Auton Robot (2007) 23: 97–111
DOI 10.1007/s10514-007-9034-y

A comparison of line extraction algorithms using 2D range data for indoor mobile robotics

Viet Nguyen · Stefan Gächter · Agostino Martinelli · Nicola Tomatis · Roland Siegwart

Received: 21 September 2006 / Revised: 14 March 2007 / Accepted: 28 March 2007 / Published online: 8 June 2007

Abstract This paper presents an experimental evaluation of different line extraction algorithms applied to 2D laser scans for indoor environments. Six popular algorithms in mobile robotics and computer vision are selected and tested. Real scan data collected from two office environments by using different platforms are used in the experiments in order to evaluate the algorithms. Several comparison criteria are proposed and discussed to highlight the advantages and drawbacks of each algorithm, including speed, complexity, correctness and precision. The results of the algorithms are compared with ground truth using standard statistical methods. An extended case study is performed to further evaluate the algorithms in a SLAM application.

Keywords Line extraction algorithm · 2D range data · Mobile robotics

V. Nguyen (✉) · S. Gächter · A. Martinelli · R. Siegwart
Autonomous Systems Laboratory (ASL), Swiss Federal Institute of Technology Zürich (ETHZ), Tannenstrasse 3, 8092 Zurich, Switzerland
e-mail: viet.nguyen@mavt.ethz.ch

S. Gächter
e-mail: stefan.gaechter@mavt.ethz.ch

A. Martinelli
e-mail: agostino.martinelli@mavt.ethz.ch

R. Siegwart
e-mail: rsiegwart@ethz.ch

N. Tomatis
BlueBotics SA, PSE-C, 1015 Lausanne, Switzerland
e-mail: nicola.tomatis@bluebotics.com

1 Introduction

It is important for a mobile robot to be able to autonomously navigate and localize itself in a known or unknown environment. A precise position estimation always serves as the heart in any navigation systems, such as localization, map building or path planning. It is well known that pure dead-reckoning methods, e.g. odometry, are prone to errors that grow unbounded over time (Iyengar and Elfes 1991). The problem gives rise to a variety of solutions using different exteroceptive sensors (sonar, infrared, laser, vision, etc.). Among different types of sensors, 2D laser range finders are becoming increasingly popular in mobile robotics. For example, laser scanners have been used in localization (Cox 1991; Jensfelt and Christensen 1998), dynamic map building (Lu and Milios 1994; Gutmann et al. 1998; Castellanos and Tardós 1996; Tomatis et al. 2003) or feature tracking (e.g. for collision avoidance) (Pears 2000). There are many advantages of laser scanners compared to other sensors: they provide dense and more accurate range measurements, they have high sampling rates, high angular resolution, good range distance and resolution.

One primary issue is how to accurately match sensed data against information in a priori map or information that has been continuously collected. There are two common matching techniques that have been used in mobile robotics: point-based matching and feature-based matching. The early work of Cox (1991) uses range data to help the robot localizing itself in a small polygonal environment. He proposes a matching algorithm between point images and target models in a priori map using an iterative least-squares minimization method. Another work done by Lu and Milios (1994) addresses the problem of self-localization in an unknown environment, not necessarily polygonal. The proposed approach is to approximate the alignment of two consecutive scans

and then iteratively improve the alignment by defining and minimizing some distance between the scans.

Instead of working directly with raw measurement points, feature-based matching first transforms the raw scans into geometric features. The extracted features are then used for matching in the next step. This approach has been studied and employed intensively in recent research on feature extraction, feature tracking, robot localization, dynamic map building, etc. (Crowley 1985; Leonard et al. 1992; Castellanos and Tardós 1996; Gutmann and Schlegel 1996; Jensfelt and Christensen 1998; Schröter et al. 2002; Brunskill and Roy 2005). Being more compact, they require much less memory and still provide rich and accurate information. Algorithms based on parameterized geometric features are therefore more efficient compared to point-based algorithms.

Among many geometric primitives, line segments are the simplest ones. It is easy to describe most office environment using line segments. Many algorithms have been proposed for mobile robotics using line features extracted from 2D range data. Castellanos and Tardós (1996) propose a line segmentation method inspired from an algorithm in computer vision, used with a priori map as an approach to robot localization. Vandorpe et al. (1996) introduce a dynamic map building algorithm based on geometrical features, e.g. lines and circles, using a laser scanner. Arras and Siegwart (1997) use a 2D scan segmentation method based on line regression in map-based localization. Jensfelt and Christensen (1998) present a technique for acquisition and tracking of the pose of a mobile robot in an office environment with a laser scanner by extracting orthogonal lines (walls). Pfister et al. (2003) suggest a line extraction algorithm using weighted line fitting for line-based map building. Finally, Brunskill and Roy (2005) propose an incremental probabilistic technique to extract segments for solving the SLAM problem.

Many work has been done on line extraction. However, there is a lack of comprehensive comparisons of the so far proposed algorithms. Selecting the best method to extract lines from scan data is the first task for anyone who is going to build a line-based navigation system using 2D laser scanners. In terms of speed, one would prefer the fastest algorithm for his real time application. In terms of quality, bad line extraction can seriously lead the system to divergence in line-based SLAM. Implementation complexity is also one of the main aspects to take into account.

The work done by Gutmann and Schlegel (1996) gives a brief comparison of three algorithms which are relatively out of date compared to ones found in recent research. Moreover, the uncertainty modeling of the used parameters is not introduced in their paper. Borges and Aldon (2004) present an extended version of split-and-merge and compare their method with a generic split-and-merge algorithm and a line tracking (incremental) algorithm. Sack and Burgard (2004) perform a comparison of three selected algorithms on range data. However, in both papers the evaluation results are indirectly observed and interpreted from the map built by the mapping process. A direct comparison of the extracted lines by the selected algorithms using probabilistic analysis is still missing.

This paper presents a throughout evaluation of six line extraction algorithms applied to laser range scans. This work is an extension of the one described in (Nguyen et al. 2005). The six selected algorithms are the most commonly used in mobile robotics and computer vision. Several comparison criteria are proposed and discussed, including speed, complexity, correctness and precision. The experiments are performed on two datasets collected from the Autonomous Systems Laboratory—EPFL, Switzerland and the Intel Jones Farms Campus, Oregon with the environment size of 80 m × 50 m and 40 m × 40 m, respectively. The results of the algorithms are compared with ground truth using standard statistical methods. One case study is performed to evaluate the maps obtained from a SLAM application when using different line extraction algorithms. Finally, the conclusions are presented.

2 Problem statement

A range scan describes a 2D slice of the environment. Points of a range scan are specified in polar coordinate system (ρ_i, θ_i) whose origin is the location of the sensor (or the robot location offset by the mounting distance). It is common to assume that the noise on range measurement follows a Gaussian distribution with zero mean and variance $\sigma^2_{\rho_i}$. We neglect the small angular uncertainty for the ease of computing the covariance matrix of line parameters (see Arras and Siegwart 1997 for more derivation detail). Note that in this work, we focus on the performance of the algorithms. We do not consider systematic errors as they mainly depend on a specific hardware and testing environment (Diosi and Kleeman 2003a, 2003b).

We choose the polar form to represent a line model:

$$x \cos\alpha + y \sin\alpha = r$$

where $-\pi < \alpha \leq \pi$ is the angle between the x axis and the normal of the line; $r \geq 0$ is the perpendicular distance of the line to the origin; (x, y) is the Cartesian coordinates of a point on the line.

The covariance matrix of line parameters is:

$$cov(r, \alpha) = \begin{bmatrix} \sigma_r^2 & \sigma_{r\alpha} \\ \sigma_{r\alpha} & \sigma_\alpha^2 \end{bmatrix}.$$

There are three main problems in line extraction in an unknown environment (Forsyth and Ponce 2003). They are:

- How many lines are there?
- Which points belong to which line?
- Given the points that belong to a line, how to estimate the line model parameters?

In implementing the algorithms, we try to use as many common routines as possible, so that the experimental results reflect mainly the differences of the algorithmic concepts. Particularly for the third problem, we use a common fitting method, called *total-least-squares*, for all the algorithms since it has been used extensively in the literature (Arras and Siegwart 1997; Jensfelt and Christensen 1998; Einsele 1997; Lu and Milios 1994; Siadat et al. 1997). Hence, the algorithms differ only in solving the first two problems. Regarding the parameter settings, we define 2 sets of parameters. The first set consists of parameters of input data (e.g. number of points per scan), the sensor model (e.g. sensor uncertainties), desired output (e.g. minimal length of a line segment, maximal uncertainty) and parameters for the common routines (e.g. clustering, merging functions). These parameters are set to the same values for all the algorithms. The second set consists of specific parameter values for individual algorithm procedure. These parameters are chosen individually for each algorithm based on experimental tunning so that the best performance is obtained among several runs.

Notice that with high frequency of the up-to-date laser range finders (up to 75 Hz for a SICK LMS 200), we assume that the scans are obtained in batches (half or full scanning cycle). We make also the assumption that the effect of the robot motion on individual scan points (in one batch) is negligible. This however holds only when the robot translation and rotation speeds are small, e.g. few meters per second. For outdoor navigation, one might have to revise this assumption.

3 Selected algorithms and related work

This section briefly presents the concepts of the six selected line extraction algorithms. Our selection is based on their performance and popularity in both mobile robotics and computer vision. Only basic versions of the algorithms are summarized even though the details may vary in different implementations and applications. Interested reader should refer to the indicated references for more details. Our implementation follows closely the pseudo-code described below when not stated otherwise.

3.1 Split-and-merge algorithm

Split-and-Merge is probably the most popular line extraction algorithm which originates from computer vision in 1974 by Pavlidis and Horowitz (1974). It has been studied and used in many robotic research (Castellanos and Tardós 1996; Einsele 1997; Siadat et al. 1997; Borges and Aldon 2004; Zhang and Ghosh 2000).

Algorithm 1: *Split-and-Merge*

1 Initial: set s_1 consists of N points. Put s_1 in a list $\mathcal{L}$
2 Fit a line to the next set s_i in $\mathcal{L}$
3 Detect point P with maximum distance d_P to the line
4 If d_P is less than a threshold, continue (go to 2)
5 Otherwise, split s_i at P into s_{i1} and s_{i2}, replace s_i in $\mathcal{L}$ by s_{i1} and s_{i2}, continue (go to 2)
6 When all sets (segments) in $\mathcal{L}$ have been checked, merge collinear segments.

In the implementation, we make a small modification to line 3 so that we search for a splitting position where two adjacent points P_1 and P_2 are at the same side to the line and both have distances to the line greater than the threshold (if only 1 such point is found, it is ignored as a noisy point). Intuitively, we should split at a point that has locally maximal distance to the line. Notice that in line 2, we use a least-squares method for line fitting.

One can implement the algorithm differently so that a line is constructed simply by connecting the first and the last points. In this case, the algorithm is called *Iterative-End-Point-Fit* as in (Duda and Hart 1973; Siadat et al. 1997; Borges and Aldon 2004; Zhang and Ghosh 2000).

3.2 Incremental algorithm

Simplicity is the main advantage of this algorithm. It has been used in many applications (Forsyth and Ponce 2003; Vandorpe et al. 1996; Taylor and Probert 1996) and is also known as *Line-Tracking* (Siadat et al. 1997).

Algorithm 2: *Incremental*

1 Start by the first 2 points, construct a line
2 Add the next point to the current line model
3 Recompute the line parameters
4 If it satisfies a condition, continue (go to 2)
5 Otherwise, put back the last point, recompute the line parameters, return the line
6 Continue with the next 2 points, go to 2

In the implementation, we add 5 measurement points at each step (line 2) to speed up the incremental process. When the line does not satisfy a predefined line condition (variances of line parameters are less than some thresholds), the last 5 points are put back and the algorithm is switched back to the normal mode (adding one measurement point at a time). Again, we use a total-least-squares method for line fitting (line 3, 5).

The incremental scheme has been used in the algorithms proposed in (Adams and Kerstens 1998) and (Pears 2000),

however they use the EKF to estimate the parameters of extracted lines which is equivalent to using the total-least-squares fitting in this paper.

3.3 Hough transform algorithm

Hough Transform tends to be most successfully applied to line finding in intensity images (Forsyth and Ponce 2003). It has been brought in to robotics for extracting line from scan images (Forsberg et al. 1995; Jensfelt and Christensen 1998; Iocchi and Nardi 2002; Pfister et al. 2003).

Algorithm 3: *Hough-Transform*

1 Initial: A set of N points
2 Initialize the accumulator array
3 Construct values for the array
4 Choose the element with max. votes $V_{\max}$
5 If $V_{\max}$ is less than a threshold, terminate
6 Otherwise, determine the inliers
7 Fit a line through the inliers and store the line
8 Remove the inliers from the set, goto 2

There are some drawbacks with *Hough Transform*:

- It is usually difficult to choose an appropriate grid size for the accumulator array.
- Basic *Hough Transform* algorithm does not take noise and uncertainty into account when estimating the line parameters.

In this implementation, we use a resolution of 1 cm and 0.4° as the grid size of range and angle, respectively. To overcome the second problem, we use a *total-least-squares* method for line fitting (line 7).

Several variants of the Hough Transform have been proposed in order to improve the performance of line extraction, such as randomized Hough Transform (Xu et al. 1990), range-weighted Hough Transform (Forsberg et al. 1995), Log-Hough Transform (Alempijevic 2004).

3.4 Line regression algorithm

This algorithm has been proposed in (Arras and Siegwart 1997) for map-based localization. The key idea is inspired from the Hough Transform algorithm so that the algorithm first transforms the line extraction problem into a search problem in model space (line parameter domain) and then applies the *Agglomerative Hierarchical Clustering (AHC)* algorithm to construct adjacent line segments. One drawback of this algorithm is that it is quite complex to implement.

Algorithm 4: *Line-Regression*

1 Initialize sliding window size N_f
2 Fit a line to every N_f consecutive points (a window)
3 Compute a line fidelity array, each is the sum of Mahalanobis distances between every 3 adjacent windows
4 Construct line segments by scanning the fidelity array for consecutive elements having values less than a threshold, using an AHC algorithm
5 Merge overlapped line segments and recompute line parameters for each segment

The optimal sliding window size N_f depends on environment and has great influence on the algorithm performance. For our benchmark, $N_f = 7$ is used. A *total-least-squares* fitting method is used in line 2.

3.5 RANSAC algorithm

RANSAC—Random Sample Consensus (Fischler and Bolles 1981) is an algorithm for robust fitting of models in the presence of data outliers. The main advantage of *RANSAC* is that it is a generic segmentation method and can be used with many types of features once we have the feature model. It is also simple to implement. This algorithm is very popular in computer vision to extract features (Forsyth and Ponce 2003). Again, the same fitting method is used in line 4, 7.

Algorithm 5: *RANSAC*

1 Initial: A set of N points
2 **repeat**
3 Choose a sample of 2 points uniformly at random
4 Fit a line through the 2 points
5 Compute the distances of other points to the line
6 Construct the inlier set
7 If there are enough inliers, recompute the line parameters, store the line, remove the inliers from the set
8 **until** *Max.N.Iterations reached or too few points left*

3.6 EM algorithm

This algorithm, *Expectation-Maximization (EM)*, is a probabilistic method and commonly used in missing variable problems. *EM* has been used as a line extraction tool in computer vision (Forsyth and Ponce 2003) and robotics (Liu et al. 2001; Pfister et al. 2003; Sack and Burgard 2004).

There are some drawbacks of *EM* algorithm:

- It can be trapped in local minima
- It is difficult to choose good initial values

Algorithm 6: *EM*

```
1  Initial: A set of N points
2  repeat
3     Randomly generate parameters for a line
4     Initialize weights for remaining points
5     repeat
6        E-Step: Compute the weights of the points from the
         line model
7        M-Step: Recompute the line model parameters
8     until Max.N.Steps reached or convergence
9  until Max.N.Trials reached or found a line
10 If found, store the line, remove the inliers, go to 2
11 Otherwise, terminate
```

3.7 Extra details

As already mentioned, we use the same total-least-squares method to compute the line parameters of a line and their covariance matrix once we have a set of inliers determined by the algorithms. This technique overcomes the well known bias problem of least-squares method where it tends to put more weight on noisy, outlying points (Forsyth and Ponce 2003). For details of the total-least-squares method, please refer to (Arras and Siegwart 1997).

We implement a simple clustering algorithm for filtering out largely noised measurement points and also coarsely dividing a raw scan into sub groups (clusters) of scan points. The algorithm works similarly to the *Successive Edge Following—SEF* algorithm (Siadat et al. 1997). Briefly, it scans the raw scan points in sequence coming from the sensors for big jumps in radial differences between consecutive points. If a radial difference is greater than a threshold, the algorithm breaks the scan sequence at this point into two sub groups. Thus, the scan is segmented into clusters of contiguous points. Clusters having too few number of points are removed. To be safe and to avoid over segmentation, we use very conservative values for the thresholds. One can improve this clustering algorithm by considering the case of over segmentation when a line segment contains one noisy point in the middle (e.g. due to noise or occlusions). In this case, the two split segments can be merged afterward (described in the following paragraph). However the risk is that if one or both of the sub segments are too small, they could be removed before reaching the merging step. Fortunately, this problem rarely occurs in our experiments.

Due to occlusions, a line may be observed and extracted as several segments. Localization algorithms usually use line parameters (r, α) in position estimation (Vandorpe et al. 1996; Arras and Siegwart 1997). Thus, it might be a good idea to merge collinear line segments into one line segment. It results in a longer, hence more reliable segment, reducing the number of lines to process and still containing the same information. Therefore, we implement a merging routine that is applied at the output of each algorithm, after segments have been extracted. The routine uses a standard statistical method, called *Chi-Square* test (Snedecor and Cochran 1989), to compute the Mahalanobis distance between each pair of line segments based on the already computed covariance matrices of line parameters. If two line segments have a statistical distance less than a threshold, they are merged. The new line parameters are recomputed from the raw scan points that constitute the two segments.

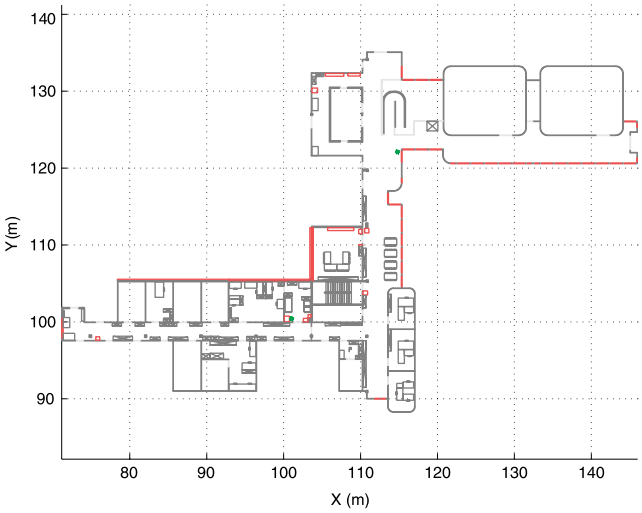

Fig. 1 The ground map of the Autonomous System Laboratory (ASL) with a size of 80 m × 50 m. This map contains only main building parts (e.g. walls, doors, windows) and stationary office furniture (e.g. cupboards, tables)

4 Experimental comparison

4.1 The experiment setup

To evaluate the performance of the algorithms, we select two laser scan datasets taken from two different environments: the hallway of the Autonomous Systems Laboratory (ASL)—EPFL, Switzerland and a part of the Intel Jones Farms Campus, Oregon. The second dataset is obtained from the Radish repository (Howard and Roy 2003).

The ground map of the ASL hallway is shown in Fig. 1. The environment is polygonal with a map size of 80 m × 50 m. The hallway contains many walls, doors and cupboards that are good targets for line extraction. There are also table legs, chair legs and glass windows. For collecting the ASL dataset, we use the mobile base of the robot RoboX (Siegwart et al. 2003) which is equipped with two laser sensors (see Fig. 2). The robot is running a real-time operating system (RTAI Linux) with an embedded obstacle avoidance system and a remote control module via wireless network. We let the robot navigate through the environment while the

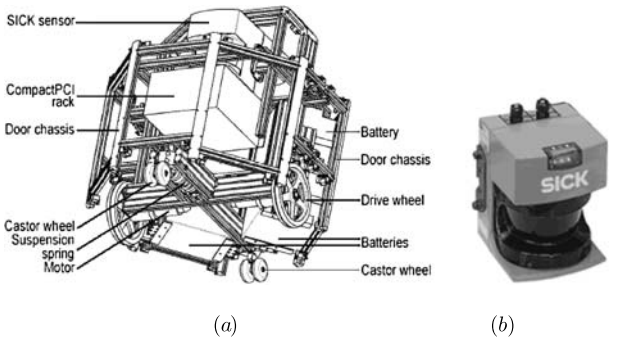

Fig. 2 **a** The mobile base of the RoboX. The two SICK sensors are placed back-to-back (one is hidden in the figure). **b** A laser range finder SICK LMS291-S05

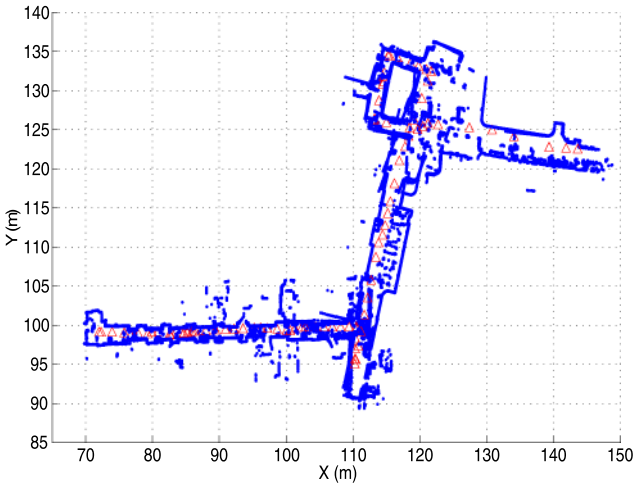

Fig. 3 The ASL hallway map obtained by using the odometry alone and the selected 100 raw scans. The red triangles represent the robot positions at which the scans are taken

direction and speed are being remotely controlled. The experiment is carried out during the working hours so that the robot observes people regularly walking nearby.

The laser sensors are two laser range finders SICK-LMS 291-S05. Each sensor has a maximum measurement range of 80 m, a range resolution of 10 mm and a statistical error standard deviation of 10 mm at normal reflectivity condition. Each sensor is able to scan an angle of 0°–180° with selectable angular resolutions of 0.25°, 0.50° or 1.00°. The maximum sampling frequency is 75 Hz. For technical specification detail, please refer to the *Technical Information LMS 200/291. SICK, Inc*. The combination of two SICK laser scanners enables the robot to scan a full 360°. In our experiment, we use a maximum scan range of 7.0 m, an angular resolution of 0.5° and a sampling rate of 3 Hz.

During the whole experiment, the robot makes 5122 observation steps. The benchmarking dataset consists of 100 scans selected every 50 observation steps. The hallway map accumulated by those 100 scans are shown in Fig. 3. Two

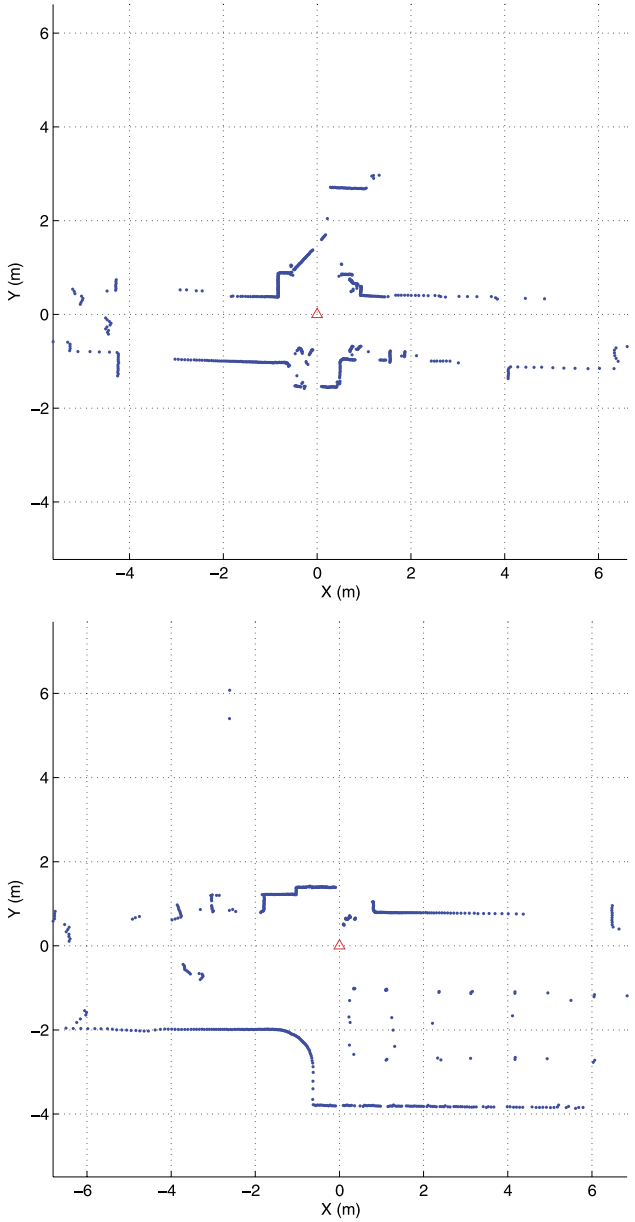

Fig. 4 Two samples from the selected raw scans of the ASL hallway. Generally, the scans are quite noisy due to the clutter and dynamics of the environment. Notice that in the *first figure*, short segments from the cupboards and several human legs are observed; in the *second figure*, the wall is observed as a broken line because of the occlusions of table legs

samples of the selected scans are shown in Fig. 4 to demonstrate the environment shape and surroundings.

In the second experiment, the scan data are taken in the Intel Jones Farms Campus, Oregon. The ground map is shown in Fig. 5. The environment is a typical office structure which has a map size of 40 m × 40 m. Again, we select 100 scans for our benchmark set among 8030 original scans (one scan is selected every 80 observation steps). Two samples of the selected scans are shown in Fig. 6. The robot

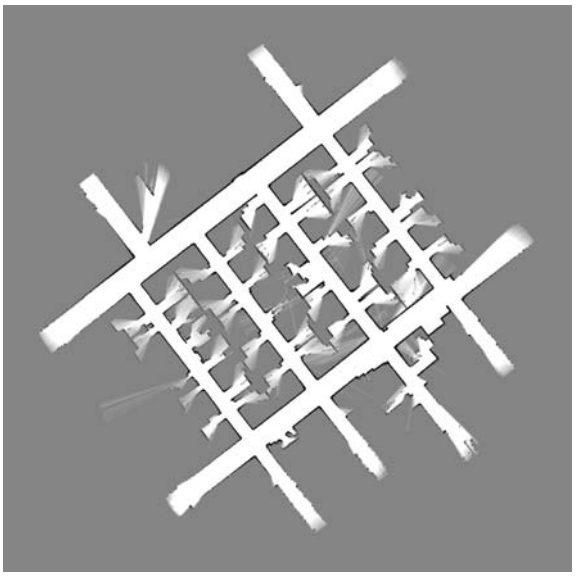

Fig. 5 The map of a part of the Intel Jones Farms Campus in Hillsboro, Oregon (source: the Radish repository http://radish.sourceforge.net/)

platform for this dataset was a Pioneer2DX (odometry) with a SICK LMS 200 (laser range finder). The SICK has a range resolution of 10 mm and a statistical error standard deviation of 5 mm at normal reflectivity condition. In this dataset, the maximum range is set to 7.0 m and the angular resolution to 1°. Thus, one SICK sensor gives 181 measurement points for each scan.

4.2 The algorithm implementation

The algorithms are programmed in C. The benchmarks are performed on a computer with one CPU PentiumIV-3.4 GHz and 2 GB of memory.

Choosing parameter values is an important task since algorithm performances are very sensitive to the values used. As already said above, we divide the parameters into two groups: one group of *common parameters* and one group of *algorithm specific parameters*. The common parameter group consists of parameters of input data, the sensor model, desired output and parameters for the common routines. These parameters are set to the same values for all the algorithms. The second set consists of specific parameter values for individual algorithm procedure. These parameters are chosen individually for each algorithm based on experimental tunning so that the best performance is obtained among several runs. Certainly to have a fair comparison, we want to use as many common parameters as possible. The following common parameters and their values are chosen according to the sensor hardware and the environment (the numbers in parentheses are used for the second dataset):

- *MinNumPoints* = 9 (7): Minimum number of scan points for a line segment.

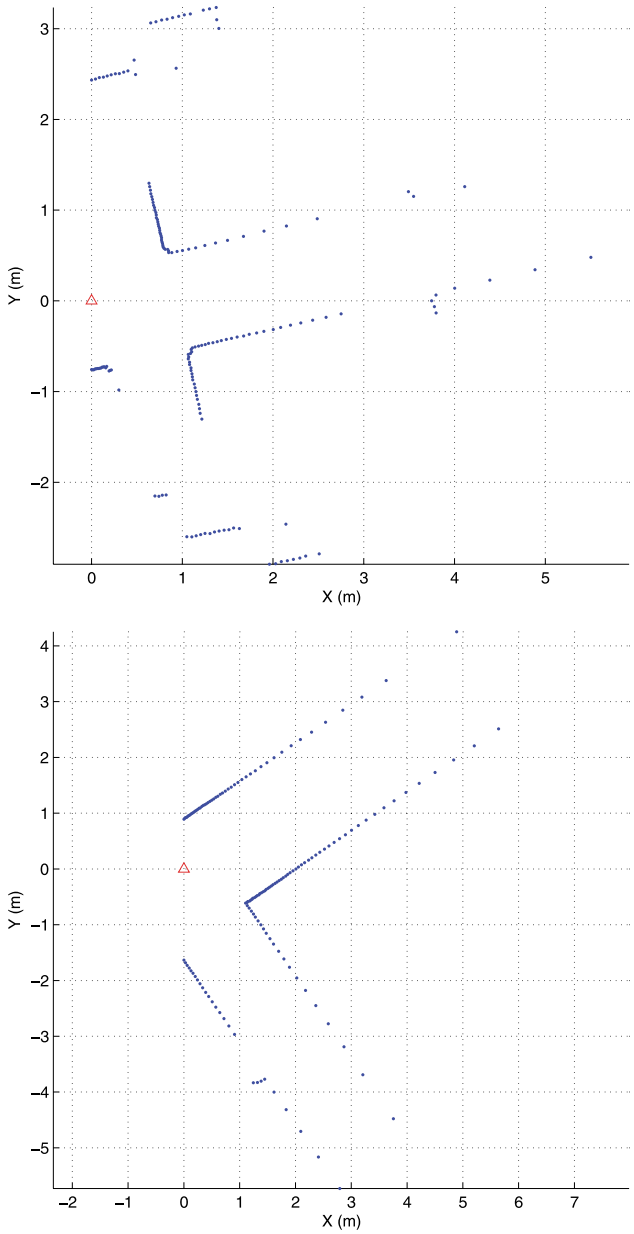

Fig. 6 Two samples from the selected raw scans of the Intel laboratory. Generally, the scans are clean and well-defined because of the good arrangement and good reflectivity of the surroundings. It makes line extraction easier

- *MinLength* = 0.4 m: Minimum physical length of an line segment.
- $\sigma_r = 0.015$ m (0.01 m): Standard deviation of range measurement uncertainty.
- *InlierThreshold* = 0.01 m: Maximum distance from a scan point to a line that the point is considered inlier to the line.
- *ValidGate* = 2.77: The threshold used in the merging routine (which corresponds to the 75% confidence interval).

Table 1 Experimental results of the ASL-hallway dataset

Algorithm	Complexity	Speed	N.Lines	Correctness		Precision	
				TruePos	FalsePos	$\sigma_{\Delta r}$	$\sigma_{\Delta\alpha}$
		[Hz]		[%]	[%]	[cm]	[deg]
Split-Merge + Clustering	$N \times \log N$	1780	614	83.9	7.2	1.76	0.69
Incremental	$S \times N^2$	469	536	76.0	3.7	1.68	0.64
Incremental + Clustering		772	549	77.6	4.0	1.70	0.74
Line Regression	$N \times N_f$	457	565	75.3	9.6	1.75	0.68
LR + Clustering		502	556	75.6	7.7	1.75	0.72
RANSAC	$S \times N \times N.Trials$	20	725	76.0	28.8	1.60	0.69
RANSAC + Clustering		91	523	70.0	9.2	1.24	0.57
Hough Transform	$S \times N \times NC + S \times NR \times NC$	6.6	368	84.1	36.0	1.55	0.68
HT + Clustering		7.6	463	80.6	12.5	1.51	0.67
EM	$S \times N1 \times N2 \times N$	0.2	893	74.4	43.4	1.86	0.83
EM + Clustering		0.2	646	77.5	18.6	1.46	0.72

The value *MinNumPoints* is used for the reason that the hallways in two cases are quite narrow, thus extracted line segments tend to have highly concentrated points. For the second dataset, since the angular resolution used is bigger (1°), this value is set to 7 as we have less scan points. We choose a quite big value for *MinLength* (0.4 m) to get rid of spurious scan points observed from moving people, e.g. human legs. The standard deviation of range measurement uncertainty is set to be

$$\sigma_r = \sigma_{\text{sensor}} + 0.005 \text{ m}$$

where the addition 0.005 m accounts for the reflectivity differences of surrounding objects. The parameter *InlierThreshold* is used in all the algorithms, mainly to decide where to start/stop a line segment and to determine the inliers of a line. The value of 0.01 m is experimentally selected since it gives the near-best performance for all the algorithms.

To determine the correctness and precision of the lines extracted by each algorithm, we define a set of "true lines" that contains manually extracted line segments of the selected scans. The values *MinNumPoints* and *MinLength* are taken into account during the manual extraction. All the true lines have the same statistical uncertainty:

$$\sigma_r^T = 0.03 \text{ m}, \qquad \sigma_\alpha^T = 0.03 \text{ rad}.$$

For the total of 100 selected scans, there are 679 true lines (≈7 lines/scan) and 412 true lines (≈4 lines/scan) for the first and second dataset, respectively. The extracted lines by the algorithms are then compared with the true lines to find the matched pairs using the Chi-Square test with a matching valid gate $MatchValidGate = 2.77$ (75% confidence interval).

4.3 The experimental results

In order to analyze the experimental results, four quantity measures are evaluated: complexity, speed, correctness and precision. The benchmark results are shown in Table 1 and Table 2. There are 11 algorithm candidates in which 6 of them are the selected algorithms combined with our simple clustering algorithm (shown as "+ Clustering"). The other 5 candidates are the basic versions of the corresponding algorithms. The terminology used in the tables is as follows (the values used are in parentheses):

- N: Number of points in an input scans (722 or 181)
- S: Number of line segments extracted by an algorithm
- N_f: Sliding window size for *Line-Regression* (7)
- $N.Trials$: Number of trials for *RANSAC* (1000)
- NR, NC: Number of rows, columns respectively for the HT accumulator array ($NR = 671$, $NC = 901$ for resolution $r_{\text{res}} = 1$ cm, $\alpha_{\text{res}} = 0.4°$)
- $N1$, $N2$: Number of trials and convergence iterations, respectively, for *EM* ($N1 = 50$, $N2 = 200$).

The common routines, e.g. clustering, total-least-squares line fitting and line merging, all have a complexity of N.

The correctness measures are defined as follows:

$$TruePos = \frac{N.Matches}{N.TrueLines}$$

$$FalsePos = \frac{N.LineExByAlgo - N.Matches}{N.LineExByAlgo}$$

where $N.LineExByAlgo$ is the number of lines extracted by an algorithm, $N.Matches$ is the number of matches and $N.TrueLines$ is the number of true lines.

Table 2 Experimental results of the Intel-Laboratory dataset

Algorithm	Complexity	Speed	N.Lines	Correctness		Precision	
				TruePos	FalsePos	$\sigma_{\Delta r}$	$\sigma_{\Delta\alpha}$
		[Hz]		[%]	[%]	[cm]	[deg]
Split-Merge + Clustering	$N \times \log N$	7353	379	90.8	1.3	0.69	0.31
Incremental	$S \times N^2$	2538	366	87.6	1.4	0.85	0.32
Incremental + Clustering		3322	368	88.1	1.4	0.72	0.31
Line Regression	$N \times N_f$	1751	329	77.4	3.0	0.68	0.37
LR + Clustering		1879	326	77.4	2.1	0.69	0.37
RANSAC	$S \times N \times N.Trials$	328	278	64.1	1.9	0.79	0.42
RANSAC + Clustering		470	265	63.1	1.9	0.51	0.30
Hough Transform	$S \times N \times NC + S \times NR \times NC$	27.3	471	86.9	24.0	1.10	0.5
HT + Clustering		22.2	435	87.1	17.5	0.78	0.37
EM	$S \times N1 \times N2 \times N$	0.8	341	73.1	11.7	1.16	0.60
EM + Clustering		1.1	325	74.8	5.2	0.67	0.36

To determine the precision, we define the following two sets of errors on line parameters:

$$\{\boldsymbol{\Delta r} : \Delta r_i = r_i - r_i^T,\ i = 1, \ldots, n\},$$
$$\{\boldsymbol{\Delta \alpha} : \Delta \alpha_i = \alpha_i - \alpha_i^T,\ i = 1, \ldots, n\}$$

where n is the number of matched pairs, r_i^T, α_i^T are line parameters of a true line, r_i, α_i are line parameters of the corresponding matched line (extracted by an algorithm). Here we make an assumption that the error distributions are Gaussian. The variances of the two distributions are computed as follows:

$$\overline{\Delta r} = \frac{1}{n}\sum \Delta r_i, \qquad \sigma_{\Delta r}^2 = \frac{1}{n-1}\sum (\Delta r_i - \overline{\Delta r})^2,$$
$$\overline{\Delta \alpha} = \frac{1}{n}\sum \Delta \alpha_i, \qquad \sigma_{\Delta \alpha}^2 = \frac{1}{n-1}\sum (\Delta \alpha_i - \overline{\Delta \alpha})^2$$

where n is approximately 400–600. (Notice that we use $\frac{1}{n-1}$ instead of $\frac{1}{n}$ for unbiased variances.)

For nondeterministic *RANSAC*-based and *EM*-based algorithm, the values shown are the average after 10 runs.

The results of the first benchmark are shown in Table 1. As seen in column 3, the first 5 algorithms, which are based on *Split-and-Merge*, *Incremental* and *Line-Regression*, perform much faster than the others. This is mainly because these 5 algorithms are based on deterministic methods and especially, they make use of the sequencing characteristic of the raw scan points. *Split-and-Merge* algorithm, being in the class of *divide-and-conquer* algorithms, takes the lead. The performance of 1780 Hz affirms with the algorithm complexity as being the fastest. Notice that with the clustering algorithm, *Incremental* performs at almost double speed.

In term of correctness, the *Incremental*-based algorithms seem to perform best since they have very low number of false positives, which is very important for SLAM. Being better in *TruePos*, *Split-and-Merge+Clustering* could be the best choice for localization with a priori map. Again, the algorithms based on *RANSAC* and *EM* perform poorly as they result in very high *FalsePos*. This can be explained by the fact that, since the algorithms do not use the sequencing property of the scan points, they often try to fit lines falsely across the scan map. This could be reduced by increasing the minimum number of points per line segment. However, short segments maybe left out.

In spite of bad speed and correctness, algorithms based on *RANSAC*, *HT* and *EM+Clustering* produce relatively more precise lines. One of the reasons is that these algorithms tend to include good inliers only, rather than to maximize number of points following the scan sequence as in other algorithms. For instance, with *RANSAC*, if more iterations are performed, the fitted line is getting closer to the stable position (local minimum), or in *HT* a badly noised inlier of a line may put its vote into an adjacent grid cell (of the cell representing the line) and thus does not get included in the set of scan points forming the line. Hence, the extracted line model parameters are not affected by the noise of bad inliers.

Table 2 shows the results of the second benchmark. The first observation is that in general, the algorithms perform better in term of correctness and precision. This is because the scans are cleaner compared with the first dataset. The algorithm speeds are much better in this case by the fact that there are only 181 scan points per batch scan.

In term of individual algorithm performance, the *Split-and-Merge* shows as the lead in most of the comparison criteria. Following is the *Incremental* (both versions) with only about 3% less of *TruePos* and about 0.1% more of *FalsePos*. The algorithms *HT*, *HT+Clustering* and *EM* again

generate quite high values of *FalsePos* as in the first benchmark. However, the *EM+Clustering* in contrast performs quite well with $TruePos = 74\%$ and $FalsePos = 5.2\%$. This can be explained that the objects are quite well-defined (see Fig. 6) so that the clustering algorithm is better in dividing the raw scan points into groups where each represents an individual line segment. In addition, the second dataset contains much less noise than the first one and it is important for the *EM+Clustering* to meet quickly the convergence condition.

In summary, the algorithms are tested with two datasets which are taken from the two environments having different structures, shapes and degrees of clutter. The comparison results are however consistent in terms of performance of individual algorithm. Note that in the first dataset (taken in the ASL hallway), while conducting the data collection, there were people walking around as they appeared in the raw scan pictures. However, they are mostly removed by the condition of minimal length of extracted line segments. Only few segments are actually extracted from human and they are classified as false positives. Due to this problem, the minimal length of an extracted line segment is set to 40 cm.

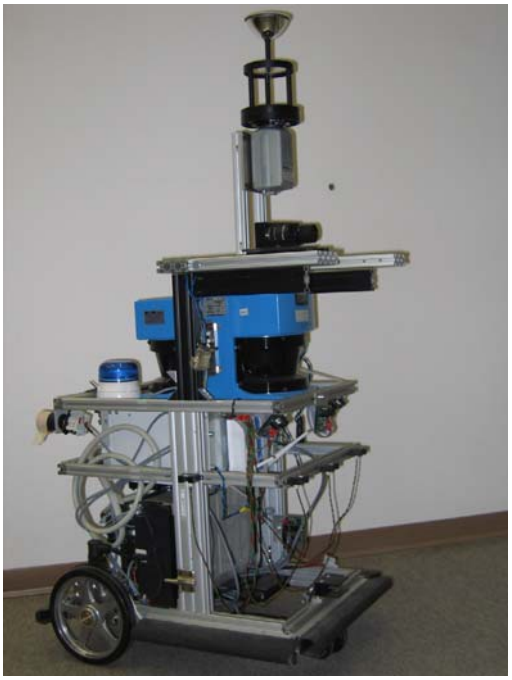

Fig. 7 The Biba robot used to perform the *OrthoSLAM* algorithm

5 Case study: SLAM application

In this Case Study, we apply the different line extraction algorithms in a SLAM application on a real robot. The performance of the algorithms will be indirectly evaluated by comparing the results obtained from the SLAM application. We select the OrthoSLAM algorithm (see (Nguyen et al. 2006) for detail) as the core algorithm to perform SLAM where the observation inputs are the extracted lines from the line extraction algorithms.

The Orthogonal SLAM (*OrthoSLAM*) is developed as a light lightweight and real-time consistent SLAM algorithm. The target of this algorithm is minimum systems embedded on simple robots. It is shown (in (Nguyen et al. 2006)) that using known techniques, it is possible to robustly and precisely perform SLAM in indoor, office-like environments by using line features and taking a simple assumption about the shape of the environment: the Orthogonality. It comes from the fact that in most indoor engineered environments, major structures, like walls, windows, cupboards, etc., can be represented by sets of lines which are parallel or perpendicular to each other. For reconstruction of the desired map, it is sufficient to extract and maintain those major lines. In fact, ignoring other lines (arbitrary oriented or non-orthogonal lines) not only does not lead to loss of valuable information, but also brings us amazing robustness on the robot orientation and filter out many dynamic objects.

By using the assumption, only observed orthogonal lines are selected so that it is able to represent line segment features using just one parameter. In fact we perform the mapping based on this constraint and in a simplified framework, rather than applying it as extra observation afterward. This is an important advantage which leads to the removal of non-linearities in the observation model and rather precise and consistent mapping. Furthermore, if an extracted line segment is not precisely orthogonal, it will be corrected "artificially" by the *OrthoSLAM*. Therefore, the *OrthoSLAM* algorithm is quite robust so that small, limited error on the extracted lines do not have strong effects on the results.

For the experiments, we use the Biba robot which has similar hardware configuration as the RoboX. The combination of 2 laser scanners enables the robot to scan a full 360°. We use a maximum scan range of 7.0 m, an angular resolution of 0.5° and a sampling rate of 4 Hz. We choose again our laboratory hallway which is a typical office environment having many line features as the testing zone. The average speed of the robot is about 30 cm/s excluding the time that it is stationary.

The robot navigates the hallway in approximately 45 minutes and performs 3643 observation steps. A sample map of raw scan points obtained using pure odometry information is plotted in Fig. 8. Notice that the resulting map is corrupted by the odometry drift. The drift is also different from the one obtained when using the RoboX because the robots have different odometry hardware. The odometry error cumulates so that when the robot arrives to the other end of the hallway, it has been totally lost by approximately 20 m from the true position.

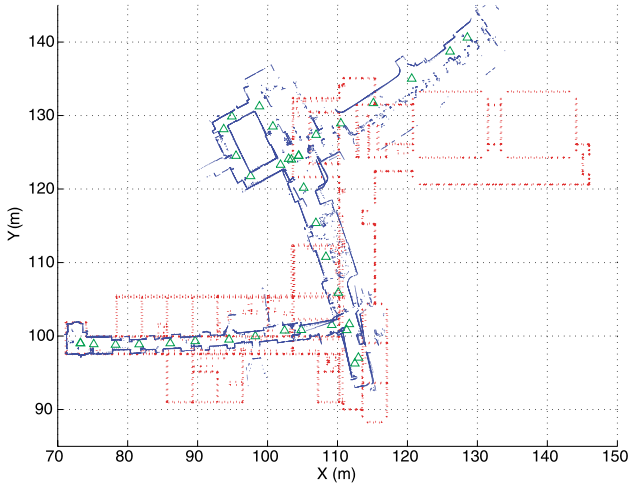

Fig. 8 The map built (*blue dotted curves*) by using raw scan points and pure odometry information is corrupted by the odometry drift. The ground truth (including office rooms) is shown in *red dotted lines*

The *OrthoSLAM* algorithm, using different line extraction algorithm each time, is performed on a laptop with a PentiumM-600 MHz using the logged data. The line extraction algorithms use the same parameters as for the previous experiments with RoboX. The summary of results is shown in Table 3. The first thing to notice is that the *OrthoSLAM* works with most of the line extraction algorithms, except for the three algorithms using *RANSAC*, *Hough Transform* and *EM*. The maps built using those three algorithms can not be used to recognize main features of the environments (cupboards, walls). For demonstration purpose, two maps obtained using the algorithms *Split-and-Merge* and *RANSAC* are shown in Figs. 9(b) and 9(c), respectively.

In term of computational speed, the results are consistent with the last results where the *Split-and-Merge* is the fastest algorithm: the *OrthoSLAM* can perform at 50 Hz which is approximately 10 times faster than the data acquisition speed.

In term of precision, the maps built by different line extraction algorithms (combined with the *OrthoSLAM*) are compared with the groundtruth which is obtained by hand measurement. Specifically, map size and absolute locations of the three main walls are used in the evaluation (see Fig. 9(a) for the selected main walls). We define the error on the map size:

$$\Delta d_x = |m_x^T - m_x|, \qquad \Delta d_x = |m_y^T - m_y|$$

where m_x^T, m_y^T and m_x, m_y are the dimensions in the x, y directions of the true and estimated map, respectively. We define the error on the absolute location:

$$\Delta r_y^1 = |^T r_y^1 - \overline{r_y^1}|, \quad \overline{r_y^1} = \frac{1}{5}\sum_{i=1}^{5} {}^i r_y^1,$$

$$\Delta r_x^2 = |^T r_x^2 - \overline{r_x^2}|, \quad \overline{r_x^2} = \frac{1}{3}\sum_{i=1}^{3} {}^i r_x^2,$$

$$\Delta r_y^3 = |^T r_y^3 - \overline{r_y^3}|, \quad \overline{r_y^3} = \frac{1}{3}\sum_{i=1}^{3} {}^i r_y^3$$

where x, y indicate the direction of the line components ${}^i r$ and T indicates the true value. Because of the orthogonality assumption, only the y direction is considered for the reference horizontal walls 1 and 3 and only the x direction is considered for the reference vertical wall 2. Because walls are usually extracted as several broken line segments (occluded by cupboards, doors), we select only the five major extracted line segments for the reference wall 1 and three major extracted line segments for the reference lines 2 and 3 to compute the mean value of the line parameters.

From Table 3, the results obtained by using the algorithms *Split-and-Merge*, *Incremental*, *Incremental*+ *Clustering* are consistently acceptable where the precision of the obtained map size is approximately 10 cm. The errors on the estimate of reference line 3 are quite large (for all algorithms). There are two reasons. First, there is a slope on the floor when the robot starts observing the line features belonging to the reference line 3. This causes a shift on the y direction. The problem of non-horizontal floor is not considered in the *OrthoSLAM* algorithm. Second, one side of the last part of the hallway (parallel to the reference line 3) is glass window where laser beams are not very well reflected. Thus, during this area the robot often observes only one horizontal wall. As a result, the *OrthoSLAM* discards the observation since it needs at least two horizontal (or vertical) lines in each observation.

The error in x direction of the map size when using the algorithm *Line Regression* is quite large. This is because of several false data association. The performance improves when it is used with Clustering algorithm. Still, the absolute location error increases as the robot departs further from the starting position.

The performance of the *OrthoSLAM* using the line extraction algorithms based on the *Hough Transform*, *RANSAC* and *EM* is quite poor compared to that when using the other algorithms. One sample of resulting map using the *RANSAC* is shown in Fig. 9(c) where one can not recognize the main walls, objects in the hallway. Similar results are obtained when using the algorithms *Hough Transform* and *EM* (the results are not included in the table). When coupled with the *Clustering* algorithm, their performance is improved significantly. However, since this group of line extraction algorithms do not use the ordering of scan points, lines are often not correctly extracted (see Table 1). Some extracted lines are crossing the map, leading to false data association.

Thus, we can state that the outcome of the case study with *OrthoSLAM* correlates with the outcome of the line

Table 3 Case Study: Result Summary

Algorithm	Number lines	Speed	Obtained map size	Map size error		Abs. location error		
				Δd_x	Δd_y	Δr_y^1	Δr_y^2	Δr_y^3
		[Hz]	[m] × [m]	[cm]	[cm]	[cm]	[cm]	[cm]
Split-Merge + Clustering	276	50	73.58 × 44.89	6	8	6	4	13
Incremental	244	37	73.55 × 45.09	3	8	9	12	16
Incremental + Clustering	235	40	73.53 × 44.94	1	3	9	9	8
Line Regression	261	40	73.09 × 45.03	43	6	6	24	37
Line Regression + Clustering	262	38	73.48 × 44.97	4	1	7	17	58
RANSAC	291	–	–	–	–	–	–	–
RANSAC + Clustering	255	16	73.48 × 45.02	4	5	89	70	32
Hough Transform	279	–	–	–	–	–	–	–
Hough Transform + Clustering	292	2	73.33 × 45.08	19	11	45	70	26
EM	287	–	–	–	–	–	–	–
EM + Clustering	346	0.2	73.34 × 44.79	18	21	23	65	116
Groundtruth	–	–	73.52 × 44.97	–	–	–	–	

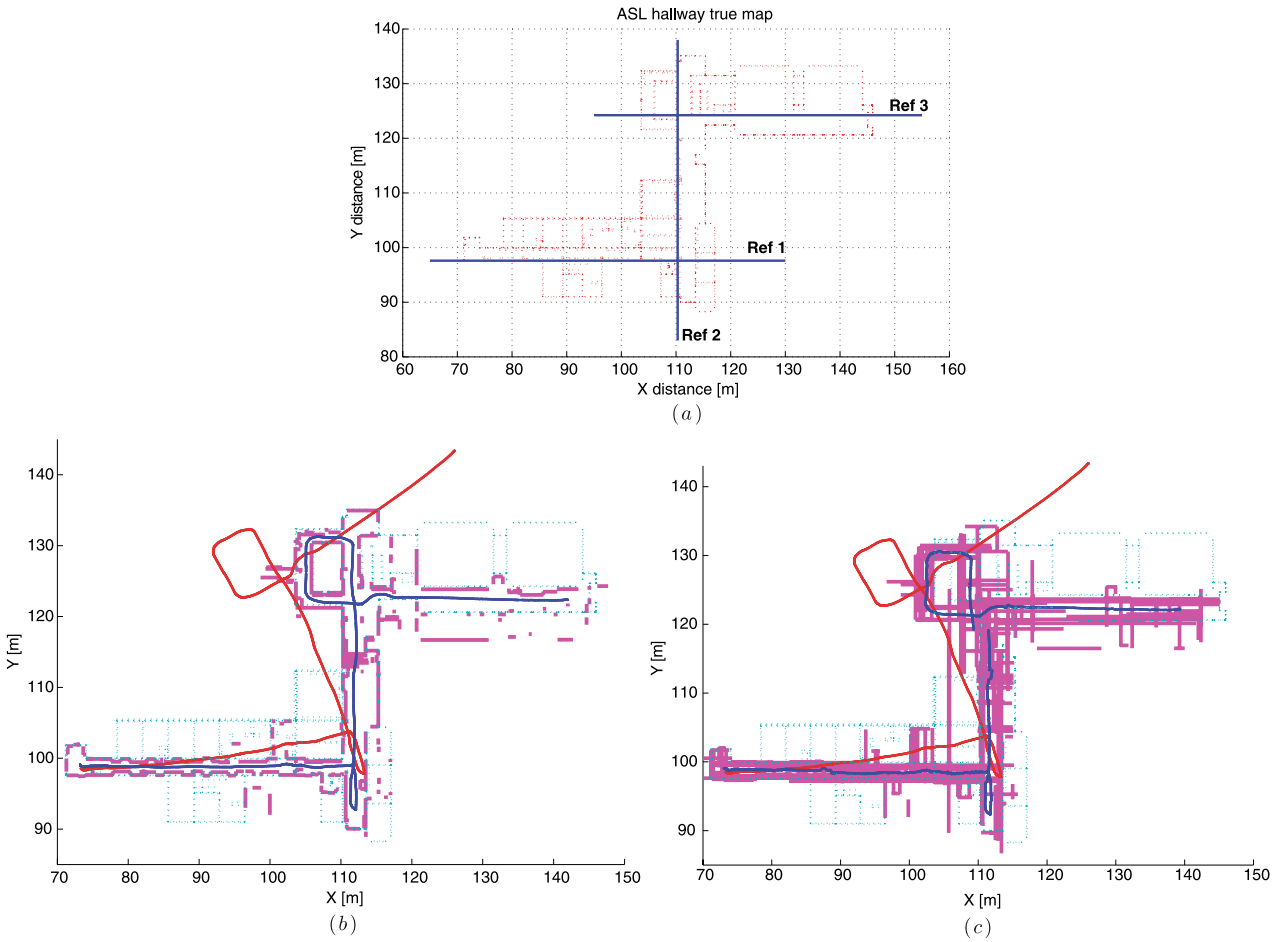

Fig. 9 **a** The 3 reference lines along the three main walls for absolute location evaluation. **b**, **c** The maps built by using the *OrthoSLAM* algorithm combined with the line extraction algorithms *Split-and-Merge* and *RANSAC*, respectively. The estimated segments (*pink solid lines*), the odometry trajectory (*red curve*), the estimated robot trajectory (*blue curve in middle of the hallway*) and the ground truth (*cyan dotted lines*). The map in (**b**), obtained using *Split-and-Merge*, shows correctly main features. The map in (**c**), obtained using *RANSAC*, is not recognizable

extraction experiments. Clustering as well as correctness have a direct impact on the map estimation. Because *OrthoSLAM* makes use of the orthogonality assumption, correctness is more important than precision. The line extraction algorithms *RANSAC*, *Hough Transform* and *EM* are outperformed by the other algorithms even though their precision is higher.

6 Conclusions

This paper has presented an experimental evaluation of the six line extraction algorithms using 2D laser scanner which are commonly used for feature extraction in mobile robotics and computer vision. The basic versions of the algorithms are implemented and tested with two datasets taken from two office environments which have different structures, shapes and degrees of clutter. Line segments extracted by the algorithms are compared with the manually extracted lines using standard statistical methods. Several comparison criteria are proposed and used to discuss in details their advantages and drawbacks. Additionally, the line extraction algorithms are tested in the OrthoSLAM application and the results obtained individually are compared. We believe this is important, particularly in mobile robotics, that techniques or algorithms should be tested and verified their performance in real life application before we can make the final choice. The comparison result with the OrthoSLAM has been shown to agree with the findings in the first part, thus again verify the conclusions.

Overall, the experimental results show that the two algorithms *Split-and-Merge* and *Incremental* have best performance because of their superior speed and correctness. For real-time applications, *Split-and-Merge* is clearly the best choice by its superior speed. It is also the first choice for localization problems with a priori map, where *FalsePos* is not very important. However, a right choice highly depends on the applications and implementation details as the case study showed, where correctness is favored over precision.

The first released version of the implementation is accessible to public at http://www.asl.ethz.ch/people/tnguyen/. The ASL dataset can also be downloaded from the same location for comparison purposes. Interested user can send inquiries to viet.nguyen@mavt.ethz.ch.

Acknowledgements This work has been supported by the Swiss National Science Foundation No. 200021-101886 and the EU project Cogniron FP6-IST-002020. We would like to thank Frédéric Pont for the support in carrying out the experiments. The Intel Oregon dataset was obtained from the Robotics Data Set Repository (Radish). Thanks go to Maxim Batalin for providing the data.

References

Adams, M. D., & Kerstens, A. (1998). Tracking naturally occurring indoor features in 2D and 3D with lidar range/amplitude data. *International Journal of Robotics Research*, *17*(9), 907–923.

Alempijevic, A. (2004). High-speed feature extraction in sensor coordinates for laser rangefinders. In *Proceedings of the Australasian conference on robotics and automation*.

Arras, K. O., & Siegwart, R. (1997). Feature extraction and scene interpretation for map-based navigation and map building. In *Proceedings of the symposium on intelligent systems and advanced manufacturing*.

Borges, G. A., & Aldon, M.-J. (2004). Line extraction in 2D range images for mobile robotics. *Journal of Intelligent and Robotic Systems*, *40*, 267–297.

Brunskill, E., & Roy, N. (2005). SLAM using incremental probabilistic PCA and dimensionality reduction. In *Proceedings of the IEEE international conference on robotics and automation*.

Castellanos, J., & Tardós, J. (1996). Laser-based segmentation and localization for a mobile robot. In *Robotics and manufacturing: recent trends in research and applications* (Vol. 6). New York: ASME.

Cox, I. J. (1991). Blanche: an experiment in guidance and navigation of an autonomous robot vehicle. *IEEE Transactions on Robotics and Automation*, *7*(2), 193–204.

Crowley, J. L. (1985). Navigation for an Intelligent mobile robot. *IEEE Journal of Robotics and Automation*, *1*(1), 31–41.

Diosi, A., & Kleeman, L. (2003a). Uncertainty of line segments extracted from static SICK PLS laser scans. In *Proceedings of the Australasian conference on robotics and automation*.

Diosi, A., & Kleeman, L. (2003b). *Uncertainty of line segments extracted from static SICK PLS laser scans*. Department of Electrical and Computer Systems Engineering, Monash University, Tech. Rep. MECSE-26-2003.

Duda, R. O., & Hart, P. E. (1973). *Pattern classification and scene analysis*. New York: Wiley.

Einsele, T. (1997). Real-time self-localization in unknown indoor environments using a panorama laser range finder. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems* (pp. 697–702).

Fischler, M., & Bolles, R. (1981). Random sample consensus: A paradigm for model fitting with application to image analysis and automated cartography. *Communications of the ACM*, *24*(6), 381–395.

Forsberg, J., Larsson, U., & Wernersson, A. (1995). Mobile robot navigation using the range-weighted hough transform. *IEEE Robotics and Automation Magazine*, *2*(1), 18–26.

Forsyth, D. A., & Ponce, J. (2003). *Computer vision: a modern approach*. New York: Prentice Hall.

Gutmann, J.-S., & Schlegel, C. (1996). AMOS: comparison of scan matching approaches for self-localization in indoor environments. In *First European workshop on advanced mobile robots (Eurobot)*.

Gutmann, J.-S., Burgard, W., Fox, D., & Konolige, K. (1998). An experimental comparison of localization methods. In *Proceedings of the IEEE/RSJ intenational conference on intelligent robots and systems, IROS*.

Howard, A., & Roy, N. (2003). The robotics data set repository (radish). Available: http://radish.sourceforge.net/.

Iocchi, L., & Nardi, D. (2002). Hough localization for mobile robots in polygonal environments. *Robotics and Autonomous Systems*, *40*, 43–58.

Iyengar, S., & Elfes, A. (1991). *Autonomous mobile robots* (Vols. 1, 2). Los Alamitos: IEEE Computer Society.

Jensfelt, P., & Christensen, H. (1998). Laser based position acquisition and tracking in an indoor environment. In *Proceedings of the IEEE international symposium on robotics and automation* (vol. 1).

Leonard, J., Cox, I. J., & Durrant-Whyte, H. (1992). Dynamic map building for an autonomous mobile robot. *International Journal of Robotics Research*, *11*(4), 286–298.

Liu, Y., Emery, R., Chakrabarti, D., Burgard, W., & Thrun, S. (2001). Using EM to learn 3D models of indoor environments with mobile robots. In *Proceedings of the IEEE international conference on machine learning (ICML)*.

Lu, F., & Milios, E. (1994). Robot pose estimation in unknown environments by matching 2D range scans. In *Proceedings of the IEEE computer society conference on computer vision and pattern recognition* (pp. 935–938).

Nguyen, V., Martinelli, A., Tomatis, N., & Siegwart, R. (2005). A comparison of line extraction algorithms using 2D laser rangefinder for indoor mobile robotics. In *Proceedings of the IEEE/RSJ intenational conference on intelligent robots and systems, IROS*, Edmonton, Canada.

Nguyen, V., Harati, A., Tomatis, N., Martinelli, A., & Siegwart, R. (2006). OrthoSLAM: a step toward lightweight indoor autonomous navigation. In *Proceedings of the IEEE/RSJ intenational conference on intelligent robots and systems, IROS*, Beijing, China.

Pavlidis, T., & Horowitz, S. L. (1974). Segmentation of plane curves. *IEEE Transactions on Computers*, *C-23*(8), 860–870.

Pears, N. (2000). Feature extraction and tracking for scanning range sensors. *Robotics and Autonomous Systems*, *33*, 43–58.

Pfister, S. T., Roumeliotis, S. I., & Burdick, J. W. (2003). Weighted line fitting algorithms for mobile robot map building and efficient data representation. In *Proceedings of the IEEE international conference on robotics and automation, ICRA* (pp. 1304–1311).

Sack, D., & Burgard, W. (2004). A comparison of methods for line extraction from range data. In *Proceedings of the 5th IFAC symposium on intelligent autonomous vehicles*.

Schröter, D., Beetz, M., & Gutmann, J.-S. (2002). RG mapping: learning compact and structured 2D line maps of indoor environments. In *Proceedings of 11th IEEE international workshop on robot and human interactive communication (ROMAN)*.

Siadat, A., Kaske, A., Klausmann, S., Dufaut, M., & Husson, R. (1997). An optimized segmentation method for a 2D laser-scanner applied to mobile robot navigation. In *Proceedings of the 3rd IFAC symposium on intelligent components and instruments for control applications*.

Siegwart, R., Arras, K. O., Bouabdallah, S., Burnier, D., Froidevaux, G., Greppin, X., Jensen, B., Lorotte, A., Mayor, L., Meisser, M., Philippsen, R., Piguet, R., Ramel, G., Terrien, G., & Tomatis, N. (2003). Robox at Expo.02: a large scale installation of personal robots. *Special issue on Socially Interactive Robots, Robotics and Autonomous Systems*, *42*, 203–222.

Snedecor, G. W., & Cochran, W. G. (1989). *Statistical methods* (8th ed.). Ames: Iowa State University Press.

Taylor, R., & Probert, P. (1996). Range finding and feature extraction by segmentation of images for mobile robot navigation. In *Proceedings of the IEEE international conference on robotics and automation, ICRA*.

Tomatis, N., Nourbakhsh, I., & Siegwart, R. (2003). Hybrid simultaneous localization and map building: a natural integration of topological and metric. *Robotics and Autonomous Systems*, *44*, 3–14.

Vandorpe, J., Brussel, H. V., & Xu, H. (1996). Exact dynamic map building for a mobile robot using geometrical primitives produced by a 2D range finder. In *Proceedings of the IEEE international conference on robotics and automation, ICRA* (pp. 901–908).

Xu, L., Oja, E., & Kultanen, P. (1990). A new curve detection method: randomized hough transform (RHT). *Pattern Recognition Letters*, *11*(5), 331–338.

Zhang, L., & Ghosh, B. K. (2000). Line segment based map building and localization using 2D laser rangefinder. In *Proceedings of the IEEE international conference on robotics and automation, ICRA*.

Viet Nguyen is a Ph.D. student at the Swiss Federal Institute of Technology Lausanne (EPFL). He received his Bachelor in Software Engineering from the Australian National University, Australia in 1999. He worked as a research assistant in the Laboratory of Artificial Intelligence at EPFL in 2001. In 2003, he joined the Autonomous Systems Laboratory where started his Ph.D. study under the supervision of Professor Siegwart. Since July 2006, he has been working at the Autonomous Systems Laboratory at the Swiss Federal Institute of Technology Zurich (ETHZ). His research covered mobile robot navigation, SLAM, feature extraction, data association, constraint satisfaction and optimization.

Stefan Gächter received the Master's degree in electrical engineering from the École Polytechnique Fédérale de Lausanne (EPFL), Switzerland, in 2001. After his master's studies, he spent three years as a research engineer in the field of active magnetic bearings at Koyo Seiko Co. Ltd, Japan. He joined the Autonomous Systems Lab (ASL), École Polytechnique Fédérale de Lausanne (EPFL), Switzerland, in 2004, where he commenced work towards a Ph.D. degree. Since July 2006, the Autonomous Systems Lab has been based at the Eidgenössische Technische Hochschule Zürich (ETHZ), Switzerland. His research interest is in computer vision for mobile robotics with emphasis on probabilistic algorithms for object classification applied to range images.

Agostino Martinelli (1971) received the M.Sc. in theoretical Physics in 1994 from the University of Rome "Tor Vergat" and the PhD in Astrophysics in 1999 from the University of Rome "La Sapienza". During his PhD he spent one year at the University of Wales in Cardiff and one year in the School of Trieste (SISSA). His research was focused on the chemical and dynamical evolution in the elliptical galaxies, in the quasars and in the intergalactic medium. He also introduced models based on the General Relativity to explain the anisotropies on the Cosmic Background Radiation. After his PhD, his interests moved on the problem of autonomous navigation. He spent two years in the University of Rome "Tor Vergata" and, in 2002, he moved to the Autonomous Systems Lab, EPFL, in Lausanne as senior researcher, where he was leading several projects on multi sensor fusion for robot localization (with particular attention to the odometry sensors), simultaneous localization and odometry error learning, and simultaneous localization and map building. Since September 2006 he is First Researcher (CR1) at the INRIA Rhone Alpes in Grenoble. He has authored/co-authored more than 30 journals and conference papers.

Nicola Tomatis (1973) received his M.Sc. in computer science in 1998 from the Swiss Federal Institute of Technology (ETH) Zurich. After working as assistant for the Institute of Robotics, ETH, he moved to the Swiss Federal Institute of Technology (EPFL) Lausanne, where he received his Ph.D. at the end of 2001. His research covered metric and topological (hybrid) mobile robot navigation, computer vision and sensor data fusion. From 2001 to 2005 he had a

part time position as senior researcher with the Autonomous Systems Lab, EPFL (now ETH), where he was leading several projects and continuing his research in navigation, man–machine interaction and robot safety and reliability. He has authored/co-authored more than 35 journals and conference papers. During 2001 he joined BlueBotics SA, a SME involved in mobile robotics. Since year 2003 he is CEO of the company. http://asl.epfl.ch, http://www.bluebotics.com

Roland Siegwart is full professor for autonomous systems at ETH Zurich since July 2006. He has a Diploma in Mechanical Engineering (1983) and PhD in Mechatronics (1989) from ETH Zurich. In 1989/90 he spent one year as postdoctoral fellow at Stanford University. After that he worked part time as R&D director at MECOS Traxler AG and as lecturer and deputy head at the Institute of Robotics, ETH Zürich. In 1996 he was appointed as associate and later full professor for autonomous microsystems and robots at the Ecole Polytechnique Fédérale de Lausanne (EPFL). During his period at EPFL he was Deputy Head of the National Competence Center for Research (NCCR) on Multimodal Information Management (IM2), co-initiator and founding Chairman of Space Center EPFL and Vice Dean of the School of Engineering. In 2005 he hold a visiting position at NASA Ames and Stanford University. Roland Siegwart is member of the Swiss Academy of Engineering Sciences and board member of the European Network of Robotics (EURON). He served as Vice President for Technical Activities (2004/05) and is currently Distinguished Lecturer (2006/07) and AdCom Member (2007–2009) of the IEEE Robotics and Automation Society. He is member of the "Bewilligungsausschuss Exzellenzinitiative" of the "Deutsche Forschungsgemeinschaft (DFG)". He is coordinator of two European projects and co-founder of several spin-off companies.