The 2009 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 11-15, 2009 St. Louis, USA

Drawing Stereo Disparity Images into Occupancy Grids: Measurement Model and Fast Implementation

Franz Andert
German Aerospace Center (DLR)
Institute of Flight Systems, Unmanned Aircraft
38108 Braunschweig, Germany
franz.andert@dlr.de

***Abstract*— Mapping the environment is necessary for navigation in unknown areas with autonomous vehicles. In this context, a method to process depth images for occupancy grid mapping is developed. Input data are images with pixel-based distance information and the corresponding camera poses. A measurement model, focusing on stereo-based depth images and their characteristics, is presented. Since an enormous amount of range data must be processed, improvements like image pyramids are used so that the image analysis is possible in real-time. Output is a grid-based image interpretation for sensor fusion, i.e. a world-centric occupancy probability array containing information stored in a single image. Different approaches to draw pixel information into a grid map are presented and discussed in terms of accuracy and performance. As a final result, 3D occupancy grids from aerial image sequences are presented.**

I. Introduction

For mapping, exploration, obstacle avoidance, and other tasks that require information of previously unknown areas, range sensors are used to recognize the environment. Since it is quite common to use occupancy grids to integrate multiple sensors and data sequences from different locations to only one obstacle map, an approach to create grid-based obstacle data is developed here. Inside robotic applications with integrated sensor fusion, mapping, localization and planning steps as mentioned in [1], the proper creation of occupancy grids from sensor data consists of two main steps:

- interpretation of sensor data to occupancy values, and
- integration of multiple readings over time into one map.

The fusion of sensor data into occupancy maps requires a representation of which map parts are occupied and which are free. In general, map cells become occupied if they are inside the field of view and if their coordinates can be derived from distance measurements. If the global sensor position and line of sight is known, world-centric grid maps can be created and multiple measurements can be combined by adding the cell values. With that, occupancy information becomes more and more significant if measured multiple times. While simple mapping approaches use grid arrays where only occupancy information is counted for each interpreted measurement [2], advanced methods include free space information and allow noise reduction and a separation between unknown and free map regions. An early work is presented in [3] where a binary 3D grid is created from range images. A quite successful approach is based on distance measurements from sonar data [4], and it turned out as very useful to store detailed occupancy probability information for each grid cell to allow fuzzy representations of obstacles, and to consider specific sensor properties, for instance, range precision. Grid maps are widely used for sensor fusion and in most cases, they are world-centric samples of the environment with cubic cells of the same size. As an addition, multi-scale maps are used to handle especially far distances with fast algorithms [5].

Conclusions from sensor data to a map are drawn with forward models (i.e. which range measurement is expected for given obstacles) or with inverse models (i.e. which grid cells are probably occupied for a given range measurement) [6]. For most sensors, forward models are given by the sensor accuracy and can be easily extended with noise and maximum-range measurements for very far distances. Sensor fusion turned out to be much easier with inverse models, and they are used in most mapping approaches. Inverse models can be derived from the forward model by using a learning procedure with simulated random maps, sensor positions and measurements [7].

A lot of approaches to interpret sensor data and to build grid maps are presented in the literature: with robots using sonar sensors, [4], laser scanning systems [8], [9] and passive systems that use stereo vision [10] or a combination of stereo vision and optical flow estimation [11]. For stereo-based approaches, sensor measurements are easily to be interpreted as object points with depth-dependent Gaussian error as described by an early approach in [12]. By considering the way of light between an object and its projection, free space between a visible obstacle and both cameras can be included [10]. This is advantageous for large baselines and high resolution maps, but it hardly improves stereo image interpretation where the cameras are close to one another and often located at the same grid cell. In this case, the interpretation of measurements as objects and free space between the object and only one camera will provide suitable results. In the more general case, images from three or more viewpoints are used to estimate distances and to build occupancy grids [13].

II. Paper Overview

This paper deals only with the sensor interpretation step mentioned in the beginning and acts as a basis for oc-

cupancy grid mapping algorithms explained in the related literature. Further use has been published separately [14]. The presented approach is optimized for a stereo camera as a special application for distance sensing, but, nevertheless, the approach can easily be adapted to other sensors if their special features are taken into account. Methods to interpret range data are developed in the following steps:

- Characteristics of stereo vision are explained to be used in the following sections. With that, depth images can be regarded as 2D range sensor output.
- A model is introduced that concludes from sensor data to occupied and free occupancy grid cells.
- Different methods of generating the grid-based occupancy values using the measurement model are presented. The resulting algorithms return a world-centric occupancy grid given depth images and corresponding camera positions and attitudes.
- The usage of image pyramids data is tested. It is a common technique to cope with a huge amount of image data.
- Finally, the different approaches are compared.

All methods of sensor interpretation are applied to full 3D environments and they are developed to be real-time capable. Occupancy grids are always 3D arrays and camera poses are given in six degrees of freedom.

The interpretation of real images to a geodetic map is presented at the end of this paper. Application context is the aerial robotics domain.

III. Stereo Geometry and Disparity Images

A fast calculation of depth images uses the pinhole camera model and requires a standard-stereoscopic view. This is achieved through a calibration of estimated internal and relative camera parameters and an image undistortion and rectification step. Theoretical fundamentals are widely explained in [15]. Depth images are two-dimensional disparity functions $d(x, y)$ that describe the correspondence between two images. Different methods to estimate the disparity function are explained e.g. in [16], they are mainly based on finding corresponding regions in both images that look similar and include improvements like filtering low-confidence measurements or speck. Distance measurements z_c are reconstructed from the disparity values through

$$z_c = \frac{b \cdot f}{d}, \tag{1}$$

with the baseline, i.e. the horizontal camera distance b. The horizontal and vertical parts of the 3D coordinate $\mathbf{p}_c = (x_c, y_c, z_c)^T$ are

$$x_c = z_c \cdot \frac{(x - x_0)}{f}, \quad \text{and} \quad y_c = z_c \cdot \frac{(y - y_0)}{f}. \tag{2}$$

In practice, d is limited to a maximal search range and a minimal distance value $z_{c\min} > 0$ can be stated for distance measurements [17]. Zero disparities lead to infinite distances and will be interpreted as free space information up to a sensor range limit. Using a known global camera orientation $(\mathbf{R}, \mathbf{t})$, geodetic coordinates of an object point $\mathbf{p}_g$ are

$$\mathbf{p}_g = \mathbf{R}^T \cdot \mathbf{p}_c + \mathbf{t}. \tag{3}$$

Since the given image coordinates are not exact, but have an uncertainty Δd, the range measurements z_c given by the disparity function have an error Δz_c called range resolution, given by

$$\Delta z_c = \frac{z_c^2}{b\, f} \cdot \Delta d. \tag{4}$$

The literature measures or presents a variety of values for pixel correlation uncertainties Δd, e.g. $\frac{1}{16}$ pixels [17] or 0.2 pixels [18], and the measurement model can be parameterized by this value. In the later sections, a very conservative value of $\Delta d = 0.5$ pixels is chosen as proposed by [16]. The uncertainty of a measured obstacle coordinate can be determined using $z_c \pm \Delta z_c$ and the results $x_c \pm \Delta x_c$ and $y_c \pm \Delta y_c$ from equation 2 with input values $x \pm 0.5$ and $y \pm 0.5$, respectively, and it is

$$\Delta x_c = \Delta y_c = z_c \cdot \frac{1}{2f}. \tag{5}$$

Objects that are far away are measured with lower accuracy than nearer objects.

IV. Inverse Measurement Models

An inverse measurement model describes the transformation from sensor data to a map. It is the interpretation of distance data and concludes which map parts are probably occupied or free if distances have been measured from a specific point of view. In the approach presented here, an inverse model is postulated without a learning procedure by taking the sensor accuracy and the assumption that obstacles are not transparent.

Basically, a single distance measurement allows the interpretation that an obstacle is supposed to be at a specific world coordinate described in the previous section. Additionally, there is usually free space in front of the measured obstacle and no information about the area behind. In the case of depth image processing, each valid depth image pixel leads to a unique ray in space. Beginning at the camera center, there is free, occupied, and unknown space.

In the measurement model described here, every depth image pixel acts as a single distance measurement along a ray in space that is defined by the pixel coordinate and the camera pose. The ray begins at the camera center and intersects with the object point. In camera coordinates, it is $\mathbf{p}_c = (x_c, y_c, z_c)^T$ and the measured distance $l_{\mathbf{p}}$ defined by the pixel is

$$l_{\mathbf{p}} = \sqrt{x_c^2 + y_c^2 + z_c^2} \tag{6}$$

with the depth-dependent uncertainty

$$\Delta l_{\mathbf{p}} = \Delta z_c \cdot \frac{l_{\mathbf{p}}}{z_c}. \tag{7}$$

It is known that $l_{\min} = z_{c\min} > 0$. To interpret the measurement, the occupancy probability for a map cell $\mathbf{s}$ at a world-centric coordinate must be defined for all points on the

ray that depends on the camera pose and the pixel coordinate. The function denoted as $P(\mathbf{s}(l))$ iterates over the distance l along the ray and defines occupancy values for cells at these coordinates. Further, log odds $L(\mathbf{s}(l))$ are calculated since they are commonly stored in grid cell arrays to allow fast additive Bayesian map updates (e.g. described in [19]). Positive log odd values refer to high occupancy probabilities and negative values to free space. Zero values are an indicator for unknown information. It is

$$L(\mathbf{s}(l)) = \ln\left(\frac{P(\mathbf{s}(l))}{1 - P(\mathbf{s}(l))}\right) \tag{8}$$

where the probabilities are stated through

$$P(\mathbf{s}(l)) = P_{\text{occ}}(l) + \left(\frac{k}{\Delta l_{\mathbf{p}}\sqrt{2\pi}} + 0.5 - P_{\text{occ}}(l)\right) e^{-\frac{1}{2}\left(\frac{l - l_{\mathbf{p}}}{\Delta l_{\mathbf{p}}}\right)^2} \tag{9}$$

with

$$P_{\text{occ}}(l) = \begin{cases} p_{\min}, & \text{if } 0 < l \leq l_{\mathbf{p}}; \\ 0.5, & \text{if } l > l_{\mathbf{p}}. \end{cases} \tag{10}$$

Unlike the modeling of 3D object point uncertainties proposed by [12], this model applies a 1D Gaussian error to the measured distance along the measurement ray and considers the other dimensions of uncertainty during the postprocessing step as described in section VII. The error increases with the measured distance. Occupancy probabilities (eq. 9) include both occupancy and free space information, and they are are modeled with respect to [4] where an inverse model for a sonar sensor is presented. The parameter k specifies the significance of a single measurement. The constraint $\forall l \geq l_{\min} : P(\mathbf{s}(l)) \in (0; 1)$ must be fulfilled to ensure that the resulting probabilities are valid. Figure 1 shows typical plots of occupancy probabilities derived from single depth image pixels. Near distance measurements become more significant than far ones since they are more accurate. Very large distances are interpreted as free space up to a maximum distance threshold.

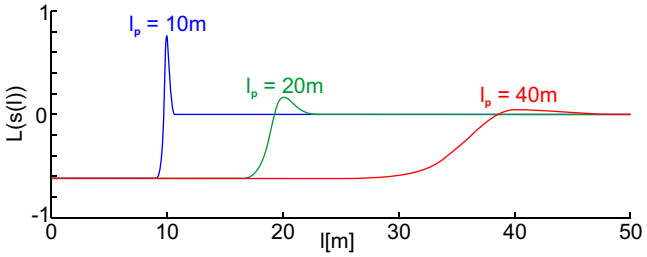

Fig. 1. Inverse Measurement Model. Occupancy values with the parameters $k = 0.1$ and $p_{\min} = 0.35$, showing measurements of 10m (blue), 20m (green) and 40m (red) along the optical axis (i.e. $l_{\mathbf{p}} = z_c$). Camera parameters are $b = 0.3$m, $f = 700$ pixels and $\Delta d = 0.5$ pixels.

In practice, maximal occupancy values in an environment around l are used since obstacles must not be skipped when taking samples of $L(\mathbf{s}(l))$ in the grid drawing process. It is

$$\hat{L}(\mathbf{s}(l)) = \max\left(L(\mathbf{s}(l)) : l \in [l_{\mathbf{p}} - c/2, l_{\mathbf{p}} + c/2]\right) \tag{11}$$

where c is the grid cell size. $\hat{L}(\mathbf{s}(l))$ is approximated by sampling $L(\mathbf{s}(l))$ in the given interval with a high frequency. Function values for distances inside the sensor range and possible measurements are stored in a lookup table for a fast grid calculation.

V. Depth Image Processing

This section describes methods to process depth image pixels so that occupancy values can be drawn into a 3D occupancy grid. Both input image and target map are discrete arrays where an iteration over the elements can be applied. Hence, there are two basic approaches for the extraction of occupancy data from depth images:

- Iteration over depth image pixels (*line drawing*). For each depth image pixel, calculate the projection ray and occupied and free coordinates of the grid. The result is a 3D line that is drawn into the grid map. Points on the line have values according to the inverse model.
- Iteration over the target grid (*ray tracing*). For each grid cell, the projection coordinate on the depth image is calculated. Using the cell position, the corresponding depth pixel value and the measurement model, an occupancy value is assigned to the grid cell.

There is an additional approach, but it ignores free space information [19]. Obstacle point clouds are directly calculated from distance measurements and set into the grid array. This point drawing method is not considered here. Even though it is simple, fast, and allows better resolution in the same computation time, point drawing does not reduce errors due to spurious measurements because the grid maps contain no information to disprove occupancy. Since the described measurement model cannot be used, the results are very different and not drawn to comparison here.

A. Line Drawing

In the case of drawing 3D occupancy grids, line drawing means to iterate over the input data, i.e. the 2D depth image, and calculate occupancy values of the 3D grid for every pixel. The algorithm consists of the following three steps:

1) For a depth image pixel (x, y) with valid depth information d, calculate the projected world coordinate $\mathbf{p}_g$ and distance $l_{\mathbf{p}}$ (eqs. 3, 6).
2) Draw a line through the camera center and $\mathbf{p}_g$ (Bresenham algorithm, extended to 3D) and,
   a) calculate l for every grid cell $\mathbf{s}$ drawn, and
   b) set the occupancy value to $\hat{L}(\mathbf{s}(l))$ (eq. 11). If $\mathbf{s}$ is already occupied with a higher value, leave it (obstacle priority). This occurs when drawn lines overlap.
3) Proceed with step 1 taking the next depth image pixel until lines are drawn for the whole image.

B. Ray Tracing

Unlike line drawing, ray tracing calculates occupancy values the other way round. The iteration is done over the 3D target grid instead of iterating over the input depth image. For every grid cell inside the viewable area, the projected pixel coordinate is calculated. If the disparity value of that pixel or its environment leads to an obstacle where the grid

cell is located, the occupancy probability of that cell is set to a high value. It is done as follows:

1) Calculate the sensor range. The view frustum is determined by the camera center and the projection of the 4 image corner pixels, taking a maximal distance threshold. For ease of use, an aligned bounding box is used in the next step.
2) For each cell $\mathbf{s} = \mathbf{p}_g$ inside the sensor range
   a) calculate Euclidean distance l of the cell from the camera,
   b) calculate image projection pixel coordinate
   c) take maximal disparity $\hat{d}$ in environment around (x, y), that is determined by the image projection of the cell cube corners $\mathbf{s} + (\text{-}0.5, \text{-}0.5, \text{-}0.5)^T, \ldots, \mathbf{s} + (0.5, 0.5, 0.5)^T$,
   d) calculate $l_{\mathbf{p}}$ using $x, y, \hat{d}$, and
   e) set the grid cell $\mathbf{s}$ to $\hat{L}(\mathbf{s}(l))$ using the inverse sensor model (eq. 11).

VI. Improvement with Depth Image Pyramids

A. Depth Images with Lower and Higher Resolution

A depth image contains a lot of measurements that must be processed. As in classical intensity images, those with lower resolution contain similar information with reduced complexity and accuracy. Since the number of pixels is smaller, image processing will be faster. Pyramidal approaches combine images with lower and higher resolution and take advantages of speed and precision.

An example is shown in figure 2. The projection of depth pixels are points in space depending on pixel position and depth value. Due to perspective transformation, the projections of neighboring pixels are closer to each other if the distance from the camera is smaller. With that, neighboring pixels may contain nearly the same information which is rasterized to the same occupancy grid cells (1). Compressed information of these pixels can be used in this case. No change is required for projections where neighboring pixels refer to neighboring grid cells (2) which is most likely if the depth values are inside a certain range described later. Additionally, if projections of neighboring pixels are too distant, images with higher resolution can be used to fill the grid without gaps (3).

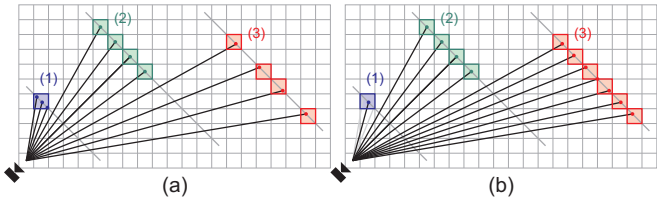

Fig. 2. 2D illustration of depth image pixels projected to the grid map without (a) or with (b) the use of image pyramids.

As it can be derived from equation 2, neighboring pixels are projected to world coordinates that have the distance

$$\Delta\mathbf{p} = z_c \cdot \frac{2^i \cdot 1 \text{ pixel}}{f} \tag{12}$$

from each other. Here, i is the pyramid layer where the neighboring pixels are taken from. The pyramid layers $i > 0$ refer to smaller images where the resolution is reduced by the factor 2^i. Vice versa, $i < 0$ refers to image enlargements. The grid cell size is denoted as c and the constraint

$$0.5 \cdot c < \Delta\mathbf{p} \leq c \tag{13}$$

is used here to define which pyramid layer is taken to project a specific distance or disparity. If $\Delta\mathbf{p} \leq 0.5 \cdot c$, an image with lower resolution fulfills the requirement and in the other direction, if $\Delta\mathbf{p} > c$, an image with higher resolution does.

Inequality (eq. 13) takes into account that a lower depth image resolution is suitable if projected neighboring pixels are very close so that they may refer to the same grid cell and, in the other direction, that a higher image resolution are used if neighboring pixels refer to coordinates too distant from each other so that image details are not lost. It is not considered whether pixels are really referring to the same grid cells since this depends on the actual camera pose and would be slow. Here, the used pyramid level is only dependent on the disparity values and can be stored in a lookup table.

Figure 3 illustrates the connection between disparity values, distances, and the image pyramid layer for an example grid with a resolution of 0.5m. Stereo camera parameters are $b = 0.3$m and $f = 700$ pixels. As shown there, lower depth image resolution is sufficient to project most distances. For example, if the distances between 6 and 25m (disparities approximatively between 8 and 32) are taken into account and higher distances are interpreted as free space up to 25m, only reduced depth image resolutions are needed to create a grid map of 0.5m cell size. Hence, the grid calculation will be fast.

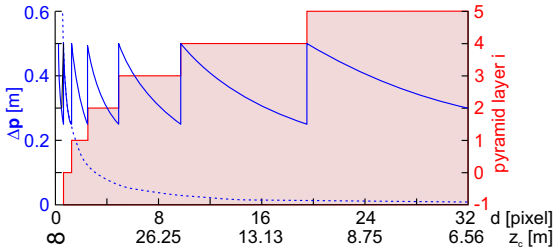

Fig. 3. Size of 1-pixel objects (dotted), 2^i-pixel objects (solid) and the used depth image pyramid layer i, dependent on the disparity value.

For line drawing methods, pyramid levels with $i < 0$ are useful for very large distances in high resolution maps so that the chance of remaining holes in the map is decreasing. In contrast to that, such levels are not useful for ray tracing methods since no additional complexity is provided and disparity values of the original depth image are suitable to collect occupancy information.

B. Pyramidal Line Drawing

For line drawing algorithms, depth image pyramids are created using the maximum distance, i.e. minimal disparity of objects represented by an image region. If the input depth image is denoted as $d_0(x, y) = d(x, y)$, it is recursive

$$d_{i+1}(x,y) = \min\{d_i(2x,2y), d_i(2x{+}1,2y), \\ d_i(2x,2y{+}1), d_i(2x{+}1,2y{+}1)\} \quad (14)$$

with $i \geq 0$. In comparison to d_i, d_{i+1} is an image with a halved width, height, and image center coordinates. Larger images with $i \leq 0$ can be generated through

$$d_{i-1}(x,y) = d_i\left(\lfloor x/2 \rfloor, \lfloor y/2 \rfloor\right) \quad (15)$$

and the image size and parameters are doubled.

The image processing step starts at the maximal pyramid level. If the distance from a given pixel value $d_i(x,y)$ is equal to or less than the valid value for this layer, draw a line. Otherwise, try to draw lines for the pixels

$$d_{i-1}(2x,2y), \quad d_{i-1}(2x+1,2y), \\ d_{i-1}(2x,2y+1), \quad d_{i-1}(2x+1,2y+1).$$

The recursion ends at the minimal existing pyramid level where lines are always drawn.

C. Pyramidal Ray Tracing

Improved ray tracing works by creating depth image pyramids using the minimal distance, i.e. maximal disparity of objects represented by an image region to ensure that obstacles are not omitted. It is

$$d_{i+1}(x,y) = \max\{d_i(2x,2y), d_i(2x{+}1,2y), \\ d_i(2x,2y{+}1), d_i(2x{+}1,2y{+}1)\} \quad (16)$$

with $i \geq 0$. While processing the grid, step 2c of the ray tracing algorithm (section V-B) is changed. For a projected image coordinate $\mathbf{q} = (x,y,1)^T$, the maximal disparity $\hat{d}$ of the environment defined by the four neighboring pixels

$$d_i(\lfloor x/2^i \rfloor, \lfloor y/2^i \rfloor), \quad d_i(\lfloor x/2^i \rfloor + 1, \lfloor y/2^i \rfloor), \\ d_i(\lfloor x/2^i \rfloor, \lfloor y/2^i \rfloor + 1), \quad d_i(\lfloor x/2^i \rfloor + 1, \lfloor y/2^i \rfloor + 1)$$

of the fitting depth image layer i is taken in order to determine the occupancy value set to the actual grid cell.

VII. Grid Postprocessing

After creating a grid map from a depth image, blurring is applied to the grid in order to consider uncertainty in the vehicle position and attitude. In comparison to a classical Gaussian blur with constant standard deviation to all grid cells, the approach described here takes into account that angular uncertainties have a greater influence on larger distance measurements.

It is assumed that translational and angular deviations (σ_{trans} and σ_{angle}, respectively) are not correlated and equal for each axis. To cope with different deviations in each dimension, the largest standard deviation value of the three axes is taken. The total deviation σ in each direction of obstacle points $\mathbf{p}_g$ caused by vehicle uncertainty is

$$\sigma = \sqrt{\sigma_{\text{trans}}^2 + (r \cdot \sigma_{\text{angle}})^2} \quad (17)$$

where r is the Euclidean distance between $\mathbf{p}_g$ and the geodetic vehicle position. With that, occupancy grid cells are blurred with different Gaussian convolution kernels parameterized by the the rounded σ value. Figure 4 shows an example how the kernel sizes can be distributed over the grid. Grid cells far from the vehicle position become more and more blurred. As an example, figure 5 illustrates the basic output of the ray tracing algorithm (left) and the influence of blurring techniques with a constant convolution mask (center) and the advanced method described here (right).

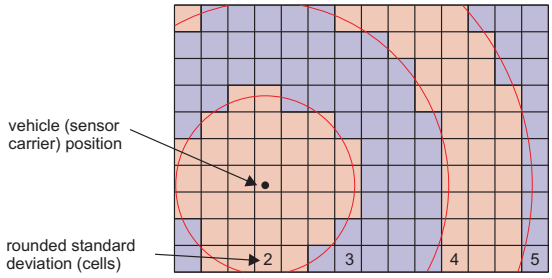

Fig. 4. Example (2D) of convolution kernel sizes dependent on the cell position.

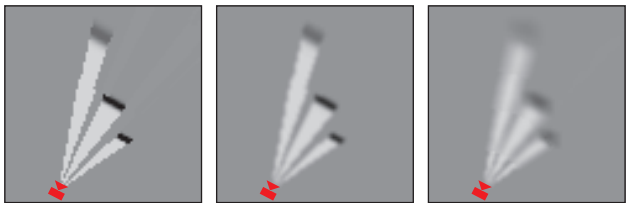

Fig. 5. Example of a grid using the ray tracing method and additional blurring (center, right), simplified 2D view. Vehicle position and viewing direction is marked, darker pixels refer to higher occupancy values.

VIII. Comparison of the Different Approaches

The presented methods are applied to a test image where grids of different resolutions are built. Figure 6 illustrates the test image (nearer distances are brighter green, invalid pixels are white), the expected map (top view, 2D) with view angle and object positions for a ground truth comparison, created maps (without postprocessing) in a simplified 2D view, and a 3D output of the segmented grid (cells from pyramidal line drawing, with an occupancy value of $L(\mathbf{s}) > 0.1$) that has been created with the pyramidal line drawing algorithm. Camera parameters are in all cases: image size 640×480, baselength 0.3m, focal length 700 pixels, disparity uncertainty 0.5 pixels. The grid array is large enough so that a view frustum with a maximal range of 25m fits in all viewing directions. It is calculated in different resolutions: with a cell size of 1.0m, 0.5m, and 0.25m.

As a qualitative result, all objects are identified with every algorithm, see the dark spots in the resulting maps. Bresenham line drawing yields aliasing effects but the map quality and precision seems to be satisfactory as visible. Ray tracing methods tend to draw larger objects into the map. Occluded areas behind objects are also larger. Pyramid usage has only a little effect, especially for line drawing.

Further, the following quantities are identified to compare the algorithms:

- Coordinate transfers: The number of coordinate transforming calculations, i.e. from pixel to grid coordinates or vice versa.
- Used image pixels: The number of memory accesses to pixels of the depth image and its pyramid. The memory accesses can be higher than the number of valid depth

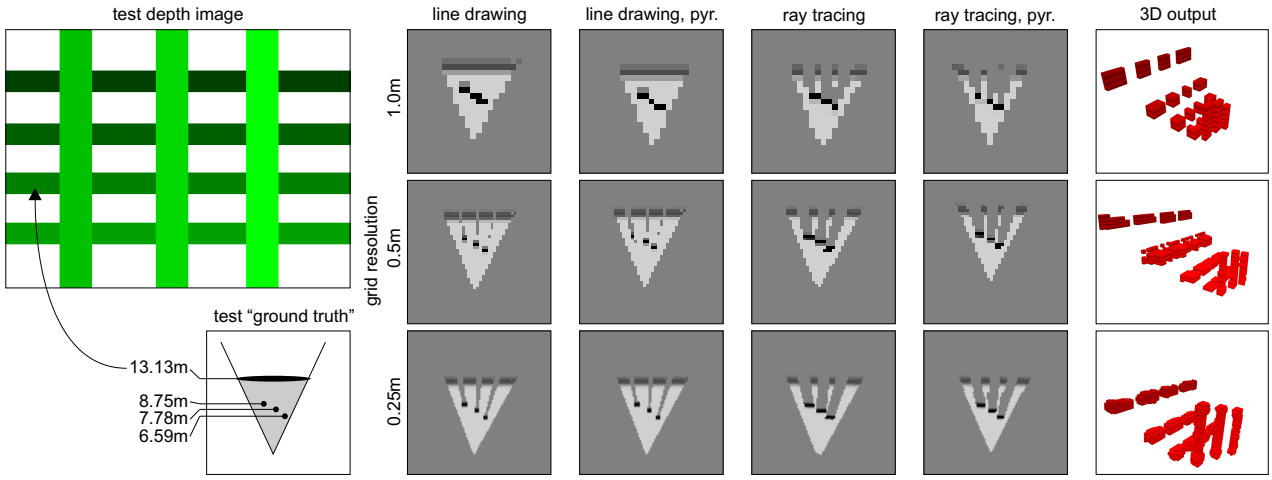

Fig. 6. Example of a testing depth image, expected map as ground truth including object distances, grid maps calculated with different resolutions (simplified 2D), and binary 3D grid representation, slant side view.

image pixels if pixels are checked multiple times, or they can be lower due to pyramid usage.

- Grid cells drawn: The number of occupancy values calculated and assigned to grid cells.
- Different grid cells: The number of grid cells that have changed at the end. Will be lower than the drawn grid cells if some cells have been updated multiple times.
- Calculation time on a 1.8 GHz processor, without blurring. No streaming CPU instructions or graphics acceleration used.

algorithm	coordinate transfers	used image pixels	grid cells drawn	different grid cells	time [ms]
1.0m grid (11400 cells):					
line drawing	194048	194047	2002640	943	207.
line drawing, pyr.	653	652	5487	921	0.62
ray tracing	11405	7939260	11400	11400	50.0
ray tracing, pyr.	11405	13752	11400	11400	6.09
0.5m grid (86400 cells):					
line drawing	194048	194047	4013218	5287	352.
line drawing, pyr.	1710	1709	27942	4739	2.50
ray tracing	86405	62875230	86400	86400	395.
ray tracing, pyr.	86405	106172	86400	86400	44.8
0.25m grid (674500 cells):					
line drawing	194048	194047	7832833	32985	632.
line drawing, pyr.	5392	5391	189446	31946	15.9
ray tracing	674505	505599749	674500	674500	3171.
ray tracing, pyr.	674505	861312	674500	674500	346.

TABLE I

RESULTS OF THE ALGORITHM COMPARISON.

Table I shows the resulting performance when the given test image is processed with the different algorithms. To interpret the facts presented in the table, comparisons between line drawing and ray tracing methods, and between pyramidal and non-pyramidal approaches are drawn. Beside that, it is trivial that a higher resolution has a negative effect on the processing time.

In line drawing approaches, all valid pixels have to be processed, independent from the grid resolution. It reduces the amount of processed image pixels, especially on low grid resolutions where high pyramid levels can be used. With that, the amount of drawn lines decreases. The number of different grid cells drawn is nearly equal as an indicator for similar outputs. Since it is much lower than the drawn grid cells, a lot of cells are drawn multiple times because drawn lines overlay. The probability of drawing lines with exactly the same start and end coordinate decreases, especially on near distances.

Pyramid usage in ray tracing has the effect that the image pixels that have to be checked is decreasing. All other numbers do not change since all cells in the sensor range are drawn once and not multiple times. For every cell, a coordinate transform is done, plus five calculations to determine the field of view.

As a conclusion, line drawing is faster, except for non-pyramidal approaches and low grid resolutions. As expected, in all cases, the calculation performance increases significantly with image pyramids. Especially when combining image pyramids with low grid resolutions, the line draw method is real-time capable. Tests on other images and sequences not shown here offer similar results. Nevertheless, there is a risk of remaining holes in the map when using line draw. Blurring can decrease the risk, but ray tracing can really exclude holes; if it is required.

IX. Application to Aerial Depth Images

The presented data interpretation method is a part of a mapping and world modeling approach developed within the ARTIS (Autonomous Rotorcraft Testbed for Intelligent Systems) research project that deals with unmanned helicopters.

The test vehicle is a model helicopter (fig. 8) with a main rotor diameter of 3 meters and a total weight of maximal 25 kg. The image acquisition and processing equipment are a 30 cm baseline stereo camera (Videre Design STOC) and a separate vision computer (Intel Pentium 4 Mobile).

Results from images taken in flight are presented in figure 7. Two scenarios are shown, the difference is mainly the amount of valid depth pixels to process. The selected images are a part of video sequences acquired during helicopter flight. Geodetic occupancy grids with 0.25m resolution are created using the pyramidal line drawing algorithm and the

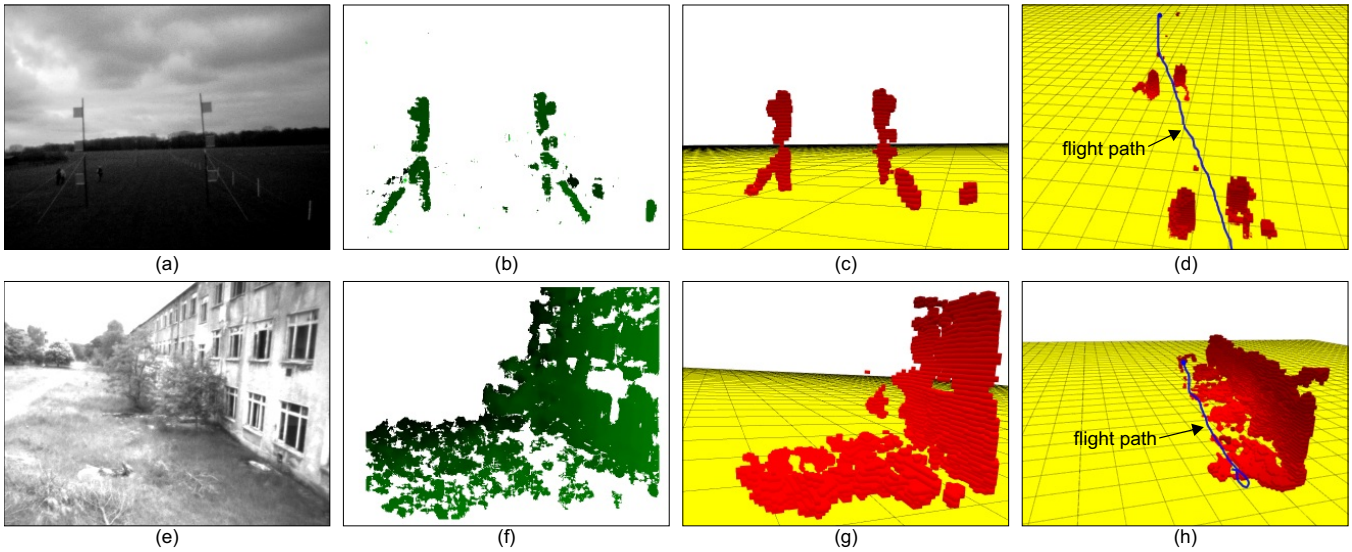

Fig. 7. Creation of 3D occupancy grids from aerial stereo image data. The first scenario illustrates a flight through obstacle posts and an example left camera image (a), depth image (b) created map ($L(s) > 0.1$) from single image (c), final map ($L(s) > 1.0$) from complete sequence (d). The second one is an urban scenario near houses and an example camera image (e), depth image (f), map from single image (g), map from complete sequence (h). Ground plane added to the grid maps for illustration.

Fig. 8. The helicopter ARTIS flying through obstacle posts.

helicopter position taken from GPS/INS-based navigation data. The image processing speed is close to the order of magnitude predicted by the test image: 10 –15 ms for images presented in the first scenario (fig. 7(a-d)) and 20 –25 ms for images of the second one (fig. 7(e-h)). With that, real-time 3D grid mapping from stereo depth data is possible using the presented algorithm.

X. Conclusion

This paper presents approaches to interpret depth images in order to be fused with occupancy grid maps. The methods are used as an update step of mapping techniques in the form of the so-called inverse sensor model. To interpret depth image pixels, a model is developed that copes with specific range resolution in stereo vision, with probabilistic grid representations, and with problems when sampling signals. To draw single pixels into a map, line drawing and ray tracing approaches are introduced and compared. In addition to that, the use of image pyramids and a post-processing step of the resulting grid is explained. It turns out that a combination of line drawing and depth image pyramids is a very efficient way for processing depth images capable of real-time use. Eventually, the approach is proved using images taken onboard an aerial vehicle.

References

[1] S. Thrun, "Learning metric-topological maps for indoor mobile robot navigation," *Artificial Intelligence*, vol. 99, pp. 21–71, 1998.

[2] 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.

[3] Y. Roth-Tabak and R. Jain, "Building an environment model using depth information," *IEEE Computer*, vol. 22, no. 6, pp. 85–90, 1989.

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

[5] M. Montemerlo and S. Thrun, "Large-scale robotic 3d mapping of urban structures," in *Int. Symposium on Experimental Robotics*, 2004.

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

[7] M. C. Martin and H. P. Moravec, "Robot evidence grids," Carnegie Mellon University, Pittsburgh, Tech. Rep. CMU-RI-TR-96-06, 1996.

[8] M. Yguel, O. Aycard, and C. Laugier, "Efficient gpu-based construction of occupancy grids using several laser range finders," in *IEEE Int. Conf. on Intelligent Robots and Systems*, 2006, pp. 105–110.

[9] C. Plagemann, K. Kersting, P. Pfaff, and W. Burgard, "Gaussian beam processes: A nonparametric bayesian measurement model for range finders," in *Robotics: Science and Systems (RSS)*, 2007.

[10] H. P. Moravec, "Robot spatial perception by stereoscopic vision and 3d evidence grids," Carnegie Mellon University, Tech. Rep., 1996.

[11] C. Braillon, C. Pradalier, K. Usher, J. Crowley, and C. Laugier, "Occupancy grids from stereo and optical flow data," in *International Symposium on Experimental Robotics*, 2006.

[12] L. Matthies and S. A. Shafer, "Error modeling in stereo navigation," *IEEE Journal of Rob. and Autom.*, vol. 3, no. 3, pp. 239–248, 1987.

[13] D. Murray and J. J. Little, "Using real-time stereo vision for mobile robot navigation," *Auton. Rob.*, vol. 8, no. 2, pp. 161–171, 2000.

[14] F. Andert and L. Goormann, "A fast and small 3-d obstacle model for autonomous applications," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2008, pp. 2883–2889.

[15] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge: University Press, 2000.

[16] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *International Journal of Computer Vision*, vol. 47, no. 1/2/3, pp. 7–42, 2002.

[17] K. Konolige, "Small vision systems: Hardware and implementation," in *Eighth International Symposium on Robotics Research*, 1997.

[18] Point Grey Research, Inc.: Stereo Products / Stereo Accuracy Chart (Website). www.ptgrey.com, June 2009.

[19] D. Hähnel, "Mapping with mobile robots," Ph.D. dissertation, Universität Freiburg, Germany, 2004.