3D Perception and Environment Map Generation for Humanoid Robot Navigation

Article *in* The International Journal of Robotics Research · October 2008

DOI: 10.1177/0278364908096316 · Source: DBLP

3 authors:

Steffen Gutmann
Google Inc.
73 PUBLICATIONS **3,352** CITATIONS
SEE PROFILE

Masahiro Fujita
Sony Corporation
70 PUBLICATIONS **3,433** CITATIONS
SEE PROFILE

Masaki Fukuchi
32 PUBLICATIONS **734** CITATIONS
SEE PROFILE

3D Perception and Environment Map Generation for Humanoid Robot Navigation

Jens-Steffen Gutmann, Masaki Fukuchi and Masahiro Fujita

Abstract

A humanoid robot that can go up and down stairs, crawl underneath obstacles, or simply walk around requires reliable perceptual capabilities for obtaining accurate and useful information about its surroundings. In this work we present a system for generating 3D environment maps from data taken by stereo vision. At the core is a method for precise segmentation of range data into planar segments based on the algorithm of *scan-line grouping* [Jiang and Bunke, 1994] extended to cope with the noise dynamics of stereo vision. In off-line experiments we demonstrate that our extensions achieve a more precise segmentation. When compared to Murray's *patchlet* method [2003], we obtain a richer segmentation with a higher accuracy while also requiring far less computations. From the obtained segmentation we then build a 3D environment map using occupancy grid and floor height maps. The resulting representation classifies areas into one of six different types while also providing height information of objects. We apply our perception method for the navigation of the humanoid robot QRIO and present experiments of the robot stepping through narrow space, walking up and down stairs, and crawling underneath a table.

keywords: Humanoid robot navigation, 3D environment perception, range image segmentation, stereo vision.

1 Introduction

Humanoid robots posses many degrees of freedom allowing them to navigate in areas where wheeled robots run into difficulties. For example they can go up and down staircases or crawl underneath obstacles such as chairs or tables. In order to find a route in such environments reliable perceptual capabilities are needed to provide the robot with accurate and meaningful information about its surroundings.

In this paper we present a system for creating an environment map that is useful for navigating a humanoid robot from a start to a goal location. At the core is a method for the precise segmentation of stereo data into planar regions based on the algorithm of *scan-line grouping* [Jiang and Bunke, 1994] extended in various ways to account for the different noise characteristics in stereo vision.

From the segmented data we then compute a floor height map and combine it with a 3D occupancy grid built from the raw range measurements. The resulting representation divides the robot's surroundings into a regular grid where cells hold information about an *environment type* with an associated height value. The environment types we wish to distinguish are designed for the navigation of the robot and are composed of types for regions where the robot can walk ordinarily (floor), require the robot to step up or down (stairs), allow the robot to crawl underneath an object (tunnel), are blocked (obstacle,) are unsafe because the robot might fall down (border), or haven't been sensed yet (unknown).

The application of this navigation map lies in path-planning and obstacle avoidance. In previous work we showed how to extend this representation in order to detect collisions between robot and environment, and how to employ a traditional A* search algorithm for path finding [Gutmann *et al.*, 2005c].

The contribution of this paper is twofold. First we present an improvement to scan line grouping with an experimental validation. Our improvements address the splitting of line segments by analyzing the distribution of points along a scan line, residual error computation for providing a statistical distance measure for region growing, and a data structure allowing for efficient plane estimation without accessing image pixels in the main algorithm. This enables the method to segment images exhibiting a high noise dynamic as noise is analyzed in a local neighborhood during line extraction and residual errors reflecting this noise steer the process of region growing. We validate our approach on two sets of range images and compare our results to a state-of-the-art method that employs RANSAC, EM and so-called *patchlets* [Murray and Little, 2004; Murray, 2003].

Our second contribution lies in a novel method for computing a detailed 3D environment map useful for the navigation of a humanoid robot. We combine a coarse 3D occupancy grid with the plane information obtained in the segmentation process. This achieves robustness to sensor noise while also providing precise floor heights. We also present an algorithm for computing an environment type for each location. Experimental results of sample navigation runs illustrate the usefulness of our system.

Part of the work presented in this paper has previously been published in a symposium [Gutmann *et al.*, 2004b] and in two conference papers [Gutmann *et al.*, 2005a; 2005b]. While this paper integrates the contributions of the previous ones, it also provides refined versions of our algorithms and more experimental results regarding the evaluation of our range segmentation method.

The rest of this paper is organized as follows. The next section discusses methods for range data segmentation. Section 3 introduces the mathematical tools for fitting a plane to a set of 3D points and presents our improvements to the algorithm of scan-line grouping. In Section 4 we experimentally evaluate our method and compare it to competing approaches. Section 5 presents our method for environment map building and classification. In Section 6 we implement our perception technology on the humanoid robot QRIO and illustrate its performance in a sample navigation run. We conclude in Section 7.

2 Plane Segmentation

Segmentation of range data is a fundamental problem in 3D image recognition. Besides its usefulness for mobile robot navigation, its applications include the generation of geometric scene descriptions [Jiang and Bunke, 1994; Murray, 2003; Hoover *et al.*, 1996], or the building of 3D models from raw sensor range data [Iocchi *et al.*, 2000; Hähnel *et al.*, 2002; Nüchter *et al.*, 2003; Thrun *et al.*, 2004]. The segmentation procedure labels a set of input range measurements into a set of segments. Range data is usually provided by laser range finders, structured light cameras, or stereo cameras and the segments of interests are often planar regions that can be fitted to the labeled data.

Jiang and Bunke [1994] presented an efficient and precise method for segmenting range data of an accurate sensor into a set of planes. Their main idea is that points on a scan line in the range image form a straight line if they belong to the same planar region. This is because in a calibrated camera each scan line defines a plane passing through the focal point. When this plane intersects with a plane of the environment, a straight line segment is observed. Based on this idea they first extract line segments in each scan line and then perform region growing using the line segments as primitives. The result is a precise segmentation close to the best results available at that time but at a significant lower computation cost than competing methods using pixel-based region growing or clustering [Hoover *et al.*, 1996].

Concerning noise, Jiang and Bunke used global parameter settings making it difficult to directly apply the method to segmenting images exhibiting a high dynamic in noise such as stereo disparity images. The range error in stereo vision increases quadratic in the distance to the observed object (assuming a constant error in disparity). Thus, points farther away from the sensor have a much larger error than close measurements. Trying to model this error is complicated by the fact that the error does not only depend on distance but also on other factors such as available texture, etc.

Recent trends in segmentation are to move away from *ad hoc* methods like region growing to methods that are mathematically more sound. Approaches using the Hough transformation [Iocchi *et al.*, 2000; Okada *et al.*, 2001], random sample consensus (RANSAC) [Nüchter *et al.*, 2003], expectation maximization (EM) [Thrun *et al.*, 2004], and the generalized principal component analysis (GPCA) [Vidal *et al.*, 2003] have been presented and provide a solid framework for segmenting parametric curves in high dimensional data. Most of these algorithms usually treat data points independently without considering neighboring pixels in an image, i.e. models have no boundaries. This can be problematic if curves are aligned in a pattern where the pattern itself can be regarded as a candidate for a curve. For example, when extracting planes in a range image of a staircase, one has to be careful not to segment all data into a single *ramp*. The algorithms also require substantial more computations than region growing as data points need to be accessed multiple times (randomized Hough transformation, RANSAC, EM [all iterative], GPCA [polynomial in number of models]) or require an expensive update of another data structure (Hough transform). This limits their application for the use in real-time systems.

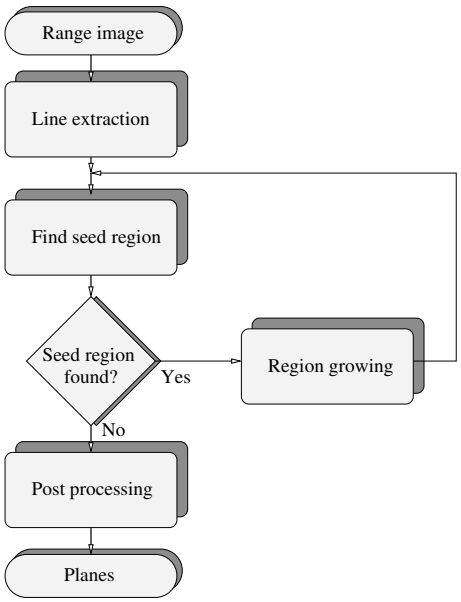

Figure 1: Flow-chart of scan-line grouping.

3 Improvements to Scan Line Grouping

A flow-chart diagram of our extended scan-line grouping method is depicted in Fig. 1. Before presenting the details about each sub system we introduce some mathematical tools that allow to compute plane parameters in an efficient way.

3.1 Fitting Planes to Range Data

Let $\mathbf{p}_i = (x_i, y_i, z_i)^T, i = 1 \dots N$ be a set of range measurements belonging to the same planar region. We represent a plane by a unit-length normal vector $\mathbf{n} = (n_x, n_y, n_z)^T$ and signed distance d to the origin such that all points $(x, y, z)^T$ on the plane satisfy:

$$x\, n_x + y\, n_y + z\, n_z - d \;=\; 0. \qquad (1)$$

Fitting a plane to the range data is achieved by least-squares estimation, i.e. we minimize the following energy function:

$$fit(\mathbf{n}, d) \quad = \quad \sum_{i=1}^{N} (\mathbf{p}_i^T \, \mathbf{n} - d)^2. \qquad (2)$$

There are several approaches for finding the optimal parameters that minimize (2), see e.g. [Weingarten *et al.*, 2004] for a discussion. For range sensors such as stereo cameras, if we choose the focal point as origin of the coordinate system then observed planes cannot pass through the origin due to visibility constraints, thus $d \neq 0$. This allows to optimize the error function divided by d^2 and by substituting $\mathbf{m} = \mathbf{n}/d$ we obtain:

$$\frac{fit(\mathbf{n}, d)}{d^2} \quad = \quad \sum_{i=1}^{N} (\mathbf{p}_i^T \, \mathbf{m} - 1)^2. \qquad (3)$$

Deriving (3) with respect to $\mathbf{m}$ and setting the result to zero leads to a linear equation system:

$$A \, \mathbf{m} \quad = \quad \mathbf{b}, \quad \text{with} \qquad (4)$$

$$A = \sum_{i=1}^{N} \mathbf{p}_i \, \mathbf{p}_i^T, \qquad \mathbf{b} = \sum_{i=1}^{N} \mathbf{p}_i, \qquad (5)$$

which can be solved for $\mathbf{m}$ using Cramer's rule. The plane parameters are then obtained by back substitution:

$$\mathbf{n} = \frac{\mathbf{m}}{\|\mathbf{m}\|}, \qquad d = \frac{1}{\|\mathbf{m}\|}. \qquad (6)$$

where $\|\mathbf{m}\|$ denotes the length of $\mathbf{m}$. Although (4)–(6) do not minimize the original least-squares problem (2), the difference to the exact solution is usually small for $|d| \gg 0$.

The advantage of this representation is that it is additive. By maintaining the quantities A and $\mathbf{b}$, a data point can be added (or removed) and plane parameters be recomputed in constant time. This also holds for a set of N data points where only information about the first two moments $E(\mathbf{p})$ and $E(\mathbf{p}\mathbf{p}^T)$ is known. Updating A and $\mathbf{b}$ is achieved by:

$$A \leftarrow A + N \, E(\mathbf{p}\mathbf{p}^T), \qquad \mathbf{b} \leftarrow \mathbf{b} + N \, E(\mathbf{p}). \qquad (7)$$

For a plane given by parameters $\mathbf{n}$ and d, the root mean square (RMS) residual is defined as:

$$rms(\mathbf{p}_1 \dots \mathbf{p}_N) \quad = \quad \sqrt{\frac{1}{N} \sum_{i=1}^{N} (\mathbf{p}_i^T \, \mathbf{n} - d)^2} \qquad (8)$$

which can again be computed from the first two moments:

$$rms(N, \mathbf{b}, A) \quad = \quad \sqrt{\frac{1}{N} (\mathbf{n}^T A \mathbf{n} - 2d\mathbf{n}^T \mathbf{b}) + d^2} \qquad (9)$$

The RMS residual is a measure of how well the set of points $\mathbf{p}_i$ fit to the plane given by $\mathbf{n}$ and d. If the plane is a least-squares fit to the data points $\mathbf{p}_i$ then the RMS residual resembles the standard deviation σ of the plane. This is the minimal RMS error we can achieve when fitting a plane to all points $\mathbf{p}_i$. By computing the ratio

$$r_\sigma(N, \mathbf{b}, A, \sigma) \quad = \quad \frac{rms(N, \mathbf{b}, A)}{\sigma} \qquad (10)$$

of the RMS residual of a set of points given by their statistics N, $\mathbf{b}$, and A and the plane error σ we obtain a statistical distance measure. If the plane is not an optimal least-squares fit then this ratio will be larger than 1. Thus, by placing a threshold on $r_\sigma(N, \mathbf{b}, A, \sigma)$ we can decide whether a set of points fits to a plane of standard deviation σ. We will make use of this distance metric in the region growing process.

3.2 Line extraction

For line extraction, the data points $\mathbf{p}_i$, $i = 1 \dots N$ of a scan line can be processed in the 2D coordinate system of the plane corresponding to the scan line. The classic algorithm for this line segmentation is the *split* algorithm of Duda and Hart [1973]:

Algorithm *split*$(\mathbf{p}_1 \dots \mathbf{p}_N)$
Input: Set of points $\mathbf{p}_1 \dots \mathbf{p}_N$ along a scan line
Output: Set of line segments approximating point curve
Sequence:
  if $N < N_{min}$ **return** $\emptyset$
  $k := \text{argmax}_{i=2\dots(N-1)} \, dist(\mathbf{p}_i, \mathbf{p}_1, \mathbf{p}_N)$
  if $dist(\mathbf{p}_k, \mathbf{p}_1, \mathbf{p}_N) < d_{cord}$ **then**
    return $\{(\mathbf{p}_1, \mathbf{p}_N)\}$
  else
    return *split*$(\mathbf{p}_1 \dots \mathbf{p}_k) \cup$ *split*$(\mathbf{p}_{k+1} \dots \mathbf{p}_N)$
  endif

Here N_{min} is the minimum number of points for defining a line segment and $dist(\mathbf{p}_i, \mathbf{p}_1, \mathbf{p}_N)$ computes the distance of $\mathbf{p}_i$ to the closest point on the line passing through $\mathbf{p}_1$ and $\mathbf{p}_N$. If this distance is below a threshold d_{cord} for all points then a line segment (defined by $\mathbf{p}_1$ and $\mathbf{p}_N$) has been found. Otherwise the point set is split at the most distant point p_k and processed in a recursive way.

The *split* algorithm is a powerful tool when noise in data is constant over the whole image. In stereo vision, however, noise can vary a lot. For example, Fig. 2 contains two data sets, each along a scan line to which a line has been fitted using linear regression. Both lines are identical and exhibit the same standard deviation. However, the segmentation in Fig. 2(a) can be improved, while the data in Fig. 2(b) contains too much noise for refining the line.

The *split* algorithm does not work well here because it is not possible to find a suitable threshold d_{cord} for deciding when to split the point set. Therefore, we propose an improved algorithm *rd-split* which like the classical one processes data in a recursive way but also analyzes the distribution of points along the fitted line similar to the run-distribution test of Fitzgibbon and Fisher [1994]:

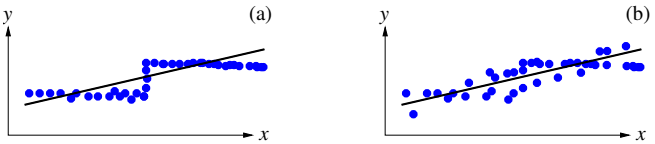

Figure 2: Line fitting to (a) precise and (b) distorted data.

Algorithm *rd-split*$(\mathbf{p}_1 \dots \mathbf{p}_N)$
Input: Set of points $\mathbf{p}_1 \dots \mathbf{p}_N$ along a scan line
Output: Set of line segments approximating point curve
Sequence:
  if $N < N_{min}$ **return** $\emptyset$
  $L :=$ *least-squares-fit*$(\mathbf{p}_1 \dots \mathbf{p}_N)$
  $C :=$ *max-count-consecutive-same-side*$(L, \mathbf{p}_1 \dots \mathbf{p}_N)$
  if $C < C_{max}$ **then**
    return $\{L\}$
  else
    $k := \text{argmax}_{i=2\dots(N-1)}\, dist(\mathbf{p}_i, \mathbf{p}_1, \mathbf{p}_N)$
    return *rd-split*$(\mathbf{p}_1 \dots \mathbf{p}_k) \cup$ *rd-split*$(\mathbf{p}_{k+1} \dots \mathbf{p}_N)$
  endif

Here *least-squares-fit*$(\mathbf{p}_1 \dots \mathbf{p}_N)$ fits a straight line to the data points using least-squares estimation [Lu, 1995, p. 42] which involves the computation of the first two moments of the data. The points along this line are then analyzed in *max-count-consecutive-same-side*$(L, \mathbf{p}_1 \dots \mathbf{p}_N)$ which returns the maximum number of consecutive points that all fall on the same side of L. If the points are randomly distributed along L (as e.g. in Fig.2(b)) then this number should be small. Thus, if C falls below a threshold C_{max} then the line L is reported. Otherwise, a large C indicates that the point set can be refined (e.g. the set of points in Fig.2(a)). We use the same criteria as in the classical *split* algorithm for finding the critical point. Note, that C might also become larger than C_{max} by coincidence since any random distribution along the line might contain a large set of consecutive points all falling on the same side. It can be shown that the probability $P(C > C_{max})$ increases with the number of points. Thus, for large N, a line that actually fits well to the data set might occasionally also be split.

Each line L found by the *rd-split* algorithm is recorded as

$$L \quad = \quad (s, b, e, N, \mathbf{b}, A, \sigma) \tag{11}$$

where s is the scan line index, b and e are the start and end indices in the scan line, $\mathbf{b} = \sum_{i=1}^{N} \mathbf{p}_i$ and $A = \sum_{i=1}^{N} \mathbf{p}_i \mathbf{p}_i^T$ are the first two moments times the number of points N, and σ is the least-squares fitting error of the line.

The major advantage of the *rd-split* algorithm is that it is independent of the employed sensor and the noise level in the data. It is a general tool for approximating a set of points with line segments without knowing the measurement errors. The only parameters to choose are N_{min} and C_{max} which correspond to the level of detail that should be achieved by the segmentation and mainly depend on the complexity of the observed scene.

3.3 Selection of seed region

We define a seed region for scan line grouping as a triple of neighboring line segments that pass the following test. Let $L_i = (s_i, b_i, e_i, N_i, \mathbf{b}_i, A_i, \sigma_i)$, $i = 1, 2, 3$ be 3 line segments where $s_2 = s_1{+}1$, $s_3 = s_2{+}1$ and $b_2 < e_1$, $b_1 < e_2$, $b_3 < e_2$, $b_2 < e_3$, i.e. the line segments are neighbors overlapping in their indices. A plane is fitted to these lines by computing

$$\mathbf{b} = \mathbf{b}_1 + \mathbf{b}_2 + \mathbf{b}_3, \qquad A = A_1 + A_2 + A_3, \tag{12}$$

solving the equation system of (4), and determining the plane parameters $\mathbf{n}$ and d according to (6). We then compute the ratio of RMS residual to plane error using (10). A seed region has been found if for all $i = 1, 2, 3$

$$r_\sigma(N_i, \mathbf{b}_i, A_i, \sigma_i) \quad < \quad r_{seed} \tag{13}$$

where r_{seed} is a preset threshold. Note that the standard deviation σ_i of a line is treated here as the plane error. In principle the plane error could be directly computed by (9). However, the different line segments might exhibit different levels of noise which is ignored when computing a global plane error over all 3 lines. Therefore we use the individual line errors as each σ_i is an upper bound to the error of a plane containing the line (the error reaches σ_i if the plane is orthogonal to the plane passing through the focal point and containing scan line s_i). In our experiments this gave better results than when using the RMS residual of (9).

3.4 Region growing

The region growing step maintains an *open* list initialized with the 3 line segments of the seed region. In an iterative way, a current line $L_c = (s_c, b_c, e_c, N_c, \mathbf{b}_c, A_c, \sigma_c)$ is chosen and removed from the *open* list until there are no more lines to process. Lines $L_i = (s_i, b_i, e_i, N_i, \mathbf{b}_i, A_i, \sigma_i)$ neighboring L_c are searched for by finding candidates in scan lines $s_c - 1$ and $s_c + 1$ for which $[b_c, e_c] \cap [b_i, e_i] \neq \emptyset$. If a candidate is found, the ratio of RMS residual to plane error is computed according to (10). If this ratio falls below a threshold r_{grow}:

$$r_\sigma(N_i, \mathbf{b}_i, A_i, \sigma_i) \quad < \quad r_{grow} \tag{14}$$

then L_i is moved into the *open* list, the equation system of the plane is updated according to (7) and parameters $\mathbf{n}$ and d are re-computed by (6). Once a line has been moved into the *open* list it is not considered as a candidate for seed selection or region growing again.

Like in the selection of the seed region, we use σ_i as an estimate for the plane error rather than the global RMS residual over all data points. The line error σ_i can be regarded as a local measure of the noise in data and, although it is only an upper bound, it is a better means for deciding membership if the image exhibits a high dynamic in noise.

3.5 Post processing

After grouping all lines into planar regions several post processing steps can be applied for improving the segmentation. Here is a summary of steps in our implementation.

Trading line segments: Since our algorithm is a *hill climbing* method, the assignment of lines to planes might not be optimal. We examine line segments at the border of a plane whether a line better fits to a neighboring plane based on the RMS residual of (9).

Trading points: The critical points found in the *rd-split* algorithm are not always optimal. By checking whether single points at the border of a plane better fit to a neighboring plane, the plane borders are refined. For this, we compute the statistics N, $\mathbf{b}$ and A in a window centered on the point and use the RMS residual for deciding the optimal plane. We also include data points which were not segmented into a line segment.

Merging planes: A pair of neighboring planes that pass a similar test as the one for finding seed regions are merged into one.

The final output of the scan-line-grouping algorithm is a set of plane regions $\{P_i\}$ where each region is defined as $P_i = (\mathbf{n}_i, d_i, N_i, \mathbf{b}_i, A_i, \mathcal{B}_i)$. Here $\mathbf{n}_i$ is the unit length normal vector of the plane, d_i the signed distance to the origin, N_i the total number of points belonging to this plane, $\mathbf{b}_i$ and A_i are the first two moments times N_i, and $\mathcal{B}_i$ describes the region border which is represented as a vector of pairs where each pair holds the start and end index of data points belonging to the plane for a given scan line. The information for $\mathcal{B}_i$ can be computed during the region growing process when adding each line to the plane and modifying it accordingly in the above post processing steps.

4 Segmentation Results

We evaluated our improvements to scan line grouping on two different range data sets and compared the results to the original UB method [Jiang and Bunke, 1994] and a recently published, different method using so called *patchlets* [Murray and Little, 2004; Murray, 2003]. The first data set contains range images from an ABW scanner (507x512 pixels) which measures range using structured light.[1] These images were previously used in a comparison of four different range segmentation methods [Hoover *et al.*, 1996]. The second data set are disparity data generated from images of a Digiclops trinocular stereo camera (285x205 pixels).[2]

Our results were obtained with the parameters listed in Table 1. For the ABW images we set $C_{max} = 10$ which gave slightly better results than the setting $C_{max} = 15$ which we used for the stereo data. We believe, this is because the ABW images contain scenes that are more complex than those of the stereo data. This requires a lower C_{max} for splitting a scan line into a larger number of segments.

Table 1: Parameters used in our experiments.

$N_{min} = 5$	$C_{max} = 10$ / 15	$r_{seed} = 2$	$r_{grow} = 3$

Our results on the ABW image data set are similar to the original scan line grouping method. Fig. 3 shows our results on ABW test image #8. Each segmented plane is indicated by a different color (gray scale) with the border drawn in black. When using the comparison framework of Hoover *et al.* [1996] our segmentation achieves a correct detection of 19 of the 20 planes with 1 region missing using a compare tolerance of 66 %. When raising the compare tolerance to 80 %, the comparison tool reports 17 as being correct while 3 are missed and 2 are considered as noise. This result compares well to the UE and UB segmentations [Hoover *et al.*, 1996, Fig. 13]. A difference of our method to the UB scan line grouping is that we do not pre-filter the image with a median filter and thus our results leave outliers unlabeled similar to the UE result.

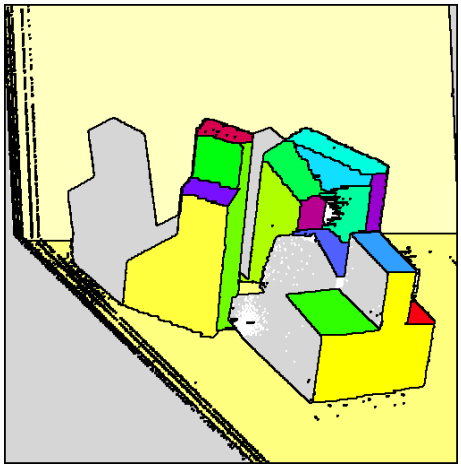

Figure 3: Segmentation on ABW test image #8.

In general, the ABW images contain range information with an almost constant error except for outliers that can be separated from the data easily, e.g. by a median filter. The UB scan line grouping method uses the classical *split* algorithm for line segmentation and a simple check for a line to decide if it belongs to a plane by placing a threshold d_{line} on the distance between both end points and the plane. This approach is very successful if noise is low and constant.

When experimenting with the UB approach on stereo images, the quality of results changes. Fig. 4(a) shows an image of a room made of planar walls with bands connecting walls and floor. The disparity data (Fig. 4(b)) shows a large range of values which indicates a high dynamic in range error. When

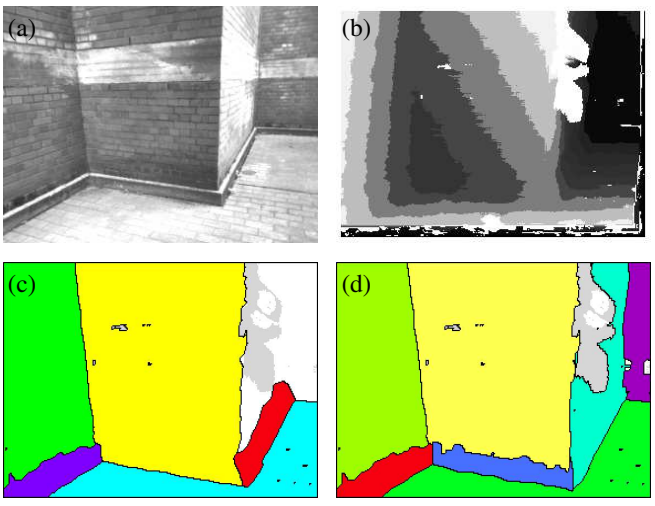

Figure 4: Stereo range data of a room. Distance to near wall is about 2 m, distance to end of the room is about 6 m. (a) camera image, (b) disparity image, (c) result of UB method, (d) our result.

[1] These images are available from the USF Range Image Database http://marathon.csee.usf.edu/range/DataBase.html.

[2] Don Murray made these images available under his *Patchlet* homepage http://www.cs.ubc.ca/spider/donm/patchlets.

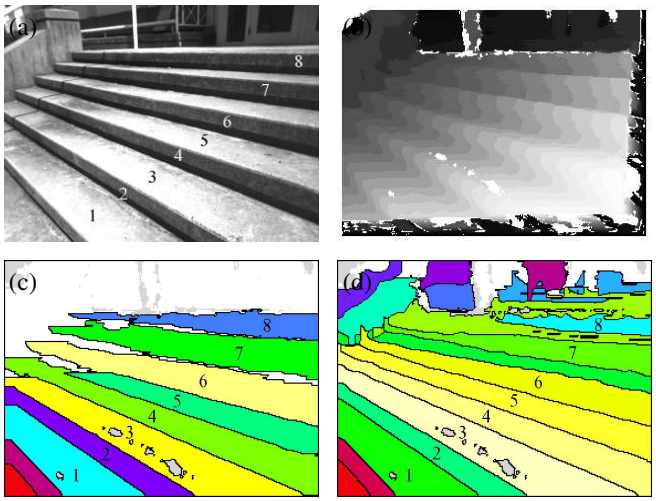

Figure 5: Stereo range data of a stair case. For the 8 labeled surfaces ground truth information is available. (a) camera image, (b) disparity image, (c) result of UB scan line grouping, (d) our result.

setting d_{cord} = 15 mm and d_{line} = 30 mm,[3] the UB method produces the segmentation shown in Fig. 4(c). The result of our method is shown in Fig. 4(d). UB fails to detect the distant walls while producing a more coarse segmentation for the closer wall than our method. Other values for d_{cord} and d_{line} can provide either finer results for closer objects or coarser results that include distant objects. However, we were unable to find parameters than can achieve both as it is impossible to reflect the noise dynamics using absolute thresholds.

Figs. 5(a)–(d) contain results on images of a stair case. Segmenting this data into planes is challenging because of the many small surfaces which could be considered as noise when segmenting the whole staircase into a single ramp. E.g., when applying the EM algorithm on this data using the result of our scan line grouping method as initial seed, we observed that EM quickly diverges. In our experiments using EM each data point is treated individually and only the distance perpendicular to an (infinite) plane is regarded. Therefore, as the seed usually contains an inclined plane and the border of segments is ignored, many points on the stairs show a closer distance to this plane than to their true planar surface. This plane is then refined to better approximate these points and the method fails. This has also been reported by Murray and Little [2004; 2003].

Scan line grouping has the potential to achieve a precise segmentation since its line segmentation searches for the points where planes intersect. Our method is again able to provide a richer segmentation than UB (using the same fixed thresholds as in the previous example). For example, the concrete balustrade is partially recovered. Some of the upper and more distant stairs are merged into an inclined plane. The noise at these locations is too large to let our method find a finer approximation.

[3]These parameters correspond to $\mathbf{T}_1$ and $\mathbf{T}_2$ in [Hoover *et al.*, 1996].

Table 2: Comparison of segmentation accuracy for Fig. 5.

	Patchlet approach		Our approach	
Surface	# pixels	% correct	# pixels	% correct
1	4313	99.9	4192	100.0
2	3164	69.7	3086	74.1
3	6532	99.1	7114	99.9
4	6490	78.8	6119	84.9
5	4283	99.4	5537	97.5
6	6649	66.7	5107	85.6
7	5714	61.9	3709	87.2
8	2415	80.9	1772	90.7
Overall	39560	82.0	36636	91.1

The images of Figs. 4 and 5 were previously used in an evaluation of a segmentation method that computes a *patchlet* containing position, orientation, size and confidence measures for each image pixel. Using RANSAC an initial set of planes is found which is then refined by the EM algorithm [Murray and Little, 2004; Murray, 2003]. While the method is able to correctly segment data of a staircase, our results are in general superior in that a more detailed segmentation containing more segments is achieved. For the room image (Fig. 4) patchlets report 5 regions (see [Murray, 2003, Fig. 7.27]) compared to 7 in our method, for the staircase (Fig. 5) patchlets compute 8 regions (see [Murray, 2003, Fig. 7.23]) compared to 18 in our method.

Murray also provides quantitative numbers on the accuracy of the patchlet method using ground truth data obtained by manual labeling of the images. The accuracy of segmentation is determined by counting the number of pixels of a segment that are supported by the ground truth [Murray, 2003, p. 136]. For the staircase scene of Fig. 5 ground truth data of 8 surfaces has been made available. The accuracy of segmentation reported by Murray [Murray, 2003, Table 7.3] and the result obtained by our method are summarized in Table 2.

As the table suggests our method in general produces a more accurate segmentation than the patchlet method. However, this result has to be taken with a grain of salt. As we allow for a coarse segmentation in areas of high noise, e.g. far regions, the overall accuracy can drop significantly if we include surfaces farther away. For example if the 3771 pixel large segment surrounding surface 8 in Fig. 5(d) is associated with the (unlabeled) ground truth surface located between 7 and 8, an accuracy of only 38.7 % is achieved, reducing the overall accuracy to 86.3 %. Nevertheless, our method is usually more precise on data closer to the sensor. In another two images containing frontal and closeup views of the staircase our method also produces competitive numbers to the patchlet approach: For the 5 surfaces of Fig. 6 [Murray, 2003, Fig. 7.36] we obtain an overall accuracy of 86.1 % (compared to 85.8 % of the patchlet method), and for the 6 surfaces of Fig. 7 [Murray, 2003, Fig. 7.34] our method has an accuracy of 95.9 % (compared to 93.0 %)

A big advantage of scan line grouping is its efficiency. Table 3 shows the processing time of our algorithm on a Pentium 4 and a MIPS R5000 CPU for the images reported in this paper. The higher run-time in post processing for the

Table 3: Processing time in milliseconds for the images reported in this paper

Image	ABW # 8 Fig. 3	Room Fig. 4	Stairs/side Fig. 5	Stairs/front Fig. 6	Stairs/close Fig. 7	Stairs/QRIO Fig. 10	Stairs/QRIO Fig. 10
Size	507x512	285x205	285x205	285x205	285x205	88x72	88x72
Processor	Pentium 4	Pentium 4	Pentium 4	Pentium 4	Pentium 4	Pentium 4	MIPS R5000
Clock	3GHz	3GHz	3GHz	3GHz	3GHz	3GHz	400MHz
Line extraction	23.5	9.4	9.3	6.0	3.9	0.4	6.2
Region growing	5.9	2.8	3.1	1.8	0.8	0.1	1.8
Line trading	1.4	1.7	3.2	1.0	0.1	0.0	1.2
Point trading	7.5	15.7	33.1	17.8	14.7	0.1	0.3
Plane merging	0.1	1.0	1.6	0.8	0.0	0.0	0.0
Total	38.4	30.6	50.3	27.4	19.5	0.6	9.5

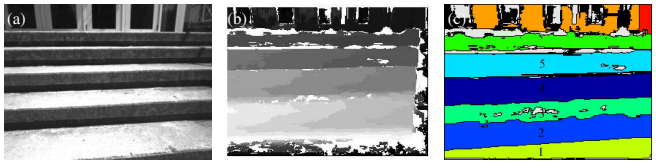

Figure 6: Front view of the stair case. (a) camera image, (b) disparity image, (c) segmentation result.

Figure 7: Closeup view of the stair case. (a) camera image, (b) disparity image, (c) segmentation result.

stereo images are the result of using a larger window size (9x9) than for the ABW images (3x3). Murray reports a runtime of several minutes for the patchlet approach [Murray, 2003, p. 147]. Although there is room for optimizations, their algorithm would probably still be two or three orders of magnitude slower than scan line grouping.

5 Map Building and Classification

We now turn to the problem of generating an environment map from sensor data that is useful for the navigation of a humanoid robot. A number of previous systems assume the world is known in advance, i.e. they only work in simulation [Shiller *et al.*, 2001; Chestnutt *et al.*, 2003; Li *et al.*, 2003]. The systems running on real robots sometimes place severe restrictions to the environment or possible robot actions. Lorch *et al.* [2002] restrict the environment to only contain box-shaped objects that can be recognized by edge detection in monocular vision. Kagami *et al.* [2003] model the environment by a 2.5D height map providing precise information about floor and obstacles heights. Such a representation while being useful for avoiding obstacles does e.g. not allow to find tables the robot could crawl underneath. For squatting and crawling, a 3D occupancy grid map can be employed where empty volumes in the grid correspond to holes in the environment [Kanehiro *et al.*, 2005]. However, the choice for the grid resolution can be difficult if not only holes but also precise information about the floor height is required.

We represent the environment by a regular 2 dimensional floor and obstacle grid map *FOG* which is created from sensor data under the following assumptions:

1. The world is separated into floor and obstacles.
2. Floor is planar and horizontal (else it is an obstacle).
3. There are no multiple floor levels at the same location.
4. The robot is able to distinguish between floor and obstacles and can estimate their relative position and height using its sensors.

Note that these assumption still allow to represent obstacles above the floor where the robot can crawl underneath. In the rare situation where there are multiple floor levels at the same location we only consider the highest one. Usually this is the one relevant for navigation. In principle the restriction to horizontal surfaces could be relaxed by also estimating roll and tilt of inclined surfaces.

It is important that the floor height can be estimated precisely since kinematic constraints of humanoid robots need accurate information to ensure stable walking. On the other hand, obstacle heights are allowed to be imprecise since our only aim is to avoid them. Under these considerations, we can base our perception system on a coarse 3D occupancy grid *OG* and a precise floor height map $Floor$:

$$OG : (x, y, z) \mapsto p, \qquad p \in [0, 1] \tag{15}$$

$$Floor : (x, y) \mapsto h, \qquad h \in R \cup \{-\infty\}. \tag{16}$$

where $-\infty$ has the special meaning of no floor information.

The combination with a 3D occupancy grid also provides robustness against sensor noise, i.e. it allows to verify the height values in the floor map (e.g. when the environment changes) and enables the detection and filtering of outliers.

From the two maps we can compute the *FOG* map that classifies each cell into a type and provides height information about the corresponding entity:

$$FOG : (x, y) \mapsto (t, h) \tag{17}$$

where t is one of the following types:

- *floor*: even surface the robot can step on,
- *stairs*: small change in floor height,
- *border*: large change in floor height,
- *tunnel*: low ceiling above the floor,
- *obstacle*: an obstacle the robot has to avoid,
- *unknown*: unclassified terrain.

and h is the associated height value.

Fig. 8 shows an outline of our map building and environment classification system. First, 3D data (x_i^C, y_i^C, z_i^C) in sensor coordinates C are segmented into planes using our approach from Section 3. Kinematic information provides the system with pose information of the sensor relative to the robot coordinate system R. We filter the segmented planes by only looking at horizontal ones and obtain labeled range data $(x_i^R, y_i^R, z_i^R, h_i^R)$ in robot coordinates where the additional value h_i^R is the height of the flat horizontal plane the 3D point (x_i^R, y_i^R, z_i^R) belongs to, or $-\infty$ if the point was not segmented into such a plane. Basically h_i^R tells whether the robot could potentially step on this point ($h_i^R \neq -\infty$), or if it refers to an obstacle ($h_i^R = -\infty$). Updating OG and $Floor$ is then performed by following the steps described below.

5.1 Robot Localization

The first step is to transform the labeled range data into grid coordinates. This can be achieved by maintaining the robot pose $\mathbf{p}_R^G = (x_R^G, y_R^G, z_R^G, \theta_R^G)$ in the grid coordinate system G. In our system we are using foot step odometry for updating the robot pose and avoid revisiting places after significant odometry error has been accumulated. This can be achieved by centering the grid maps on the robot location and erasing areas far from the robot, e.g. by only maintaining and updating a window around the current position [Sabe *et al.*, 2004]. After transforming to grid coordinates we obtain labeled point data $\mathbf{p}_i^G = (x_i^G, y_i^G, z_i^G, h_i^G)$, where $h_i^G = h_i^R + z_R^G$. Note that $h_i^G = -\infty$, if $h_i^R = -\infty$.

5.2 3D Occupancy Grid

The 3D occupancy grid OG is updated by ray tracing. Given the robot pose $\mathbf{p}_R^G$ and relative camera pose $\mathbf{p}_C^R$, we compute the camera pose $\mathbf{p}_C^G$ in grid coordinates. Then, for each labeled data point $\mathbf{p}_i^G$, we draw a straight line from $\mathbf{p}_C^G$ to (x_i^G, y_i^G, z_i^G) and update all cells along this line according to a sensor model of the range data. Typically this updates the cells between sensor and data point as being empty and cells at the data point as being occupied. We employ the Bayesian approach for updating the probability $P(c) = OG(x, y, z)$ of each grid cell c along the line according to:

$$P(c) \leftarrow P(c \mid \mathbf{p}_i^G) = \frac{P(\mathbf{p}_i^G \mid c)P(c)}{P(\mathbf{p}_i^G \mid c)P(c) + P(\mathbf{p}_i^G \mid \neg c)(1 - P(c))} \quad (18)$$

where $P(\mathbf{p}_i^G \mid c)$ and $P(\mathbf{p}_i^G \mid \neg c)$ are the sensor model and define the likelihood of the measurement $\mathbf{p}_i^G$ given the cell is occupied or empty respectively.

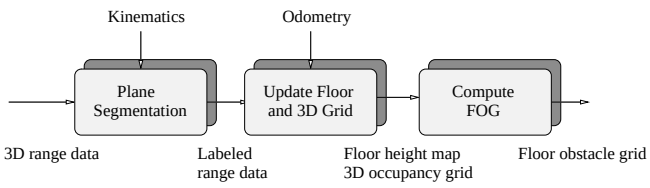

Figure 8: System of floor and obstacle map generation.

5.3 2.5D Floor Height Map

For obtaining reliable floor height information robust to sensor noise, we verify the floor height map and segmented planes by examining the corresponding cells in the 3D occupancy grid. We require that a cell has accumulated enough probability before floor height information can be considered reliable. Formally, floor height h at position (x, y) is considered reliable if the following inequality holds:

$$OG(x, y, h) > P_{occ} \quad (19)$$

where $P_{occ} > 0.5$ is a sufficient large threshold.

Since cells in the occupancy grid might have dropped below this threshold after updating the 3D grid, we need to verify the state of the existing floor map in the updated area. Each cell $Floor(x, y)$ is recomputed as:

$$Floor(x, y) \leftarrow \begin{cases} Floor(x, y), & \text{if } P_{Floor}(x, y) > P_{occ} \\ -\infty, & \text{otherwise} \end{cases} \quad (20)$$

where $P_{Floor}(x, y) = OG(x, y, Floor(x, y))$. This verification of the floor map has the advantage that it exploits the 3D structure of the range sensor encoded in the 3D grid.

We are now ready to insert the plane segmented sensor data into the height map. For each labeled data point $(x_i^G, y_i^G, z_i^G, h_i^G)$, if $OG(x_i^G, y_i^G, h_i^G) > P_{occ}$ then the floor grid is updated as:

$$Floor(x_i^G, y_i^G) \leftarrow \begin{cases} Floor(x_i^G, y_i^G), & \text{if } h_i^G < Floor(x_i^G, y_i^G) - \varepsilon \\ h_i^G, & \text{if } h_i^G > Floor(x_i^G, y_i^G) + \varepsilon \\ \nu Floor(x_i^G, y_i^G) + (1 - \nu)h_i^G, & \text{otherwise} \end{cases} \quad (21)$$

where ε is a small gate threshold for testing if floor map and measurement refer to the same floor surface, and $\nu \in [0, 1]$ is a smoothing factor that filters readings over time. This method only keeps an estimate of the highest floor surface in situations with multiple floor heights at the same position. The smoothing is possible since we know that existing floor height and measurement are both reliable.

5.4 Floor and Obstacle Grid (FOG)

The final step in our system is to compute the FOG representation. In order to determine the environment type of each cell, we compute the following quantities from the 3D occupancy grid and floor height map:

Obstacle detection: The highest occupied cell above a given floor height h_f defines the obstacle height at (x, y):

$$Obstacle(x, y, h_f) = \max\{z > h_f \mid OG(x, y, z) > P_{occ}\} \quad (22)$$

If no cells exceed P_{occ} then $Obstacle(x, y, h_f) = -\infty$.

Ceiling detection: Likewise, the lowest occupied cell above h_f defines the height of the ceiling at (x, y):

$$Ceiling(x, y, h_f) = \min\{z > h_f \mid OG(x, y, z) > P_{occ}\} \quad (23)$$

If no cells exceed P_{occ} then $Ceiling(x, y, h_f) = -\infty$.

Floor change detection: In a neighborhood $U(x, y)$, the floor change $\Delta h(x, y)$ is the maximum height difference to the floor height $h_f = Floor(x, y)$:

$$\Delta h(x, y) = \max\{|h_f - Floor(x', y')| \mid (x', y') \in U(x, y), Floor(x', y') \neq -\infty\} \quad (24)$$

From these quantities, the cells are classified using the following algorithm:

Algorithm *classify*(x, y)
Input: cell indices x, y
Output: type t of environment cell
Sequence:
  $f := Floor(x, y)$
  $o := Obstacle(x, y, f)$
  if $o > f$ **then**
    if $f = -\infty$ **return** *obstacle*
    $c := Ceiling(x, y, f)$
    if $c > f + h_{robot}$ **return** *floor*
    if $c > f + h_{tunnel}$ **return** *tunnel*
    return *obstacle*
  else
    if $f = -\infty$ **return** *unknown*
    $\delta := \Delta h(x, y)$
    if $\delta < h_{floor}$ **return** *floor*
    if $\delta < h_{stairs}$ **return** *stairs*
    return *border*
  endif

Here, h_{robot} is the robot height, h_{tunnel} the minimum ceiling height for crawling underneath, h_{floor} the maximum change in floor height the robot can still walk on ordinarily, and h_{stairs} the maximum stair height the robot can climb.

Last but not least, the height associated to an environment type t with floor height $h_f = Floor(x, y)$ is computed as:

$$height(x, y) = \begin{cases} -\infty, & \text{if } t = \textit{unknown} \\ Obstacle(x, y, h_f), & \text{if } t = \textit{obstacle} \\ h_f, & \text{otherwise.} \end{cases} \quad (25)$$

This completes the construction of the FOG map.

5.5 Simulated example

Fig. 9 illustrates our approach with an example. A simulated world containing 3 different floor levels and 2 obstacles exhibiting a rough surface at their top is fully observed by a perfect sensor. The lower right obstacle resembles a part of a table showing a large unoccupied volume between floor and obstacle (Fig. 9(a)). The floor height map captures the precise floor height but is unable to represent any obstacles since the data from the rough surfaces could not be segmented into planes. Therefore, areas corresponding to the obstacles contain a value of *inv* = $-\infty$ except where floor below the obstacle was detected (Fig. 9(b)). The 3D occupancy grid (Fig. 9(c)) contains a complete representation of the scene but the floor height information is only coarse. The top of the obstacles show the same flat surface as the floor levels. Our combination (Fig. 9(d)) shows the classification of the environment cells with their associated height. Floor is drawn in white, yellow (light gray) and green (gray), stairs are marked with crosses, border is filled in solid brown (dark gray), tunnel is marked by + signs, and obstacles are black. Note that obstacle heights are coarse while all other heights are precise.

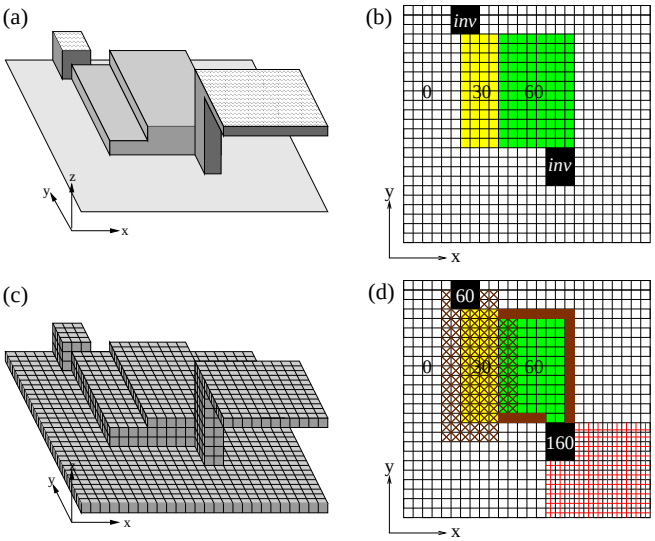

Figure 9: Sample result in a simulated environment. (a) World fully sensed by system. (b) Floor height map. (c) 3D occupancy grid. (d) Floor obstacle grid (FOG).

6 Results on QRIO

We implemented our approach on QRIO, Sony's small humanoid robot with 38 degrees of freedom and powered by 3 MIPS R5000 CPU's clocked at 400 MHz [Fujita *et al.*, 2003]. The robot's hardware and stabilization control allow it to walk ordinarily on regular and irregular terrain with a maximum change in floor height of about h_{floor} = 1.5 cm. For stair climbing, the maximum step height is h_{stairs} = 5 cm. When crawling, the robot needs at least h_{tunnel} = 30 cm vertical space. When standing, the robot is h_{robot} = 58 cm tall.

A stereo camera module with a 45° field of view mounted in the robot's head provides disparity images (176x144) at 12.5 fps [Sabe *et al.*, 2004]. We utilize sub-sampled 88x72 images which allows to process all data in real-time on the robot. The disparities are first transformed into 3D measurements and then processed by our plane segmentation algorithm. Fig. 10 shows the left image of a sample image pair and the segmentation achieved by our method.

The grid maps are sized 100x100 cells with a 4 cm side length. Additionally, the 3D grid uses 16 layers (each 4 cm) in the z direction. The maps are centered on the robot such that they reflect a 4x4 m area around the robot. As the robot moves, the grids are translated in x and y according to the odometry information reported by the kinematic module.

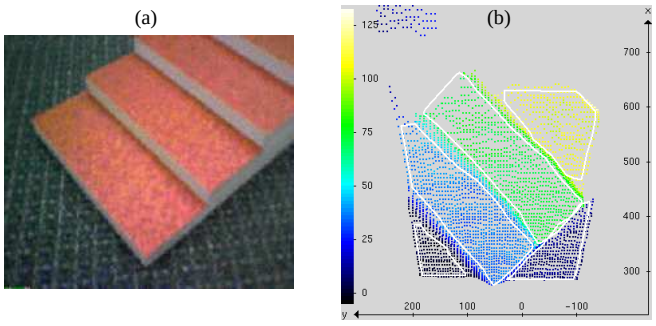

Figure 10: Plane segmentation on QRIO. (a) Camera image. (b) Segmented range data in robot coordinates.

Cells at the grid border are erased.

For testing our perception technology, we set up an obstacle course containing obstacles of different size and height, a staircase for stepping up and down, and a table for crawling underneath. Fig. 11 shows the scenario with the robot in the start position at the far end. After the robot obtained an initial model of its close surroundings by looking around, we utilized a *path navigator* [Gutmann *et al.*, 2005b] for finding and following a path to a goal location about 4 meters ahead (at the other side of the table).

Snapshots of the generated environment models as well as the paths found by the navigator are depicted in Figs. 12 to 15. In each figure, floor is drawn in different colors (gray scales) showing the floor height according to the indicator bar on the right. Obstacles are marked in black with a cross indicating the coarse height. Brown (dark gray) crosses mark stairs whereas solid brown (dark gray) cells are border. A tunnel is displayed using + signs with red (gray) ones indicating the actual tunnel and black ones an enlarged area.

In the beginning the robot navigates around two obstacles (Fig. 12). Our perception system estimates the height of the upper left obstacle to be at most 8 cm while the lower right one is at least 20 cm and therefore the robot chooses a path with a slightly larger distance to the taller obstacle.

After passing the obstacles, the robot detects the staircase with stair heights of about 3 and 6 cm (Fig. 13). Note

Figure 11: QRIO walks around obstacles, stairs and a table.

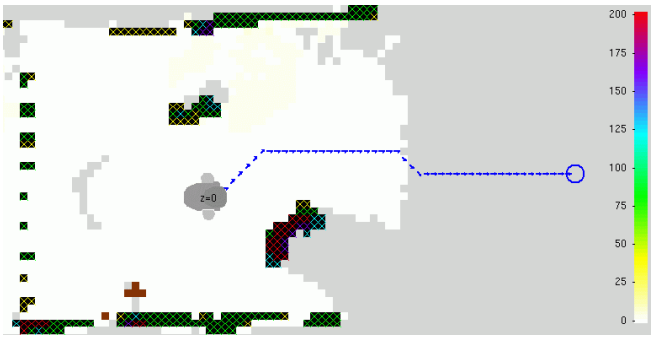

Figure 12: QRIO walks around two obstacles.

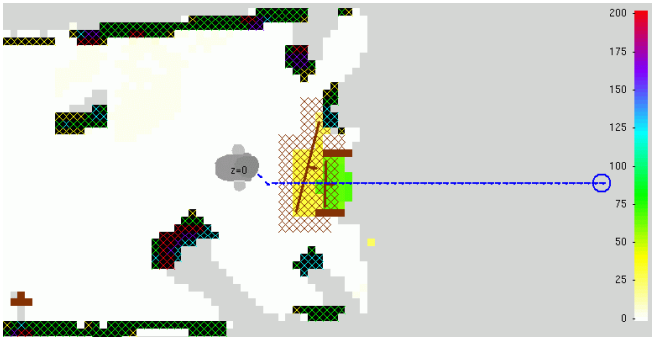

Figure 13: QRIO detects stairs area and starts climbing.

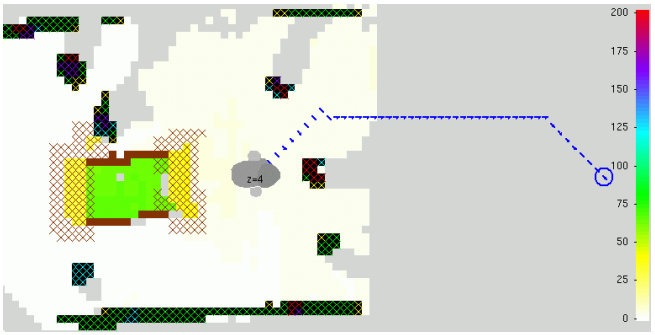

Figure 14: In narrow area the robot walks sideways.

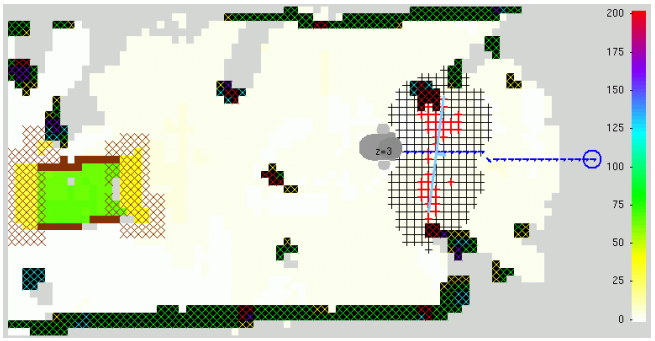

Figure 15: The table is classified as a tunnel.

that the robot has not yet seen the end of the staircase and thus the classification of this area is unknown. For climbing over the stairs the robot utilizes a specifically developed stair-climbing module that directly utilizes extracted plane information [Gutmann *et al.*, 2004a]. After the robot reached the top of the stairs, new observations refine the classification of cells into stairs and the robot again uses the stair-climbing module for descending to the ground.

After walking up and down the staircase, the robot finds three obstacles, two of them are about 20 cm and one about 8 cm in height (Fig. 14). The robot passes this area by stepping sideways between the two taller obstacles.

At the last stage, the robot moves towards the table and detects it as a tunnel (Fig. 15). From here, the robot switches to a crawling motion for reaching the goal.

A similar run of our system can be viewed in the video provided as Extension 1.

7 Conclusion

We presented a 3D perception system for the navigation of a humanoid robot. Range data obtained by a stereo camera is segmented into planes and integrated into a floor height and 3D occupancy grid from which in turn environment cells are classified and associated with a height value.

For plane extraction we employed the method of scan line grouping and improved it for segmenting stereo range data. Our improvements analyze the distribution of points along each scan line and compute statistics reflecting the noise locally present in the data. The statistics are then used for region growing where a notion of residual error is used for deciding whether a line can be merged into a plane. Our experiments show that we obtain a richer segmentation than the original method on images exhibiting a high dynamic in noise. In a further comparison with a state-of-the-art approach using RANSAC, EM, and patchlets [Murray and Little, 2004; Murray, 2003], our method outperforms the competitor in the number of reported segments, segmentation accuracy and run-time for computing results. Our method accesses each data point only during line extraction and when trading points in the post processing steps. Therefore, the algorithm is very efficient and can be employed in real-time systems with reasonably large image sizes.

In our environment map building approach we are able to distinguish between 6 different environment types. Precise height information is provided for floor, stairs and tunnel classifications, whereas coarse height is available for obstacles. In an experiment, we showed how the humanoid robot QRIO uses this approach for the navigation on an obstacle course containing obstacles, stairs and a table.

The maps generated by our approach can be employed in a variety of existing path planning systems that have been developed in the past for simulated 2.5D worlds [Shiller *et al.*, 2001; Chestnutt *et al.*, 2003; Li *et al.*, 2003] and for real robot systems [Gutmann *et al.*, 2005c]. Thus, our approach fills in the gap between path-planning in simulation and real robots by providing detailed floor and obstacle maps of real world environments.

There are several limitations in our approach. Our method relies on dense range data, i.e. the environment contains enough texture for obtaining reliable disparity data. Including a matching score, that is typically obtained during stereo image matching, for estimating lines and planes in a weighted least-squares version of our algorithm would allow to segment data with varying texture quality. However, surfaces not containing any texture still have to be addressed using a different approach.

Our approach only considers horizontal planes for navigation. While it is possible to extend the representation to include inclined surfaces and estimate roll and pitch of planes, it is unclear how well such an estimation would perform in practice since the additional two degrees of freedom are subject to incremental estimation errors.

Furthermore, considering only one floor level at a location can be too restrictive, e.g. when there is a staircase leading up to a stage where the robot could choose to either walk up the stairs or crawl underneath the stage. Integrating the work of [Triebel *et al.*, 2006] would be a possible way to improve our representation for multiple planes.

Our system also relies on a good localization of the robot. Techniques from visual SLAM, e.g. [Karlsson *et al.*, 2005; Davison, 2003] could be applied to improve position estimates. Such an integration is behind the scope of this paper.

Finally, our environment classification can only distinguish between six different environment types and these are predesigned and set into stone in our *classify* algorithm. A generative model where environment types are learned by the robot would be desirable. Future work addresses these issues.

Acknowledgment

We would like to thank Don Murray from the University of British Columbia in Vancouver, Canada for providing us with his experimental data. We also thank all developers of the QRIO robot in Sony for making this work possible.

References

[Chestnutt *et al.*, 2003] Joel Chestnutt, James Kuffner, Koichi Nishiwaki, and Satoshi Kagami. Planning biped navigation strategies in complex environments. In *Int. Conf. on Humanoid Robotics (Humanoids)*, 2003.

[Davison, 2003] A.J. Davison. Real-time simultaneous localization and mapping with a single camera. In *Int. Conf. on Computer Vision (ICCV)*, 2003.

[Duda and Hart, 1973] R.O. Duda and P.E. Hart. *Pattern classification and scene analysis*. Wiley, New York, 1973.

[Fitzgibbon and Fisher, 1994] Andrew W. Fitzgibbon and Robert B. Fisher. Lack-of-fit detection using the run-distribution test. In *European Conf. on Computer Vision (ECCV)*, 1994.

[Fujita *et al.*, 2003] M. Fujita, Y. Kuroki, T. Ishida, and T.T. Doi. A small humanoid robot SDR-4X for entertainment applications. In *Int. Conf. on Advanced Intelligent Mechatronics (AIM)*, Kobe, Japan, 2003.

[Gutmann *et al.*, 2004a] J.-S. Gutmann, M. Fukuchi, and M. Fujita. Stair climbing for humanoid robots using stereo

vision. In *Int. Conf. on Intelligent Robots and Systems (IROS)*, Sendai, Japan, 2004.

[Gutmann *et al.*, 2004b] J.-S. Gutmann, K. Kawamoto, K. Sabe, and M. Fukuchi. Multiple plane segmentation with the application of stair climbing for humanoid robots. In *Symposium on Intelligent Autonomous Vehicles (IAV)*, Lisbon, Portugal, 2004.

[Gutmann *et al.*, 2005a] J.-S. Gutmann, M. Fukuchi, and M. Fujita. A floor and obstacle height map for 3d navigation of a humanoid robot. In *Int. Conf. on Robotics and Automation (ICRA)*, Barcelona, Spain, 2005.

[Gutmann *et al.*, 2005b] J.-S. Gutmann, M. Fukuchi, and M. Fujita. A modular architecture for humanoid robot navigation. In *Int. Conf. on Humanoid Robotics (Humanoids)*, Tsukuba, Japan, 2005.

[Gutmann *et al.*, 2005c] J.-S. Gutmann, M. Fukuchi, and M. Fujita. Real-time path planning for humanoid robot navigation. In *Int. Joint Conference on Artificial Intelligence (IJCAI)*, Edinburgh, Scotland, 2005.

[Hähnel *et al.*, 2002] D. Hähnel, W. Burgard, and S. Thrun. Learning compact 3D models of indoor and outdoor environments with a mobile robot. *Robotics and Autonomous Systems*, 2002.

[Hoover *et al.*, 1996] Adam Hoover, Gillian Jean-Baptiste, Xiaoyi Jiang, Patrick J. Flynn, Horst Bunke, Dmitry B. Goldgof, Kevin K. Bowyer, David W. Eggert, Andrew W. Fitzgibbon, and Robert B. Fisher. An experimental comparison of range image segmentation algorithms. *Transactions on Pattern Analysis and Machine Intelligence*, 18(7):673 – 689, 1996.

[Iocchi *et al.*, 2000] L. Iocchi, K. Konolige, and M. Bajracharya. Visually realistic mapping of a planar environment with stereo. In *Int. Symposium on Experimental Robotics (ISER)*, Hawaii, 2000.

[Jiang and Bunke, 1994] X.-Y. Jiang and H. Bunke. Fast segmentation of range images into planar regions by scan line grouping. *Machine Vision and Applications*, 7(2):115 – 122, 1994.

[Kagami *et al.*, 2003] Satoshi Kagami, Koichi Nishiwaki, James J. Kuffner, Kei Okada, Masayuki Inaba, and Hirochika Inoue. Vision-based 2.5D terrain modeling for humanoid locomotion. In *Int. Conf. on Robotics and Automation (ICRA)*, pages 2141–2146, Taipei, Taiwan, September 2003.

[Kanehiro *et al.*, 2005] F. Kanehiro, T. Yoshimi, S. Kajita, M. Morisawa K. Fujiwara, K. Harada, K. Kaneko, H. Hirukawa, and F. Tomita. Whole body locomotion planning of humanoid robots based on a 3D grid map. In *Int. Conf. on Robotics and Automation (ICRA)*, Barcelona, Spain, 2005.

[Karlsson *et al.*, 2005] N. Karlsson, N. di Bernardo, E. Ostrowski, J. Goncalves, P. Pirjanian, and M.E. Munich. The vSLAM algorithm for robust localization and mapping. In *Int. Conf. on Robotics and Automation (ICRA)*, Barcelona, Spain, 2005.

[Li *et al.*, 2003] T.-Y. Li, P.-F. Chen, and P.-Z. Huang. Motion planning for humanoid walking in a layered environment. In *Int. Conf. on Robotics and Automation (ICRA)*, Taipei, Taiwan, 2003.

[Lorch *et al.*, 2002] O. Lorch, A. Albert, J. Denk, M. Gerecke, R. Cupec, J. F. Seara, W. Gerth, and G. Schmidt. Experiments in vision-guided biped walking. In *Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 2484–2490, Lausanne, Switzerland, October 2002.

[Lu, 1995] Feng Lu. *Shape Registration using Optimization for Mobile Robot Navigation*. PhD thesis, University of Toronto, 1995.

[Murray and Little, 2004] D. Murray and J.J. Little. Environment modeling with stereo vision. In *Int. Conf. on Intelligent Robots and Systems (IROS)*, Sendai, Japan, 2004.

[Murray, 2003] Donald Murray. *Patchlets: a method for interpreting correlation stereo 3D data*. PhD thesis, University of British Columbia, Vancouver, 2003.

[Nüchter *et al.*, 2003] Andreas Nüchter, Hartmut Surmann, and Joachim Hertzberg. Automatic model refinement for 3D reconstruction with mobile robots. In *Int. Conf. on Recent Advances in 3D Digital Imaging and Modeling (3DIM)*, pages 394 – 401, Banff, Canada, October 2003.

[Okada *et al.*, 2001] Kei Okada, Satoshi Kagami, Masayuki Inaba, and Hirochika Inoue. Plane segment finder: Algorithm, implementation, and applications. In *Int. Conf. on Robotics and Automation (ICRA)*, pages 2120–2125, Seoul, Korea, May 2001.

[Sabe *et al.*, 2004] K. Sabe, M. Fukuchi, J.-S. Gutmann, T. Ohashi, K. Kawamoto, and T. Yoshigahara. Obstacle avoidance and path planning for humanoid robots using stereo vision. In *Int. Conf. on Robotics and Automation (ICRA)*, New Orleans, 2004.

[Shiller *et al.*, 2001] Zvi Shiller, Katsu Yamane, and Yoshihiko Nakamura. Planning motion patterns of human figures using a multi-layered grid and the dynamics filter. In *Int. Conf. on Robotics and Automation (ICRA)*, Seoul, Korea, May 2001.

[Thrun *et al.*, 2004] S. Thrun, C. Martin, Y. Liu, D. Hähnel, R. Emery-Montemerlo, D. Chakrabarti, and W. Burgard. A real-time expectation maximization algorithm for acquiring multi-planar maps of indoor environments with mobile robots. *Transactions on Robotics and Automation*, 20(3):433 – 442, 2004.

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

[Vidal *et al.*, 2003] R. Vidal, Y. Ma, and S. Sastry. Generalized principal component analysis (GPCA). In *Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2003.

[Weingarten *et al.*, 2004] J.W. Weingarten, G. Gruener, and R. Siegwart. Probabilistic plane fitting in 3D and an application to robotic mapping. In *Int. Conf. on Intelligent Robots and Systems (IROS)*, Sendai, Japan, 2004.