3D Perception and Environment Map Generation for Humanoid Robot Navigation
Jens-Steffen Gutmann, Masaki Fukuchi and Masahiro Fujita
The International Journal of Robotics Research 2008 27: 1117
DOI: 10.1177/0278364908096316

The online version of this article can be found at:
http://ijr.sagepub.com/content/27/10/1117
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 three-dimensional (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 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 a previously developed patchlet method, 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 object height information. 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.

KEY WORDS—humanoid robot navigation, 3D environment perception, range image segmentation, stereo vision

1. Introduction

Humanoid robots possess 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 within 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 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 three-dimensional (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 of 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 yet been sensed (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 provides a method of segmenting im-
images exhibiting a high noise dynamic as noise is analyzed in a local neighborhood during line extraction. 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 random sample consensus (RANSAC), expectation maximization (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. 2004) and in two conference papers (Gutmann et al. 2005a,b). While this paper integrates the contributions of the previous work, 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 (Quest for cuRIOsity), 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. As well as its usefulness for mobile robot navigation, its applications include the generation of geometric scene descriptions (Jiang and Bunke 1994; Hoover et al. 1996; Murray 2003), or the building of 3D models from raw sensor range data (locchi et al. 2000; Hahnel et al. 2003; Nuchter 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 interest 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 significantly 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 quadratically with 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 depends not only on distance but also on other factors such as available texture.

Recent trends in segmentation are to move away from ad hoc methods such as region growing to methods that are mathematically more sound. Approaches using the Hough transformation (locchi et al. 2000; Okada et al. 2001), RANSAC (Nüchter et al. 2003), EM (Thrun et al. 2004; Lakaemper and Latecki 2006) and the generalized principal component analysis (GPCA) (Vidal et al. 2005) 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 substantially 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.

3. Improvements to Scan-line Grouping

A flowchart of our extended scan-line grouping method is depicted in Figure 1. Before presenting the details about each subsystem we introduce some mathematical tools that allow us to compute plane parameters in an efficient way.

3.1. Fitting Planes to Range Data

Let \( p_i = (x_i, y_i, z_i)^T, i = 1, \ldots, N \) 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:

\[
x n_x + y n_y + z n_z - d = 0.
\]

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

\[
\text{fit}(\mathbf{n}, d) = \sum_{i=1}^{N} (\mathbf{p}_i^T \mathbf{n} - d)^2.
\]

There are several approaches for finding the optimal parameters that minimize Equation (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 the origin of the coordinate system then observed planes cannot pass through the origin due to visibility constraints, thus \( d \neq 0 \). This allows us to optimize the error function divided by \( d^2 \) and by substituting \( \mathbf{m} = \mathbf{n}/d \) we obtain:

\[
\frac{\text{fit}(\mathbf{n}, d)}{d^2} = \sum_{i=1}^{N} (\mathbf{p}_i^T \mathbf{m} - 1)^2.
\]

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

\[
A \mathbf{m} = \mathbf{b}, \quad \text{with}
\]

\[
A = \sum_{i=1}^{N} \mathbf{p}_i \mathbf{p}_i^T, \quad \mathbf{b} = \sum_{i=1}^{N} \mathbf{p}_i,
\]

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

\[
\mathbf{n} = \frac{\mathbf{m}}{\|\mathbf{m}\|}, \quad d = \frac{1}{\|\mathbf{m}\|}
\]

where \( \|\mathbf{m}\| \) denotes the length of \( \mathbf{m} \). Although Equations (4)–(6) do not minimize the original least-squares problem (Equation (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 \( \mathbf{A} \) 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 \( \mathbf{A} \) and \( \mathbf{b} \) is achieved by:

\[
\mathbf{A} \leftarrow \mathbf{A} + N \mathbf{E}(\mathbf{p}\mathbf{p}^T), \quad \mathbf{b} \leftarrow \mathbf{b} + N \mathbf{E}(\mathbf{p}).
\]

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

\[
\text{RMS}(\mathbf{p}_1, \ldots, \mathbf{p}_N) = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (\mathbf{p}_i^T \mathbf{n} - d)^2}
\]

which can again be computed from the first two moments:

\[
\text{RMS}(N, \mathbf{b}, A) = \sqrt{\frac{1}{N} (\mathbf{n}^T A \mathbf{n} - 2d/\mathbf{n}^T \mathbf{b}) + d^2}.
\]

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 \( \sigma \) 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

\[
\frac{\text{RMS}(N, \mathbf{b}, A, \sigma)}{\sigma} = \frac{\text{RMS}(N, \mathbf{b}, A)}{\sigma}
\]

of the RMS residual of a set of points given by their statistics \( N, \mathbf{b} \) and \( A \) and the plane error \( \sigma \), 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 a plane of standard deviation \( \sigma \). We will make use of this distance metric in the region-growing process.

### 3.2. Line Extraction

For line extraction, the data points \( \mathbf{p}_i, i = 1, \ldots, N \) 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 (see algorithm 1) of Duda and Hart (1973).

Here, \( N_{\text{min}} \) is the minimum number of points for defining a line segment and \( dist(p_i, p_1, p_N) \) computes the distance of \( p_i \) to the closest point on the line passing through \( p_1 \) and \( p_N \). If this distance is below a threshold \( d_{\text{cord}} \) for all points then a line segment (defined by \( p_1 \) and \( p_N \)) 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, Figure 2 contains two datasets, each along a scan line to which a line has been fitted 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 \( L \) which returns the maximum number of consecutive points that all fall on the same side of the line. If the points are randomly distributed along the line (e.g. in Figure 2(b)) then this number should be small. Thus, if \( C \) falls below a threshold \( C_{\text{max}} \) then the line is reported. Otherwise, a large \( C \) indicates that the point set can be refined (e.g. the set of points in Figure 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 \( C_{\text{max}} \) 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 \( P(C > C_{\text{max}}) \) increases with the number of points. Thus, for large \( N \), a line that actually fits well to the dataset might occasionally also be split. For a more detailed analysis, we refer to the run-distribution test and its variations as presented by Fitzgibbon and Fisher (1994).

Each line \( L \) found by the rd-split algorithm is recorded as

\[
L = (s, b, e, N, b, A, \sigma)
\]

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

### Algorithm 1 split\((p_1, \ldots, p_N)\)

**Input:** Set of points \( p_1, \ldots, p_N \) along a scan line  
**Output:** Set of line segments approximating point curve  
**Sequence:**

\[
\text{if } N < N_{\text{min}} \text{ return } \emptyset \\
\text{k := arg max}_{i=2, \ldots, (N-1)} \text{dist}(p_i, p_1, p_N) \\
\text{if dist}(p_k, p_1, p_N) < d_{\text{cord}} \text{ then} \\
\quad \text{return } \{(p_1, p_N)\} \\
\text{else } \\
\quad \text{return } \text{split}(p_1, \ldots, p_k) \cup \text{split}(p_{k+1}, \ldots, p_N) \\
\text{endif}
\]

### Algorithm 2 rd-split\((p_1, \ldots, p_N)\)

**Input:** Set of points \( p_1, \ldots, p_N \) along a scan line  
**Output:** Set of line segments approximating point curve  
**Sequence:**

\[
\text{if } N < N_{\text{min}} \text{ return } \emptyset \\
L := \text{least-squares-fit}(p_1, \ldots, p_N) \\
C := \text{max-count-consecutive-same-side}(L, p_1, \ldots, p_N) \\
\text{if } C < C_{\text{max}} \text{ then} \\
\quad \text{return } \{L\} \\
\text{else } \\
\quad k := \text{arg max}_{i=2, \ldots, (N-1)} \text{dist}(p_i, p_1, p_N) \\
\quad \text{return } \text{rd-split}(p_1, \ldots, p_k) \cup \text{rd-split}(p_{k+1}, \ldots, p_N) \\
\text{endif}
\]

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

![Fig. 2. Line fitting to (a) precise and (b) distorted data.](image-url)
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 $N_{\max}$ 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 $L_i = (s_i, b_i, e_i, N_i, b_i, A_i, \sigma_i)$, $i = 1, 2, 3$ be three 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

$$b = b_1 + b_2 + b_3, \quad A = A_1 + A_2 + A_3,$$  
(12)

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

$$r_\sigma(N_i, b_i, A_i, \sigma_i) < r_{\text{seed}}$$  
(13)

where $r_{\text{seed}}$ is a preset threshold. Note that the standard deviation $\sigma_i$ of a line is treated here as the plane error. In principle, the plane error could be directly computed by Equation (9). However, the different line segments might exhibit different levels of noise which is ignored when computing a global plane error over all three lines. We therefore use the individual line errors as each $\sigma_i$ is an upper bound to the error of a plane containing the line (the error reaches $\sigma_i$ 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 Equation (9).

### 3.4. Region Growing

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

$$r_\sigma(N_i, b_i, A_i, \sigma_i) < r_{\text{grow}}$$  
(14)

then $L_i$ is moved into the open list, the equation system of the plane is updated according to Equation (7) and parameters $n$ and $d$ are re-computed by Equation (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.

As with the selection of the seed region, we use $\sigma_i$ as an estimate for the plane error rather than the global RMS residual over all data points. The line error $\sigma_i$ 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. A summary of the steps involved in our implementation is as follows.

**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 fits better to a neighboring plane based on the RMS residual of Equation (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 fit better to a neighboring plane, the plane borders are refined. For this, we compute the statistics $N$, $b$ 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.

**Optimizing planes:** Points at the border of a plane usually contain more noise than inner pixels. By removing border points from $b$ and $A$ of a plane’s equation system and recomputing the plane parameters $n$ and $d$, a slightly improved estimate can be achieved.

The final output of the scan-line grouping algorithm is a set of plane regions $\{P_i\}$ where each region is defined as $P_i = (n_i, d_i, N_i, b_i, A_i, B_i)$. Here $n_i$ 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, $b_i$ and $A_i$ are the first two moments times $N_i$, and $B_i$ 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 $B_i$ 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 datasets and compared the results to the original UB (University of Bern) method (Jiang and Bunke 1994) and a recently published, different method using so-called patchlets (Murray 2003; Murray and Little 2004). The first dataset contains range images from an ABW scanner (507 × 512 pixels) which measures range using structured light of a Digiclops trinocular stereo camera (285 × 205 pixels). These images were previously used in a comparison of four different range segmentation methods (Hoover et al. 1996). The second dataset are disparity data generated from images of a Digiclops trinocular stereo camera (285 × 205 pixels) which measures range using structured light (http://marathon.csee.usf.edu/range/DataBase.html). These images were previously used in a comparison of four different range segmentation methods (Hoover et al. 1996).

Our results were obtained with the parameters listed in Table 1. For the ABW images, we set $C_{\text{max}} = 10$ which gave slightly better results than the setting $C_{\text{max}} = 15$ 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 $C_{\text{max}}$ for splitting a scan line into a larger number of segments.

Our results on the ABW image dataset are similar to the original scan-line grouping method. Figure 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, figure 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.

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 $d_{\text{line}}$ 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. Figure 4(a) shows an image of a room made of planar walls with bands connecting walls and floor. The disparity data (Figure 4(b)) shows a large range of values which indicates a high dynamic in range error. When setting $d_{\text{cond}} = 15$ mm and $d_{\text{line}} = 30$ mm (corresponding to $T_1$ and $T_2$ in Hoover et al. 1996) the UB method produces the segmentation shown in Figure 4(c). The result of our method is shown in Figure 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 $d_{\text{cond}}$ and $d_{\text{line}}$ 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.

Figures 5(a)–(d) contain results on images of a staircase. 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. For example, 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 a (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 (2003) and Murray and Little (2004).

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 con-

<table>
<thead>
<tr>
<th>$N_{\text{min}}$</th>
<th>$C_{\text{max}}$</th>
<th>$r_{\text{seed}}$</th>
<th>$r_{\text{grow}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>10/15</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

*Fig. 3. Segmentation on ABW test image #8.*
Fig. 4. Stereo range data of a room. The distance to the near wall is about 2 m and the distance to the end of the room is about 6 m: (a) camera image; (b) disparity image; (c) result of UB method; (d) our result.

Fig. 5. Stereo range data of a staircase. For the eight labeled surfaces ground truth information is available: (a) camera image; (b) disparity image; (c) result of UB scan-line grouping; (d) our result.
crete 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.

The images of Figures 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 2003; Murray and Little 2004). 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 (Figure 4), patchlets report 5 regions (see Murray 2003, figure 7.27) compared to 7 in our method. For the staircase (Figure 5), patchlets compute 8 regions (see Murray 2003, figure 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 Figure 5, ground truth data of 8 surfaces has been made available. The accuracy of segmentation reported by Murray (2003, table 7.3) and the result obtained by our method are summarized in Table 2.

As the table suggests, our method generally produces a more accurate segmentation than the patchlet method. However, this result cannot be taken too literally. 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 which are farther away. For example, if the 3771 pixel large segment surrounding surface 8 in Figure 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 Figure 6 (Murray 2003, figure 7.36) we obtain an overall accuracy of 86.1% (compared to 85.8% of the patchlet method). For the 6 surfaces of Figure 7 (Murray 2003, figure 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 runtime in post-processing for the stereo images are the result of using a larger window size ($9 \times 9$) than for the ABW images ($3 \times 3$). 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.
Fig. 7. Close-up view of the staircase: (a) camera image; (b) disparity image; (c) segmentation result.

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

<table>
<thead>
<tr>
<th>Image</th>
<th>ABW # 8</th>
<th>Room</th>
<th>Stairs/side</th>
<th>Stairs/front</th>
<th>Stairs/close</th>
<th>Stairs/QRI0</th>
<th>Stairs/QRI0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 3</td>
<td>257</td>
<td>285</td>
<td>285</td>
<td>285</td>
<td>285</td>
<td>88</td>
<td>88</td>
</tr>
<tr>
<td>Figure 4</td>
<td>512</td>
<td>205</td>
<td>205</td>
<td>205</td>
<td>205</td>
<td>88</td>
<td>88</td>
</tr>
<tr>
<td>Size</td>
<td>507 × 512</td>
<td>285 × 205</td>
<td>285 × 205</td>
<td>285 × 205</td>
<td>285 × 205</td>
<td>88 × 72</td>
<td>88 × 72</td>
</tr>
<tr>
<td>Processor</td>
<td>Pentium 4</td>
<td>Pentium 4</td>
<td>Pentium 4</td>
<td>Pentium 4</td>
<td>Pentium 4</td>
<td>Pentium 4</td>
<td>MIPS R5000</td>
</tr>
<tr>
<td>Clock</td>
<td>3 GHz</td>
<td>3 GHz</td>
<td>3 GHz</td>
<td>3 GHz</td>
<td>3 GHz</td>
<td>3 GHz</td>
<td>400 MHz</td>
</tr>
<tr>
<td>Line extraction</td>
<td>23.5</td>
<td>9.4</td>
<td>9.3</td>
<td>6.0</td>
<td>3.9</td>
<td>0.4</td>
<td>6.2</td>
</tr>
<tr>
<td>Region growing</td>
<td>5.9</td>
<td>2.8</td>
<td>3.1</td>
<td>1.8</td>
<td>0.8</td>
<td>0.1</td>
<td>1.8</td>
</tr>
<tr>
<td>Line trading</td>
<td>1.4</td>
<td>1.7</td>
<td>3.2</td>
<td>1.0</td>
<td>0.1</td>
<td>0.0</td>
<td>1.2</td>
</tr>
<tr>
<td>Point trading</td>
<td>7.5</td>
<td>15.7</td>
<td>33.1</td>
<td>17.8</td>
<td>14.7</td>
<td>0.1</td>
<td>0.3</td>
</tr>
<tr>
<td>Plane merging</td>
<td>0.1</td>
<td>1.0</td>
<td>1.6</td>
<td>0.8</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Total</td>
<td>38.4</td>
<td>30.6</td>
<td>50.3</td>
<td>27.4</td>
<td>19.5</td>
<td>0.6</td>
<td>9.5</td>
</tr>
</tbody>
</table>

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. Michel et al. (2007) employ GPU-accelerated tracking of an a priori model of a staircase which is climbed by the HRP-2 humanoid robot. 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 not allow finding tables which the robot could crawl underneath, for example. 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 precise information about the floor height is required as well as about holes.

We represent the environment by a regular 2D 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. The floor is planar and horizontal (or 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 the representation of 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 $Floor$:

$$OG: (x, y, z) \mapsto p, \quad p \in [0, 1]$$

$$Floor: (x, y) \mapsto h, \quad h \in R \cup \{-\infty\}$$

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 the verification of 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:

$$FOG: (x, y) \mapsto (t, h)$$

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.

Figure 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 which 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 $h_i^R$ determines 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 $p_G = (x_G^R, y_G^R, z_G^R, t_G^R)$ in the grid coordinate system $G$. In our system we are using foot step odometry for updating the robot pose and to 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 $p_G^G = (x_i^G, y_i^G, z_i^G, h_i^G)$, where $h_i^G = h_i^R + z_i^G$. 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 $p_G^G$ and relative camera pose $p_C^G$, we compute the camera pose $p_C^G$ in grid coordinates. Then, for each labeled data point $p_G^G$, we draw a straight line from $p_C^G$ 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 to being empty and cells at the data point to 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:
where $\varepsilon$ is a small gate threshold for testing if floor map and measurement refer to the same floor surface and $v \in [0, 1]$ 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 (first two cases in equation). In the last case of the equation, the floor height is smoothed over time.

### 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 $(x, y)$:

$$\text{Obstacle}(x, y, h_f) = \max\{z > h_f \mid OG(x, y, z) > P_{\text{occ}}\}. \quad (24)$$

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

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

$$\text{Ceiling}(x, y, h_f) = \min\{z > h_f \mid OG(x, y, z) > P_{\text{occ}}\}. \quad (25)$$

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

**Floor change detection**: In the 8-neighborhood $U(x, y)$, the floor change $\Delta h(x, y)$ is the maximum height difference to the floor height $h_f = \text{Floor}(x, y)$:

$$\Delta h(x, y) = \max\{h_f - \text{Floor}(x', y') \mid (x', y') \in U(x, y), \text{Floor}(x', y') \neq -\infty\}. \quad (26)$$

From these quantities, the cells are classified using the classify logic shown in Algorithm 3.

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

Finally, the height associated to an environment type $t$ with floor height $h_f = \text{Floor}(x, y)$ is computed as:

$$\text{height}(x, y) = \begin{cases} 
-\infty, & \text{if } t = \text{unknown} \\
\text{Obstacle}(x, y, h_f), & \text{if } t = \text{obstacle} \\
h_f, & \text{otherwise.}
\end{cases} \quad (27)$$

This completes the construction of the FOG map.
Fig. 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).

**Algorithm 3 classify**(x, y)

**Input:** cell indices x, y  
**Output:** type t of environment cell

**Sequence:**

\[ f := \text{Floor}(x, y) \]
\[ o := \text{Obstacle}(x, y, f) \]

**if** o > f **then**

\[ \text{if } f = -\infty \text{ return } \text{obstacle} \]
\[ c := \text{Ceiling}(x, y, f) \]
\[ \text{if } c > f + h_{\text{robot}} \text{ return } \text{floor} \]
\[ \text{if } c > f + h_{\text{tunnel}} \text{ return } \text{tunnel} \]

{return } \text{obstacle} 

**else**

\[ \text{if } f = -\infty \text{ return } \text{unknown} \]
\[ \delta := \Delta h(x, y) \]
\[ \text{if } \delta < h_{\text{floor}} \text{ return } \text{floor} \]
\[ \text{if } \delta < h_{\text{stairs}} \text{ return } \text{stairs} \]

{return } \text{border} 

**endif**

### 5.5. Simulated Example

Figure 9 illustrates our approach with an example. A simulated world containing three different floor levels and two 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 (Figure 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 \( \text{inv} = -\infty \) except where floor below the obstacle was detected (Figure 9(b)). The 3D occupancy grid (Figure 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 (Figure 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.
6. Results on QRI0

We implemented our approach on QRI0, Sony’s small humanoid robot with 38 degrees of freedom and powered by three MIPS R5000 CPUs 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 $h_{\text{floor}} = 1.5$ cm. For stair climbing, the maximum step height is $h_{\text{stairs}} = 5$ cm. When crawling, the robot needs at least $h_{\text{tunnel}} = 30$ cm vertical space. When standing, the robot is $h_{\text{robot}} = 58$ cm tall.

A stereo camera module with a 45° field of view mounted in the robot’s head provides disparity images (176×1534) at 12.5 frames per second (fps) (Sabe et al. 2004). We utilize subsampled 88×72 images which allow the processing of all data in real-time on the robot. The disparities are first transformed into 3D measurements and then processed by our plane segmentation algorithm. Figure 10 shows the left image of a sample image pair and the segmentation achieved by our method.

The grid maps are sized 100×100 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 4×4 m area around the robot. (Note that the numbers used here were found by experimentation and are a trade-off between accuracy, area coverage, data size and computation time. More investigations are needed in order to tune these parameters systematically.)

As the robot moves, the grids are translated in $x$ and $y$ according to the odometry information reported by the kinematic module. 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. Figure 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 m 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 Figures 12–15. In each figure, floor is drawn in different colors (grayscale) 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) signs indicating the actual tunnel and black signs an enlarged area.

Initially, the robot navigates around two obstacles (Figure 12). Our perception system estimates the height of the upper left obstacle to be at most 8 cm while the lower right obstacle 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 cm and 6 cm (Figure 13). Note 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. 2004). 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 another about 8 cm in height (Figure 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 (Figure 15). From here, the robot switches to a crawling motion for reaching the goal.
Fig. 11. QRIO walks around obstacles, stairs and a table.

Fig. 12. QRIO walks around two obstacles.

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

7. Conclusions

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 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. Since the early work of Jiang and Bunke, other variants of scan-line grouping have been presented (Checchin et al. 1997; Haindl and Zid 1998) providing similar or better results than the original approach. However, both methods also use fixed thresholds during segmentation and it is therefore unclear how they perform on stereo range data.

In a further comparison with a state-of-the-art approach using RANSAC, EM, and patchlets (Murray 2003; Murray and Little 2004), 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. The algorithm is therefore very efficient and can be employed in real-time systems with reasonably large image sizes. We believe that in order to further improve segmentation results, techniques like those found in robust statistics and estimators could be employed with the drawback of increased processing times due to their nonlinear nature.

In our environment map-building approach we are able to distinguish between six 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 experiment indicates a reasonably accurate terrain classification. When regarding obstacles and floor only, results are comparable to previous methods (Kagami et al. 2003; Sabe et al. 2004). However, more
experiments are needed in order to fully evaluate the classification accuracy of our approach.

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). Our approach therefore fills 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 to our approach. Our method relies on dense range data, i.e. the environment contains enough texture for obtaining reliable disparity data. By including a matching score (typically obtained during stereo image matching) for estimating lines and planes in a weighted least-squares version of our algorithm, data of varying texture quality could be segmented. However, surfaces not containing any texture still have to be addressed using a different approach.

Although plane segmentation returns planes with arbitrary orientation, we only consider horizontal planes for navigation. (Restricting the segmentation process to horizontal planes could further improve results but requires camera orientation to be known with good accuracy.) We believe these are most important for the walking of a humanoid robot. 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. Davison 2003; Karlsson et al. 2005) could be applied to improve position estimates. There are also promising results for detecting larger structures in the feature map of visual SLAM (Gee et al. 2007). Such an integration is behind the scope of this paper.

Finally, our environment classification can only distinguish between six different environment types which have been pre-designed in our classify algorithm. A generative model where environment types are learned by the robot would be desirable. Future work addresses these issues.

Acknowledgments

We would like to thank Don Murray of 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.

Appendix: Index to Multimedia Extensions

The multimedia extensions to this article can be found at http://www.ijrr.org.

<table>
<thead>
<tr>
<th>Extension</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Video</td>
<td>QRIO walking on obstacle course</td>
</tr>
</tbody>
</table>

References

Conference on Recent Advances in 3-D Digital Imaging and Modeling (3DIM), pp. 156–163.


Gee, A., Chekhlov, D., Mayol-Cuevas, W., and Calway, A. (2007). Discovering planes and collapsing the state space in visual SLAM. In British Machine Vision Conference (BMVC), University of Warwick, UK.


Nüchter, A., Surmann, H., and Hertzberg, J. (2003). Automatic model refinement for 3D reconstruction with mobile ro-


