Proceedings of the 2007 IEEE/RSJ International
Conference on Intelligent Robots and Systems
San Diego, CA, USA, Oct 29 - Nov 2, 2007

Combined Grid and Feature-Based Occupancy Map Building in Large Outdoor Environments

Franz Andert and Lukas Goormann
German Aerospace Center, Institute of Flight Systems
38108 Braunschweig, Germany
{*franz.andert, lukas.goormann*}*@dlr.de*

***Abstract*— This paper presents an approach to create three-dimensional occupancy maps from an aerial vehicle with stereo vision. The main idea is to create an occupancy grid that moves along with the vehicle and extract features into a fixed global map. Vice versa, global features or a-priori knowledge can be inserted into the grid. The maps are calculated onboard to be used for autonomous behavior like path planning and obstacle avoidance. With the described method, maps are created and updated in real-time, and due to its flexibility, the vehicle is not restricted to a pre-defined area. The developed approach has been demonstrated in flights with a small unmanned helicopter.**

I. Introduction

Mapping the world with the help of environmental sensors is one of the main tasks in robotics. Maps are used for autonomous applications like path planning and obstacle avoidance and in contrast to classical cartography, these maps must be calculated onboard in real-time. For more than 20 years, occupancy grids have been used for world modeling [1], [2]. There are numerous advanced research applications using different techniques reported in the literature, for example using sonar sensors [3], [4], [5], [6], laser scanning systems [7], [8], [9] and stereo vision based on real-time image processing made possible by powerful computer systems [10], [11], [12], [13]. Grid maps can work with different kinds of sensors allowing simple and efficient data fusion to integrate multiple sensors, insert complete data sequences and detect or remove measurement errors and noise. Unlike fixed objects, moving ones are hard to detect and often not recognized, though. Since data is stored for every cell, a lot of memory is needed, and the map has to be limited to a specified region whose maximal size will depend on the cell size and the available memory for data storage.

On the other hand, many applications use feature-based maps, where environmental properties are identified and stored separately. For example, the world can be represented by topological graphs [14], velocity obstacles [15], or polygonal objects. In the later case, simplifications are often used, e.g. vertical walls are assumed [16]. Feature maps take much less memory, dependent on the amount of objects, and they are not limited to static boundaries since item lists are used instead of cell arrays.

It is a straightforward procedure to generate a grid map from sensor data and extract features out of it. But outdoor scenarios can be too large to store the whole scene in a data array of an accurate resolution. Additionally, the area boundaries may be unknown before mapping.

The approach presented in this paper tackles this problem. It concentrates on detecting static objects and uses the fact that the actual sensor information has no influence on regions that are invisible or out of the robot's sensor range. In these regions, an occupancy grid for data fusion is not needed. Hence, occupancy grids are only used in a small region around the sensor. Characteristic features are recognized and inserted to a global map that will take less memory and is easily expandable. This map is not restricted to the sensor environment and is used for path planning and other applications.

The robot used in this investigation is called *maxiARTIS* (Autonomous Rotorcraft Testbed for Intelligent Systems), see figure 1. It is an autonomous helicopter with a main rotor diameter of 3 meters and a total weight of maximal 25 kg [17]. Flights of more than 30 minutes are possible.

Fig. 1. The helicopter *maxiARTIS*.

For the application described in this paper, it is equipped with a stereo camera and a separate vision computer for camera control and image processing. Additionally, the helicopter's position and attitude is provided in six degrees of freedom by a navigation solution using a GPS sensor, a magnetometer and an inertial measurement unit. Figure 2 illustrates the connection of these components.

II. Environmental Perception

For depth measurement, a stereo camera with a baseline of 30 cm and a field of view of approximately $51° \times 40°$ is used. It creates images with 640×480 pixels and has an inbuilt FPGA processor that calculates a depth image out of

1-4244-0912-8/07/$25.00 ©2007 IEEE.

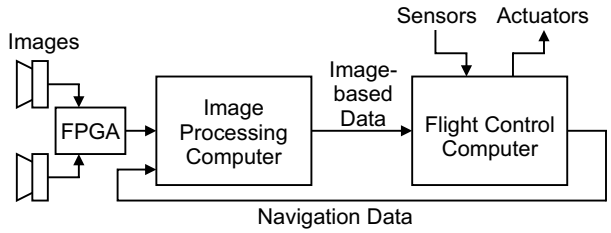

Fig. 2. Overview of the onboard hardware for vision applications, including the connection to the flight controller.

the two input images in realtime with 30 Hz. This image, see figure 3, is a result of a complex processing step where regions of the two camera images are matched [18], and it acts as a depth sensor in the mapping process. The depth image is a discrete 2-D function $D : d(u, v)$ with distance information about an object projected to the pixel at (u, v) on the image plane.

As it can be derived from theoretical considerations, the range of depth measurements is limited to $d \in [d_{\min}, \infty]$ with a minimal distance $d_{\min} > 0$. The error Δd is given by the range resolution

$$\Delta d = \frac{d^2}{b\, f} \cdot \sigma \tag{1}$$

based on the camera parameters baseline (b), focal length (f) and a parameter σ that describes the uncertainty of a region match between the left and right image. For the parameter σ, the depth estimation algorithm provides an accuracy of 1/16 pixel. However, a quarter pixel size is assumed to increase the robustness to noise.

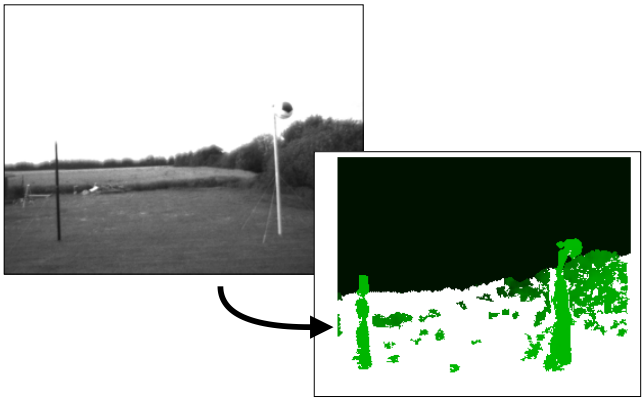

Fig. 3. Left camera image and generated depth image of this stereo pair. Darker green values represent greater distances, missing values are marked white.

The depth function may be undefined for some pixel coordinates due to bad or indeterminable depth estimation at that point. Here, missing information for a depth image pixel is either ignored or filled up with infinite distance if the original image coordinate leads to an empty *sky* region where no objects are assumed. Since sky is usually bright and low-textured, stereo-based depth estimation is quite poor and a filter adapted from [19] is used for a fast detection of image regions with the attributes mentioned.

III. Occupancy Grid Mapping

A. The Basic Approach

An occupancy grid M is a discrete 3-D sample of the space with occupied and free areas. The literature presents and discusses many probabilistic ways how sensor data is regarded for map building, including noisy sensor models and pose estimation. For simplicity, a grid map is an array of cells $m(x, y, z)$ with likelihood values that describe the presence of objects at that world coordinate. Positive values represent occupied regions. Their absolutes can be regarded as a significance of being occupied or free. Hence, zero values represent unknown regions.

At time t, the map M_t is built out of the current and all previous sensor information, i.e.

$$M_t = \bigcup_{\tau=1}^{t} \langle D_\tau, \mathbf{p}_\tau \rangle, \tag{2}$$

where $\langle D_\tau, \mathbf{p}_\tau \rangle$ is an interpretation of a depth image D recorded from a camera pose $\mathbf{p} = [x, y, z, \phi, \theta, \psi]$ at time τ. The pose is known through the navigation data and a constant camera misalignment on the carrier.

A single depth image contains information about free and occupied areas that are in the sight pyramid of the actual camera pose. A grid $\widehat{M}$ is built that only has information about the actual image. It is

$$\langle D, \mathbf{p} \rangle = \widehat{M}. \tag{3}$$

B. Sensor Interpretation

The calculation of $\widehat{M}$ out of a depth image is done by interpreting every image pixel as one measurement. A depth image value $d(u, v)$ refers to objects on a plane perpendicular to the camera axis and with a distance d to the camera center. Since every valid pixel leads to a unique ray in space using the pinhole camera model, the object coordinates are reconstructed by intersecting the ray and the depth plane using projective geometry.

Figure 4 illustrates the interpretation of one pixel and its effect to an empty occupancy grid. A depth measurement leads to the obstacle found at the specific distance and, additionally, to free space in front of the obstacle. The area behind the obstacle is unknown [2]. Higher likelihood values (top) are an indicator for a high confidence that a cell is occupied and are represented by dark cells in the grid (bottom). Vice versa, lower values lead to free areas, illustrated by brighter cells.

Since the image data is not perfect, enhancements are added to get a more realistic interpretation:

- The minimal distance $d_{\min}$ and furthermore, a maximal distance $d_{\max}$ is applied so that too far distance measurements lead to free areas only up to this maximum. Boundary values for our camera system are 8m and 50m, respectively.
- Smooth transitions between *free*, *occupied* and *unknown* information to consider an uncertainty Δd in distance

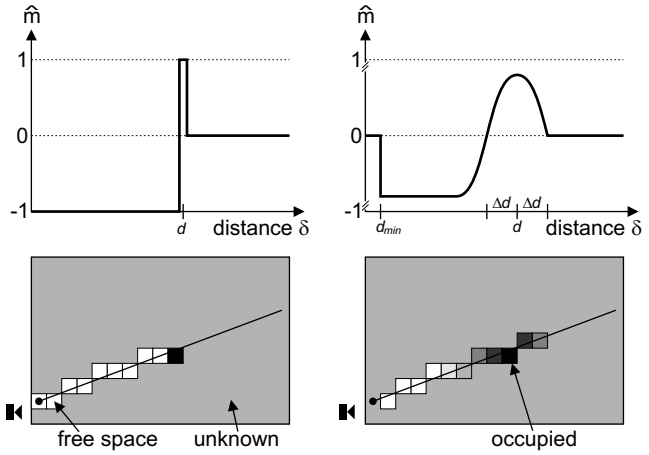

Fig. 4. Interpretation of an ideal depth image pixel (left) and a more realistic model(right).

measurement which is given by equation (1). Δd increases disproportionately with larger distance values when using stereo vision.

Finally, a set of linear equations is used for the sensor model to get a fast implementation. The definition is

$$\widehat{m}(\delta) = -1 \tag{4}$$

for $d_{\min} \leq \delta < d - 2\Delta d$ and

$$\widehat{m}(\delta) = 1 - \frac{|\delta - d|}{\Delta d} \tag{5}$$

for $d - 2\Delta d \leq \delta < d + \Delta d$. Otherwise, it is $\widehat{m}(\delta) = 0$ so that these ray sections have no influence on the map.

In the equations (4) and (5), $\widehat{m}(\delta)$ is the cell of the grid $\widehat{M}$ next to the intersection coordinate between the ray defined by the pixel and a depth plane with the distance δ. To draw the whole ray, δ is iterated from $d_{\min}$ to $d_{\max}$.

C. Local Mapping

The actual grid $\widehat{M}$ is initialized with cell values of 0 and added with the rays of each pixel using the interpretation model of the previous section. Each ray is transformed into global coordinates using the camera pose at the time the image has been recorded and a line is drawn into the grid using a modified Bresenham algorithm [20]. As a result, grid cells inside the sight pyramid of the camera are filled with occupancy likelihood values. If some cells are filled more than once, the maximal value is written to enforce the mapping of obstacles. After inserting all lines, gaussian blur is applied to the local grid.

Figure 5 shows an example of a local mapping. Corresponding obstacles in the image and in the map are marked. Contrary to many indoor robotics, the maps are built in three dimensions, so that a separate floor plane detection is not needed.

The data fusion between succeeding image frames is done using equations (2) and (3), it is given by

$$M_t = \bigcup_{\tau=1}^{t} \widehat{M}_\tau, \tag{6}$$

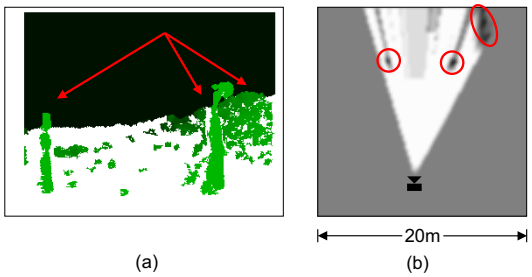

Fig. 5. Depth image (a) and a 2-D view of the local map that was built out of it (b).

or recursively $M_t = M_{t-1} \cup \widehat{M}_t$ with an initialization of M_0: $\forall x, y, z : m_0(x, y, z) = 0$. The unification is done by adding the cell values, i.e.

$$m_t(x, y, z) = m_{t-1}(x, y, z) + \widehat{m}_t(x, y, z). \tag{7}$$

Similar to other approaches, free or occupied cells become more confident if measured several times. With that, especially unmoved objects can be determined easily. In practice, the map values are truncated to a specified range so that integers can be used for the cell array. The range must be large enough to ensure the robustness to failures.

D. Creating Global Maps

The data fusion of the map M with one image will only affect the grid cells inside a small environment of the actual position. There is no need to store values of a local map $\widehat{M}$ outside this environment.

To insert the whole sensor data into $\widehat{M}$, its definition range must be large enough so that the sight pyramid fits. To set the camera at the center for any heading or pitch angle, the zone must be a cube with an edge length of a little more than $2\,d_{\max}$, dependent on the camera aperture angles. If the size is smaller, far measurements may be outside and they are omitted in this case. In practice, the environment size in x- and y-direction should not limit the maximal sensor range. The size in z-direction can be smaller since a lower vertical speed of the helicopter is assumed and there is no need to store objects far above or below the actual flight altitude.

Finally, this environmental grid $\widehat{M}$ is inserted into the global map M using equation (7). Hence, only those global map parts need to be grid-based which are interacting with $\widehat{M}$. Further, the grid array that stores M is defined in only a small environment.

To avoid shifting a large number of map cells when moving, the implementation divides the global map into cuboidal zones. Their position is fixed. The size of each zone is equal to the sensor environment so this is overlapping with maximal eight zones of the global 3-D map. The environmental cuboid $\widehat{M}$ moves through M with the movement of the camera, i.e. with the helicopter. Only those global zones overlapping with $\widehat{M}$ are represented by an occupancy grid.

The actual camera and environment position is shown in figure 6 (a), the other graphics show possible effects on the map, caused by the next measurement. The camera stays next to the same grid cell, thus not affecting the definition range of $\widehat{M}$, despite rotations and small movements of the

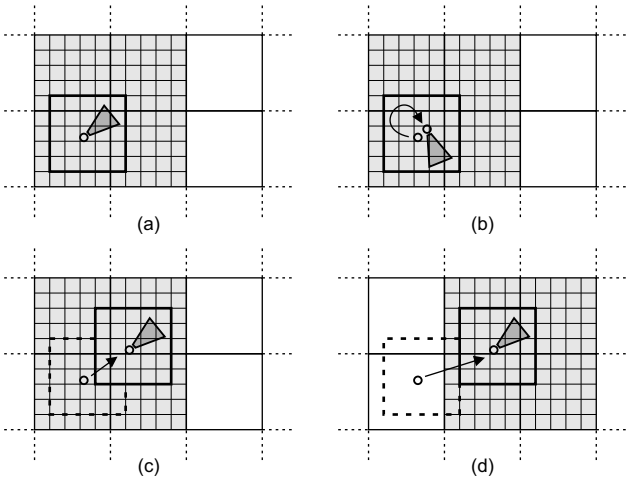

Fig. 6. 2-D view of the global map that is divided into zones. For zones around the actual camera position, an occupancy grid representation exists.

helicopter (b). If the movement is larger (c), the environment $\widehat{M}$ moves so that its center will always be the cell next to the camera position.

If global zone boundaries are crossed (d), new parts of M are allocated for these zones. Grid information is discarded for zones of M that fall outside.

IV. Feature Mapping

A. Creating Bounding Boxes and Object Shapes

In every map zone where a grid exists, features are detected by segmenting the grid into occupied and free areas, applying a threshold. A single object is a set of occupied cells that are connected in a 6-neighborhood of the 3-D array. These objects are recognized with a flood fill algorithm. By saving the minimal and maximal values of the x, y, and z-coordinates of cells belonging to the object, the bounding box is calculated and put into the global map as it is illustrated in figures 7 and 8. Unlike the cell array, these boxes are saved when the occupancy grid moves and the features are available to further applications, independent of the existence of a grid in that zone. The features are detected separately for each map zone, so that a bounding box will always be inside the boundaries of one zone. Objects that belong physically to more than one map zone are represented by multiple features.

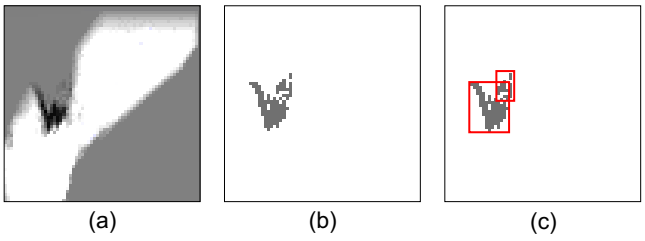

Fig. 7. Extracting features from an occupancy grid (a) with thresholding (b) and bounding box calculation (c). Bounding boxes can overlap.

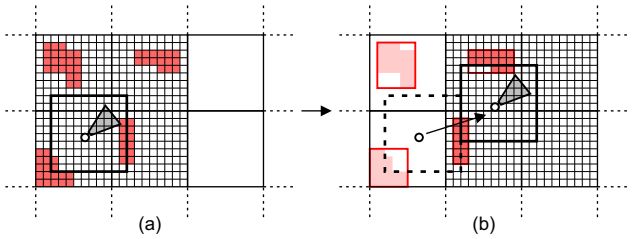

Fig. 8. Features are stored when the grid data is discarded due to helicopter movement.

Saving the bounding boxes leads to a great data reduction since they do not contain the object shapes. But they will only be a useful object representation in rather unoccupied environments, e.g. outdoor scenarios with large free spaces and single obstacles like trees. Free paths inside bounding boxes of tunnel or canyon walls will not be found in the feature map since the whole box is regarded as occupied in the feature map.

As an improvement, shape detection algorithms are applied, and the shape of each object is stored together with the box. Let the output of the flood fill algorithm, i.e. the grid-based shape of an object be denoted as an array $s(x, y, z)$ where $s = 0$ represent free and $s = 1$ occupied cells. By assuming vertical walls, objects can be modeled as prisms with a ground plane that is represented by a 2-D array s'. Its elements are calculated using

$$s'(x,y) = \begin{cases} 0, & \text{if } \sum_z s(x,y,z) = 0; \\ 1, & \text{otherwise.} \end{cases} \tag{8}$$

As a flexible solution, a quadtree of s' is built and stored together with the bounding box so that the object height and size in z-direction is not lost.

B. Re-Integration of Features into the Grid

To complete the exchange of elements between grid and feature-based map representations, single objects of the global map must be inserted into the grid. This is done when a new grid zone is allocated and some world map objects exist there. First, the global map is searched for bounding boxes. Then, those grid cells that intersect with an object shape are marked as occupied by giving them a high likelihood value. The other cells remain unexplored.

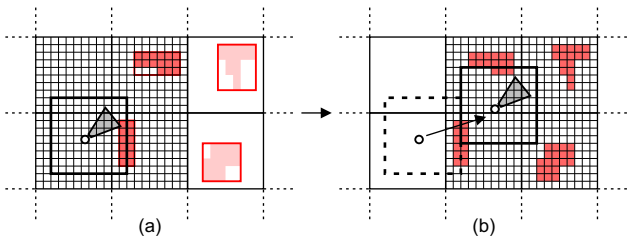

Fig. 9. Inserting features to the grid if there are objects inside a new zone.

V. Flight Tests and Results

The map building algorithm is tested on video sequences that have been recorded by the onboard stereo camera. The paper presents results of three scenarios.

At first, the aim is to detect the obstacles as seen in figure 10. There is a straight line of bushes between the fields and a small aircraft on the left side. The mapping is done at a rather low resolution for fast processing. A grid with a cell size of 1 m and zones with each $80 \times 80 \times 10$ cells are used. These small-sized zones do not integrate all depth measurements, but increase the calculation speed and allow a demonstration of allocating new grid parts in shorter flights.

Fig. 10. Left stereo camera image, recorded during a flight.

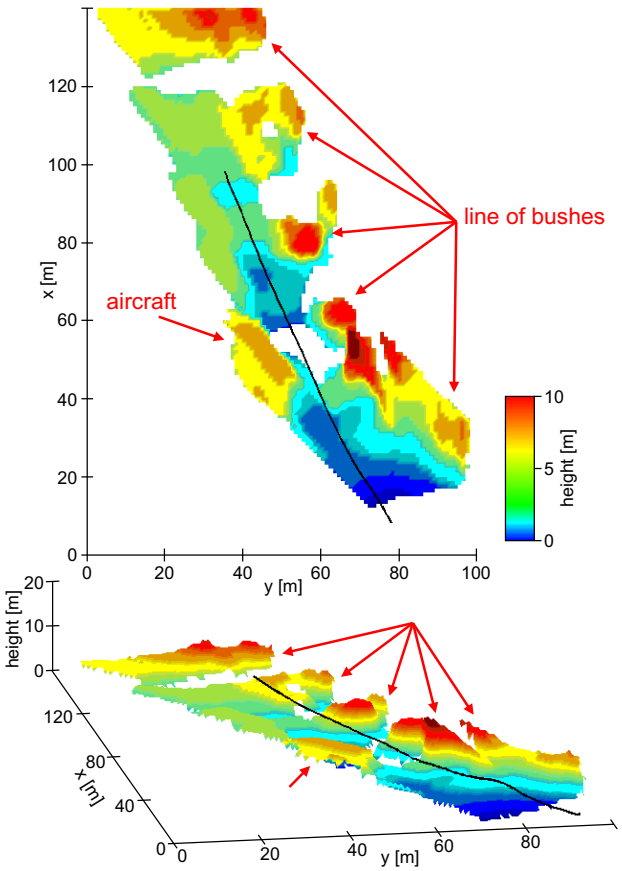

Fig. 11. Height profile map and flight trajectory in a 2-D (top) and 3-D view (bottom).

For viewing purposes, a height profile representation of the map is used. It shows a grid in x- (north) and y-direction (east). For each 2-D cell of that profile, the occupied map cell at this (x, y)-coordinate with maximal height is drawn with a height-specific color. However, holes are discarded in this representation but in the cases shown here, the information loss is negligible. Figure 11 shows the output of the first flight test. The bushes and the aircraft have been detected, and the floor too. Some missing data remains due to noise and the mapping size: objects of 5 meters below the helicopter are not recorded to the map. When looking at the end of the flight path, objects with a distance of 40 meters can be detected correctly. On a 3.0 GHz computer, the calculation speed is approximately 15 frames per second when using the original-sized camera images with 640×480 pixels.

In the second flight (figures 12 and 13), a line of trees is crossed and the mapping is done at a higher resolution of 0.5 m and with a larger environmental grid of $160 \times 160 \times 80$ cells. Here, no holes from missing data are remaining in the mapped area, the white space in the curve is due to the higher flight altitude there. The line of trees has been detected correctly and the marked higher trees are also mapped.

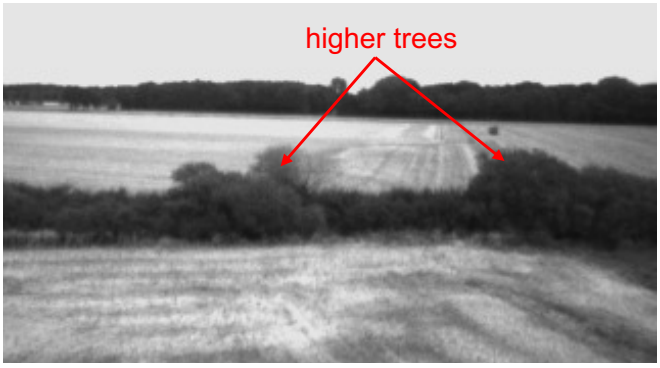

Fig. 12. A line of trees near the flight field.

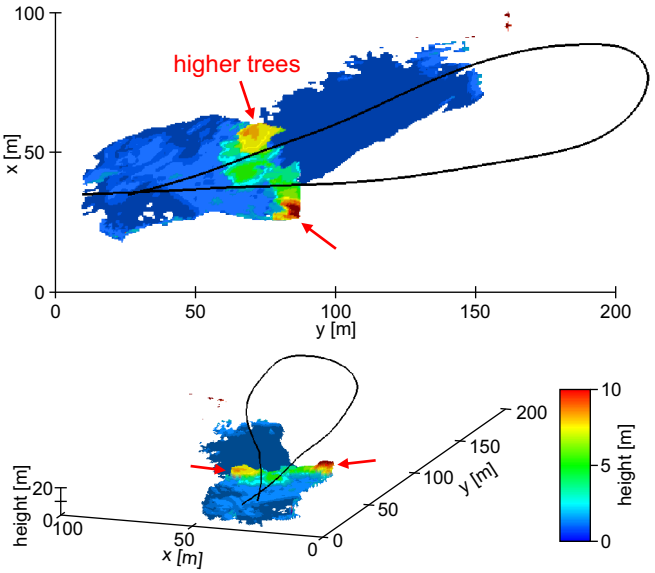

Fig. 13. Flight trajectory and height profile map.

For the third flight test (fig. 14), two posts are positioned on the flight field. They are spaced at intervals of 7 meters. The mapping is done with $160 \times 160 \times 20$ cells and a resolution of 0.5 m. Two flights are done as seen in figures 15 and 16, a fly-through and a pirouette around while heading the nose to the center. In the map images, the posts are visible but there is a lot of missing data in the free area that describes the floor height. Although the floor detection is not optimal in this case, the posts have been detected correctly and the detected positions are nearly the same in both flights. They are marked in the height profile images.

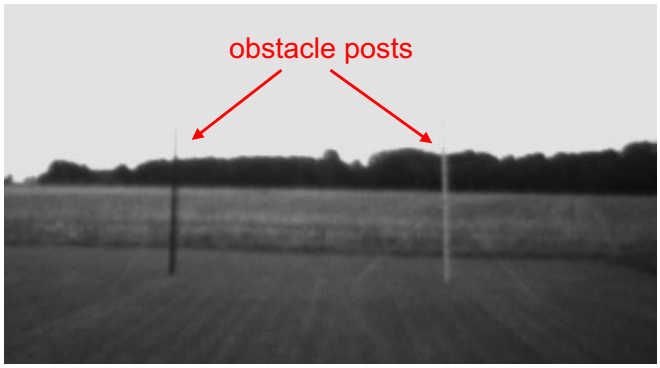

Fig. 14. Two posts positioned on the flight field.

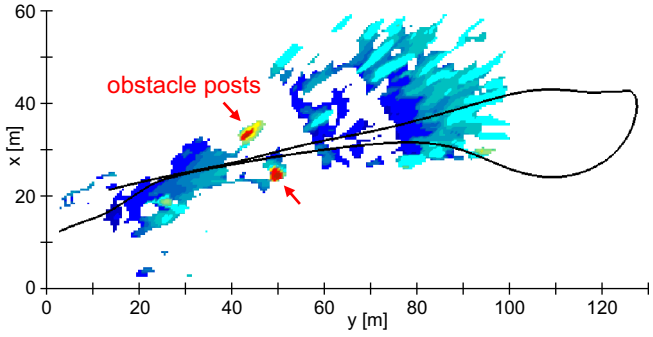

Fig. 15. Mapping the posts by flying through.

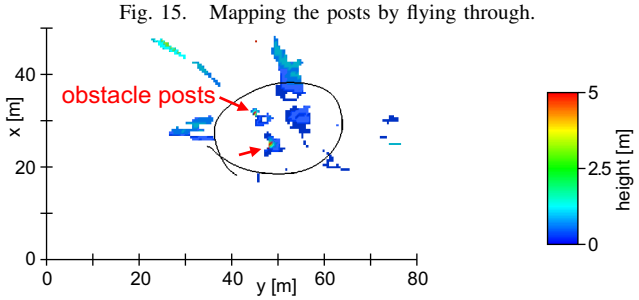

Fig. 16. Mapping the posts by flying around.

VI. Conclusion

The paper shows a technique for obstacle mapping. With the help of an occupancy grid, free areas and fixed obstacles can be detected in unknown scenarios. The grid map is available in only a small environment around the helicopter and moves along with the helicopter movement. Connected occupied cells are regarded as single objects. Their bounding box and their shape is put into a feature-based map. In contrast to the occupancy grid, the feature map is global. The investigations reported in this paper have shown that the image quality is sufficient to build maps of the given resolution and that localization with navigation data works. The map calculation is fast enough to be an input for further real-time applications.

Future work will concentrate on improvements to the global map. The recognition of polygonal shapes and the merging of large objects is one task. Another challenge is the tracking of features to handle moving objects. Applications like autonomous path planning are currently developed. The goal is an on-board obstacle avoidance system that considers actual map updates for an immediate replanning of the flight trajectory.

References

[1] H. P. Moravec and A. Elfes, "High resolution maps from wide angle sonar," in *IEEE International Conference on Robotics and Automation*, vol. 2, March 1985, pp. 116–121.

[2] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *IEEE Computer*, vol. 22, no. 6, pp. 46–57, 1989.

[3] U. Raschke and J. Borenstein, "A comparison of grid-type map-building techniques by index of performance," in *IEEE International Conference on Robotics and Automation*, vol. 3, 1990, pp. 1828–1832.

[4] J. Borenstein and Y. Koren, "The vector field histogram –fast obstacle avoidance for mobile robots," *IEEE Journal of Robotics and Automation*, vol. 7, no. 3, pp. 278–288, 1991.

[5] K. Konolige, "Improved occupancy grids for map building," *Autonomous Robots*, vol. 4, pp. 351–367, 1997.

[6] S. Thrun, "Learning occupancy grids with forward sensor models," *Autonomous Robots*, vol. 15, pp. 111–127, 2003.

[7] S. Thrun, W. Burgard, and D. Fox, "A real-time algorithm for mobile robot mapping with applications to multi-robot and 3d mapping," in *IEEE International Conference on Robotics and Automation*, 2000.

[8] S. Thrun, M. Diel, and D. Hähnel, "Scan alignment and 3-d surface modeling with a helicopter platform," in *International Conference on Field and Service Robotics*, 2003.

[9] S. Scherer, S. Singh, L. Chamberlain, and S. Saripalli, "Flying fast and low among obstacles," in *IEEE International Conference on Robotics and Automation*, 2007, pp. 2023–2029.

[10] C. Jennings and D. Murray, "Stereo vision based mapping and navigation for mobile robots," in *IEEE International Conference on Robotics and Automation*, 1997, pp. 1694–1699.

[11] K. Sabe, M. Fukuchi, J.-S. Gutmann, T. Ohashi, K. Kawamoto, and T. Yoshigahara, "Obstacle avoidance and path planning for humanoid robots using stereo vision," in *IEEE International Conference of Robotics and Automation*, 2004, pp. 592–597.

[12] K. Konolige, M. Agrawal, R. C. Bolles, C. Cowan, M. Fischler, and B. Gerkey, "Outdoor mapping and navigation using stereo vision," in *Proc. of Intl. Symp. on Experimental Robotics (ISER)*, 2006.

[13] D. Wooden, "A guide to vision-based map building," *IEEE Robotics and Automation Magazine*, vol. 13, no. 2, pp. 94–98, June 2006.

[14] S. Thrun and A. Bücken, "Learning maps for indoor mobile robot navigation," Carnegie Mellon University, Tech. Rep. CMU-CS-96-121, 1996.

[15] C. Fulgenzi, A. Spalanzani, and C. Laugier, "Dynamic obstacle avoidance in uncertain environment combining pvos and occupancy grid," in *IEEE International Conference on Robotics and Automation*, 2007, pp. 1610–1616.

[16] L. Iocchi, K. Konolige, and M. Bajracharya, "Visually realistic mapping of a planar environment with stereo," in *Seventh International Symposium on Experimental Robotics*, 2000.

[17] J. Dittrich, A. Bernatz, and F. Thielecke, "Intelligent systems research using a small autonomous rotorcraft testbed." in *2nd AIAA Unmanned Unlimited–Systems, Technologies and Operations–Aerospace*, 2003.

[18] D. Scharstein and R. Szeliski, "A taxonomy of dense two-frame stereo correspondence algorithms," Microsoft Research, Tech. Rep. MSR-TR-2001-81, November 2001.

[19] T. D. Cornall and G. K. Egan, "Measuring horizon angle from video on a small unmanned air vehicle," in *2nd International Conference on Autonomous Robots and Agents*, 2004, pp. 339–344.

[20] J. E. Bresenham, "Algorithm for computer control of a digital plotter," *IBM Systems Journal*, vol. 4, no. 1, pp. 25–30, 1965.