An Extended Line Tracking Algorithm

Conference Paper · November 2013

DOI: 10.1109/ROPEC.2013.6702752

3 authors:

Leonardo Romero
Universidad Michoacana de San Nicolás de Hidalgo
43 PUBLICATIONS **156** CITATIONS
SEE PROFILE

Moises Garcia
Universidad Michoacana de San Nicolás de Hidalgo
20 PUBLICATIONS **28** CITATIONS
SEE PROFILE

Carlos Lara-Alvarez
Centro de Investigación en Matemáticas (CIMAT)
52 PUBLICATIONS **120** CITATIONS
SEE PROFILE

An Extended Line Tracking Algorithm

Leonardo Romero Muñoz
Facultad de Ingeniería Eléctrica
UMSNH
Morelia, Mich., Mexico
Email: lromero@umich.mx

Moises García Villanueva
Facultad de Ingeniería Eléctrica
UMSNH
Morelia, Mich., Mexico
Email: moises@correo.fie.umich.mx

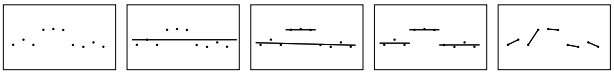
Carlos Alberto Lara Álvarez
CINVESTAV, IPN
Guadalajara, Jal., Mexico
Email: carlos_lara24@yahoo.com

***Abstract*—This paper introduces an extension of the classical Line Tracking Algorithm to solve the problem of fitting multiple straight–lines to a sequence of 2D points. We also analyze the performance of the extended algorithm to solve a classical problem in robotics: finding a map of lines from laser measurements; tests show that the extended algorithm obtains reliable models.**

I. Introduction

Let $Z = (z_1, \ldots, z_N)$ be a sequence of N measurements or points where each point $z_i = \langle x_i, y_i \rangle$ is represented by its rectangular coordinates; the multiple lines fitting problem consists of finding the set of lines $\Theta = \{\theta_1, \ldots, \theta_M\}$ that best represents Z; To find Θ, we need to discover the number of lines, M, and the parameters of each line. The multiple lines fitting problem occurs in several fields of science and engineering; for instance, a mobile robot can build a line map using data sensed from an indoor environment and use it to navigate. In man-made environments (both indoors and outdoors), planar surfaces are very common and they are typically modeled by line segments. Some examples include corridor or room walls, doors, tables, etc.

Finding the best set of lines is sometimes hard; Figure 1 shows a set of points and four hypothetical line sets with 1, 2, 3 and 5 lines, respectively. If the sum of square distance from each point to the associated line is used to decide which model is the best, then the five–lines model shown in Figure 1e is chosen because the total error is zero. This case illustrates the over fitting problem where a very complex model has an error close to zero, but it is not generally useful. Often simple models are preferable than complex models.

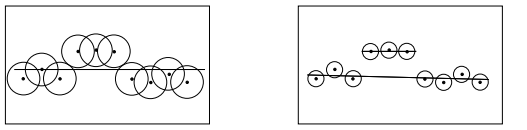


(a) Points Z (b) 1 line (c) 2 lines (d) 3 lines (e) 5 lines

Fig. 1. A data set Z and some hypothetical models to represent it.

The accuracy of the measurements is another factor that must be considered when selecting the best model Θ. Figure 2 represents the uncertainty with circles; a single line model is preferable when the uncertainty is big (Fig 2a), and the two–lines model is selected when the uncertainty is small (Fig 2b).

This work focuses on finding lines form laser data, the resulting set of lines can be used for indoor and structured outdoor applications. Lasers have several advantages with respect to other sensors: they are accurate, have a high sampling rate, and range measurements are easier to interpret than images from cameras, for example. Laser data are usually a sequence of points $R = (\langle \rho_1, \phi_1 \rangle, \ldots, \langle \rho_N, \phi_N \rangle)$ where $\langle \rho_i, \phi_i \rangle$ are the polar coordinates of the i-th scan point, where the location of the sensor is the origin of this polar coordinate system. The laser, using a fixed angular resolution, acquires points in an ordered sequence, starting at direction ϕ_{min} and stopping at ϕ_{max} (0° and 180° for the SICK LMS209–S02).



(a) Big uncertainty, 1 line (b) Low uncertainty, 2 lines

Fig. 2. Line models considering different uncertainty.

This paper develops an extended version of the Line Tracking (ELT) Algorithm that improves the accuracy of the lines as well as merging similar lines into a single line. The rest of the paper is organized as follows: Section II briefly reviews existing line extraction methods, specially the Line Tracking (LT) Method. Section III presents the extensions to the LT Method. Experimental results are presented in Section IV. Finally, Section V presents the main conclusions of this work.

II. Line extraction algorithms

This section is based in the revision of existing line extraction algorithms given in [6]. Probably, the most straightforward example of a line extractor is the *Successive Edge Following* (SEF) algorithm [9]. Briefly, it considers that a new line begins when the distance between two consecutive scan points exceeds a certain threshold.

Another algorithm that it is remarkable not only for being uncomplicated but also for its reduced time complexity is the Line Tracking (LT) algorithm [3], also known as Incremental. Basically, it starts off by building a line model which passes through the first and second scan points, successively adding a new scan point if a line criterion is validated. Otherwise, the line is terminated and a new one is started, repeating the algorithm until the end of the dataset is reached.

Unlike the two previous methods, the Iterative End Point Fit algorithm (IEPF) [5] is recursive. It begins by constructing a line using the first and last scan points. Next, it finds the

978-1-4799-2370-0/13/$31.00 ©2013 IEEE

most distant point p_k from the line, and if it is far enough, two subsets are created taking p_k as the splitting point. This procedure is repeated recursively for all the subsets until the validation criterion fails.

The Split & Merge (S&M) algorithm [8] is twofold. Its first phase (split) is similar to the IEPF method. Nevertheless, it differs from it in that it adds another phase (merge) in which collinear segments are fused together if the angle between them is sufficiently small.

In contrast to the previously discussed methods, the RANdom SAmple Consensus (RANSAC) algorithm [2] makes use of a probabilistic approach, and it robustly fits models in the presence of data outliers. Firstly, it constructs a line R using two randomly chosen points from the initial point set. Secondly, a consensus set is created, formed by line inliers, and if it is big enough, the line R is readjusted to the points included in the consensus set. Otherwise, the algorithm is repeated until a proper consensus set is found or the maximum number of loop iterations is reached.

Another robust method to find lines is the Hough Transform (HT) [4]. The basic theory of the HT is that any point could be part of some set of possible lines. If we parameterize each line by its polar coordinates $\langle r, \phi \rangle$, a point is transformed to a locus of points in the $\langle r, \phi \rangle$ plane corresponding to all the lines passing though that point. If the plane $\langle r, \phi \rangle$ is discretized and the contributions of each point are added to the plane $\langle r, \phi \rangle$, then lines that appear in the set of points will appear as local maxima in the plane $\langle r, \phi \rangle$. In [6], a method named REHOLT (Reduced Hough transform Line Tracker) is developed for detecting lines. It is a technique that combines the speed of the LT algorithm and the robustness of the HT. Using a reduced HT, lines are more precisely fitted to their corresponding set of points, since the HT is based on a robust voting strategy.

A main disadvantage of SEF, LT, IEPF and S&M is that they fit lines to a set of points using fast but non-robust methods, such as least squares for example, which is known to have problems with outliers [2]. On the other hand, RANSAC is a robust method, but its processing time and results are not always the same, because it is a nondeterministic algorithm. The HT is also a robust method, but it takes a significant amount of processing time and the precision of the results depend of the discretization of the plane $\langle r, \phi \rangle$.

A. The Line Tracking Algorithm

The LT algorithm is remarkable not only for being simple but also because it is very fast. The method is based on computing a line θ that fits the last $n-1$ points. Then, a distance $d(z_n, \theta)$ from the point z_n to line θ is computed. If the distance is greater than a fixed threshold value T_{max}, a new line is started. Otherwise, a new line is computed with all points, including z_n. The algorithm starts joining the first two points with a line.

A line is represented by its normal form,

$$\theta_j = \langle r_j, \phi_j \rangle \tag{1}$$

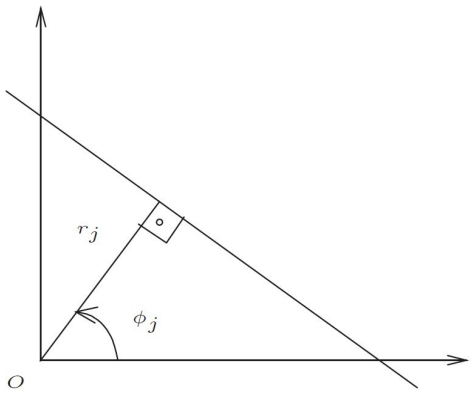


Fig. 3. Line parameters in the polar form.

where r_j and ϕ_j are the length and the angle of inclination of the normal, respectively. As shown in Figure 3, the normal is the shortest segment between the line and the origin of a given coordinate frame. Points $z = \langle x, y \rangle$ that are on the line $\theta_j = \langle r_j, \phi_j \rangle$ satisfy $r_j = x \cos \phi_j + y \sin \phi_j$.

The orthogonal distance from a point $z_i = \langle x_i, y_i \rangle$ to the line θ_j is given by

$$d_\perp(z_i, \theta_j) = r_j - x_i \cos \phi_j - y_i \sin \phi_j. \tag{2}$$

This equation can be easily derived by considering the line $r_i = x_i \cos \phi_i + y_i \sin \phi_i$; such line passes through the point z_i and is parallel to line θ_j when $\phi_i = \phi_j$; Therefore, $d_\perp(z_i, \theta_j) = r_j - r_i$.

Under the assumption of the same known normal uncertainty in directions x and y, the best line θ with parameters $\langle r, \phi \rangle$ minimizes the sum of squared perpendicular distances from the points to the line. From n points $\langle x_i, y_i \rangle (i = 1, \ldots, n)$, it can be shown that the solution is given by [1]

$$\begin{bmatrix} r \\ \phi \end{bmatrix} = \begin{bmatrix} \overline{x} \cos \phi + \overline{y} \sin \phi \\ \frac{1}{2} \arctan \frac{-2 s_{xy}}{s_{yy} - s_{xx}} \end{bmatrix} \tag{3}$$

Where

$$\overline{x} = \frac{1}{n} \sum_{i=1}^{n} x_i, \qquad \overline{y} = \frac{1}{n} \sum_{i=1}^{n} y_i,$$

$$s_{xx} = \sum_{i=1}^{n} (x_i - \overline{x})^2, \qquad s_{yy} = \sum_{i=1}^{n} (y_i - \overline{y})^2,$$

$$s_{xy} = \sum_{i=1}^{n} (x_i - \overline{x})(y_i - \overline{y}).$$

The Line Tracking algorithm is presented in Algorithm 1 and it computes the set of lines as well as the indexes of the points that form each line.

III. An extended Line Tracking Algorithm

After using the LT algorithm, we add two steps: a Backtracking Step and a Merge Step.

Algorithm 1 Line Tracking Algorithm

INPUT: Z, a sequence of points $(z_1, \ldots, z_N)$ and T_{max}
OUTPUT: Θ, a sequence of lines $(\theta_1, \ldots, \theta_M)$ and I, a sequence of indexes for Θ

$i \leftarrow 1$, $j \leftarrow 1$, $l \leftarrow 1$, $\Theta \leftarrow ()$, $I \leftarrow ()$
while $j < N - 2$ **do**
  $\theta_l \leftarrow$ best line that fits $(z_i, \ldots, z_{j+1})$ // (eq. 3)
  $T \leftarrow d(z_{j+2}, \theta_l)$ // (eq. 2)
  if $T > T_{max}$ **then**
    Add θ_l to Θ
    Add $\langle i, j+1 \rangle$ to I // indexes of the line θ_l
    $l \leftarrow l + 1$
    $i \leftarrow j + 2$
    $j \leftarrow i$
  else
    $j \leftarrow j + 1$
  end if
end while
return Θ and I

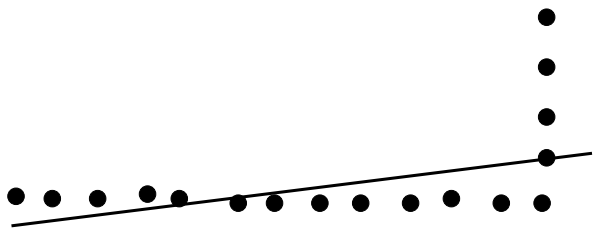


Fig. 4. LT does not estimate accurately line parameters.

A. A Backtracking Step

When the LT algorithm is adding new points to a line, it is possible that some of the final points (before reach the T_{max} value) belong to the next line. This situation is shown in Figure 4, where there are two lines (one horizontal and one vertical), but the horizontal line includes vertical points, so it is slightly pull up by these points. The backtracking step reviews if it is better to associate the last points of a line θ_l to line θ_{l+1} and it considers two cases:

1) The line θ_l has 3 or more points. In this case, the last point is associated to the closest line: θ_l or θ_{l+1}. If the point is associated to θ_{l+1}, the line θ_{l+1} is recomputed with the new point. This step is repeated until an endpoint belongs to line θ_l. In this way, this step removes the wrong associations of the LT algorithm.
2) The line θ_l has 2 points. In this case, if the distance from the second point of θ_l to line θ_{l+1} is less than or equal to T_{max}, that point is associated to line θ_{l+1}. In that case, the same revision is executed with the first point of θ_l. At the end, the line θ_l can be two, one or zero points.

The Backtracking algorithm is shown in Algorithm 2, where the outputs of Algorithm 1 are the inputs of Algorithm 2.

Algorithm 2 Backtracking Algorithm

INPUT: Z $(z_1, \ldots, z_N)$, T_{max}, Θ $(\theta_1, \ldots, \theta_M)$ and I
OUTPUT: Θ and I
for $i = 1, \ldots, M - 1$ **do**
  $\langle s_1, e_1 \rangle \leftarrow I(i)$ // indexes of line θ_i
  $\langle s_2, e_2 \rangle \leftarrow I(i+1)$ // indexes of line θ_{i+1}
  repeat
    $d_2 \leftarrow d(z_{e_1}, \theta_{i+1})$
    if $e_1 - s_1 + 1 > 2$ **then**
      $\theta_i^{'} \leftarrow$ best line that fits $(z_{s_1}, \ldots, z_{e_1 - 1})$
      $d_1 \leftarrow d(z_{e_1}, \theta_i^{'})$ // $\theta_i^{'}$ does not include z_{e_1}
    else
      $d_1 = T_{max}$
    end if
    if $d_2 < d_1$ **then**
      $e_1 \leftarrow e_1 - 1$, $s_2 \leftarrow s_2 - 1$ // adjust indexes
      Update indexes of lines θ_i and θ_{i+1} in I
      recompute lines θ_i and θ_{i+1} and update Θ
    else
      break **repeat**
    end if
  until $s_1 = e_1$
end for
Delete from Θ and I, lines with no points
return Θ and I

B. A Merge Step

After the Backtracking Step, a Merge Step is executed. The idea is to merge two sets of points if every point has a distance to the new line (computed using both sets of points) less than or equal to T_{max}. In this way, two o more sets of points can be represented by using a single line.

Algorithm 3 implements this step, where the outputs of Algorithm 2 and the inputs of Algorithm 1 are the inputs for this algorithm. The search in the algorithm can be exhaustive or reduced to data sets with similar line parameters. With the Merge step, we obtain a better feature model and also avoid the over-fitting problem.

IV. Experimental Results

To show how the Backtracking and Backtracking + Merge algorithms are working, we use a synthetic data set and a real laser scan (361 measurements from 0° to 180° for the SICK LMS209-S02).

Figure 5 shows the result of the LT algorithm for the synthetic data. Note how two lines, the horizontal line (at the bottom right part of the figure) and a vertical line (at the top right part of the figure) are not accurately computed, because they include point from the next line. Figure 6 shows the result of the Backtracking algorithm. Now both lines are horizontal and vertical and also there are three single points. We can consider these points as outliers because they are far away from any line segment. Figure 7 shows the result of the Backtracking and the Merge algorithm. There are only three

Algorithm 3 Merge Algorithm

INPUT: Z $(z_1, \ldots, z_N)$, T_{max}, Θ $(\theta_1, \ldots, \theta_M)$ and I
OUTPUT: Θ and D (a set of set of points)

Compute $D = \{D_1, \ldots, D_M\}$, where D_i is the set of points of line θ_i
loop
Search the pair D_a, $D_b \in D \mid D_a \neq D_b$, where a single line can represent both sets (given T_{max})
if (successful search) **then**
$D_a \leftarrow D_a \cup D_b$
$D \leftarrow D \setminus \{D_b\}$ // delete D_b from D
$M \leftarrow M - 1$
else
break **loop**
end if
end loop
Recompute line parameters Θ from D
return Θ and D

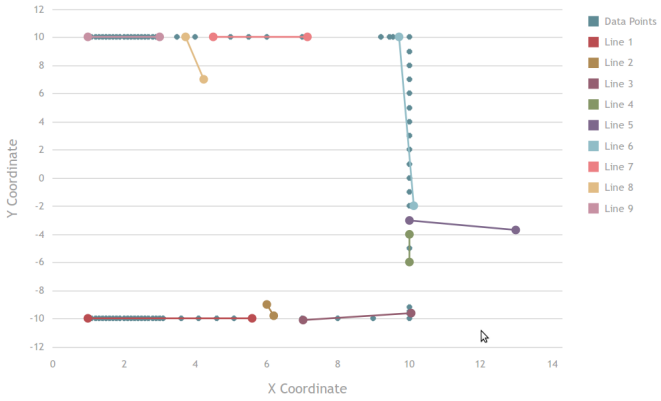


Fig. 5. LT algorithm with synthetic data.

lines, because the other 3 lines are reduced to single points. The results from the Backtracking algorithm are similar to the REHOLT method [6] without using the Hough Transform.

Figures 8, 9 and 10 show the result of the application of LT, LT + Backtracking and LT + Backtracking and Merge algorithms (with $T_{max} = 6cm$) to a laser scan. To avoid spurious lines, only lines with 5 o more points are considered. Results are similar to the case of synthetic data.

Figure 11 shows the result of the application of the RANSAC algorithm to the same laser scan (also with $T_{max} = 6cm$ and lines with 5 o more points). In this case, T_{max} is the maximum distance from a point to the line, to consider the point as an inlier. In this case the RANSAC algorithm made 500 iterations to get a line; then its inliers are removed from the data set, and the algorithm is applied to the reduced data set to get new lines. Results from RANSAC and the extended LT algorithm are very similar.

Table I shows the time required by each algorithm. The Line Tracking (LT) and the Line Tracking + Backtracking and Merge (LT-BT-M) are much faster than RANSAC. This

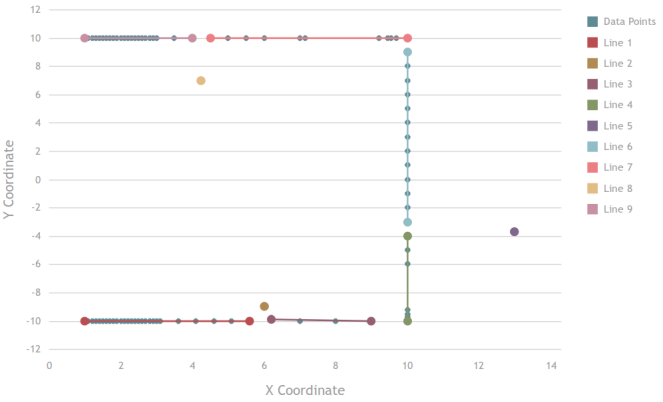


Fig. 6. LT + Backtracking algorithm with synthetic data.

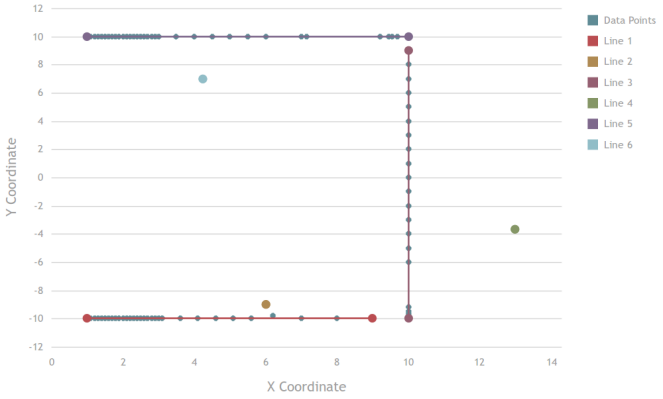


Fig. 7. LT + Backtracking and Merge algorithms with synthetic data.

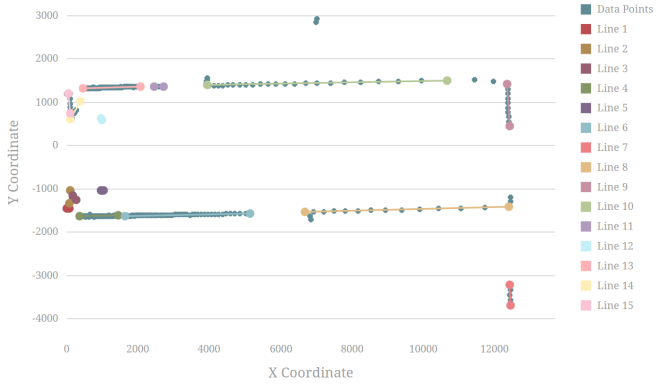


Fig. 8. LT algorithm with a laser scan

is mainly because LT and LT-BT-M algorithms are based on deterministic methods and especially, they take advantage of the sequence of points of the laser scan. A comprehensive experimental evaluation of line extraction algorithms (including LT, RANSAC, Hough, and S&M) using 2D laser rangefinder for indoor mobile robotics is shown in [7]. They report a speed of 344 Hz for the LT algorithm against 29 Hz for RANSAC, with a better behavior of LT. RANSAC is much slower than the LT.

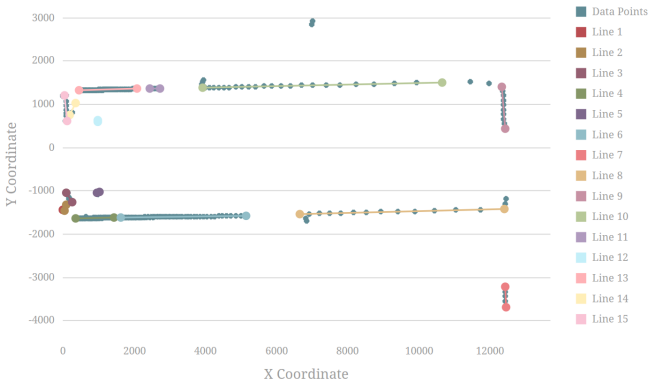


Fig. 9. LT + Backtracking algorithm with a laser scan

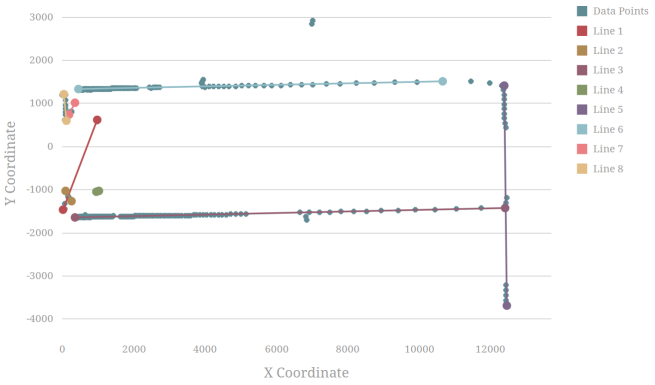


Fig. 10. LT + Backtracking and Merge algorithms with a laser scan

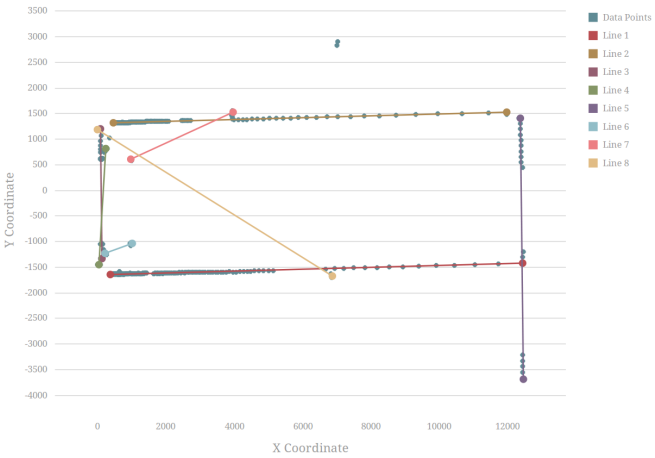


Fig. 11. RANSAC algorithm with a laser scan

V. Conclusions and Future Work

We have presented an Extended Line Tracking algorithm to find lines given a sequence of points. This method adds a Backtracking step and a Merge step to the standard Line Tracking algorithm. The Backtracking step tries to remove wrong associations of points from adjacent lines and the Merge step decide between merging set of points (associated to lines) or not. To merge two set of points, it uses the same criterion used in the Line Tracking algorithm.

This Extended Line Tracking algorithm is simple, accurate and fast, allowing the algorithm to be used for real-time line extraction. If lines are accurately determined, then the robot has better features of the environment.

TABLE I
Speed and lines found under common parameters.

Algorithm	Time (ms)	Lines found
LT	30	15
LT-BT-M	49	8
RANSAC	1027	8

References

[1] Kai Oliver Arras and Roland Y. Siegwart. Feature extraction and scene interpretation for map-based navigation and map building. In *Proc. of SPIE, Mobile Robotics XII*, pages 42–53, 1997.

[2] Robert C. Bolles and Martin A. Fischler. A RANSAC-based approach to model fitting and its application to finding cylinders in range data. In *IJCAI*, pages 637–643, 1981.

[3] G.A. Borges and M.-J. Aldon. Line extraction in 2d range images for mobile robotics. *Journal of Intelligent and Robotic Systems*, 40:267–297, 2004.

[4] R. O. Duda and P. E. Hart. Use of the hough transformation to detec lines and curves in pictures. *Communications of the Association for Computing Machinery*, 15:11–15, 1972.

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

[6] Carlos Fernandez, Vidal Moreno, Belen Curto, and J. Andres Vicente. Clustering and line detection in laser range measurements. *Robotics and Autonomous Systems*, 40:720–726, 2010.

[7] Viet Nguyen, Stefan Gächter, Agostino Martinelli, Nicola Tomatis, and Roland Siegwart. A Comparison of Line Extraction Algorithms using 2D Range Data for Indoor Mobile Robotics. *Autonomous Robots*, 23(2):97–111, August 2007.

[8] T. Pavlidis and S.L. Horowitz. Segmentation of plane curves. *IEEE Transactions on Computers*, 23:860–970, 1974.

[9] A. Siadat, A. Kaske, S. Klausmann, M. Dufaut, and R. Husson. An optimized segmentation method for a 2d laser-scanner applied to mobile robot navigation. In *3rd IFAC Symposium on Intelligent Components and Instruments for Control Applications*, pages 153–158, 1997.