracy of our approach, we placed artifical objects on the ground at known locations. We measured their locations manually with a measuring tape (up to an accuracy of approximately 3 cm). The distance in the x coordinate between neighboring landmarks is 5 m and 1.5 m in the y direction. The six

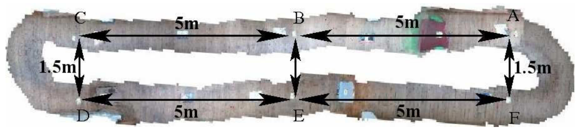

Fig. 6. Top view of the map of the indoor experiment. The image shows the map after least square error minimization. The labels A to F present six landmarks for which we determined the ground truth location manually to evaluate the accuracy of our approach.

TABLE I
ACCURACY OF THE RELATIVE POSE ESTIMATE BETWEEN LANDMARKS

landmarks	A-B	B-C	C-D	D-E	E-F	F-A	loop
mean error [m]	0.18	0.26	0.11	0.20	0.21	0.12	1.10
sigma [m]	0.21	0.32	0.12	0.39	0.3	0.15	1.25
error [%]	3.6	5.2	7.3	4.0	4.2	8.0	4.8

landmarks are labeled A to F in Figure 6. We used these six known locations as ground truth, which allowed us to measure the accuracy of our mapping technique. Figure 6 depicts a resulting map after applying the least square error minimization approach. We repeated the experiment 10 times and measured the relative distance between them.

Table I summarizes this experiment. As can be seen, the error of the relative pose estimates is always below 8% and typically around 5% compared to the true difference. This results mainly from the error in our self-made and comparably low quality stereo setup. To our opinion, this is an accurate estimate for a system consisting of two cheap cameras and an IMU, lacking sonar, laser range data, and real odometry information.

C. *Experiments with a Blimp*

The third experiment is also a statistical analysis carried out with our blimp. The blimp has only one camera looking downwards. Instead of the stereo setup, we mounted a sonar sensor to measure its altitude. Furthermore, no attitude sensor was available and we therefore assumed the roll and pitch angle to be zero (which is an acceptable approximation given the smooth motion of a blimp). We placed two landmarks on the ground with a distance of 5 m and flew 10 times over the scene. The mean estimated distance between the two landmarks was 4.91 m with a standard deviation of 0.11 m. Thus, the real position was within the 1σ interval.

The next experiment in this paper is designed to illustrate that such a visual map can be used for navigation. We constructed the map shown in Figure 7 with our blimp. During this task, the blimp was instructed to return always to the same location and was repeatedly pushed away several meters. The blimp was always able to register its current camera image against the map constructed so far and in this way kept track of its location relative to the map. This enabled the controller of the blimp to steer the air vehicle to the desired location. The experiment lasted 18 min and the blimp recorded during that time around 10,800 images. The robot processed around 500,000 features and the map was constructed online.

D. *Experiments with a Light-weight Helicopter*

We finally mounted an analog RF-camera on our light-weight helicopter depicted in Figure 1. This helicopter is not equipped with an attitude sensor nor with a sonar sensor to measure its altitude. Since neither stereo information nor the elevation of the helicopter is known, the scale of the visual map was determined by a known size of one landmark (a book lying on the ground). Furthermore, the attitude was assumed to be zero which is a quite rough approximation

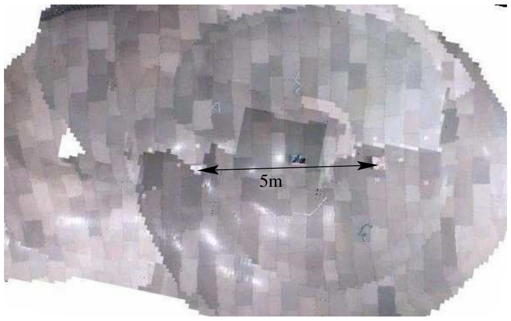

Fig. 7. Map constructed by the blimp. The ground truth distance between both landmarks is 5 m and the estimated distance was 4.91 m with 0.11 m standard deviation (10 runs). The map was used to autonomously steer the blimp to user specified locations.

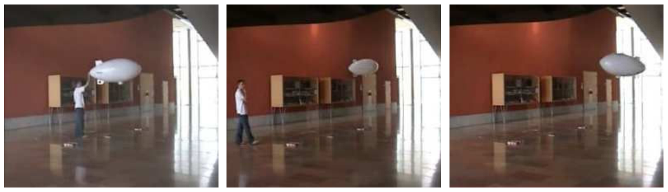

Fig. 8. A person pushes the blimp away. The blimp is able to localize itself and navigate back using the map shown in Figure 7 (see video material).

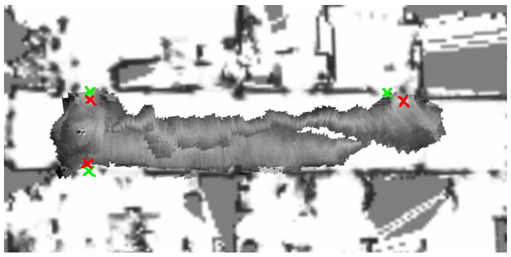

Fig. 9. Visual map build with a helicopter overlayed on a 2D grid map constructed from laser range finder data recorded with a wheeled robot.

for a helicopter. We recorded a dataset by flying the helicopter and overlayed the resulting map with an occupancy grid map recorded from laser range data with a wheeled robot. Figure 9 depicts the result. The red and green crosses indicate the same locations in the occupancy grid map and the visual map. Even under the hard sensory limitations, our approach was able to estimate its position in a quite accurate manner. The helicopter flew a distance of around 35 m and the map has an error in the landmark locations that varies between 20 cm and 60 cm.

VI. Conclusions

In this paper, we presented a robust and practical approach to learn visual maps based on down looking cameras and an attitude sensor. Our approach applies a robust feature matching technique based on a variant of the PROSAC algorithm in combination with SURF features. The main advantages of the proposed methods are that it can operate with monocular or with a stereo camera system, that it is easy to implement, and that it is robust to noise in the camera images.

We presented a series of real world experiments carried out with a small-size helicopter, a blimp, and by manually carrying a sensor platform. Different statistical evaluations of our approach show its ability to learn consistent maps of comparably large indoor and outdoor environments. We furthermore illustrated that such maps can be used for navigation tasks of air vehicles.

Acknowledgment

This work has partly been supported by the DFG under contract number SFB/TR-8 and by the EC under contract numbers FP6-2005-IST-6-RAWSEEDS and FP6-IST-34120-muFly.

References

[1] H. Andreasson, T. Duckett, and A. Lilienthal. Mini-SLAM: Minimalistic visual SLAM in large-scale environments based on a new interpretation of image similarity. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Rome, Italy, 2007.

[2] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2006.

[3] D. Chekhlov, M. Pupilli, W. Mayol-Cuevas, and Calway A. Robust real-time visual slam using scale prediction and exemplar based feature description. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, Minneapolis, usa, 2007.

[4] O. Chum and J. Matas. Matching with PROSAC - progressive sample consensus. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, Los Alamitos, USA, 2005.

[5] L.A. Clemente, A. Davison, I. Reid, J. Neira, and J.D. Tardós. Mapping large loops with a single hand-held camera. In *Proc. of Robotics: Science and Systems (RSS)*, Atlanta, GA, USA, 2007.

[6] A. Davison, I. Reid, , N. Molton, and O. Stasse. MonoSLAM:real time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6), 2007.

[7] R.M. Eustice, H. Singh, J.J. Leonard, and M.R. Walter. Visually mapping the RMS Titanic: conservative covariance estimates for SLAM information filters. *Int. Journal of Robotics Research*, 25(12), 2006.

[8] G. Grisetti, S. Grzonka, C. Stachniss, P. Pfaff, and W. Burgard. Efficient estimation of accurate maximum likelihood maps in 3D. In *Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS)*, San Diego, CA, USA, 2007.

[9] G. Grisetti, D. Lodi Rizzini, C. Stachniss, E. Olson, and W Burgard. Online constraint network optimization for efficient maximum likelihood map learning. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Pasadena, CA, USA, 2008.

[10] P. Jensfelt, D. Kragic, J. Folkesson, and M. Björkman. A framework for vision based bearing only 3D SLAM. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Orlando, CA, 2006.

[11] I. Jung and S. Lacroix. High resolution terrain mapping using low altitude stereo imagery. In *Proc. of the Int. Conf. on Computer Vision (ICCV)*, Nice, France, 2003.

[12] K. Konolige and M. Agrawal. Frame-frame matching for realtime consistent visual mapping. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Rome, Italy, 2007.

[13] D.G. Lowe. Object recognition from local scale-invariant features. In *Proc. of the Int. Conf. on Computer Vision (ICCV)*, Krekyra, Greece, 1999.

[14] J.M. Montiel, J. Civera, and A.J. Davison. Unified inverse depth parameterization for monocular SLAM. In *Proc. of Robotics: Science and Systems (RSS)*, Cambridge, MA, USA, 2006.

[15] A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann. 6d SLAM with approximate data association. In *Proc. of the 12th Int. Conference on Advanced Robotics (ICAR)*, pages 242–249, 2005.

[16] E. Olson, J. Leonard, and S. Teller. Fast iterative optimization of pose graphs with poor initial estimates. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 2262–2269, 2006.

[17] P. Piniés, T. Lupton, S. Sukkarieh, and J. D. Tardós. Inertial aiding of inverse depth slam using a monocular camera. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 2797–2802, Rome, Italy, 2007.

[18] L. Quan and Z.-D. Lan. Linear N-Point camera pose determination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(8):774–780, 1999.

[19] B. Steder, G. Grisetti, S. Grzonka, C. Stachniss, A. Rottmann, and W. Burgard. Learning maps in 3D using attitude and noisy vision sensors. In *Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS)*, San Diego, CA, USA, 2007.

[20] S. Thrun. An online mapping algorithm for teams of mobile robots. *Int. Journal of Robotics Research*, 20(5):335–363, 2001.

[21] G.D. Tipaldi, G. Grisetti, and W. Burgard. Approximate covariance estimation in graphical approaches to SLAM. In *Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS)*, San Diego, CA, USA, 2007.

[22] R. Triebel, P. Pfaff, and W. Burgard. Multi-level surface maps for outdoor terrain mapping and loop closing. In *Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS)*, Beijing, China, 2006.