Abstract—Exploration is a challenging task in mobile robotics, which has been discussed to a great extent for ground robots. Its application on Micro Aerial Vehicles (MAVs), however, produces unique challenges. In this paper we present an approach for map-exploration designed for a quadrotor helicopter navigating through GPS-denied indoor environments using a single fisheye camera as primary sensor. Based on an occupancy grid map, obstacles are avoided safely, while the environment is mapped in 3 dimensions using a sparse point cloud gathered by the camera images. This algorithm has been tested successfully in textured environments. Results from simulation and real world experiments are presented at the end of this paper.

I. INTRODUCTION

The growing interests of using MAVs in highly scattered environments, such as for urban or indoor application, induces an increasing effort of the research community in exploration and mapping of an unknown environment. For both, civil and military purposes, MAVs are of high interests. The widespread applications range from surveillance tasks in unfriendly or dangerous areas to search and rescue operations, or reconnaissance of indoor areas. Those applications mainly rely on the MAV’s ability to explore the environment and send an accurate representation of it to the end-user. Therefore, exploring and mapping the environment is one of the fundamental tasks of MAVs. Meeting this challenge is therefore one very important issue in the field of research in mobile robotics.

The currently used models of MAVs mainly rely on GPS data to navigate within their workspaces. However, this practical navigation aid can not be used, when operating in near-earth or even indoor areas, where too many unforeseen obstacles may intersect the way path of the MAV and receiving GPS signal is not guaranteed. For both, exploration and navigation purposes, other ways have to be found to overcome this issue. For such applications, it is unavoidable to address the problem of obstacle avoidance in the context of exploration. The MAV has to fully rely on its own onboard sensors to safely navigate and localize itself.

This issue has been tackled using different combinations of sensors. Mainly laser sensors[1] or in combination with stereo vision[2]. However, all those approaches have the disadvantage of great weight and high power consumption, whereas for MAVs lightweight energy-saving systems are of great importance. To tackle this issues, we use one omnidirectional camera pointing downward as single exteroceptive sensor. The use of a single, and even passive sensor allows to save both, energy and weight to a great extent. The data acquired by camera images can be used for Simultaneous Localization And Mapping (SLAM), which allows to complete navigation and mapping tasks. The here presented approach uses the Autonomous Systems Lab1 (ASL) visual SLAM framework, which is based on the algorithm of Klein et al.[3] modified by Bloesch et al.[4]. It solves the SLAM problem while mapping features from the camera images into the 3 dimensional space and recompute the camera’s pose, which is rigidly attached to the helicopter frame. The points form a full map of the environment in 3 dimensions. Creating a triangulation of the received points provides an accurate three dimensional mesh of the observed environment.

Based on this mesh informations, we developed an exploration algorithm, which allows to enlarge the map and to safely avoid obstacles laying in the waypath. The navigation is based on an occupancy grid map, created by making use of the given mesh informations. The grid map spans the same area as the mesh, while a cell is called occupied, if the mesh’s height within this cell exceeds a given threshold. The helicopter travels through the free cells in a spiraling mode. When approaching the map’s border, new points will be observed, what leads to an expansion of the mesh map and consequently of the grid map. As soon as the spiraling mode can not be proceeded due to an obstacle, a free cell is searched and approached using a wavefront algorithm. If the search fails, no free cell is available and the entire reachable environment is observed.

The paper is organized as follows: in Section II, we review the work in this area; in Section III, we present the background of this work and in Section IV we describe our approach. Finally, in Sections V and V-A we present simulation and experimental results and draw the conclusions in Section VII.

II. RELATED WORK

A. Vision Based MAV Navigation

The use of onboard vision as input for MAV navigation tasks is a widespread application, especially for obstacle detection and avoidance. Optical flow based strategies have been proved working for lateral obstacle avoidance by Serres et al.[5] and Hrabar[6], who used two cameras pointing ±90° to the side. Zingg et al.[7] used a single downward looking omnidirectional camera for optical flow based wall detection.

1www.asl.ethz.ch
Muratet et al. [8] implemented a frontal obstacle detection using optical flow information from a camera pointing in direction of travel. Detecting a divergence of the optical flow field from one point it is possible to compute the time to impact. If this falls below a threshold, the MAV executes a 180° turn to avoid collision. However, all these approaches are just navigation aids, not able to maneuver the MAV relying only on vision inputs. Zufferey et al. [9] presented an approach on a 10g microflyer combining frontal and lateral obstacle detection and using vision as primary input. An additional, downward pointing sensor allows ground clearance.

This applications basically provide the robot with information about the distances to potential obstacles. This, however, are not enough informations for stabilizing a helicopter, where the full knowledge about the current six degree of freedom position and rotation is needed. Using sensor fusion of vision and IMU informations, Cheviron et al. successfully extracted the MAV’s pose [10]. Pure vision inputs have been used by Tournier et al. [11], who extracted all six degrees of freedom of the camera’s pose observing a known moire pattern. Bloesch et al. [4] showed successfully stabilization and navigation of a quadrotor helicopter in unstructured environment by solving the SLAM problem based on the algorithm of Klein et al. [3]. It allows to simultaneously determine the helicopter’s pose and create a sparse point map of the local surroundings. This approach serves as basis of the here presented exploration algorithm.

**B. Autonomous Exploration**

The issue of exploring unknown environments using autonomous vehicles has been tackled in various ways [12]. Yamauchi presented the frontier based method [13], where one distinguishes between observed and unobserved environment. The robot moves toward the map’s frontier, where the boundary between known and unknown area is constantly pushed back while the map is successively enlarged. A possible map representation is presented by Stachniss et al. [14], who used a coverage grid-map. For each cell, a posterior about the amount of the covered area is stored. Using a decision-theoretic approach, the map is built and the uncertainty about the cell coverage is minimized. A simpler representation of the environment would be a occupancy grid map, where each cell is called either occupied or free, this approach is widely used [15], [16].

These algorithms have been used mostly on ground robots. The use of MAVs for exploration tasks, however has become more and more popular, recently. Bachrach et al. [17] presented a fully autonomous exploration for a quadrotor helicopter equipped with a laser-sensor and stereo vision cameras using a frontier-based algorithm. He showed successful tests of exploring indoor environments at the MIT. Exploration of outdoor environments have been carried out by Shim et al. [12], who used laser-sensor equipped helicopter moving in an urban-like environment. It autonomously builds a local obstacle map and navigates using model predictive control algorithm to the target position.

The above presented approaches all rely on multiple or heavy sensors, which is unfavorable for MAVs that have a very limited payload. Therefore, our approach uses one single lightweight camera as single sensor to execute exploration of an unknown indoor environment.

**III. BACKGROUND**

**A. ASL Visual SLAM Framework**

The ASL Visual SLAM framework is based on the research of Bloesch et al. [4] which is the basis of creating the three dimensional mesh map and navigating the MAV. The fundamental is the algorithm presented by Klein and Murray [3], allowing to build a local three dimensional point map of the environment using features extracted from images taken by a single camera. Simultaneously it estimates all six degrees of freedom of the camera’s pose up to an arbitrary scaling parameter depending on the map initialization. It runs using two threads, one for mapping and one for tracking. Therefore, the map has not to be updated at every image acquired but only at certain keyframes, while the pose estimation is done at every image, which saves a lot of computational power. If the camera is moved to a new location and a new keyframe is built, the points are stored in the current map. The map is aligned according to the first position of the camera.

Bloesch used this algorithm to estimate the pose of a quadrotor helicopter, equipped with a down looking camera. An additional LQR controller allows to navigate the MAV in a textured environment with user inputs. To overcome the problem with the unknown scaling parameter, the map is initialized at a given height over a flat surface with the axis of the camera pointing in vertical direction. At a framerate of approximately 16 frames per second, the MAV can be stabilized safely. Computation is done off-board on an up-to-date computer.

The framework additionally provides the user with a triangulation of the most reliable points, which allows to compute a mesh map of the environment [18]. Texture from the images may be put on the map to give it a nice three dimensional look of the environment (see Fig. 1).
Fig. 2. The µEye camera from IDS with a 150° lens recording grayscale images. The right image depicts a snapshot of the experimental environment taken by the µEye camera, where the opening angle ranges from 90° to 150°.

B. Equipment

The flying platform is the Hummingbird quadrotor helicopter from Ascending Technologies\(^2\), having an overall diameter of 53 cm with a payload of 200g. The operational flying time varies between 23 minutes without payload and 12 minutes with full payload. Additionally, the MAV is equipped with a fully working Inertial and Measurement Unit (IMU) providing information about the pitch, roll, and yaw angle of the helicopter. Its built in highlevel controller allows us to regulate the overall thrust, the angular velocity of the yaw angle and the positions of the pitch and roll angles. For communication purposes, a ZigB communication board is used to send control inputs to the helicopter and to grab IMU data. Gray scaled images are provided by a µEye camera from IDS\(^3\), having a resolution of 752 x 480 pixels, while the maximal possible frame rate lays at 87 frames per second. Its lens has an opening angle ranging from 90° to 150° (see Fig. 2).

Computation is done on an up-to-date laptop running with 2GHz processor and 2GB RAM.

IV. Exploration Algorithm

Based on the visual SLAM framework, we developed an exploration algorithm that allows to enlarge the mesh map which can later be used for the end-user. However, for the robot itself, a mesh map contains too much information, therefore as internal representation for the environment, a grid map with square cells is chosen. During the procedure, all free cells have to be visited. To visit a cell, the MAV has to pass through the cell’s center.

A. Notation

$$\begin{pmatrix} x \\ y \\ z \\ x_c \\ y_c \\
\end{pmatrix}$$ real world coordinates

$$\begin{pmatrix} x_c \\ y_c \\
\end{pmatrix}$$ integer cell coordinates

$$c_{x_c,y_c}$$ cell at coordinates \((x_c,y_c)^T\)

$$c_{x_n,y_n}$$ cell visited in step \(n\)

$$h_{x_n,y_n}$$ coordinates of cell visited in step \(n\)

$$A_{x_c,y_c}$$ area spanned by the cell at coordinates \((x_c,y_c)^T\)

\(o_{x_c,y_c}\) is 1 if the cell is occupied, otherwise 0

\(w_{c}\) width of the cell

B. Planar Exploration

For planar exploration (i.e. exploration of a planar surface without obstacles), a mode is chosen in which the MAV moves outward from a given initial position in a spirally manner where each cell is visited, keeping a constant altitude. This procedure allows high loop closure, a very important property keeping the map’s quality high when working with a visual SLAM algorithm.

Each time the SLAM algorithm provides new features, they are used to expand the current grid map. Based on the triangulation the grid map is built (see Fig. 3). The expansion is therefore only done in regions with features, this ensures to merely explore environments with enough features to reliably compute the pose of the camera. The altitude at which the exploration is executed is the same altitude the MAV has when the algorithm is initialized. The grid map is aligned with the initial position, i.e. the initial position defines the center of the first cell \(c_{0,0}^1\) at position \((0,0)\).

Whenever a cell \(c_{x_n,y_n}^n\) is visited, the next cell \(c_{x_{n+1},y_{n+1}}^{n+1}\) to visit is determined. The following constraints allow to remain in the spiral and therefore ensure highest possible loop closure while continuously exploring new areas. Assume a coordinate system with its main axis \(x\) and \(y\) aligned with the grid. From each cell only those neighboring cells in \(\pm (0\ 1)^T\) and \(\pm (1\ 0)^T\) direction may be reached in the next step. It is therefore not possible to move diagonal to the grid alignment, ensuring that no other cells are touched than the current and the next to be visited. Further, only previously not visited cells are taken into consideration, such that primarily new areas are
explored. From the remaining candidates the one is chosen with the highest number of visited neighbors. For counting the neighboring visited cells, all eight surrounding cells are taken into account, contrary to the path planning, where only the four cells in x and y directions are considered as neighbors. If there are multiple candidate cells with equal number of visited neighbors, the one that remains most likely within the spiral is chosen. To achieve this requirement check first the possibility of moving in the direction of the current rotation of the spiral.

\[
\begin{pmatrix}
  x_{n+1} \\
  y_{n+1}
\end{pmatrix} = \begin{pmatrix}
  x_n \\
  y_n
\end{pmatrix} + A \cdot \left( \begin{pmatrix}
  x_{n-1} \\
  y_{n-1}
\end{pmatrix} - \begin{pmatrix}
  x_n \\
  y_n
\end{pmatrix} \right)
\]

(1)

Where \( A = \begin{pmatrix}
  0 & -1 \\
  1 & 0
\end{pmatrix} \) if the spiral rotation is clockwise, or \( A = \begin{pmatrix}
  0 & 1 \\
  -1 & 0
\end{pmatrix} \) if the spiral rotation is counterclockwise. If this is no candidate, as second option check to move in the same direction as before.

\[
\begin{pmatrix}
  x_{n+1} \\
  y_{n+1}
\end{pmatrix} = \begin{pmatrix}
  x_n \\
  y_n
\end{pmatrix} + \left( \begin{pmatrix}
  x_{n-1} \\
  y_{n-1}
\end{pmatrix} - \begin{pmatrix}
  x_n \\
  y_n
\end{pmatrix} \right)
\]

(2)

If this candidate is not free as well, there is only one option remaining, moving contrarily to the spiral rotation.

In Figure 4, the current cell at step \( n \) is at position \( \begin{pmatrix}
  x_n \\
  y_n
\end{pmatrix}^T = \begin{pmatrix}
  2 \\
  1
\end{pmatrix}^T \). Candidates for the next step are

\[
\begin{pmatrix}
  x_{n+1} \\
  y_{n+1}
\end{pmatrix} = \begin{pmatrix}
  2 \\
  2
\end{pmatrix} \quad \text{and} \quad \begin{pmatrix}
  x_{n+1} \\
  y_{n+1}
\end{pmatrix} = \begin{pmatrix}
  3 \\
  1
\end{pmatrix}
\]

Both candidates have the equal number of visited neighbors. Using equation 1, the most desired following cell is:

\[
\begin{pmatrix}
  x_{n+1} \\
  y_{n+1}
\end{pmatrix} = \begin{pmatrix}
  2 \\
  1
\end{pmatrix} + \begin{pmatrix}
  0 & -1 \\
  1 & 0
\end{pmatrix} \cdot \begin{pmatrix}
  0 \\
  1
\end{pmatrix} = \begin{pmatrix}
  1 \\
  1
\end{pmatrix}
\]

Since this is not a candidate, using equation 2, the next desired cell is:

\[
\begin{pmatrix}
  x_{n+1} \\
  y_{n+1}
\end{pmatrix} = \begin{pmatrix}
  2 \\
  1
\end{pmatrix}
\times \begin{pmatrix}
  1 \\
  1
\end{pmatrix} = \begin{pmatrix}
  2 \\
  2
\end{pmatrix}
\]

This is a candidate cell, therefore it is chosen as next cell to be visited. When continuing path planning in this form, the trajectory forms a rectangular spiral.

For expanding the grid map, we chose a discretization of the mesh information. The visual SLAM framework returns for each mesh element three points \( p_i = (x_i, y_i, z_i)^T \) for \( i = \{1, 2, 3\} \) forming a triangle. Discretization is done using \( p_1 \) as starting point and moving toward \( p_2 \) in a given step size. At each step, the current position \( A \), which lays between \( p_1 \) and \( p_2 \), is used as starting point to move, again in discrete steps, parallel to the direction from \( p_2 \) to \( p_3 \), reaching position \( B \). Applying the intercept theorem, this is expanded until

\[
\frac{|p_3 - p_2|}{|p_2 - p_1|} \cdot (|A - p_1|) = (|B - A|)
\]

At each position in the discretization process, the related grid-cell is computed and is added if it is not part of the current grid-map. In order to be sure not to skip cells, the discretization step size is chosen equal to the width of the cells \( w_x \). This procedure allows to expand the grid map according to the received mesh information (see Figure 5).

\[\text{Fig. 5.}\]

C. Obstacle Detection

The above mentioned procedure is working only for planar exploration, however moving in a three dimensional environment meets the challenge of detecting and avoiding obstacles that possibly cross the desired waypath. To tackle this additional difficulty, each cell is given an associated height value \( h_{x,y,t} \). This height is extracted from the mesh information provided by the SLAM framework. It is chosen according to
Using the above derived occupancy information in the grid map, it is possible to adapt the planar exploration mode into a three dimensional environment containing obstacles. To

\[
h_{x_c,y_c} = \max_{(x,y) \in A_{x_c,y_c}} \{ z_{\text{mesh}}(x,y) \}
\]

Where \( z_{\text{mesh}}(x,y) \) depicts the height of the mesh at position \((x,y)^T\).

In practice, it is chosen according to the discretized mesh information used to build the grid map. The smaller the discretization steps are chosen, the more accurate are the height informations.

This feature allows us to gain a rough estimation of the surrounding’s topography, what may be used as condition to decide if a potential obstacle lays within the area of a cell. Obstacle detection is done using a simple threshold criterion. If the height \( h_{x_c,y_c} \) exceeds a threshold \( t \), the cell is occupied and therefore not visitable in the future.

\[
o_{x_c,y_c} = \begin{cases} 
1 & \text{if } h_{x_c,y_c} \geq t \\
0 & \text{if } h_{x_c,y_c} < t
\end{cases}
\]

Where \( o_{x_c,y_c} \) denotes the occupancy of the cell at \((x_c,y_c)^T\)

\(t\) while \(t\) is the chosen threshold value.

Since the primary application is in indoor or urban environments, for the most part obstacles have a vertical structure, as it can be observed at walls, pillars and the like. Therefore this simple assumption holds in general. Choosing the threshold \(t\) depends on multiple parameters such as the altitude at which exploration is done, the cell size or the opening angle of the camera.

D. Three Dimensional Exploration

Using the above derived occupancy information in the grid map, it is possible to adapt the planar exploration mode into a three dimensional environment containing obstacles. To
Fig. 10. As soon as there is no unvisited free cell in the grid map, the exploration is completed.

Fig. 11. The flow chart of the exploration algorithm. After the start of the algorithm, the grid map is updated according to the current mesh map received from the SLAM algorithm. New cells may be created or cells may be called occupied. If the MAV has reached the desired position, the grid map is updated according to this position, where each cell’s number of visited neighbors is counted and the visited cells are updated. Is the algorithm not in search mode, the next cell is searched using the wavefront algorithm. If no next cell is found, the map is fully explored. In the other case, the wavefront algorithm is again applied to create the search map, containing the number of steps to reach the desired cell. The next cell is found in this search map. The function find next cell is decomposed in the following way: In the first place, the neighboring cells are scanned for potential candidates as next cells. If there are candidates, the next cell is determined. Otherwise, if there are no free unvisited cells, the search mode is started. First, the desired cell is searched using the wavefront algorithm. If no desired cell is found, the map is fully explored. In the other case, the wavefront algorithm is again applied to create the search map, containing the number of steps to reach the desired cell. The next cell is found in this search map.

reach this, the obstacle information is fully integrated in the exploration mode.

If an occupied cell appears as a possible succeeding cell of the current cell, it will not appear in the list of possible candidates. Therefore the conditions on which cells are possibly succeeding is adapted and additional to the current restrictions of being unvisited and laying in $x$ or $y$ direction, the property of being not occupied is necessary. This might lead to the situation, where the next cell is chosen to lay in the opposing direction of the spiral’s rotation. In such a case, the rotation of the spiral is changed, such that the waypath follows the previous trajectory.

Exemplary shown in Figure 7, the current cell at step $n$ is at position $(x_n, y_n)^T = (-2, 2)^T$. There is only one single candidate for the next step:

$$
(\begin{array}{c}
 x_{n+1} \\
 y_{n+1}
\end{array}) = (\begin{array}{c}
 -2 \\
 3
\end{array})
$$

It has the highest number of visited neighbors, observing all possible not previously visited and unoccupied cells. Therefore it is chosen as the next cell, although it lays contrary to the rotation of the spiral. In this case, the rotation of the spiral is changed from counterclockwise to clockwise. This change in the spiral direction allows to keep loop closure high due to remaining close to the previous trajectory. In combination with the condition of always proceeding to the cell with the most visited neighbors, it prevents from adopting a wall-following behavior, which would be essentially bad for loop closure.

This procedure does not hold for a situation in which there is no potential candidate cell for a next step according to the above presented conditions (see Figure 8). For such a situation, a search mode is used, where a wavefront algorithm searches the nearest free cell that has previously not been visited and guides the MAV on the shortest path to this particular cell.
Spreading out from the current cell, the wavefront algorithm is looking out for an unvisited free cell. Its expansion is again only parallel to the $x$ and $y$ direction. In detail, the algorithm starts at the initial position (i.e. the current cell) explores the neighboring cells in $x$ and $y$ direction. When discovering an unvisited free cell, it is stored as the next to be visited. If none is found, the expansion is continued at the frontiers of the by the wavefront explored environment until a unvisited cell appears. The found cell is then used as target cell for a path planning algorithm, again based on the wavefront method. This time, the algorithm starts at the target cell, looking for the current cell. Additionally, for each explored cell, the number of steps to the target cell is stored, such that as soon as the current cell is found, the next cell to travel to is the one with the lowest number of steps to the target cell. This forms the shortest path from the current cell to the desired cell. Once the target cell is reached, the algorithm continues in the spiral mode as presented above.

Using this exploration algorithm, it is possible to reach every cell in the created grid map. As soon as there is no unvisited free cell remaining, the whole area is explored and a complete three dimensional mesh map of the environment is created.

To increase safeness another feature is added. It possibly appears, that the detection of an obstacle is done too late, a truly occupied cell is falsely called free and this particular cell is the next cell to travel to. In such a case, the algorithm immediately returns to the previous cell, as soon as the obstacle
is detected in this cell. This sorely reduces the possibility of crashing. Flow charts of the complete exploration algorithm are shown in Figure 11 and Figure 12.

V. SIMULATION

A. Setup

The above presented algorithm has been simulated in a virtual environment created in MATLAB\(^4\) by drawing up different indoor environments using a sparse point cloud, similar to the points extracted by the visual SLAM framework. The here presented results are based on an environment containing of two 5m x 5m rooms linked by a 2m wide corridor. The walls have a height of 4m.

The camera is simulated with having a field of view formed as a cone with an opening angle of 90° which is equal to the smallest opening angle of the camera used in the real world experiments.

The environment is created in a challenging way for the algorithm. The walls are not aligned with the grid structure but tilted 45° toward the x respectively y direction. Therefore, obstacle detection has to be maintained up to a distance of \(\sqrt{2}\cdot 1.5 \cdot w_c\).

The cell width is chosen to be \(w_c = 0.7\text{m}\), which corresponds approximately to the size of the Hummingbird helicopter. It is designed such that the MAV can safely remain at the center of the cell without touching other cells. This property is of high importance, since surrounding cells might be occupied. Based on this parameter, the hight, at which obstacles are detected is adapted. As previously mentioned the maximum distance at which obstacles have to be detected is \(\sqrt{2}\cdot 1.5 \cdot w_c\). Therefore, feature points have to be detected at this distance above the threshold set for obstacle detection. With an opening angle of the camera of 90°, the maximal height of the observed points at the maximum distance is

\[
\text{alt} - \tan(90^\circ/2) \cdot \sqrt{2} \cdot 1.5 \cdot w_c = \text{alt} - 1.48\text{m}
\]

where \(\text{alt}\) denotes the altitude at which the exploration algorithm is executed. Assuming an altitude of \(\text{alt} = 4\text{m}\), the height at which feature points are detected is 3.52 meter. The threshold for obstacle detection is therefore chosen at 3.5 meter.

B. Results

In this simulation, the exploration is launched, after the MAV has reached an altitude of 4m above ground at position \(x = 0\) and \(y = 0\), where it continues to autonomously travel through the virtual environment (see Figure 13). The simulation show an accurate and reliable obstacle detection to safely maneuver the MAV away from walls. It allows to explore the whole environment and moves through a narrower passage linking both rooms. Principally working, exploration is not finished in the first room before moving through the corridor to the second room. This happens, since there are several unknown areas in the map, while the algorithm always explores the nearest unknown area first by steering to the nearest unvisited cell.

\(^{4}\text{www.mathworks.com}\)

VI. EXPERIMENTS

A. Setup

To perform the experiments, an environment has been built containing features on the walls and on the ground (see Figure 14). Its movable walls allow to change the shape of the environment and therefore test different settings. The parameters for cell size and obstacle detection are chosen similar to those presented in the simulation. However, since the altitude is chosen at 2m, the threshold used to detect surrounding obstacles is lowered to 0.5m.

Due to a current adaption of the SLAM framework to a different operating system and a software update on the helicopter, the experiments could not be done on the real MAV, whereas a handheld camera has been used.

B. Results

Experiments have been carried out in two different environments, a room with an approximate size of 4m x 4m (see Figure 15) and a corridor of 2.5m width (see Figure 16). In each setup, the grid and the walls were chosen both, aligned and in arbitrary orientation. The algorithm was able to cope with the given situations, where it proofed to be capable of early obstacle detection in most cases. Whereas target cells were discovered to be occupied, immediate return to the previous cell was ordered as soon as the cells status was corrected.

After all, this algorithm proofed working under the given conditions.

VII. CONCLUSIONS

In this study we addressed the problem of exploration of a GPS-denied environment using a MAV quadrotor helicopter based on a visual SLAM framework, which provides a three dimensional mesh map using visual feature points from the local surroundings.

Indoor environments equipped with numerous visual features were regarded as primary working range. Relying on
an occupancy grid map, where each free grid cell has to be visited during the exploration process this algorithm is able to explore even complex shaped environments. This grid map is expanded wherever the mesh map of the SLAM framework is created. Precisely this mesh map is the fundamental for obstacle detection, where each grid cell is occupied in which the mesh exceeds a threshold in height. The grid map is run through primarily in spiral behavior, which ensures high loop closure.

The applications of this algorithm are restricted to environments containing enough visual features to ensure reliable pose estimation by the visual SLAM framework. Additionally, obstacles that are lower than the helicopter altitude but still higher than the threshold for obstacle avoidance are threatened as laying within the waypath although they are not in reality. Furthermore, obstacles without texture are not even detected. Openings smaller than twice the diagonal of a cell might not be detected as free. Therefore, the application is restricted to rooms and corridors, without passing through small openings like doors.

Nevertheless, simulation and experimental results showed that this algorithm is capable of exploring indoor environments equipped with sufficient visual features on the ground and on the walls, where obstacles have a vertical structure.

ACKNOWLEDGMENT

The authors would like to thank Markus Achtelik and Laurent Kneip for their great support and contribution.

REFERENCES

Exploration of a 2.5m wide corridor using a handheld camera. On the left hand side, the mesh map of the surrounding points is shown, while the right hand side visualizes the grid map and the waypath of the camera.

The upper image depicts a situation in which the grid is aligned with the wall structure, while in the lower image, wall and grid orientation have been chosen arbitrarily. These more challenging prerequisites are managed in a proper way, too. Finally, both situations have been successfully explored.