We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

2

Combining Occupancy Grids with a Polygonal Obstacle World Model for Autonomous Flights

Franz Andert and Lukas Goormann
Institute of Flight Systems, Unmanned Aircraft, German Aerospace Center (DLR)
Germany

1. Introduction

1.1 Overview

This chapter presents a mapping process that can be applied to autonomous systems for obstacle avoidance and trajectory planning. It is an improvement over commonly applied obstacle mapping techniques, such as occupancy grids. Problems encountered in large outdoor scenarios are tackled and a compressed map that can be sent on low-bandwidth networks is produced. The approach is real-time capable and works in full 3-D environments. The efficiency of the proposed approach is demonstrated under real operational conditions on an unmanned aerial vehicle using stereo vision for distance measurement.

1.2 The Problem of Mapping and Obstacle Representation

To be autonomous, vehicles must know the environment in which they are to move. Like humans or animals, they have to sense, understand and remember at least those parts of the environment that are in the vicinity. Only then can the vehicles operate in their environment – without manual remote control. Successful results of autonomous flights in urban areas with unmanned aircraft have been presented in the past (Hrabar et al., 2005; Zufferey & Floreano, 2005; Griffiths et al., 2007; Scherer et al., 2007). Beside that, the majority of obstacle detection, mapping, and avoidance research are carried out with ground vehicles and many approaches used in flight applications are based on practices derived from that wealth of knowledge.

A main requirement for autonomous vehicles is to detect obstacles and to generate environmental maps from sensor data and a large number of approaches have been developed over the years (Thrun, 2002). One approach to represent the environment is the use of grid-based maps (Moravec & Elfes, 1985; Konolige, 1997). They allow an easy fusion of data from different sensors; including noise reduction and simultaneous pose estimation, but they have large memory requirements. Further, they do not separate single objects. A second approach, called feature-based maps, focuses on individual objects. An early work (Chatila & Laumond, 1985) uses lines to represent the world in 2-D. Later approaches use planar (e.g. Hähnel et al., 2003) or rectangular (Martin & Thrun, 2002) surfaces for 3-D modeling – but mostly to rebuild the world with details and possible texture mapping. A suitable model for autonomous behavior is the velocity obstacle paradigm (Fiorini & Shiller, 1998) that can be added with the introduced specifications on how to measure the obstacles.

These map types, and others not discussed here, have their respective advantages and disadvantages. As a result, there is a need to have different map types for different robot tasks (Kuipers, 2000). In many cases, it is advantageous to use grid-based maps for sensor fusion and feature-based polygonal metric maps for local planning, e.g. in order to avoid obstacles (Fulgenzi et al., 2007). Additionally, non-metric topological maps are most suitable for global search tasks like route planning. In a complex scenario, a robot must deal with all of these different maps and keep them updated. These tasks need the usual information exchange.
To generate maps from sensor data that are applicable to autonomous applications, it is a straightforward procedure to generate a grid map from sensor data and extract out the features. Outdoor scenarios, however, can be too large to store the whole scene in a data array with reasonably accurate resolution. Additionally, the area boundaries may be unknown before mapping.
The approach presented here combines grid maps and polygonal obstacle representations and tackles the problem of large environments by using small grid maps that cover only essential parts of the environment for sensor fusion. Characteristic features are recognized, their shapes are calculated, and inserted to a global map that takes less memory and is easily expandable. This map is not restricted to the sensor environment and is used for path planning and other applications.

2. Flight Testbed

2.1 ARTIS – A Flying Robot

The presented mapping and world modeling approach is developed within the ARTIS (Autonomous Rotorcraft Testbed for Intelligent Systems) research project that deals with mid-sized unmanned helicopters (Dittrich et al., 2003). One of the helicopters is shown in figure 1. It has a main rotor diameter of 3 meters and a total weight of up to 25 kg. Flights of more than 30 minutes are possible.

Figure 1. The unmanned helicopter ARTIS

The 5 kW turbine engine has enough power to carry more than six kilograms of experimental payload in addition to the avionics system and power supply. The actual configuration is a dedicated image processing computer and a stereo camera system weighing 2 kg so that additional sensors like multiple cameras or laser scanners can be used in future applications.

2.2 Sense and Avoid Setup

For applications that need environmental sensing capabilities, a vision system separated from the flight controller is installed at the helicopter. The actual configuration uses only

cameras because they are lightweight, passive, and have low power consumption. A dedicated computer is used to process image information. This results in improved speed and no influence on the real-time behavior of the flight control computer. For interaction between image-based results and flight control, data exchange is provided via a local network. A mission planning and automatic control system is installed on the flight control computer (Dittrich et al., 2008). It calculates trajectories around obstacles, considers changes due to actual image-based map updates, and instructs the helicopter to fly these paths. The autopilot ensures stabilized hover so that the vehicle can completely fly autonomously.

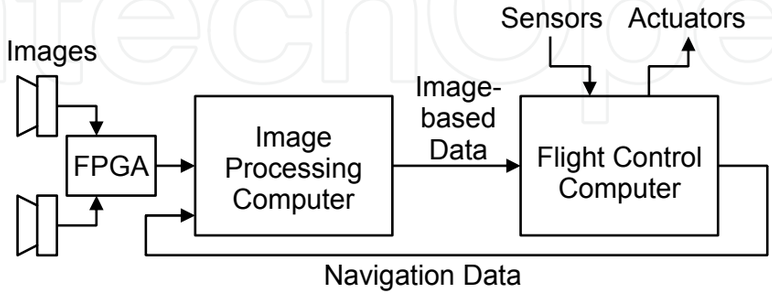


Figure 2. Overview of the onboard hardware for vision applications

Figure 2 illustrates the connection between vision hardware and flight controller. Since obstacle mapping and other image-based algorithms require flight information, a navigation solution provides the global position and attitude of the helicopter.
A stereo camera (Videre Design STOC, fig. 3) with a baseline of 30 cm and a field of view of approximately 51° x 40° is used. It creates images with 640 x 480 pixels and has an inbuilt FPGA processor that calculates a depth image out of the two input images in real-time with 30 Hz. This image is a result of a complex processing step where regions of the two camera images are matched (e.g. Scharstein & Szeliski, 2002), and it acts as a depth sensor in the mapping process. Basic pre-processing to enhance the depth image quality is already done by the camera. In the depth images shown in the figures of this article, near distances are represented by light colors and farther distances by darker colors. White space indicates that depth values are missing or have been filtered due to bad image quality, e.g. low texturing.

Figure 3. Stereo camera mounted at the helicopter (left), left onboard camera image (center), depth image (right)

The helicopter's position and attitude is provided in six degrees of freedom by the flight control computer using a differential GPS sensor, a magnetometer and an inertial measurement unit. The raw data of these sensors are integrated by an Extended Kalman filter (Koch et al., 2006) to provide an accurate solution in all six degrees of freedom. Filtered navigation data is sent with a rate of 100 Hz to the vision computer. All computer clocks are

synchronized so that a fitting recording pose of an image with a given timestamp can be obtained.

3. Model Overview

3.1 Grid Maps

The basic method to interpret data from depth image sequences follows classical approaches with occupancy grids (Moravec & Elfes, 1985; Raschke & Borenstein, 1990). These grids have turned out to be very useful for obstacle mapping since they allow easy sensor fusion, reduce sensor noise, and are also applicable to multiple vehicles. Here, a world-centric 3-D grid represents the map. Each cell consists of a value describing the presence of obstacles: the log odd of the occupancy probability. Higher values refer to a higher probability of the cell being occupied. The map is created incrementally by starting with an empty grid and writing the actual sensor information with each new depth image.

3.2 Feature Maps

As already denoted, occupancy grid maps are advantageous for sensor fusion and will be used to interpret data from depth image sequences. In addition to that, feature maps are built out of these occupancy grids to store global obstacle information in a compressed way and to be an input for applications that use the map.

For this reason, it is a primary requirement to determine the required level of detailing to represent objects in a feature map. For obstacle avoidance applications, the important criteria are:

1. Small details are not needed,
2. an identification of planes for texture projection is not needed, and
3. real-time capabilities are more important than centimeter accuracy.

These criteria imply that it is sufficient to mark a box around an object and simply avoid this area.

The simplest way of modeling potentially danger areas is the usage of cuboidal bounding boxes that are aligned with the coordinate axes. Unfortunately, they are too rough for object modeling, e.g. a set of objects like houses with a rectangular base shape and an arbitrary angle to the coordinate axes. Aligned bounding boxes will be larger than the houses and narrow paths between them will not be found because they exist inside the area marked by the occupied box.

In a real scenario like a city, objects can have any ground shape from the top view, but it is likely that they have vertical walls. This assumption is made in a lot of mapping approaches (e.g. Iocchi et al., 2000). For this reason, prism shapes are used here. They do not require an axis alignment like the bounding boxes. Additionally, they can deal with any ground shape without dramatically increasing the complexity. If walls are not vertical like roofs, the prism's volume will be much larger than the real object. To improve the modeling of objects, they can be subdivided vertically and represented by multiple prisms. In other words, the world is split into layers in different heights, and each layer has its own polygonal 2-D obstacle map. This is sufficient for a lot of applications like flight trajectory planning in urban scenarios (Dittrich et al., 2007).

Nevertheless, the prism model can only handle complex shapes in 2-D as seen from the top view. A gabled roof or a tunnel with an up- or downhill road must be modeled by multiple

obstacle prisms. These scenarios are more complex than a simplified urban canyon with a horizontal ground plane and obstacles with vertical walls. Avoiding roofs that are represented by bounding prisms will be only a small restriction and a tunnel, however, will be an unusual case, at least with respect for flying robots.

3.3 The Modeling Process

As already mentioned, advantages of both grid and feature maps are combined, see figure 4. The mapping process works as follows:

1. Create an occupancy grid around the vehicle's position if not existing in the map.
2. If a new grid is allocated or the grid is extended, check whether previously stored features can be inserted.
3. Insert the actual sensor data information to the grid.
4. Find clusters of occupied grid cells and mark them as single obstacle features.
5. Find out which obstacle features are new and which are updates of objects that have already been identified in the previous loop cycle. Preexisting objects may also be removed.
6. Calculate the shape of each new or updated feature.
7. To insert the next sensor data, go back to step 1.

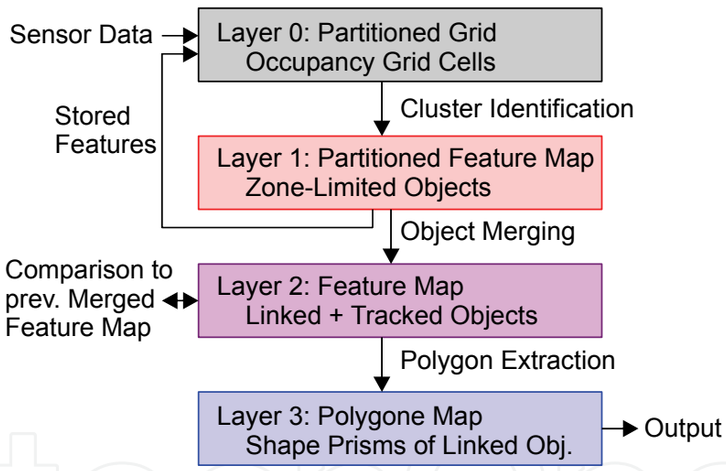


Figure 4. The general mapping process with different map layers

This approach uses different map types as illustrated. The occupancy grid map for sensor inputs (layer 0) is partitioned into zones and each zone is searched for obstacles separately (layer 1). This method is described in the following section. Next, a map with separate obstacles, but without zone separation is generated (layer 2), and finally, the prism shapes are extracted out of them (layer 3). Grid resolution and zone sizes are user-defined but may not change over time when processing image series.

4. Occupancy Grid Mapping

4.1 Sensor Interpretation and Local Mapping

To create an occupancy grid incrementally out of depth image data sequences, a local grid map is built. This local map is world-centric like the global output maps but includes only

the occupancy data of a single depth image taken from its corresponding camera position and attitude.
The local map is the interpretation of a single depth image as follows: Using projective geometry and the pinhole camera model, each pixel refers to an object coordinate in the world. By assuming no transparent objects in front of the object, there is free space along a ray between the camera center and the object coordinate. Behind the opaque obstacle, no information is available. If the distance encoded by a pixel exceeds a threshold, this image point leads to free space. A line is drawn for each pixel, illustrated by figure 5 that shows the interpretation of an ideal sensor (left) or considers measurement noise, respectively (right). The uncertainty of stereo-based depth measurement increases quadratically with higher distance values. With this information, the viewable area is spanned and the obstacles are drawn.

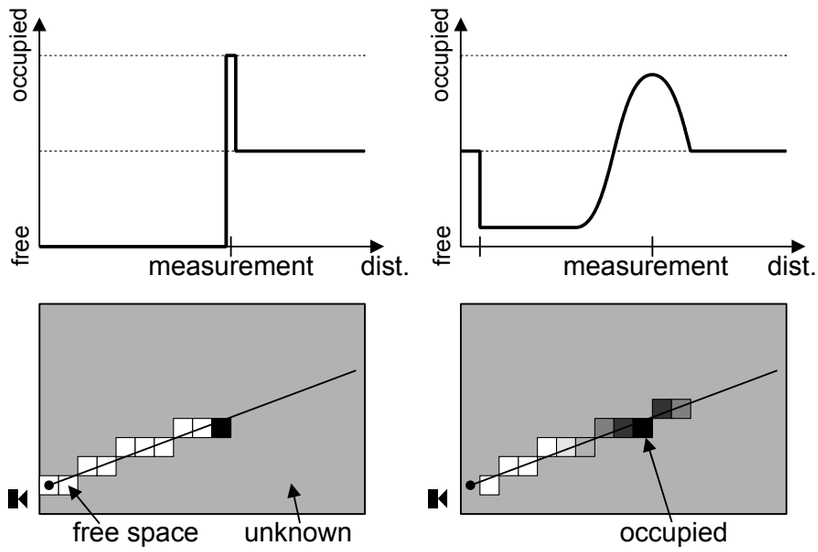


Figure 5. Interpretation of an ideal depth image pixel (left) and a more realistic model that considers depth uncertainty (right)

The result of processing all rays of a single image is illustrated in figure 6. Obstacles and free areas of the image are also visible in the corresponding map. The thickness of an obstacle cannot be derived from a single image so it depends on the depth uncertainty first. Information about the true size and the area behind obstacles are included when the vehicle moves there.
The size of the local grid is determined by considering that the data fusion of the empty local map with one image will only affect grid cells inside the sensor range, i.e. a small environment of the actual position. There is no need to update cell values outside this area. Hence, it is satisfactory to have only an occupancy grid representation of the map inside the sensor range. Further, the local map definition is independent to the outer camera orientation angles to get a fast implementation. With that, the camera is set to the center of the local map, with maximal half a grid cell deviation to keep the grid cells aligned to a global rasterization. To give an example, a sensor range of 30m implies zones with a size of each 60 x 60 x 60m to cover all attitude angles without truncating far distances. In applications where looking and moving straight upwards or downwards is restricted,

smaller zones with a height of e.g. 15m are satisfactory. If the grid resolution is set to 0.5m, grid arrays with 120 x 120 x 30 cells are created to represent a local map.

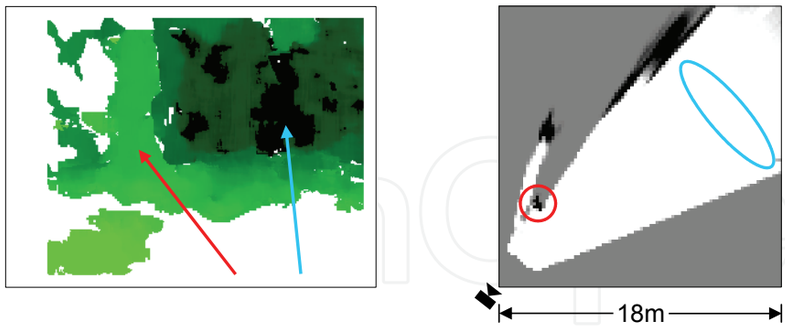


Figure 6. Depth image example (a) and a 2-D view of the local map that was built out of it (b)

4.2 Creating Global Maps

The data fusion between succeeding image frames is done by integrating the local grid with a global map by adding the values of cells at the same global position. Similar to other approaches, free or occupied cells become more significant if measured several times. Noise is filtered when occupancy information from one image is disproved by free space information from another image. With that, especially static objects can be easily determined. 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.

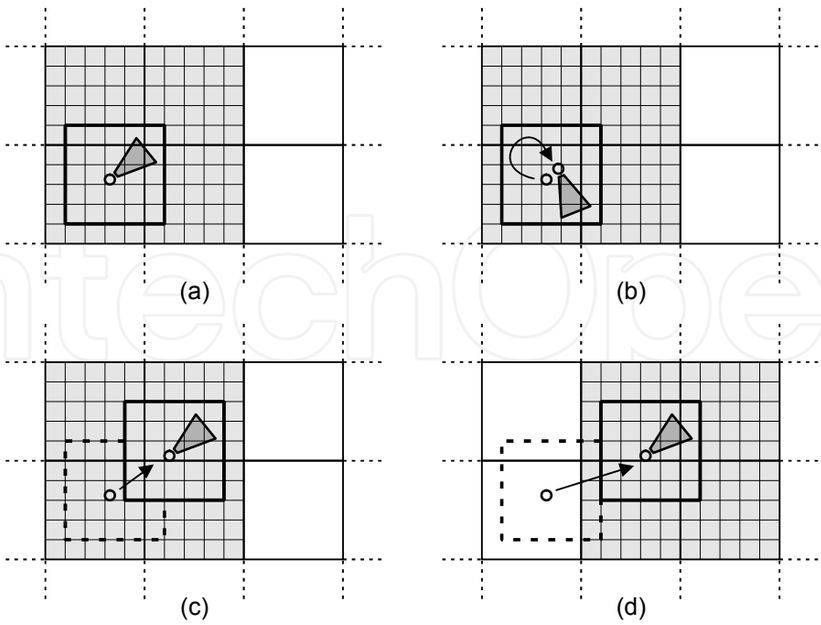


Figure 7. 2-D view of the global map that is divided into zones. For zones around the actual camera position, an occupancy grid representation exists

Since grids are usually stored as continuous data blocks in the computer memory, the boundaries of the map must be known a priori. If the vehicle moves outside, extensive reallocation or shifting methods must be applied because the map boundaries change.
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 local map so each zone overlaps a maximum of eight zones in the global 3-D map. The environmental cuboid moves through the global map with the movement of the camera, i.e. with the helicopter. Only those global zones overlapping with the local grid are represented by an occupancy grid array.
Figure 7 illustrates how the zone partitioning works. The actual camera and local map position is shown in figure 7a, the other graphics show possible effects on the map, caused by the next measurement. Often, rotations (7b) or movements (7c) will not have an effect on the zone boundaries. But if the movement is larger so that boundaries are crossed (7d), new memory is allocated for these zones. Grid information is discarded for zones that fall outside the immediate vicinity.

5. Extracting Objects from the Grid Map

5.1 Determining Separate Objects and Bounding Boxes

Object features are detected by segmenting the global map into occupied and free areas applying a threshold. A single object is a cluster of connected occupied cells of the 3-D array. These objects are recognized with a flood fill algorithm. By saving the minimal and maximal values of the coordinates of cells belonging to the object, the bounding box is calculated and put into the global feature map as it is illustrated in figure 8. Unlike the cell array, these boxes need much less memory and it is easy to make them available to further applications, independent of the existence of a grid. For each object, the binary cell shape is stored in addition to the bounding box so that this shape can be re-inserted. Compression with an oct-tree structure is applicable here. The polygonal shape is calculated later.

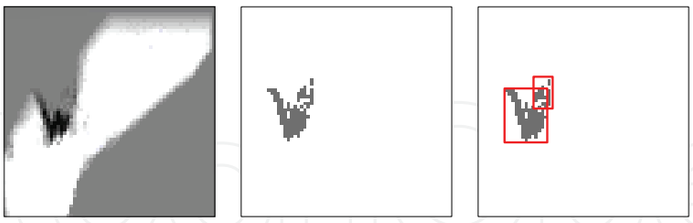


Figure 8. Extracting features from an occupancy grid (a) with thresholding (b) and bounding box calculation (c). Bounding boxes can overlap

5.2 Integration Between Features and Grid

Unlike the occupancy grid, the feature map is not limited to an environment around the actual vehicle position. Objects are stored independently from the presence of a grid. If some grid data is removed, the corresponding features will remain (fig. 9). Vice versa, objects can be inserted into the grid (fig. 10). It is possible to include a-priori knowledge about obstacles here.

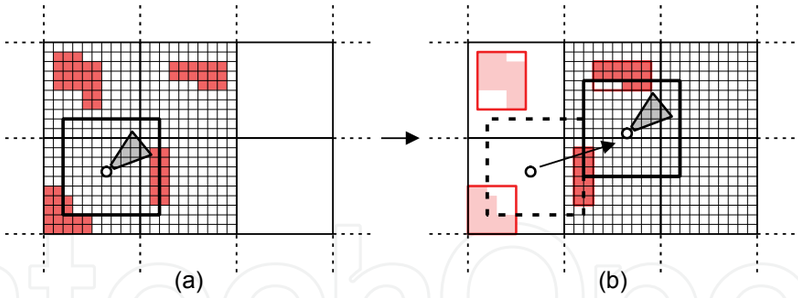


Figure 9. Vehicle movement from (a) to (b). Features are stored when the grid data is discarded due to helicopter movement

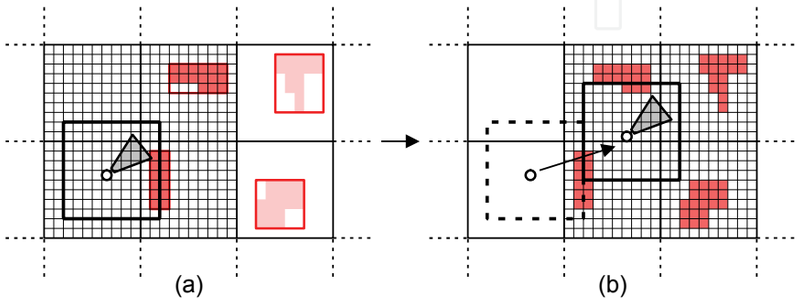


Figure 10. Vehicle movement from (a) to (b). Inserting features to the grid if there are objects inside a new zone

5.3 Merging and Tracking Obstacles

Single objects found in the temporary grid zones are limited to the zone boundaries since each zone is processed separately. To build an output map for the application that has no zone partitioning, the features of different zones are merged if a connection exists as illustrated in figure 11. First, it is checked whether the bounding box of an object is located at a zone boundary. If another object box is located at the same boundary from a different zone, the cell shapes of both objects are tested for connection. Connected objects are linked together, and the linkage is not limited to only two objects.

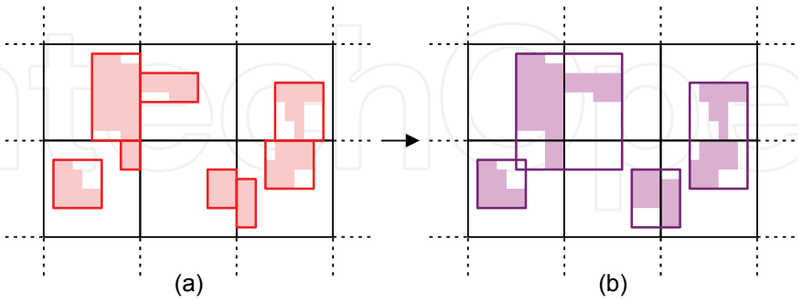


Figure 11. Merging tangent objects in the partitioned map (a) to linked objects (b) that are not partitioned into zones. Simplified 2-D view

After an update step where sensor data is inserted, the zones are checked for obstacles again without reference to the previous state. Since it is useful to know which specific object has been updated with the actual sensor information, the linked objects are tracked. This is done as follows:

1. Calculate the center of mass of each linked object.
2. For each linked object, try to find a matching object in the map with linked objects of the previous state. This is the object with the nearest center of mass determined by the Euclidean distance (Prassler et al., 2000). To avoid the matching of objects that are too far away, distances above a threshold will not result in a match.
3. If no match exists for an object of the actual state, give it a new unique ID. Otherwise, copy the ID from the matched object.

With object tracking, each linked obstacle in the world gets a unique ID that remains constant over time, if the tracking is successful. The merging and tracking step can be skipped for linked objects whose coordinates are completely outside the zones where a grid exists since there will be no update.

Due to sensor updates it is possible that a linked object becomes split. Usually, one of the new objects gets the old ID, the others receive the new IDs. If multiple linked objects are merged together over time, one ID is kept and the others removed.

5.4 Horizontal Shape Slicing

The approach presented in this paper models the shape of each linked object with a prism (see section 5.5) with horizontal bases. Since a bounding prism of complex shapes may be too rough, the cell-based shape $s(x, y, z)$ is partitioned into horizontal slices with the height of one cell. Similar slices are merged and a polygonal shape prism is calculated for each multi-slice.

A single slice in height z is denoted as $s_z(x, y)$ with $s_z(x, y) = s(x, y, z)$. Without loss of generality, z is valid from 1 to n. Similar consecutive slices are put together to multi-slices S using the following algorithm:

1. Set $i = 1$, set $z = 1$.
2. Create a multi-slice S_i: $S_i = \{s_z\}$.
3. If $z = n$, break.
4. If the slices s_z and s_{z+1} are similar given by the equation

$$\sum_x \sum_y | s_z(x,y) - s_{z+1}(x,y) | < t \tag{1}$$

with a similarity threshold t,
set $S_i = S_i \cup \{s_{z+1}\}$,
set $z = z + 1$ and go to step 3.
Otherwise,
set $i = i + 1$;
set $z = z + 1$ and go to step 2.

The result is a set of multi-slices $\{S_1, \ldots, S_m\}$. The parameter t of equation 1 controls the degree of similarity consecutive single slices must have in order to be unified. The extreme case $t = 0$ puts every single slice into a separate multi-slice and $t = \infty$ merges all single slices together. As illustrated in figure 12, grid-based object shapes are splitted into parts that have approximatively vertical walls. They can be approximated by their 2-D shape from top view and easily modeled by prisms without forfeiting large irregularities in the shape.

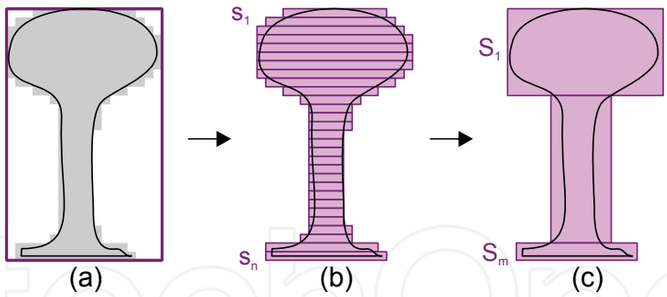


Figure 12. Grid shape and bounding box of linked feature (a), single slices (b) and merged multi-slices (c). Side view of a tree as an example

In the next step, the 2-D shape is calculated for each multi-slice S_i ($1 \le i \le m$). Occupied cells of a multi-slice are calculated through the logical disjunction of all single slices s_z inside S_i. It is

$$S_i(x,y) = \begin{cases} 0, \text{ if } \sum_{s_z \in S_i} s_z(x,y) = 0; \\ 1, \text{ otherwise.} \end{cases} \quad (2)$$

5.5 Extraction and Approximation of the Polygonal Shape

Now, algorithms developed for binary images can be applied to a multi-slice S_i since its shape is represented by a binary 2-D array. Figure 13 illustrates the process. The polygonal extraction is done in two steps. First, the contour is calculated with a tracing algorithm (Ren et al., 2002). The main idea is to start at one edge pixel and search incrementally for the next edge pixel until the whole contour is covered. Its output is a list of pixels sorted counterclockwise for outer contours and clockwise for inner contours. Multiple lists are possible. Each contour pixel can be interpreted as a polygon vertex by using the corresponding grid cell center coordinate.

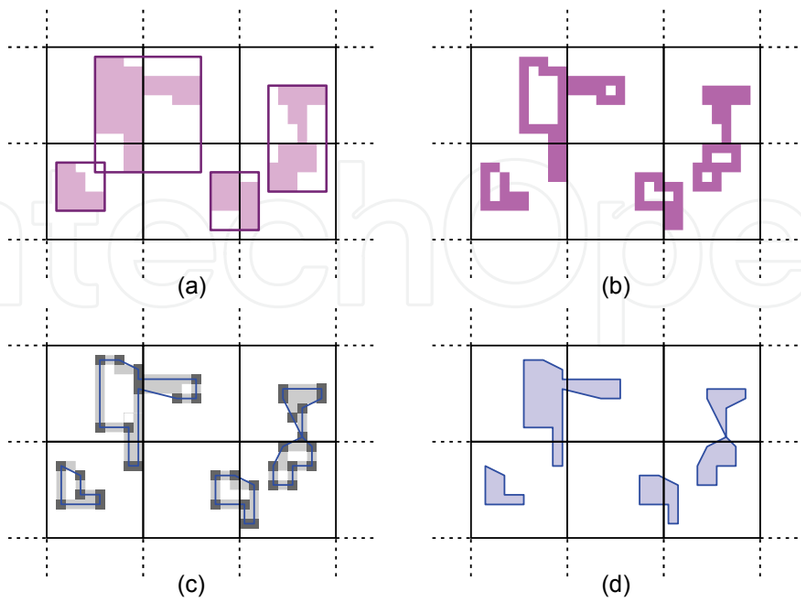


Figure 13. Shape Extraction, top view. Cell-based shapes (a), contours (b), approximated polygons with its vertices (c) and final ground shapes (d)

The second step is for data compression and to accelerate later applications. A polygon with a lower number of vertices is calculated here with the approximation algorithm presented by Ramer (1972). It is parameterized by a maximal distance value that specifies the accuracy of the algorithm. Every point of the original shape has this maximal distance to the lines defining the approximated shape.
This 2-D polygon acts as the ground shape of the right prism that is calculated for each S_i. The prism's height and vertical position is determined by the span of single slices $s_z \in S_i$. As seen in figure 13, the prism shapes can be smaller than the grid-based shapes since the center coordinates of the grid cells are used for the shape vertices and an approximation is performed. In obstacle avoidance applications, a safety distance to the objects is added; it must be larger than half a cell size plus the approximation accuracy to ensure that the grid cells of each object are completely covered by the polygon hull.

6. Estimating Ground Planes

6.1 The Height Histogram

In addition to the prism-based shape detection of obstacles, the floor plane is extracted out of the sensor data. This simplifies the resulting world model. The floor will not be a shape prism that has to be tracked. Furthermore, obstacles and the floor are not merged which helps to identify objects that hit the ground.
The floor plane detection is similar to classical Hough Transform based approaches (e.g. Okada et al., 2001) with the limitation that only the horizontal planes are searched.
The actual sensor data leads to a cloud of points (x, y, z) in global coordinates. The vehicle's position must be known and a calibrated pitch and roll angle measurement is assumed to ensure that a horizontal ground will lead to a horizontal plane in map coordinates. A sampled histogram $n(z)$ of all z-values of these points is built. The sampling should be more precise than the occupancy grid. Points are inserted with blurring considering their depth-dependent uncertainty.
Since all measurements of a horizontal plane have approximately the same z-value, planes lead to significant peaks in the histogram and can be detected there. If more than one plane is found, the floor is the one with the lowest height, i.e. with the largest z-value. There will be no ground plane if no peak with a high confidence is found.
It can be useful to increase the floor plane height, e.g. to force a minimal distance to the ground in urban flight scenarios. First, the peak maximum $n_{\max} = n(z_{\max})$ is not taken. Rather, z is decreased while $n(z) > t \cdot n_{\max}$, starting at $z = z_{\max}$. The result is denoted as z_{ground}. In the experiments, t is set to 0.5. Second, an offset $z_{\text{offset}} < 0$ can be added to this height.

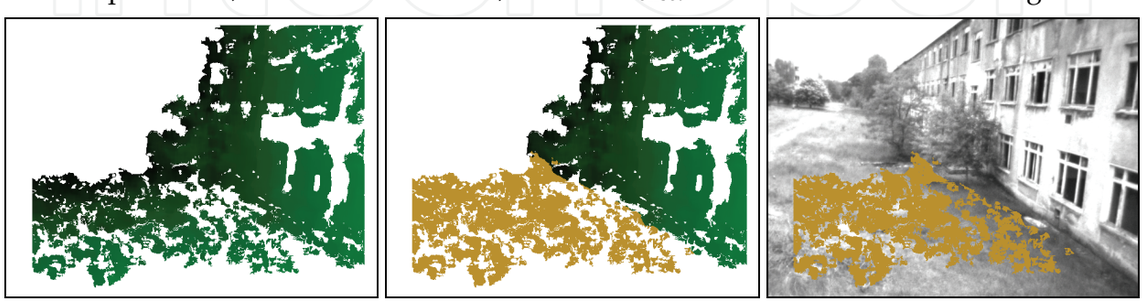


Figure 14. Depth image (left), depth image with marked floor pixels (center) and original camera image with floor pixels for comparison (right)

Figure 14 shows an example of the ground estimation. Pixels that lead to a larger z-value are marked in the depth image and for comparison in the original camera image. As seen in the right subfigure, the floor is marked contrary to the house at the right side. It is not necessary to insert the marked measurements into the obstacle map.

6.2 A Floor for each Zone

To combine the data of multiple images, the ground plane estimation is done separately for map zones. The map partitioning as described in section 4.2 is used but without the vertical separation. This leads to rectangular 2-D zones from the top view and each zone can have a floor plane height. A height histogram is calculated for each zone by accumulating these sensor data elements that lead to object points in that zone. The histogram is built incrementally over time so that multiple sensor updates can be stored in one zone if the vehicle stays there. The result is a kind of an elevation map with an estimated height value for each zone.

7. Tests and Results

A first experiment tests the mapping algorithm in a simulation environment where two views are generated and captured with cameras (fig. 15). The grid resolution is 0.25m and each map zone has a size (x, y, height) of 128 x 128 x 32 cells. Figures 15 and 16 show the result of the simulation experiment where two obstacle arches of 6 x 6m size are placed with a distance of 50m.

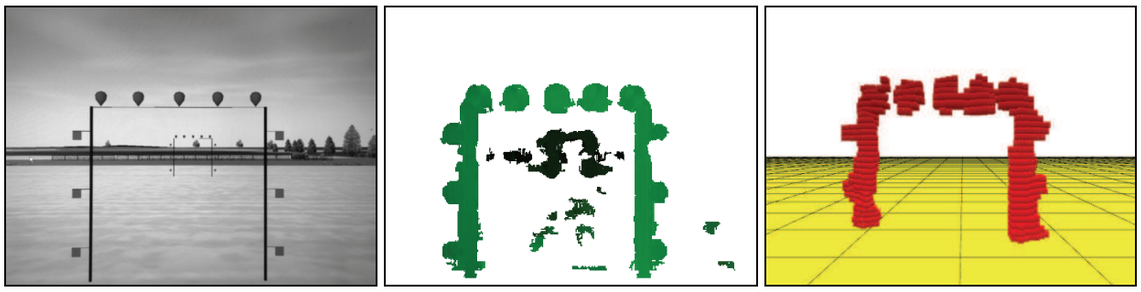


Figure 15. Original image from left camera (left), depth image (center) and extracted occupancy grid (right)

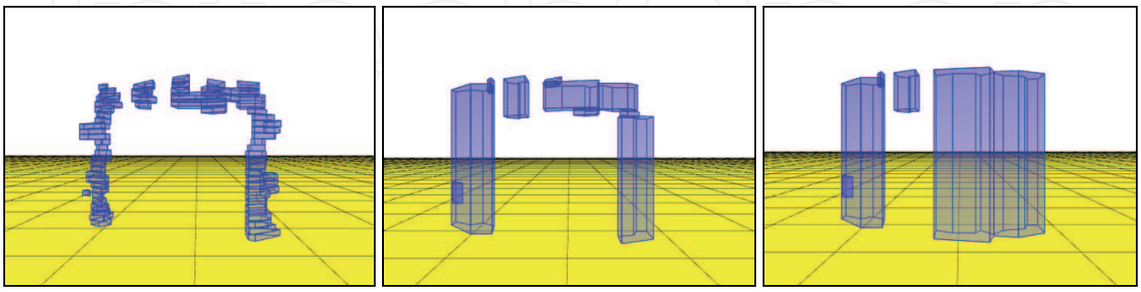


Figure 16. Prism shapes calculated from occupancy grid of figure 15 with different similarity thresholds for shape slice merging: zero (left), 100 (center) and infinite (right)

As shown in the image, the first arch is mapped while the second arch in the background is filtered due to its far distance from the viewpoint. The subfigures include the occupancy

grid map (fig. 15, right) and the resulting shape prisms (fig. 16). As shown in the figures, the final map representation is only a small set of simple shapes. The similarity threshold for horizontal shapes should not be too large to allow multiple prisms for features generated from grid cell clusters in order to represent non-vertical walls. For a better illustration, floor planes marked with lines every 8 meters are shown in the images. The computation speed on the 3 GHz test computer was generally in the interval from 15 to 20 frames per second.
In a second test series, helicopter flights are performed outdoors. The grid array size is the same as in the simulation and its resolution is 0.5m, so that the map size of each zone is doubled. The helicopter is manually directed through obstacle posts and in an urban environment near house walls.

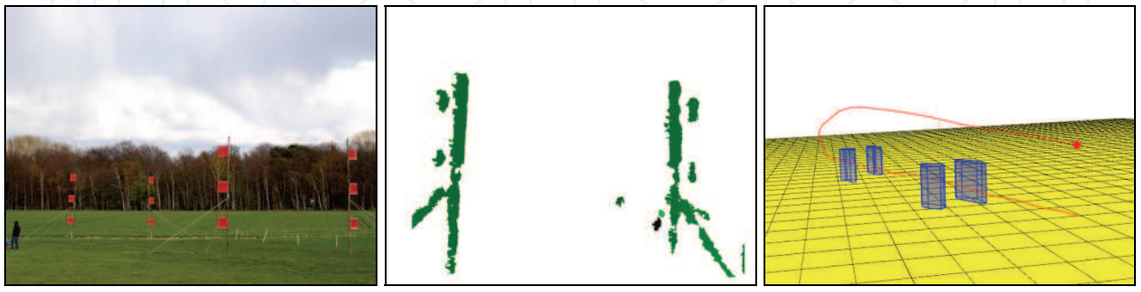


Figure 17. Obstacle posts on a flight field (left), example of a depth image (center), and the resulting map with obstacles and flight path (right)

Figure 18. An urban environment (left), example of a depth image (center), and the resulting map (right) with obstacles, detected ground planes, and flight path

As seen in figure 17, obstacle posts are detected and each post can be represented by just one polygon. The ground plane was not detected here; the image shows only a plane in the height the helicopter took off. The prisms are larger than the real obstacles because they include the wires that hold the posts. The flight trajectory is marked red. Figure 18 shows the result of mapping houses and as seen in the resulting map, the walls and the gap between the houses have been recognized correctly. Ground plane detection was enabled and successful.

8. Conclusions

This work describes a method to extract obstacles from sensor data to be used for autonomous applications. The focus is on the creation of a compact representation of the bounding shapes required for obstacle avoidance, as opposed to detail object representation. Sensor data fusion is performed with an occupancy grid map, and this map is the basis for the shape extraction. A shape defined through one or multiple right prisms is calculated for

each cluster of occupied grid cells. The shapes are calculated with each new sensor data in real-time. In addition to that, ground planes are identified. The output of this algorithm is very compact map representations of obstacles, and it is possible to send actual map updates through low-bandwidth networks which can serve as an input to other external applications.

To prove the approach, tests were performed in a simulation environment in a laboratory and under real conditions where a helicopter flies through obstacle posts and in an urban environment. The results show that it is possible to map obstacles with GPS/INS-based positioning and stereo vision, and that feature extraction functions such that the resulting map is suitable for obstacle avoidance.

9. References

Chatila, R. & Laumond, J.-P. (1985). Position referencing and consistent world modeling for mobile robots. In *IEEE International Conference on Robotics and Automation*, pp. 138-145.

Dittrich, J.; Bernatz, A. & Thielecke, F. (2003). Intelligent systems research using a small autonomous rotorcraft testbed. In *2nd AIAA Unmanned Unlimited Conference, Workshop and Exhibit*, paper 6561, San Diego.

Dittrich, J.; Adolf, F.; Langer, A. & Thielecke, F. (2007). Mission planning for small VTOL UAV systems in unknown environments. In *AHS International Specialists' Meeting on Unmanned Rotorcraft*, Chandler.

Dittrich, J.; Andert, F. & Adolf, F. (2008). An obstacle avoidance concept for small unmanned rotorcraft in urban environments using stereo vision. In *64th Annual Forum of the American Helicopter Society*, Montréal.

Fiorini, P & Shiller, Z. (1998). Robot motion planning in dynamic environments using velocity obstacles. *International Journal of Robotics Research*, Vol. 17, No. 7, pp. 760-772. ISSN 0278-3649.

Fulgenzi, C.; Spalanzani, A.; Laugier, C. (2007). Dynamic obstacle avoidance in uncertain environment combining PVOs and occupancy grid. In *IEEE International Conference on Robotics and Automation*, pp. 1610-1616, Roma.

Griffiths, J.; Saunders, A.; Barber, B.; McLain, T.; Beard, R. (2007). Obstacle and terrain avoidance for miniature aerial vehicles. Valavanis, K. P. (ed.), *Advances in Unmanned Aerial Vehicles*, pp. 213-244. ISBN 978-1-4020-6113-4.

Hähnel, D.; Burgard, W. & Thrun, S. (2003). Learning compact 3D models of indoor and outdoor environments with a mobile robot. *Robotics and Autonomous Systems*, Vol. 44, No. 1, pp. 15-27. ISSN 0921-8890.

Hrabar, S.; Corke, P.; Sukhatme, G.; Usher, K. & Roberts, J. (2005). Combined optic flow and stereo-based navigation of urban canyons for a UAV. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 302-309, Edmonton.

Iocchi, L.; Konolige, K. & Bajracharya, M. (2000). Visually realistic mapping of a planar environment with stereo. In *Seventh International Symposium on Experimental Robotics*, Waikiki.

Koch, A.; Wittich, H. & Thielecke, F. (2006). A vision-based navigation algorithm for a VTOL UAV. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, paper 6546, Keystone.

Konolige, K. (1997). Improved occupancy grids for map building. *Autonomous Robots*, Vol. 4, pp. 351-367. ISSN 0929-5593.

Kuipers, B. J. (2000). The spatial semantic hierarchy. *Artificial Intelligence*, Vol. 119, pp. 191-233. ISSN 0004-3702.

Martin, C. & Thrun, S. (2002). Real-time acquisition of compact volumetric 3D maps with mobile robots. In *IEEE International Conference on Robotics and Automation*, pp. 311-316, Washington D.C.

Moravec, H. P. & Elfes, A. (1985). High resolution maps from wide angle sonar. In *IEEE International Conference on Robotics and Automation*, pp. 116-121.

Okada, K.; Kagami, S.; Inaba, M. & Inoue, H. (2001). Plane segment finder: Algorithm, implementation and applications. In *IEEE International Conference on Robotics and Automation*, pp. 2120-2125, Seoul.

Prassler, E.; Scholz, J. & Elfes, A. (2000). Tracking multiple moving objects for real-time robot navigation. *Autonomous Robots*, Vol. 8, No. 2, pp. 105-116. ISSN 0929-5593.

Ramer, U. (1972). An iterative procedure for the polygonal approximation of plane curves. *Computer Graphics and Image Processing*, Vol. 1, No. 3, pp. 244-256. ISSN 0146-664X.

Raschke, U. & Borenstein, J. (1990). A comparison of grid-type map-building techniques by index of performance. In *IEEE International Conference on Robotics and Automation*, pp. 1828-1832, Cincinnati.

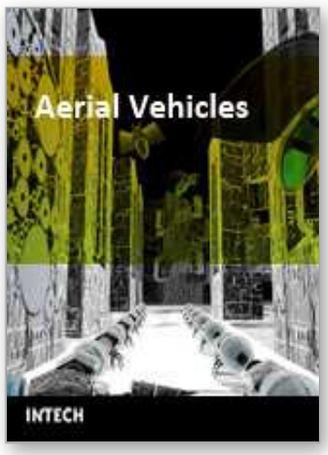
Ren, M.; Yang, J. & Sun, H. (2002). Tracing boundary contours in a binary image. *Image and Vision Computing*, Vol. 20, pp. 125-131. ISSN 0262-8856.

Scharstein, D. & Szeliski, R. (2002). A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, Vol. 47, pp. 7-42. ISSN 0920-5691.

Scherer, S.; Singh, S.; Chamberlain, L. & Saripalli, S. (2007). Flying fast and low among obstacles. In *IEEE International Conference on Robotics and Automation*, pp. 2023-2029, Roma.

Thrun, S. (2002). *Robotic mapping: A survey*. Tech. Rep., Carnegie Mellon University. No. CMU-CS-02-111.

Zufferey, J.-C. & Floreano, D. (2005). Toward 30-gram autonomous indoor aircraft: Vision-based obstacle avoidance and altitude control. In *IEEE International Conference on Robotics and Automation*, pp. 2594-2599, Barcelona.



Aerial Vehicles

Edited by Thanh Mung Lam

ISBN 978-953-7619-41-1
Hard cover, 320 pages
Publisher InTech
Published online 01, January, 2009
Published in print edition January, 2009

This book contains 35 chapters written by experts in developing techniques for making aerial vehicles more intelligent, more reliable, more flexible in use, and safer in operation.It will also serve as an inspiration for further improvement of the design and application of aeral vehicles. The advanced techniques and research described here may also be applicable to other high-tech areas such as robotics, avionics, vetronics, and space.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen