Line Extraction in 2D Range Images for Mobile Robotics

Geovany Araujo Borges (gaborges@ene.unb.br)*
Electrical Engineering Department - University of Brasilia (UnB)
Caixa Postal 04591 - Asa Norte - Brasilia - CEP 70910-900 - Brazil

Marie-José Aldon (aldon@lirmm.fr)
Robotics Department - LIRMM
UMR CNRS/Université Montpellier II, n°. C55060
161, rue ADA. 34392 - Montpellier - Cedex 5 - France

March, 2004

Abstract. This paper presents a geometrical feature detection framework for use with conventional 2D laser rangefinders. This framework is composed of three main procedures: data pre-processing, breakpoint detection and line extraction. In data pre-processing, low-level data organization and processing are discussed, with emphasis to sensor bias compensation. Breakpoint detection allows to determine sequences of measurements which are not interrupted by scanning surface changing. Two breakpoint detectors are investigated, one based on adaptive thresholding, and the other on Kalman filtering. Implementation and tuning of both detectors are also investigated. Line extraction is performed to each continuous scan sequence in a range image by applying line kernels. We have investigated two classic kernels, commonly used in mobile robots, and our Split-and-Merge Fuzzy (SMF) line extractor. SMF employs fuzzy clustering in a split-and-merge framework without the need to guess the number of clusters. Qualitative and quantitative comparisons using simulated and real images illustrate the main characteristics of the framework when using different methods for breakpoint and line detection. These comparisons illustrate the characteristics of each estimator, which can be exploited according to the platform computing power and the application accuracy requirements.

Keywords: Line extraction, breakpoint detection, 2D range images, local environments, environment modeling, mobile robotics, Fuzzy C-means, linear Kalman filter
Category: Other (Environment Modeling/Feature Extraction)

1. Introduction

General pattern recognition procedures and concepts have been applied for sensor data processing areas such as robotics and computer vision, requiring for some cases adaptations to deal with specific sensors and data organization. This is the case of geometrical primitives extraction in 2D range images provided by laser rangefinders, useful for local and global environment modeling in mobile robotics. For vehicles navigating in structured indoor environments, laser rangefinders are

* During this work G. A. Borges was partially supported by CAPES - Brasília, Brazil, under grant BEX2280/97-3.

well suitable sensors, capable to sample the local environment contour with a reasonable resolution and accuracy. Simple methods have been investigated and broadly used to support mobile robot navigation using line (Einsele, 1997) (Siadat et al., 1997)(Vandorpe et al., 1996)(Zhang and Ghosh, 2000) or point (Castellanos and Tardós, 1996)(Skrzypczynski, 1995) features extracted from range images. Nevertheless, adverse phenomena (e.g., clutter, false measurements on surface limits) inject incorrect information in the measurements, making feature extraction a very difficult task which cannot be solved by such methods. More robust segmentation methods using the Hough transform (Duda and Hart, 1973)(Shpilman and Brailovsky, 1999) have been proposed in the literature for infinite lines extraction in 2D range images. In order to obtain line segments, infinite lines should be delimited, as in (Arras et al., 2001)(Forsberg et al., 1995)(Jensfelt and Christensen, 1998). In (Pears, 2000) Pears presented a line segments extraction method which considers sensor motion. Motion is taken into account using a stochastic filtering technique. The detection of unconventional primitives is the original contribution of (Kwon and Lee, 1999) and (Skrzypczynski, 1997) who have investigated measurement clouds corresponding to artifacts exploitable for robot localization. In a previous paper (Borges and Aldon, 2000), we have introduced the Split-and-Merge Fuzzy (SMF) line extractor, which employs the fuzzy clustering approach presented in Section 5.3 in a split-and-merge framework. This algorithm achieves more accurate results by iterated partitioning of the set of measurements. An interesting approach for complex polygonal model extraction applied in mobile robot localization is presented in (Kämpke and Strobel, 2001).

In this system, there are two features of interest in 2D range images: breakpoints and line segments. Such features collect information about the environment, as follows :

- Breakpoints are scan measurements associated to discontinuities in the scanning process. Such discontinuities may arise from two distinct sources: (i) the absence of obstacles in the scanning direction, or (ii) change of surface being scanned by the sensor. In the sequel, the scan points resulting from the first source are sometimes referred to as *rupture points*;

- Line segments are supported by sets of points, (often) ordered consecutively according to the acquisition process. These features result from the scan of planar surfaces in the environment (e.g., walls).

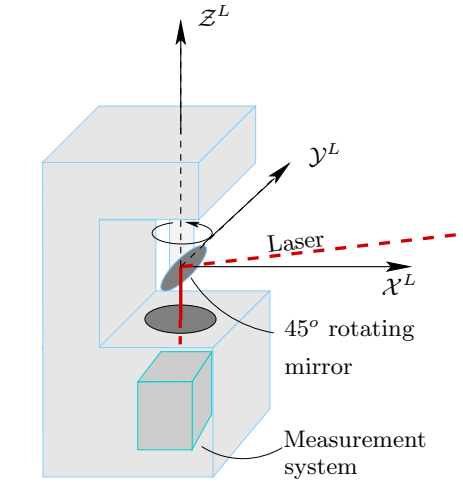
In what follows, we present a geometrical feature detection framework for use with conventional 2D laser rangefinders. This framework is composed of three main procedures: data pre-processing, breakpoint detection and line extraction, and has been in part inspired from (Castellanos and Tardós, 1996). For breakpoint detection, two algorithms are investigated: (i) an adaptive and (ii) a Kalman filter–based breakpoint detector. It is further investigated the use of classic (Duda and Hart, 1973; Siadat et al., 1997; Vandorpe et al., 1996) and SMF algorithms as line extraction kernels. Implementation details and tuning are discussed. Qualitative and quantitative comparisons using simulated and real images illustrate the main characteristics of the framework when using different methods for breakpoint and line detection. These comparisons illustrate the characteristics of each estimator, which can be exploited according to the platform computing power and accuracy requirements of the application. For applications such as navigation based on environmental features, accuracy may not be a determinig factor if feature uncertainties are correctly estimated. On the other hand, accuracy on feature extraction is an essential requirement for environment map building. Indeed, wrong feature detection may inject bias in the map update process and lead the map to diverge from the real environment.

This paper is organized as follows. Data acquisition and pre-processing are discussed in section 2. Section 3 presents the breakpoint detectors, followed by the line extraction procedures in section 4. Extensive simulations and experimental analysis are presented in sections 6 and 7, respectively.

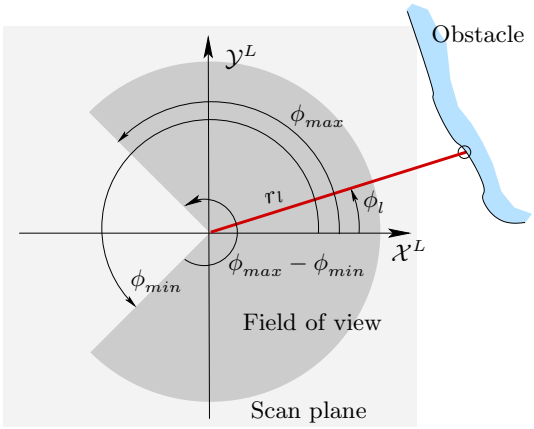
2. Acquisition and data pre-processing

The most common 2D laser rangefinder available in the market today are based on a fixed time-of-flight laser measurement system, which can only measure distances in a given direction. In order to allow measurements in a plane, a 45° rotating mirror redirects the laser beam through the scanning plane (see Figure 1(a)). For each scanning direction in the plane, when the emitted laser beam meets an obstacle, it is reflected back to the sensor. The time-of-flight of the laser beam is measured, allowing to estimate the distance of the obstacle to the sensor. The entire set of measurements taken at all sampling directions in the scanning plane composes a range image.

Usually, range images provided by commercial laser rangefinders are in the form $\mathcal{R} = \{(r_l, \phi_l) \mid l = 1, \ldots, N_{\mathcal{R}}\}$, on which (r_l, ϕ_l) are the polar coordinates of the *l-th* scan point (see Figure 1(b)). In physical terms,



(a)



(b)

Figure 1. (a) Laser rangefinder system and geometry. (b) Scan reference frame variables.

r_l is the measured distance of an obstacle to the sensor rotating axis at direction ϕ_l. The points are sequentially acquired by the laser range finder with a given angular resolution $\phi_l - \phi_{l-1} = \Delta\phi$. The scan starts at direction $\phi_{\min} = \phi_1$ and stops at $\phi_{\max} = \phi_{N_\mathcal{R}}$. As any physical system, r_l is perturbed by noise measurement ε_r, usually assumed to follow a Gaussian distribution with zero mean and variance σ_r^2. This approximation can be employed only if the sensor measurement bias ε_b, which depends of the real range, has been characterized. In our sensor, measurement characterization considers that r_m, the measured

distance, and r_v, the true obstacle distance, are related by

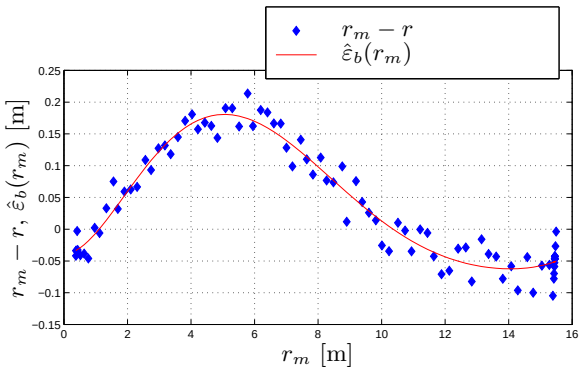
$$r_m = r_v + \varepsilon_b(r_m) + \varepsilon_r. \quad (1)$$

Using an auxiliary distance measurement system, measurements r of the true obstacle distance r_v are used to provide estimates $r_m - r$ of the sensor bias $\varepsilon_b(r_m)$, as shown in Figure 2(a). With our laser rangefinder, a 2D30 model from IBEO Lasertechnik Gmbh, we have observed a bias of about 25 cm for obstacles placed at 5 m. Such a bias is approximated by $\hat{\varepsilon}_b(r_m)$, a sixth-order polynomial which fits $r_m - r$ in the least-squares sense. This polynomial is used for direct compensation of each measurement according to model (1). It should be pointed out that the polynomial approximation has no physical root. The sixth-order polynomial was chosen since it well fitted the estimated bias, better than polynomials of lower order. The physical meaning of bias in laser rangefinders is related to internal automatic gain amplifier circuitry, which is responsible for amplitude-induced range error (Borenstein et al., 1996). Indeed, such amplifiers are a necessary part of the measurement system, given the large change in signal amplitude given the obstacle distance. The residual noise $\varepsilon_r = r_m - r_v - \hat{\varepsilon}_b(r_m)$ after bias correction is shown in Figure 2(b). The residual variance seems to slightly increase with the distance r_m, but is still compatible with the value $\sigma_r = 0.03$ m provided in the rangefinder datasheet.

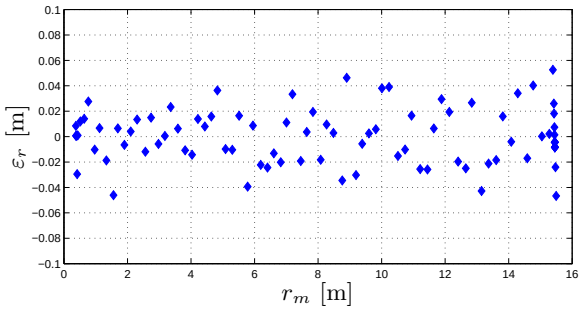
When range images are taken with the robot in motion, they may be deformed given the important scanning time. In such cases, other compensation procedures based on estimates of the robot motion should be applied. For instance, motion correction (Arras et al., 2001; Borges et al., 2001), data synchronization by software and computing time compensation (Borges and Aldon, 2001) are some of them. If there is no obstacle in the maximum sensor range at direction ϕ_l, $\bar{r}_l$ corresponds to a non-valid measure, *e.g.* a negative value or zero.

The enhancement of the range images after the pre-processing procedures of bias correction and motion compensation is illustrated in Figure 3**.** In Figure 3(a) we can see a set of range images acquired with the robot navigating in a real environment. The robot pose was estimated accurately using odometry and a high precision laser gyrometer. The environment walls in Figure 3(a) captured by range data are not well defined. It is clear that after bias compensation in Figure 3(b) and motion correction in Figure 3(c) the some environment details are accentuated. Further, consecutive feature extraction from these images are likely to preserve spatial location.

After the above procedures, images are stored in an equivalent form $\mathcal{C} = \{(p_n, \kappa_n^r, \kappa_n^b) \mid n = 1, \ldots, N_{\mathcal{C}}\}$, with $p_n = (x_n = r_{l^*} \cdot \cos(\phi_{l^*}), y_n =$



(a) Measured bias and estimated bias polynomial



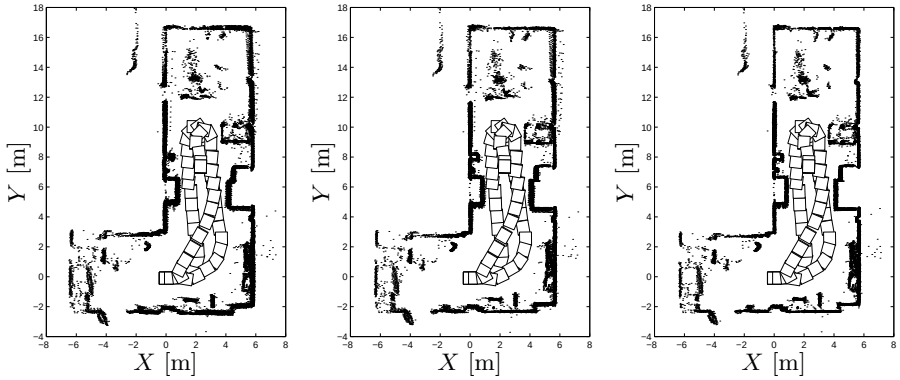
(b) Residual noise after bias compensation

Figure 2. Laser rangefinder measurements characterization curves.

$r_{l^*} \cdot \sin(\phi_{l^*})$) being the Cartesian coordinates of valid scan points whose index is l^*. Variables κ_n^r and κ_n^b are discontinuity flags. κ_n^r is a rupture flag, indicating the boundaries of a sequence of invalid range measurements. For instance, if in a range scan only the range points from index 21 to 25 are invalid, thus we have $N_\mathcal{R} - 5$ valid points with indices $l^* = 1, \ldots, 20, 26, \ldots, N_\mathcal{R}$. These points are used to generate the $N_\mathcal{C} = N_\mathcal{R} - 5$ of $\mathcal{C}$ with indices $n = 1, \ldots, 20, 21, \ldots, N_\mathcal{C}$, and only κ_{20}^r and κ_{21}^r are set to `TRUE`. κ_n^b is a flag corresponding to a breakpoint, as discussed in the next section.

3. Breakpoint detection

In this work, the terms rupture and breakpoint refer to points in the range image on which some kind of discontinuity has been detected. A rupture point indicates a discontinuity during the measurement. Evidently, the system cannot make inferences in regions where ruptures have occurred. On the other hand, breakpoints are detected by making



(a) Raw images (b) Bias compensation (c) Motion correction

Figure 3. Illustrative example of range image enhancement with pre-processing procedures: (a) A set of superimposed raw range images, followed by results from (b) bias compensation and (c) motion correction, in this order.

inferences about the possible presence of discontinuities in a sequence of valid range data.

Breakpoint detection is a procedure of great importance for the proposed line extraction system. Without such a task, the line extractor tends to connect two neighbor linear clusters, even if a large discontinuity exists between them. In real a environment, such discontinuity may correspond to an open door which should be detected. Thus, the purpose of the breakpoint detector is to test if there exists a discontinuity between consecutive range scan points, say p_n and p_{n-1}. In such a case, flags κ_n^b and κ_{n-1}^b should be set to `TRUE`.

Next, we discuss two algorithms which can be used as breakpoint detectors. The main interest on the use of such algorithms is on their simplicity and intuitive appeal.

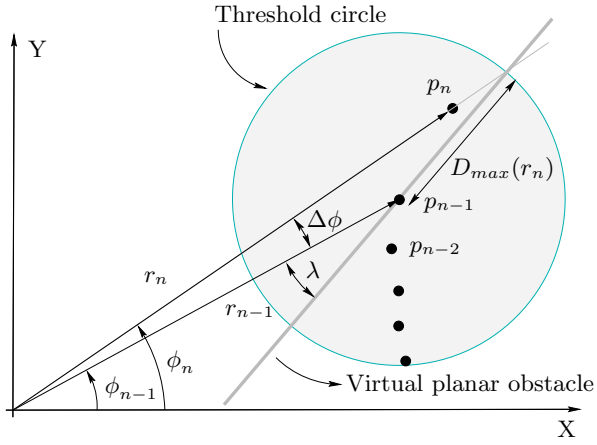
3.1. Adaptive breakpoint detector

The simplest breakpoint detector one can imagine is of the form

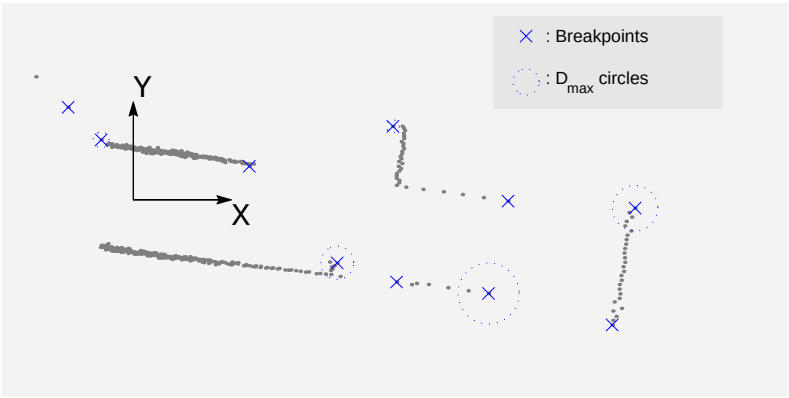
$$\text{if } \|p_n - p_{n-1}\| > D_{\max}, \text{ then } \kappa_n^b := \texttt{TRUE} \text{ and } \kappa_{n-1}^b := \texttt{TRUE},$$

where $\|p_n - p_{n-1}\|$ is the Euclidean distance between points p_n and p_{n-1}. We call this a constant threshold naïve detector. Such a detector and variants have been discussed in (Skrzypczynski, 1995). The main difficulty with these approaches is on the computation of the threshold $D_{\max}$. Of course, such a threshold should depend on the range scan distance r_n, since the sensor angular step is constant.

In this section, we propose an intuitive methodology to determine $D_{\max}$ according to r_n, as shown in Figure 4(a). In this methodology,



(a) Adaptive detector



(b) Real scan example

Figure 4. Breakpoint detection using the adaptive algorithm.

we define a virtual line passing on the scan point p_{n-1}, representing an extremum case where an environment line can be reliably detected. Such a virtual line makes an angle λ with respect to the scanning direction ϕ_{n-1} and aims to extrapolate the worst acceptable range point p_n. Under this constraint, a hypothetical range distance r_n^h for the n^{th} point is related to r_{n-1} by the following equation:

$$r_{n-1} \cdot \sin(\lambda) = r_n^h \cdot \sin(\lambda - \Delta\phi). \tag{2}$$

Further mathematical development results in

$$\left\| p_n^h - p_{n-1} \right\| = r_{n-1} \cdot \frac{\sin(\Delta\phi)}{\sin(\lambda - \Delta\phi)}. \tag{3}$$

One can use $\left\|p_n^h - p_{n-1}\right\|$ as a threshold for breakpoint test. However, since it does not take into account the noise associated to r_n, this test can fail for obstacles close to the sensor frame origin, *i.e.,* r_{n-1} is small. Thus, in our implementation, we have used

$$D_{\max} = \left\|p_n^h - p_{n-1}\right\| + 3\sigma_r \tag{4}$$

in order to encompass the stochastic behavior of r_n. Intuitively, the parameter λ corresponds to the worst case of incidence angle of the laser scan ray with respect to a line for which the scan points are still reliable. This can be determined with user experience. Thus, such a detector generates a threshold circle which is centered at p_{n-1} with radius $D_{\max}$ (see Figure 4(a)), which adapts according to r_n. If the next scan point p_n is outside this circle, p_n and p_{n-1} are both marked as breakpoints.

The implementation of the adaptive detector in shown below:

$\kappa_1^b \leftarrow$`FALSE`
for $n = 2$ to $N_{\mathcal{C}}$ **do**
    $D_{\max} \leftarrow r_{n-1} \cdot \frac{\sin(\Delta\phi)}{\sin(\lambda - \Delta\phi)} + 3\sigma_r$
    if $\|\mathbf{p}_n - \mathbf{p}_{n-1}\| > D_{\max}$ **then**
        /* *Breakpoints have been detected* */
        $\kappa_n^b \leftarrow$`TRUE`
        $\kappa_{n-1}^b \leftarrow$`TRUE`
    else
        $\kappa_n^b \leftarrow$`FALSE`
    end
end

In Figure 4(b) one has a real range scan on which the adaptive detector has been applied. The parameters used are $\sigma_r = 0.03\ m$, provided by the laser scan manufacturer, and $\lambda = 10^\circ$ which has shown to be satisfactory. Indeed, for our purposes, there is no interest in considering for line extraction scan points with an incidence angle smaller then $\lambda = 10^\circ$. Even if these results seem satisfactory, a more complete evaluation is presented in section 7.

3.2. KF-based breakpoint detector

A stochastic breakpoint detector has been briefly discussed by Castellanos *et al.* in (Castellanos and Tardós, 1996). In this paper, on which the authors refer to a procedure for finding homogeneous regions, a

Kalman filter is used in conjunction with a statistical test to verify whether two consecutive range points belong to the same region. The problem is described as tracking an approximated kinematic model from observations r_n. If the tracker fails, a new region has been detected. With few modifications, the second breakpoint detector presented in this paper is derived from (Castellanos and Tardós, 1996).

In this detector, a discrete system model is used to describe the dynamic behavior of the range measurements r_n as

$$\mathbf{x}_n = \mathbf{A}\mathbf{x}_{n-1} + \mathbf{w}_n, \tag{5}$$

$$z_n = \mathbf{C}\mathbf{x}_n + \mathbf{v}_n, \tag{6}$$

with $\mathbf{x}_n = (\, r_n \;\; \frac{dr_n}{d\phi} \,)^T$ being the system state, z_n being the measurement variable, and $\mathbf{w}_n$ and $\mathbf{v}_n$ being uncorrelated Gaussian noise with zero mean and covariance $\mathbf{Q}_n$ and R_n, respectively. Matrices $\mathbf{A}$ and $\mathbf{C}$ correspond to

$$\mathbf{A} = \begin{pmatrix} 1 & \Delta\phi \\ 0 & 1 \end{pmatrix}, \mathbf{C} = (\, 1 \;\; 0 \,) \, .$$

It can be pointed out that the transition model (5) is the simpler model of motion which can be used without prior knowledge about the trajectory to be performed by the laser scan. Target trackers of this form are discussed in (Bar-Shalom and Fortmann, 1987) assuming constant target velocity or acceleration. In the case of (5) such assumptions are fair, even for well structured environments, and this is reflected in the presence of non-modeled motion components. In such a case, the $\mathbf{Q}_n$ matrix is of great importance in keeping model consistency, *i.e.*, it has to be large enough to encompass the non-modeled motion components of the target. If this is not the case, excessive false breakpoints are detected and few line segments can be extracted. On the other hand, over-conservative $\mathbf{Q}_n$ matrix may result in no breakpoint detection at all. Thus, we propose the following KF-based algorithm for breakpoint detection:

$n \leftarrow 1$ /* *Initialize the current point index* */
$n_i \leftarrow 1$ /* *Initialize the preceding detected breakpoint point index* */
$\kappa_n^b \leftarrow$ `FALSE` for $n = 1, \ldots, N_{\mathcal{C}}$ /* *Initialize labels* */
while $n \leq N_{\mathcal{C}}$ **do**
    if $n_i = n$ **then**
        $\hat{\mathbf{x}}_n \leftarrow \mathbf{x}_0$ /* *Filter reset* */
        $\hat{\mathbf{P}}_n \leftarrow \mathbf{P}_0$
    else
        $\hat{\mathbf{x}}_{n|n-1} \leftarrow \hat{\mathbf{x}}_{n-1}$ /* *Prediction* */

$\hat{\mathbf{P}}_{n|n-1} \leftarrow \mathbf{A} \cdot \hat{\mathbf{P}}_n \cdot \mathbf{A}^T + \mathbf{Q}_n$
$v_n \leftarrow r_n - \mathbf{C} \cdot \hat{\mathbf{x}}_{n|n-1}$ /* *Measurement* */
$S_n \leftarrow \mathbf{C} \cdot \hat{\mathbf{P}}_{n|n-1} \cdot \mathbf{C}^T + \mathbf{R}_n$
if $v_n^2/S_n \geq 3.84$ /* χ_1^2 *compatibility test* */ **then**
/ Breakpoins have been detected */*
$\kappa_n^b \leftarrow$`TRUE` and $\kappa_{n-1}^b \leftarrow$`TRUE`
$n_i \leftarrow n$
$n \leftarrow n - 1$
else
/ The model matches. Perform state update */*
$\mathbf{K}_n \leftarrow \hat{\mathbf{P}}_{n|n-1} \cdot \mathbf{C}^T \cdot S_n^{-1}$
$\hat{\mathbf{x}}_n \leftarrow \hat{\mathbf{x}}_{n|n-1} + \mathbf{K}_n \cdot v_n$
$\hat{\mathbf{P}}_n \leftarrow \hat{\mathbf{P}}_{n|n-1} - \mathbf{K}_n \cdot \mathbf{C} \cdot \hat{\mathbf{P}}_{n|n-1}$
end
end
$n \leftarrow n + 1$
end

In the above algorithm, a χ_1^2 test is used to find measurements which do not satisfy in the stochastic sense the predicted model. If the test fails, a breakpoint is detected and the filter is re-initialized. As discussed above, keeping filter consistency is fundamental for good breakpoint detection. A known property of the linear Kalman filter is the conservativeness on propagation of the estimated covariance matrix given conservative $\mathbf{P}_0$, $\mathbf{Q}_n$ and R_n. It is given by the following theorem, adapted from (Jazwinski, 1970):

Theorem 1. If $\mathbf{P}_0 \geq \mathbf{P}_0^a$ and $\mathbf{Q}_n \geq \mathbf{Q}_n^a$, $R_n \geq R_n^a$, for all n, then $\hat{\mathbf{P}}_n \geq \mathbf{P}_n^a$ and $\hat{\mathbf{P}}_{n|n-1} \geq \mathbf{P}_{n|n-1}^a$ for all n, where $\mathbf{P}_0^a$, $\mathbf{Q}_n^a$, and R_n^a are the actual error covariance matrix, actual process covariance matrix and actual measurement covariance matrix, respectively.

In this application, R_n can be well modeled using sensor measurements. On the other hand, given the unknown behavior of r_n for arbitrary environments, we cannot guarantee the conservativeness of $\mathbf{Q}_n$. However, we have verified that the initialization of the system covariance matrix using a conservative $\mathbf{P}_0$ has a direct effect on the moment on which $\hat{\mathbf{P}}_n$ becomes inconsistent. We propose to apply the same concept of virtual line to compute a conservative $\mathbf{P}_0$ associated to the scan point n on which a breakpoint has been detected. To do so, we assume that the n^{th} scan point is on the virtual line that makes an angle λ with the direction ϕ_n. The polar coordinates (r_n, ϕ_n) of the n^{th}

scan point satisfy

$$r_n \cdot \cos(\alpha - \phi_n) = \text{const.},$$

with α being the direction angle of the line normal vector. Since $\alpha - \phi_n = \lambda + \pi/2$, we have

$$\frac{dr_n}{d\phi} = \cot(\lambda) \cdot r_n. \tag{7}$$

Eq. (7) can be seen as an upper bound for $dr_n/d\phi$. The lower bound is $-\cot(\lambda) \cdot r_n$. Given such bounds, the next scan point $n+1$ should be in the interval $[r_n - \cot(\lambda) \cdot r_n \cdot \Delta\phi \, , r_n + \cot(\lambda) \cdot r_n \cdot \Delta\phi]$. Without prior knowledge of the behavior of the scan sequence starting from the n^{th} point, we guess the initial state as

$$\mathbf{x}_0 = (\ r_n \ \ 0\ )^T.$$

Given the bounds obtained from the virtual line, we propose an empirical initialization of the covariance matrix as

$$\mathbf{P}_0 = \frac{1}{9} \begin{pmatrix} (\cot(\lambda) \cdot r_n \cdot \Delta\phi)^2 & 0 \\ 0 & (\cot(\lambda) \cdot r_n)^2 \end{pmatrix} \tag{8}$$

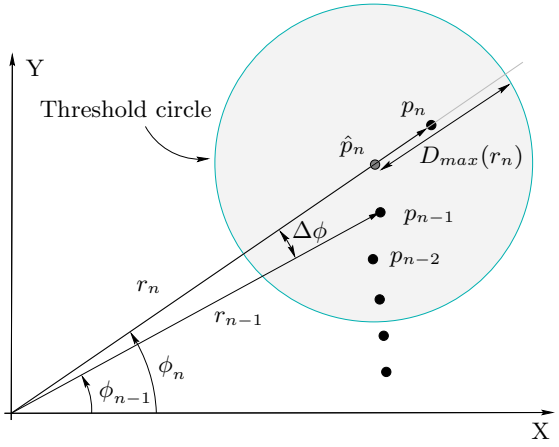
which is a conservative initialization of the system model based on the bound (7). In the final implementation, we have used $\mathbf{Q}_n = 0.1^2 \cdot \mathbf{I}_2$, where $\mathbf{I}_2$ is the identity matrix of size 2×2.

This detector generates a threshold circle which is centered at $\hat{p}_n = (\hat{r}_n, \phi_n)$ with radius $D_{\max} = \sqrt{3.84 \cdot S_n}$ (see Figure 5(a)). Figure 5(b) shows a real range scan on which the KF-based algorithm has been applied. The parameters used are $R_n = \sigma_r^2 = (0.03\ m)^2$, and $\lambda = 10^\circ$, and the results seem satisfactory, even if different from that presented by the adaptive detector. A more complete evaluation is presented in section 7.

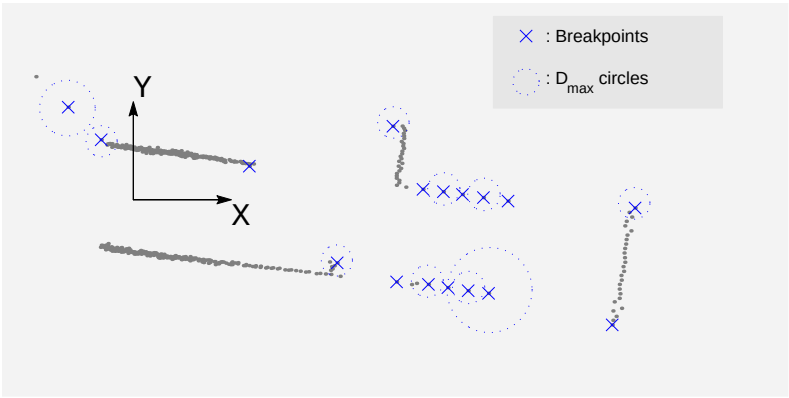
4. Line extraction

In this system, the useful *divide and conquer* concept is used in order to gain computing time and avoid connecting close linear clusters when applying classical line extraction algorithms. With the range discontinuities detected, line extraction is performed within the regions of points between two discontinuities, as in (Castellanos and Tardós, 1996). Therefore, a list of extracted lines $\mathcal{L} = \{\mathbf{l}_k \mid k = 1, \ldots, N_\mathcal{L}\}$ is incrementally updated during the extraction process. The line parameter vector $\mathbf{l}_k$ is given by

$$\mathbf{l}_k = (\rho_k, \alpha_k, \mathbf{C}_k, x_k^a, y_k^a, x_k^b, y_k^b), \tag{9}$$



(a) KF-based detector



(b) Real scan example

Figure 5. Breakpoint detection using the KF-based algorithm.

where ρ_k and α_k are the polar parameters of the *k-th* line such that $\rho_k \geq 0$ and $0 \leq \alpha_k \leq 360°$, $\mathbf{C}_k$ is the covariance matrix associated to the vector $(\rho_k, \alpha_k)^T$. The extremities of the *k-th* line have as coordinates (x_k^a, y_k^a) and (x_k^b, y_k^b).

The task of building the list L by extracting lines only on regions of continuous surfaces, previously identified using breakpoint detection, can be performed with the algorithm below:

/* *Initialization* */
$\mathcal{L} \leftarrow \emptyset$
$n_e \leftarrow 1$
/* *Main loop* */
while $n_e \leq N_\mathcal{C}$ **do**

$n_i \leftarrow n_e$ /* *Index of the first point of the region* */
$n_e \leftarrow n_i + 1$ /* *Search the last point of the region from* $n_i + 1$ */
while $\kappa^b_{n_e} =$ `FALSE` **and** $\kappa^r_{n_e} =$ `FALSE` **do** /* *Continuous region* */
  $n_e \leftarrow n_e + 1$
  if $n_e = N_{\mathcal{C}}$ **then** /* *Take care of end of image* */
    break
  end
end
if $(n_e - n_i + 1) > N_{\min}$ **then**
  $\mathcal{L}_* \leftarrow \Phi(I^T, n_i, n_e)$ /* *Extract lines from the current region* */
  $\mathcal{L} \leftarrow \Omega^S \cup \Omega^S_*$ /* *Add the lines to the main list* */
end
end
SELECTION($\mathcal{L}$) /* *Apply selection criteria* */

In this algorithm, the kernel Φ is in fact a commonly used line extractor, which may be applied for the whole scan despite of the existence of breakpoints. However, as discussed above, reducing computing time as well as avoiding to fuse close linear clusters are the main objectives of this system. With the framework presented here, classical line extractors can be used without the need to change its execution strategy in order to adapt them for avoiding close linear clusters fusion. Some kernels for line extraction are presented in section 5.

Once all lines have been extracted, a line selection criteria (i.e., SELECTION) is applied. This procedure preserves only the lines satisfying the following criteria:

- *Minimum length criterion*: all line segments must have a minimum length of $l_{\min}$;
- *Minimum number of support points criterion*: all line segments must have a minimum number of $N_{\min}$ support points;
- *Continuity criterion*: the sequence of support points of each line must not have breakpoints. This criterion is verified by applying one of the breakpoint detectors to the support points.

5. Kernels for line extraction

The kernels for line extraction investigated in this paper compute the line parameters ρ_k and α_k as the ones that minimize the following weighted cost function:

$$J_k(\rho_k, \alpha_k, \mathcal{Q}, \mathcal{U}) = \sum_{n|q_n=k} u_n^m d^2(x_n, y_n, \rho_k, \alpha_k), \tag{10}$$

where m is a positive constant and

$$d(x_n, y_n, \rho_k, \alpha_k) = \rho_k - x_n \cos(\alpha_k) - y_n \sin(\alpha_k), \tag{11}$$

is the signed orthogonal distance between the *k-th* line, represented by the parameters (ρ_k,α_k), and the scan point (x_n, y_n). The line extraction kernel also provides the entries of $\mathcal{Q} = \{q_n, n = 1, \ldots, N_\mathcal{C}\}$ and $\mathcal{U} = \{u_n, n = 1, \ldots, N_\mathcal{C}\}$. In $\mathcal{Q}$, q_n contains the index of the line extracted using the *n-th* valid scan point. If $q_n = -1$, it indicates that the *n-th* scan point is not used as support point of any extracted line. In $\mathcal{U}$, u_n is the weight associated to the correspondence between the *n-th* scan point and the q_n*-th* line.

The optimization-based line extractors used in this work computes the line parameters using an equation of the form (10). The use of such unified form allows us to directly compute the line parameters (ρ_k,α_k) and the associated covariance matrix $\mathbf{C}_k$. Thus, by solving $\partial J_k/\partial\rho_k = \partial J_k/\partial\alpha_k = 0$ we have

$$\rho_k = \tilde{x}_k \cos(\alpha_i) + \tilde{y}_k \sin(\alpha_i), \tag{12}$$

$$\alpha_k = \frac{1}{2} \arctan\left(\frac{-2\tilde{S}_{xy_k}}{\tilde{S}_{yy_k} - \tilde{S}_{xx_k}}\right), \tag{13}$$

where

$$\begin{aligned} \tilde{x}_k &= \frac{\sum_{n|q_n=k} u_n^m x_n}{\sum_{n|q_n=k} u_n^m}, \quad \tilde{y}_k = \frac{\sum_{n|q_n=k} u_n^m y_n}{\sum_{n|q_n=k} u_n^m}, \\ \tilde{S}_{xx_k} &= \sum_{n|q_n=k} u_n^m (x_n - \tilde{x}_k)^2, \quad \tilde{S}_{yy_k} = \sum_{n|q_n=k} u_n^m (y_j - \tilde{y}_k)^2, \\ \tilde{S}_{xy_k} &= \sum_{n|q_n=k} u_n^m (x_n - \tilde{x}_k) \cdot (y_n - \tilde{y}_k). \end{aligned} \tag{14}$$

In Eqs. (14), $\tilde{x}_k$ and $\tilde{y}_k$ are the weighted center of gravity coordinates of the cluster of points corresponding to the *k-th* line. It must be pointed out that for the special case of all u_n's and m equal to 1, eqs. (12) and (13) are the same as for the classical non-robust least-squares line fit.

The covariance matrix $\mathbf{C}_k$ associated to the line parameters vector $(\rho_k, \alpha_k)^T$ is computed using the Haralick's method (Haralick, 1994). Such a method is directly adapted to propagate the variances of regressors (*i.e.* input variables (x_n, y_n)) to the estimates which minimize a cost function of the form (10). It also allows to take into account the estimation residuals, which are also modeled as random Gaussian variables with zero mean and covariance estimated by sampling the residuals as

$$\mathbf{C}_{\mathbf{d}_k} = \frac{\sum_{n|q_n=k} u_n^m d^2(x_n, y_n, \rho_k, \alpha_k)}{\sum_{n|q_n=k} u_n^m} = \frac{J_k(\rho_k, \alpha_k, \mathcal{Q}, \mathcal{U})}{\sum_{n|q_n=k} u_n^m}.$$

The covariance matrix $\mathbf{C}_{\mathbf{p}_n}$ associated to the scan point is computed from the range scan variance σ_r^2 as

$$\mathbf{C}_{\mathbf{p}_n} = \sigma_r^2 \cdot \begin{pmatrix} \cos(\alpha_n) \\ \sin(\alpha_n) \end{pmatrix} \cdot \begin{pmatrix} \cos(\alpha_n) & \sin(\alpha_n) \end{pmatrix},$$

with $\alpha_n = \arctan(y_n/x_n)$. Thus we consider only the errors associated to the range measurement, since the errors on ϕ_n can be neglected w.r.t. to that compared to r_n.

Hence, according with (Haralick, 1994), we have

$$\mathbf{C}_k = \left(\frac{\partial \mathbf{g}_k}{\partial(\rho_k, \alpha_k)^T}\right)^{-1} \cdot (\mathbf{\Lambda_d} + \mathbf{\Lambda_p}) \cdot \left(\frac{\partial \mathbf{g}_k}{\partial(\rho_k, \alpha_k)^T}\right)^{-1}, \qquad (15)$$

with $\mathbf{g}_k = \partial J_k / \partial(\rho_k, \alpha_k)^T$ and

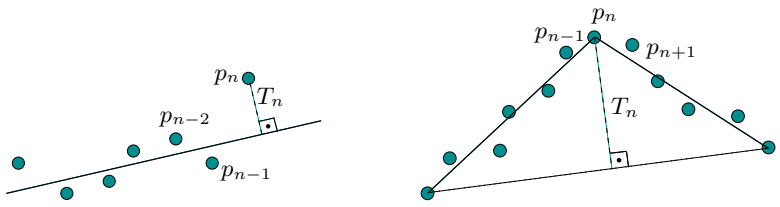
$$\mathbf{\Lambda_d} = \sum_{n|q_n=k} \frac{\partial \mathbf{g}_k}{\partial d(x_n, y_n, \rho_k, \alpha_k)} \cdot \mathbf{C}_{\mathbf{d}_k} \cdot \frac{\partial \mathbf{g}_k^T}{\partial d(x_n, y_n, \rho_k, \alpha_k)}, \quad (16)$$

$$\mathbf{\Lambda_f} = \sum_{n|q_n=k} \frac{\partial \mathbf{g}_k}{\partial \mathbf{p}_n} \cdot \mathbf{C}_{\mathbf{p}_n} \cdot \frac{\partial \mathbf{g}_k^T}{\partial \mathbf{p}_n}. \qquad (17)$$

In the sequel, we discuss the kernels for line extraction.

5.1. Line tracking (LT)

The line tracking (LT) kernel process the data points sequentially (Siadat et al., 1997; Vandorpe et al., 1996). Starting from the first two points, it computes a first guess for the *k-th* line formed by the support points $\mathbf{p}_{n_i}$ and $\mathbf{p}_{n_i+1}$ (c.f., Fig. 6(a)). If the distance T_{n_i+2} of the point $\mathbf{p}_{n_i+2}$ to the line is smaller than a threshold $T_{\max}$, then $\mathbf{p}_{n_i+2}$ is accepted as a support point of the *k-th* line, and its parameters are updated from all confirmed support points. The *k-th* is terminated when



(a) LT algorithm (b) IEPF algorithm

Figure 6. Classical algorithms for line extraction.

for some point $\mathbf{p}_{n_a}$, $T_{n_a} > T_{\max}$, and a new line starts with support points $\mathbf{p}_{n_a}$ and $\mathbf{p}_{n_a+1}$. The algorithm finishes when $\mathbf{p}_{n_e}$ is reached.

The algorithm described above is a very fast and adaptive approach for line extraction. However, it is very difficult to choose the threshold $T_{\max}$. Since it does not take into account the residuals of the already fitted line, it tends to add scan points obtained from curvilinear surfaces as support points of a given line. Evidently, this is not a suitable behavior for a line extraction kernel.

5.2. Iterative end-point fit (IEPF)

A well known recursive algorithm for line extraction is the iterative end point fit (IEPF) (Duda and Hart, 1973). This algorithm recursively splits the set of points $\mathcal{P} = \{p_{n_i}, p_{n_i+1}, \ldots, p_{n_e}\}$ into two subsets $\mathcal{P}' = \{p_{n_i}, ..., p_{n_a}\}$ and $\mathcal{P}'' = \{p_{n_a}, ..., p_{n_e}\}$ if a validation criterion is not satisfied. p_{n_a} is the point for which the distance T_{n_a} to the line formed by the extreme points of $\mathcal{P}$ is maximum (c.f., Fig. 6(b)). The criterion validates lines with $T_{n_a} \leq T_{\max}$. This is a recursive algorithm because the same procedure is repeated to $\mathcal{P}\,'$ and $\mathcal{P}\,''$ until all validation criteria are satisfied. Due to its good performance and being computationally inexpensive, the IEPF and its derivatives are commonly used for line extraction in range images (Einsele, 1997).

5.3. Prototype-based fuzzy clustering

The main objective of prototype-based fuzzy clustering algorithms is to reduce iteratively a cost function J. Most algorithms use a cost function given by

$$J(\beta, \mathcal{U}, \mathcal{Z}) = \sum_{i=1}^{C} \sum_{j=1}^{N} u_{ij}^m d^2(x_j, y_j, \beta_i). \tag{18}$$

In eq. (18), β_i represents the parameters of the *i-th* prototype and u_{ij}^m is the grade of membership of the *j-th* point (x_j, y_j) to the prototype β_i. $d(x_j, y_j, \beta_i)$ is a distance function between the point (x_j, y_j) and the

prototype β_i and m is a constant. The fuzzy clustering algorithm also imposes the following constraints:

$$u_{ij} \in [0,1], \quad 0 < \sum_{j=1}^{N} u_{ij} < N \text{ for all } i, \tag{19}$$

$$\sum_{i=1}^{C} u_{ij} = 1 \text{ for all } j, \tag{20}$$

For the purpose of straight lines extraction, there exist different representations for the prototypes β_i based on the cluster covariance matrix (Bezdek, 1981; Frigui and Krishnapuram, 1999). However, the new approach proposed in this paper uses a more compact representation. This representation is given by $\beta = \{(\rho_i, \alpha_i) \mid i = 1, \ldots, C\}$, where $\beta_i = (\rho_i, \alpha_i)$ are the polar parameters of the *i-th* straight line. From the polar representation of a line $\rho = x\cos(\alpha) + y\sin(\alpha)$ (Young and Fu, 1986), a suitable choice for the distance function d is

$$d^2(x_j, y_j, \beta_i) = f^2(x_j, y_j, \beta_i) + g^2(x_j, y_j, \beta_i), \tag{21}$$

where $f(x_j, y_j, \beta_i) = (\rho_i - x_j\cos(\alpha_i) - y_j\sin(\alpha_i))$, and $g(x_j, y_j, \beta_i)$ is a penalty function for points which are very far from the cluster center.

Partitioning the data set $\mathcal{Z}$ into C prototypes β is accomplished by minimizing the objective function J (eq. (18)). In order to consider the constraints (20), the Lagrange multipliers method is used and the problem becomes to minimize

$$V(\beta, \mathcal{U}, \mathcal{Z}) = J(\beta, \mathcal{U}, \mathcal{Z}) + \sum_{j=1}^{N} \lambda_j \left(1 - \sum_{i=1}^{C} u_{ij}\right), \tag{22}$$

where the λ_j's are the Lagrange's multipliers. V is minimized by using the alternating optimization method. Therefore, by considering β as constant, u_{ij} is determined as the one that satisfies $\partial V(\beta, \mathcal{U}, \mathcal{Z})/\partial u_{ij} = 0$ and $\partial V(\beta, \mathcal{U}, \mathcal{Z})/\partial \lambda_j = 0$. Thus, u_{ij} is given by

$$u_{ij} = \frac{1}{\sum_{k=1}^{C} \left[\frac{d^2(x_j, y_j, \beta_i)}{d^2(x_j, y_j, \beta_k)}\right]^{\frac{1}{m-1}}}. \tag{23}$$

In order to determine the parameters of the prototype β_i, the membership measures u_{ij} are considered as constants and the following equations set is solved: $\partial J(\beta, \mathcal{U}, \mathcal{Z})/\partial \beta_i = \mathbf{0}$. Therefore, using eqs. (18) and (21), it can be shown that ρ_i and α_i are given by eqs. (12) and (13). In the above development, the penalty function g was considered as a constant and given by

$$g^2(x_j, y_j, \beta_i) = (x_j - \tilde{x}_i)^2 + (y_j - \tilde{y}_i)^2, \tag{24}$$

where $\tilde{x}_i$ and $\tilde{y}_i$ are the weighted center of gravity coordinates of the *i-th* prototype from the last iteration (eq. (14)).

Given the data set $\mathcal{Z}$, the number of lines C and the initial prototypes β, the prototype-based fuzzy line extraction algorithm is summarized as follows:

$i = 0$ /* *Iteration counter* */
do
Update u_{ij} for $i = 1, \ldots, C$ and $j = 1, \ldots, N$, using eq. (23)
Update β_i for $i = 1, \ldots, C$, using eqs. (12) and (13)
$i \leftarrow i + 1$
while change in prototypes greater than a given threshold **and** $i \leq i_{\max}$

The main drawbacks of the fuzzy C-means are

- *The sensitivity to the initial prototypes* β. This subject is treated by Peña *et al.* (Peña et al., 1999) in a work presenting four initialization methodologies;
- *The non robustness to outliers*, in part due to the constraints (19)-(20). The possibilistic version of C-means (Barni et al., 1996) seems to be more robust since there are no strong constraints applied to the partitions. Some techniques from robust statistics have been also applied to increase robustness to outliers (Jolion et al., 1991)(Davé and Krishnapuram, 1997) ;
- *The number C of prototypes must be known a priori.* Nevertheless, in most applications C is not known in advance. It must be estimated or adapted as the algorithm iterates (Davé and Krishnapuram, 1997).

5.4. Split-and-merge fuzzy (SMF)

In (Borges and Aldon, 2000) we have introduced the Split-and-Merge Fuzzy (SMF) line extractor, which employs the fuzzy clustering approach presented in Section 5.3 in a split-and-merge framework. The main advantage of this framework is that the number of clusters need not to be known in advance, as in fuzzy C-means.

SMF is divided into consecutive split and merge phases. The split phase was inspired on *IEPF*, where for each iteration the set of scan points $\mathcal{P} = \{p_{n_i}, p_{n_i+1}, \ldots, p_{n_e}\}$ is divided into two sets $\mathcal{P}'$ and $\mathcal{P}''$ if a criterion is not satisfied. In the SMF, $\mathcal{P}'$ and $\mathcal{P}''$ are obtained

by applying the prototype-based fuzzy clustering with $C = 2$. The initial prototypes are initialized as follows: β_1 is given by the line which passes by the two first points of $\mathcal{P}$, and β_2 is obtained in the same manner with the two last points. In this way, $\mathcal{P}$ is divided into $\mathcal{P}'$ and $\mathcal{P}''$, which are determined based on the grade memberships u_{1n} and u_{2n}: $\mathbf{p}_n$ belongs to $\mathcal{P}'$ if $u_{1n} \geq u_{2n}$, or to $\mathcal{P}''$ on the contrary. Differently from *IEPF*, it should be observed that the sets $\mathcal{P}'$ and $\mathcal{P}''$ may be fragments of $\mathcal{P}$ which do not respect the order as points have been acquired. For instance, given $\mathcal{P}' = \{p_1, \ldots, p_{10}\}$ we may have $\mathcal{P}' = \{p_1, \ldots, p_3, p_8, \ldots, p_{10}\}$ and $\mathcal{P}'' = \{p_4, \ldots, p_7\}$. This allows to reject incompatible measurements present in cluttered environments.

The split procedure is applied once again to P' if the dispersion measure with respect to β_1 given by

$$\sigma^2 = \frac{\sum_{n|\mathbf{p}_n \in \mathcal{P}'} \left(\rho_1 - x_n \cos(\alpha_1) - y_n \sin(\alpha_1)\right)^2}{\sum_{n|\mathbf{p}_n \in \mathcal{P}'}}, \tag{25}$$

does not satisfy $\sigma \leq \sigma_{\max}$. The same can be done with P'' if its dispersion test with respect to β_2 is not satisfied. This process continues until all prototypes satisfy (25). Thus, the only user supplied parameter is the threshold $\sigma_{\max}$, which may be determined given a statistical analysis of the range sensor errors. At the end of the split phase, it is common to obtain an important number of line segments. The main advantage of this phase is that even small line segments are detected, which allows to identify parts of occluded lines in cluttered environments, which are of very difficult observation.

In the merge phase, for each line $\mathbf{l}_k$ in $\mathcal{L}$ two other lines $\mathbf{l}_a$ and $\mathbf{l}_b$ are chosen as fusion candidates. These candidates are the two closest lines to $\mathbf{l}_k$ given the distance of their center of gravity coordinates. Let the fusion of two lines $\mathbf{l}'$ and $\mathbf{l}''$ be represented by $\mathbf{l} = \mathbf{l}' \oplus \mathbf{l}''$ and defined as the line $\mathbf{l}$ fit by the support points of $\mathbf{l}'$ and $\mathbf{l}''$. The fused line $\mathbf{l}_f$ is the one that gives the smallest dispersion between $\mathbf{l}_k \oplus \mathbf{l}_a$ and $\mathbf{l}_k \oplus \mathbf{l}_b$ and that also satisfies the dispersion criterion $\sigma_f \leq \sigma_{\max}$, as well no discontinuities exist in them. The discontinuity test is done using a breakpoint detector. The merge phase exits when there is no more possible fusions.

6. Simulated data evaluation

This evaluation is based on the simulation of a laser rangefinder mounted on a mobile robot navigating in a structured environment as shown in Figure 7. It aims to provide evidence on the use of the different line

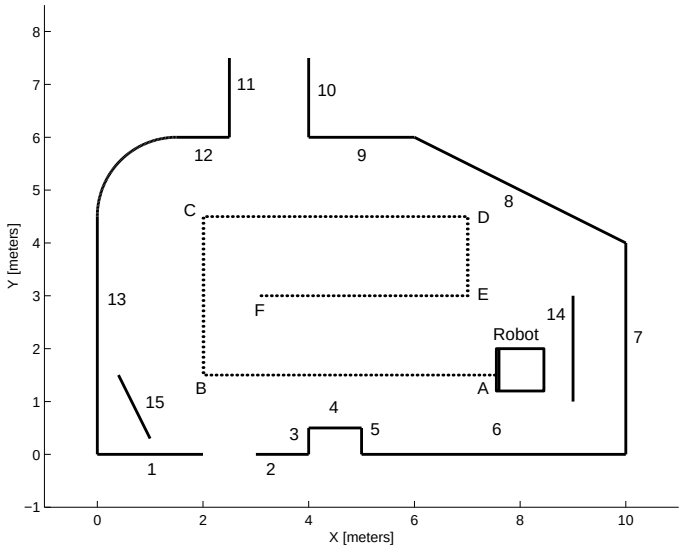


Figure 7. Simulation environment.

extraction kernels in the proposed system, as well as on the robustness of the different approaches with respect to the sensor noise. This is why two simulations were performed: one assuming a laser rangefinder with $\sigma_r = 0.03\ m$, and another with $\sigma_r = 0.05\ m$. The KF-based breakpoint detector was used. Simulations using the adaptive breakpoint detector were carried out, but the results obtained were very close **to** that given with the KF-based breakpoint detector. However, the main differences were verified between the line extractors.

In both simulations, a mobile robot equipped with a laser range finder moves following the trajectory *ABCDEF* among the obstacles represented by the lines numbered from *1* to *15* (Figure 7). During the overall trajectory, 1000 synthetic range images were generated at intermediary positions. For each range image, the three line extraction kernels LT, IEPF and SMF were executed. The range finder has an angular field of view of 270^o and a resolution of 0.55^o. These sensor parameters are the same as those of the *Ladar 2D30 IBEO Lasertechnik*. The measured distances are contaminated with a Gaussian random noise whose standard deviation is σ_r. It should be pointed out that spurious measures due to angular aperture of the laser ray were not simulated, *i.e.*, at each direction only one obstacle was observed. The parameters employed in the simulations were $T_{\max} = 10\ cm$ for the LT and IEPF algorithms, and $\sigma_{\max} = \sigma_r$ for the SMF algorithm.

In order to illustrate the simulation results, four quality measures were evaluated. These measures are statistics related to the lines ex-

Table I. Results on $\bar{N}_l$, $\bar{\eta}_l$, $\bar{\varepsilon}_\rho$, and $\bar{\varepsilon}_\alpha$ for the simulation using $\sigma_r = 0.03$ m.

e_l	$\bar{N}_l$			$\bar{\eta}_l$			$\bar{\varepsilon}_\rho$ in cm			$\bar{\varepsilon}_\alpha$ in degrees		
index	LT	IEPF	SMF	LT	IEPF	SMF	LT	IEPF	SMF	LT	IEPF	SMF
1	**1.00**	1.02	**1.00**	0.57	0.57	**0.58**	**1.45**	1.67	1.49	**0.61**	0.68	0.62
2	**1.01**	1.06	1.02	**0.61**	0.55	0.58	**8.13**	8.33	8.26	**1.36**	1.41	1.38
3	**1.00**	1.01	**1.00**	0.86	**0.98**	0.97	10.14	**2.28**	3.74	8.59	**3.13**	4.10
4	**1.02**	1.06	1.04	**0.99**	0.98	0.98	9.88	**8.03**	8.14	1.27	**1.03**	1.04
5	**1.00**	1.01	1.01	0.91	**1.00**	0.97	4.85	1.95	**1.34**	9.02	**3.46**	3.49
6	1.07	1.65	**1.05**	**0.49**	0.41	0.46	2.64	6.51	**2.56**	0.24	0.60	**0.23**
7	**1.06**	1.15	1.07	**1.00**	0.99	**1.00**	6.71	6.68	**6.29**	1.26	1.28	**1.19**
8	**1.04**	1.76	1.16	**0.98**	0.94	**0.98**	4.54	5.92	**3.63**	0.66	0.93	**0.57**
9	**1.04**	1.22	1.09	**0.99**	0.95	0.98	**4.49**	5.31	4.60	**0.52**	0.61	0.53
10	**1.00**	**1.00**	**1.00**	**1.00**	0.99	**1.00**	**4.29**	5.55	5.77	**0.36**	0.47	0.49
11	**1.00**	**1.00**	**1.00**	0.97	0.99	**1.00**	13.41	**7.37**	8.04	1.11	**0.62**	0.68
12	**1.03**	1.13	1.19	**0.98**	**0.98**	0.93	14.06	**7.18**	7.94	3.73	**2.00**	2.24
13	**1.30**	1.98	1.38	0.24	**0.37**	0.33	**1.52**	6.94	2.60	**0.35**	1.31	0.58
14	**1.02**	1.04	**1.02**	**1.00**	**1.00**	**1.00**	**2.19**	2.51	2.33	**0.56**	0.66	0.60
15	**1.02**	1.08	1.06	**1.00**	**1.00**	**1.00**	**0.97**	1.31	1.23	**0.87**	1.16	1.11

tracted from each environment line $\mathbf{e}_l = (\rho_l^e, \alpha_l^e)$, $l = 1, \ldots, 15$. Let $\mathcal{L}_l \in \mathcal{L}$ be a sub-list of N_l extracted lines with support points generated by $\mathbf{e}_l$. An extracted line cannot belong to more than one set $\mathcal{L}_l$. However a given extracted line $\mathbf{l}_k$ may have support points generated from different environment lines. In this case, $\mathbf{l}_k$ is associated with the sub-list of the more similar environment line in the polar parameters sense. As a given environment line $\mathbf{e}_l$ may be associated to one or more extracted lines as consequence of the algorithm behavior, it is interesting to measure the validity of such extracted lines. In doing so, a given line $\mathbf{l}_k$ extracted from an environment line $\mathbf{e}_l$ is valid iff $\varepsilon_\rho \leq 0.5$ m and $\varepsilon_\alpha \leq 20^o$, with $\varepsilon_\rho = \|\rho_l^e - \rho_k\|$ and $\varepsilon_\alpha = \|\alpha_l^e - \alpha_k\|$ being the absolute errors on the line parameters. Therefore, M_l being the number of valid extracted lines from the sub-list $\mathcal{L}_l$, let define the validation ratio $\eta_l = M_l/N_l$, with $0 \leq \eta_l \leq 1$. η_l is close to 1 when the extracted lines look like the corresponding environment line $\mathbf{e}_l$.

Table I presents the mean values $\bar{N}_l$, $\bar{\eta}_l$, $\bar{\varepsilon}_\rho$ and $\bar{\varepsilon}_\alpha$ of N_l, η_l, ε_ρ and ε_α, respectively, computed from the extracted lines of each line extraction kernel for the run with $\sigma_r = 0.03$ m. The values in bold font indicate the best performances for each quality measure (in some cases, more than one algorithm gave the best result). In this simulation, all algorithms presented similar performance measures $\bar{N}_l$ and $\bar{\eta}_l$. This

Table II. Results on $\bar{N}_l$, $\bar{\eta}_l$, $\bar{\varepsilon}_\rho$, and $\bar{\varepsilon}_\alpha$ for the simulation using $\sigma_r = 0.05\ m$.

e_l	$\bar{N}_l$			$\bar{\eta}_l$			$\bar{\varepsilon}_\rho$ in cm			$\bar{\varepsilon}_\alpha$ in degrees		
index	LT	IEPF	SMF	LT	IEPF	SMF	LT	IEPF	SMF	LT	IEPF	SMF
1	1.11	1.14	**1.01**	0.61	**0.62**	**0.62**	4.82	6.32	**2.62**	1.96	2.62	**1.08**
2	1.39	1.39	**1.08**	0.43	0.39	**0.59**	19.78	20.04	**15.93**	3.39	3.43	**2.70**
3	**1.01**	1.02	1.03	0.93	**0.97**	0.87	7.10	**3.25**	9.02	7.15	**5.30**	8.35
4	1.36	1.38	**1.11**	0.76	0.66	**0.92**	17.44	17.46	**13.28**	2.24	2.25	**1.71**
5	1.02	1.02	**1.01**	0.88	**0.94**	0.92	**3.94**	4.04	4.13	7.30	**5.02**	6.80
6	2.74	3.18	**1.17**	0.32	0.26	**0.44**	12.86	18.71	**5.61**	1.15	1.62	**0.47**
7	1.23	1.25	**1.09**	0.90	0.84	**0.98**	15.81	19.07	**9.90**	3.09	3.90	**1.92**
8	2.59	3.30	**1.36**	0.76	0.66	**0.97**	12.23	17.78	**5.91**	1.91	2.88	**0.89**
9	1.77	2.04	**1.16**	0.74	0.60	**0.94**	14.22	17.95	**8.25**	1.68	2.11	**0.96**
10	**1.01**	1.06	**1.01**	**0.98**	0.95	0.95	**8.91**	10.56	13.37	**0.76**	0.90	1.14
11	**1.00**	1.01	1.02	0.89	**0.94**	**0.94**	14.90	**11.86**	13.55	1.26	**1.01**	1.14
12	1.33	1.36	**1.20**	0.88	0.87	**0.91**	14.93	12.66	**12.01**	4.46	4.41	**3.28**
13	2.53	2.84	**1.33**	**0.35**	**0.35**	0.30	14.52	20.74	**3.73**	3.05	4.12	**0.79**
14	1.26	1.36	**1.02**	0.97	0.95	**0.99**	8.33	10.17	**3.33**	2.46	3.29	**0.87**
15	1.43	1.41	**1.09**	0.91	0.89	**0.99**	4.27	5.08	**1.87**	4.45	5.09	**1.83**

means that with a relatively accurate rangefinder, the environment lines present almost the same detectability independently of the employed line detector. With respect to the mean absolute errors on the extracted line parameters, we conclude that there is no better algorithm at all.

In Table II, we have the performance measures for the run with $\sigma_r = 0.05\ m$. In most cases the SMF algorithm generated less spurious lines than the LT and IEPF algorithms, as indicated by the values of $\bar{N}_l$. This indicates that the SMF algorithm breaks less environment lines than the other ones. However, it is also possible that a fine tuning of the $T_{\max}$ parameter in the LT and IEPF algorithms would give better results. For $\bar{\eta}_l$, the SMF also presented the best results, except for the environment lines *1*, *3*, *5*, *10* and *13*. For the other lines, $\bar{\eta}_l$ was close to the best performance. With respect to the mean absolute errors $\bar{\varepsilon}_\rho$ and $\bar{\varepsilon}_\alpha$, for most cases the SMF presented a smaller degeneration with respect to the results in Table I. This is a robustness feature derived from the fuzzy clustering which uses non-crisp membership values u_{ij}.

In a first time, the results presented in Table I with low noise level in the measurements do not conclude which is the best algorithm. However, for *all* cases, we verify that when the SMF does not give the best performance, its results are close to the best one. However, when the sensor noise level is higher, SMF outperforms LT and IEPF

Table III. Results on $\bar{N}_l$, $\bar{\eta}_l$, $\bar{\varepsilon}_\rho$, and $\bar{\varepsilon}_\alpha$ for the simulation using $\sigma_r = 0.07\ m$.

e_l	$\bar{N}_l$			$\bar{\eta}_l$			$\bar{\varepsilon}_\rho$ in cm			$\bar{\varepsilon}_\alpha$ in degrees		
index	LT	IEPF	SMF	LT	IEPF	SMF	LT	IEPF	SMF	LT	IEPF	SMF
1	1.33	1.20	**1.08**	**0.61**	0.60	0.56	10.75	9.92	**5.01**	4.45	4.35	**2.06**
2	1.83	1.32	**1.13**	0.19	0.20	**0.53**	26.12	22.45	**18.09**	4.45	3.84	**3.05**
3	1.03	1.03	**1.02**	0.87	**0.93**	0.84	4.93	**4.85**	6.36	**7.14**	7.15	7.17
4	1.53	1.48	**1.14**	0.47	0.41	**0.82**	19.63	18.63	**17.29**	2.51	2.38	**2.21**
5	1.07	1.05	**1.03**	**0.86**	0.84	0.81	5.68	9.13	**5.33**	8.11	8.01	**7.46**
6	3.14	2.44	**1.79**	0.18	0.17	**0.43**	19.25	20.25	**11.77**	1.67	1.70	**0.99**
7	1.28	1.20	**1.13**	0.64	0.59	**0.93**	24.08	23.31	**13.71**	6.03	5.87	**2.80**
8	3.45	3.03	**1.85**	0.54	0.48	**0.90**	18.74	21.44	**10.25**	3.10	3.43	**1.50**
9	2.28	1.82	**1.38**	0.42	0.32	**0.86**	20.91	23.23	**13.43**	2.44	2.75	**1.59**
10	1.15	1.15	**1.11**	**0.84**	0.78	0.82	**15.16**	16.47	16.15	**1.29**	1.39	1.37
11	1.04	1.06	**1.03**	0.77	**0.80**	**0.80**	18.68	**17.74**	17.82	1.60	1.51	**1.50**
12	1.49	1.26	**1.21**	0.72	0.71	**0.87**	15.71	16.38	**14.14**	5.93	6.22	**3.95**
13	2.76	2.09	**1.74**	0.27	0.23	**0.31**	20.85	25.68	**8.06**	3.88	4.83	**1.75**
14	1.42	1.27	**1.16**	0.90	0.87	**0.96**	16.50	15.99	**6.55**	6.34	6.94	**1.86**
15	1.65	1.36	**1.16**	0.67	0.71	**0.95**	6.93	7.96	**3.08**	7.49	8.54	**2.93**

as seen in Table II. A carefull comparison of the Tables I and II allows us to verify that the performance of SMF has not changed as much of that of (non robust) LT and IEPF. Thus, we conclude that SMF may be more robust to the increase of sensor noise level than the LT and IEPF algorithms. This is confirmed when repeating the same evaluation with $\sigma_r = 0.07\ m$, as can be verified in results presented in Table III. Once more SMF outperforms the other algorithms when a high level of sensor noise is present.

7. Real data evaluation

In order to evaluate the feature extraction framework, we propose two experimental evaluations of this framework using the *LT*, *IEPF* and *SMF* kernels. Both evaluations were carried out with data sets obtained during the navigation of our Mobile robot in a real indoor environment. In the evaluation presented in section 7.1, the robot pose is given by an accurate reference positioning technique. From the entire set of range images, lines extracted using the proposed algorithms are superimposed in a commom frame using the reference robot positioning. Section 7.2 presents the second experimental evaluation using the same data set

for map building. Three environment maps are compared, one for each line extraction alrgorithm. The errors which appear in the environment maps are further discussed.

7.1. Performance comparison with superimposed extracted lines

The range images used in this experimentation were acquired with our robot in a navigation experiment. In this experiment, 48 images were acquired in 96.6 seconds. Accurate pose estimation provided by a dead-reckoning method fusing odometric and laser gyrometer data allows to present all range images in a unique global reference frame. First, we present in Figures 8 and 9 the lines extracted in all scans superimposed in the global frame. These figures present the results obtained with the different line extraction kernels, using respectively the adaptive and the KF-based breakpoint detectors (the small squares indicate the robot positions). Both breakpoint detectors seem to work quite well, in a way that they had a little impact in the extracted lines. Nevertheless, a visual inspection of some images allows us to verify the behaviour of each extractor in some real circumstances.

The images used in this comparison were taken from the above experimentation. The comparison consists of case analysis, shown in Figure 10, for which the line extraction kernels have presented wrong results, indicated with arrows. Rupture and breakpoints are indicated by the symbols (+) and (×), respectively. The selection procedure used $l_{\min} = 0,5\ m$ and $N_{\min} = 10$. For the kernels, $T_{\max} = 10\ cm$ for LT and IEPF, and $\sigma_{\max} = \sigma_r = 0.03\ m$ for *SMF*, which are the same values used in one of the simulations of section 6. The maximum number of iterations in the prototype-based fuzzy C-means, used in SMF, has been fixed to $i_{\max} = 15$. The KF-based breakpoint detectors have been used.

We start with the case (a), for which using an *IEPF* kernel gave a segment that does not really correspond to a real surface. Indeed, the support points of this feature correspond to the scan of a fire extinguisher attached to the wall. This was not observed using LT and SMF kernels; the former represented the wall as two distinct surfaces, and the second extracted only one line, as the wall, and the obstacle was not detected. ¿From our own perspective, the two interpretations are acceptable. On the other hand, LT presented in case (b) segments which do not correspond to environmental surfaces. In this case, IEPF and SMF gave better results since they did not presented such an anomaly. In some real experiments, we have observed some cases with the SMF

kernel presenting superfluous segments. Nevertheless, these were less frequent than with LT and IEPF kernels.

In case (c), a wall with a closed door is scanned by the sensor, resulting in a relief which is almost imperceptible in the image (see arrow). In such case, the three kernels presented different results: LT did not determined correctly the door endpoints, IEPF generated a segment not corresponding to the door, and with SMF the door has been neglected with only one line detected corresponding to the wall. This is an extreme case, where the door relief may be of a size comparable to that of the sensor noise. Nevertheless, in the image in case (d) the robot has advanced few meters, which made the door perceivable, at least when using LT and SMF kernels. With SMF, the door extremes have been better detected. Case (e) illustrates the capabilities of each kernel to discern surface intersections. For this image, the sensor captures an open door. The detection of the extremities indicated by an arrow has been more accurate with the SMF kernel. On the other hand, this case demonstrates an enormous limitation of LT kernel, which is the bad detection of surface extremities.

In order to complete this evaluation, the computing times of different combinations of breakpoint and line detectors used in the whole data set is presented in Figure 11. For the total of 48 images, Figure 11(a) gives the execution times of our framework for each image using the adaptive breakpoint detector and the different kernels. With the KF-based breakpoint detector, the results are shown in Figure 11(b). The maximum computing time, measured on a PII 450 MHz-based computer, was of 31 ms, which is satisfactory for real-time use in robots equipped with equivalent processing power. It should be noted that with both LT and IEPF kernels, we have obtained an almost constant computing time. This is an important factor for satisfying the deteminism requirements of platforms using tight real-time kernels. On the other hand, the SMF kernel has outperformed the other kernels in qualitative results. Thus, a trade-off between results quality and determinism has to be taken into account when choosing one of these kernels in a design.

7.2. Performance comparison in a map-building framework

In (Borges and Aldon, 2002) we present a map building approach based on the propagation of stochastic constraints among the map features. Even though propagating stochastic constraints through the map updating process help to reduce map divergence, the quality and consistency of features extracted from sensor data are equaly important. The results presented in this experimental comparison are focused

on features extraction only. The reader can find further details about the map building process in (Borges and Aldon, 2002).

In this section, the same experimental data used in the preceeding evaluation was used for building a map of the environment off-line. It means that, whatever the algorithm used for line extraction, the experimental conditions are the same. In this experiment, the robot pose is predicted using odometry and corrected using a specialized robust M-estimator IWLS-Huber algorithm from (Borges and Aldon, 2003)). Every time the map update process is executed, a local map is built using features captured by both the laser range finder and a video camera. The local map is matched against the current environment map for robot pose correction. After pose correction, the environment map is updated using the recently corrected pose and the local map. Thus, cumulative errors in robot pose estimation and feature extraction can lead to map divergence. In (Borges and Aldon, 2003) the robustness of IWLS-Huber algorithm was verified with a great number of experiments and evaluation conditions. In this section, the evaluations carried out compare environment maps built using the three lines extraction kernels. The setup of the other modules of the map building process is kept the same. The kernels are not used for video features extraction, only for range images. Although the map building process is rather complex, we expect this evaluation will illustrate the influence of each line extraction kernel.

Figure 12 shows the maps obtained using LT, IEPF and SMF just as line extraction kernels. Breakpoint detection was done using the KF-based detector. The kernel parameters are the same that in the previous evaluation. The robot initial position is labeled 1. The following ones correspond to places where pairs of range and video images were acquired. The symbos used for the structures in the maps are as follows: (▬) for semiplanes, (⊗) for edges, (⊙) for corners, (◬) for photometric edges, and the robot pose is illustrated by (↦). From these structures, only semiplanes are composed of lines extracted from range images. The length of the semiplanes is updated each time a local observation is captured in the local map. The fusion process between a semiplane and the localy observed line updates only the ρ and α parameters; the end-points are just projections of the original features on the fused infinite line. Edges, corners and photometric edges are not really relevant for this comparison. Futher details about map features are given in (Borges and Aldon, 2002)(Borges and Aldon, 2003).

Since the robot has performed a closed trajectory following two different paths, the position labeled 40 is a critical one. Indeed, at this position the robot sensors start to capture environment features which were already catchen in the begining of the run. Hence, any

map divergence can be visualy verified at this point whether the newer map features do not match the older ones. Figure 12(a) shows the map obtained using LT algorithm. Even though the map presented small divergence, some structures indicated by arrows presented incoherent behaviour during the mapping process. Figure 12(b) shows the unsatisfactory results obtained using the IEPF kernel. On the other hand, SMF presented the best results, with satisfactory map convergence and only one arrow indicating a cluster of structures correponding to the same wall.

8. Conclusion

This paper has presented a framework for geometrical feature detection in 2D-range images. The framework has been inspired from (Castellanos and Tardós, 1996), which first detects homogeneous regions in the range image, and apply line extraction kernels to each region. In our method, the regions are determined using breakpoint detectors, with two simple detectors presented in the paper. The line extraction kernels may use classic algorithms, such as LT and IEPF, or more sophisticated ones such as our SMF algorithm, introduced previously in (Borges and Aldon, 2000). The main paper contribution was focused on qualitative and quantitative comparisons using simulated and real data, as well as on evaluating the framework when different methods are used for breakpoint and line detection.

Special care has been taken when implementing such algorithms. Nevertheless, we cannot expect that all detected features are correct. Experimental results have shown the limitations of each algorithm, even when SMF has outperformed LT and IEPF for line extraction. Indeed, in every segmentation algorithm, robustness can only be expected up to a certain level of outliers, which commonly occurs in real cluttered environments. Thus, reliability in local map building using segmentation should be improved with complementary reasoning at a higher level. In the context of environment map building, section 7.2 presented an evaluation of the use of the different line extractors in our environment map building system (Borges and Aldon, 2002). SMF has presented satisfactory results, reducing map divergence.

References

Arras, K. O., Tomatis, N., Jensen, B. T. and Siegwart, R.: Multisensor on-the-fly localization: Precision and reliability for applications, *Robotics and Autonomous Systems* **34** (2001), pp. 131–143.

Barni, M., Cappellini, V., Paoli, A. and Mecocci, A.: Unsupervised Detection Of Straight Lines Through Possibilistic Clustering, *In IEEE International Conference on Image Processing*, 1996, pp. 963–966.

Bar-Shalom, Y. and Fortmann, T. E.: *Tracking and Data Association*, Academic Press, London, UK, 1987.

Bezdek, J. C.: *Pattern Recognition With Fuzzy Objective Function Algorithms*, Plenum Press, New York, USA, 1981.

Borenstein, J., Everett, H. R. and Feng, L.: *Navigating Mobile Robots: Systems and Techniques*, A K Peters, Wellesley, Massachusetts, USA, 1996.

Borges, G. A. and Aldon, M.-J.: A Split-and-Merge segmentation algorithm for line extractions in 2-D range images, *In Proc of* 15^{th} *International Conference on Pattern Recognition*, 2000.

Borges, G. A., Aldon, M.-J. and Gil, T.: An Optimal Pose Estimator for Map-based Mobile Robot Dynamic Localization: Experimental Comparison with the EKF, *IEEE International Conference on Robotics and Automation*, 2001.

Borges, G. A. and Aldon, M.-J.: Design of a Robust Real-Time Dynamic Localization System for Mobile Robots, *In 9th International Symposium on Intelligent Robotic Systems*, Toulouse, France, 2001.

Borges, G. A. and Aldon, M.-J.: A Decoupled Approach for Simultaneous Stochastic Mapping and Mobile Robot Localization, *In IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2002.

Borges, G. A. and Aldon, M.-J.: Robustified estimation algorithms for mobile robot localization based on geometrical environment maps, *In Robotics and Autonomous Systems*, Elsevier Science, **45(3-4)** (2003), pp. 131–159.

Castellanos, J. A. and Tardós, J. D.: Laser-based Segmentation and Localization for a Mobile Robot, *Robotics and Manufacturing: Recent Trends in Research and Applications*, Editors: M. Jamshidi, F. Pin and P. Dauchez. Volume 6, ASME Press, 1996.

Davé, R. N. and Krishnapuram, R.: Robust Clustering Methods: A Unified View, *IEEE Transactions on Fuzzy Systems* **5(7)** (1997), pp. 270–293.

Duda, R. O. and Hart, P. E.: *Pattern Classification and Scene Analysis*, John Wiley & Sons, New York, USA, 1973.

Einsele, T.: Real-Time Self-Localization in Unknown Indoor Environments using a Panorama Laser Range Finder, *In IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1997, pp. 697–702.

Forsberg, J., Larsson, U. and Wernersson, Å.: Mobile robot navigation using the range-weighted Hough transform, *IEEE Robotics and Automation Magazine* March (1995), pp. 18–26.

Frigui, H. and Krishnapuram, R.: A Robust Competitive Clustering Algorithm With Applications in Computer Vision, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **21(5)** (1999), pp. 450–465.

Haralick, R. M.: Propagating Covariances in Computer Vision, *In International Conference on Pattern Recognition*, 1994, pp. 493–498.

Jazwinski, A. H.: *Stochastic Processes and Filtering Theory* Academic Press, New York, USA, 1970.

Jensfelt, P. and Christensen, H. I.: Laser based position acquisition and tracking in an indoor environment *In International Symposium on Robotics and Automation*, 1998, pp. 331–338.

Jolion, J. M., Meer, P. and Bataouche, S;: Robust Clustering with Applications in Computer Vision, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **13(8)** August (1991), pp. 791–802.

Kämpke, T. and Strobel, M.: Polygonal Model Fitting, *Journal of Intelligent and Robotic Systems* **30** (2001), pp. 279–310.

Kwon, Y. D. and Lee, J. S.: A Stochastic Environment Map Building Method for Mobile Robot using 2-D Laser Range Finder, *Autonomous Robots* **7** (1999), pp. 187–200.

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

Peña, J. M., Lozano, J. A. and Larrañaga, P.: An empirical comparison of four initialization methods for the K-means algorithm, *Pattern Recognition Letters* **20** (1995), pp. 1027–1040.

Shpilman, R. and Brailovsky, V.: Fast and robust techniques for detecting straight line segments using local models, *Pattern Recognition Letters* **20(8)**, August (1999), pp. 865–877.

Siadat, A. and Dufaut, M.: Real Time and Dynamic Local Map Building by Using a 2D Laser Scanner, *In AVCS* , 1998, pp. 307–312.

Siadat, A., Kaske, A., Klausmann, S., Dufaut, M. and Husson, R.: An Optimized Segmentation Method for a 2D Laser-scanner Applied to Mobile Robot Navigation, *In 3rd IFAC Symp. on Intelligent Components and Instruments for Control Applications* , France, 1997, pp. 153–158.

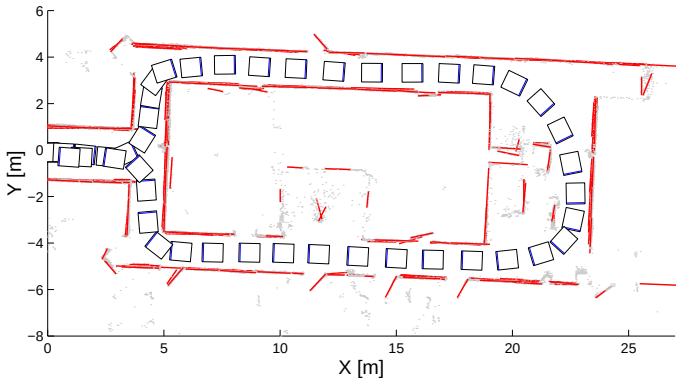
Skrzypczynski, P.: Building Geometrical Map of Environment Using IR Range Finder Data, *In Intelligent Autonomous Systems*, 1995, pp. 408–412.

Skrzypczynski, P.: Environment modelling using optical scanner data *In IFAC Symposium on Robot Control*, 1997, pp. 187–192.

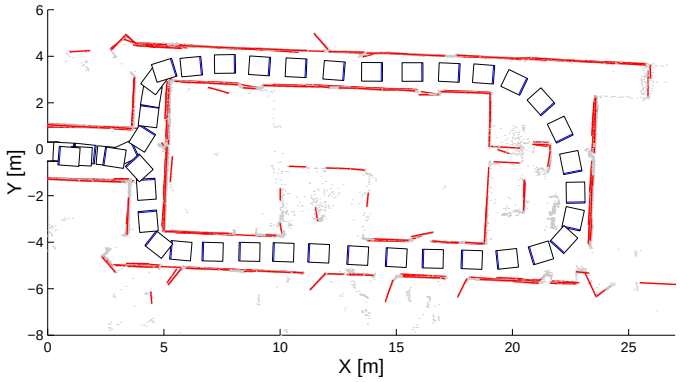
Vandorpe, J., van Brussel, H. and Xu, H.: Exact Dynamic Map Building for a Mobile Robot using Geometrical Primitives Produced by a 2D Range Finder, *In IEEE International Conference on Robotics and Automation*, Minneapolis, Minesota, USA, 1996, pp. 901–908.

Young, T. Y. and Fu, K.-S.: Handbook of Pattern Recognition and Image Processing, Academic Press, London, UK, 1986.

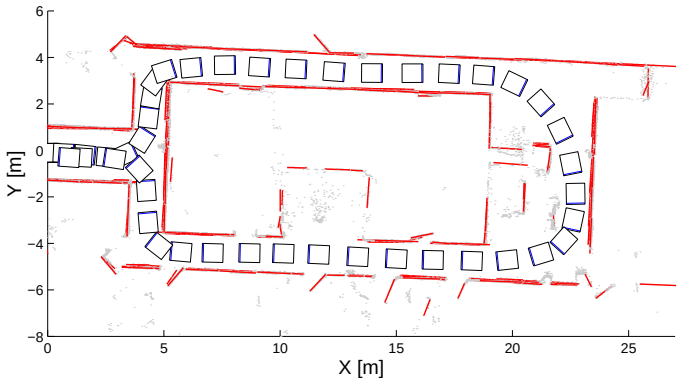
Zhang, L. and Ghosh, B. K.: Line segment based map building and localization using 2D Laser rangefinder, *In IEEE International Conference on Robotics and Automation*, 2000, pp. 2538–2543.



(a) LT

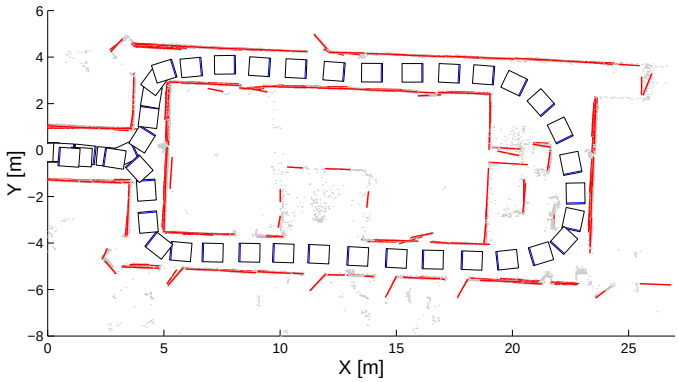


(b) IEPF

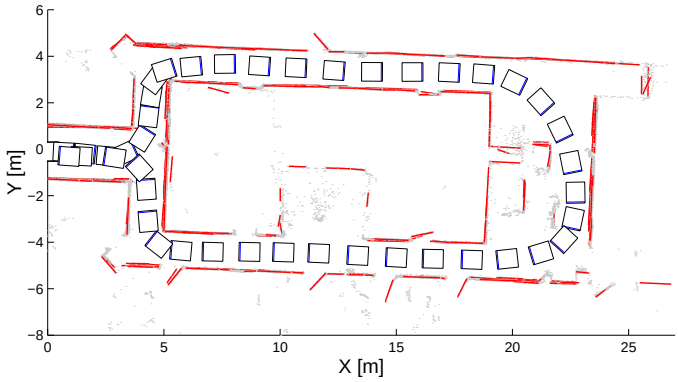


(c) SMF

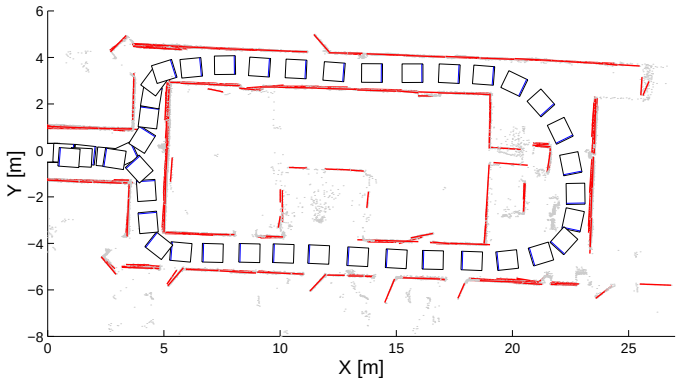
Figure 8. Experimental evaluation of the different line extraction kernels using the adaptive breakpoint detector.



(a) LT



(b) IEPF



(c) SMF

Figure 9. Experimental evaluation of the different line extraction kernels using the KF-based breakpoint detector.

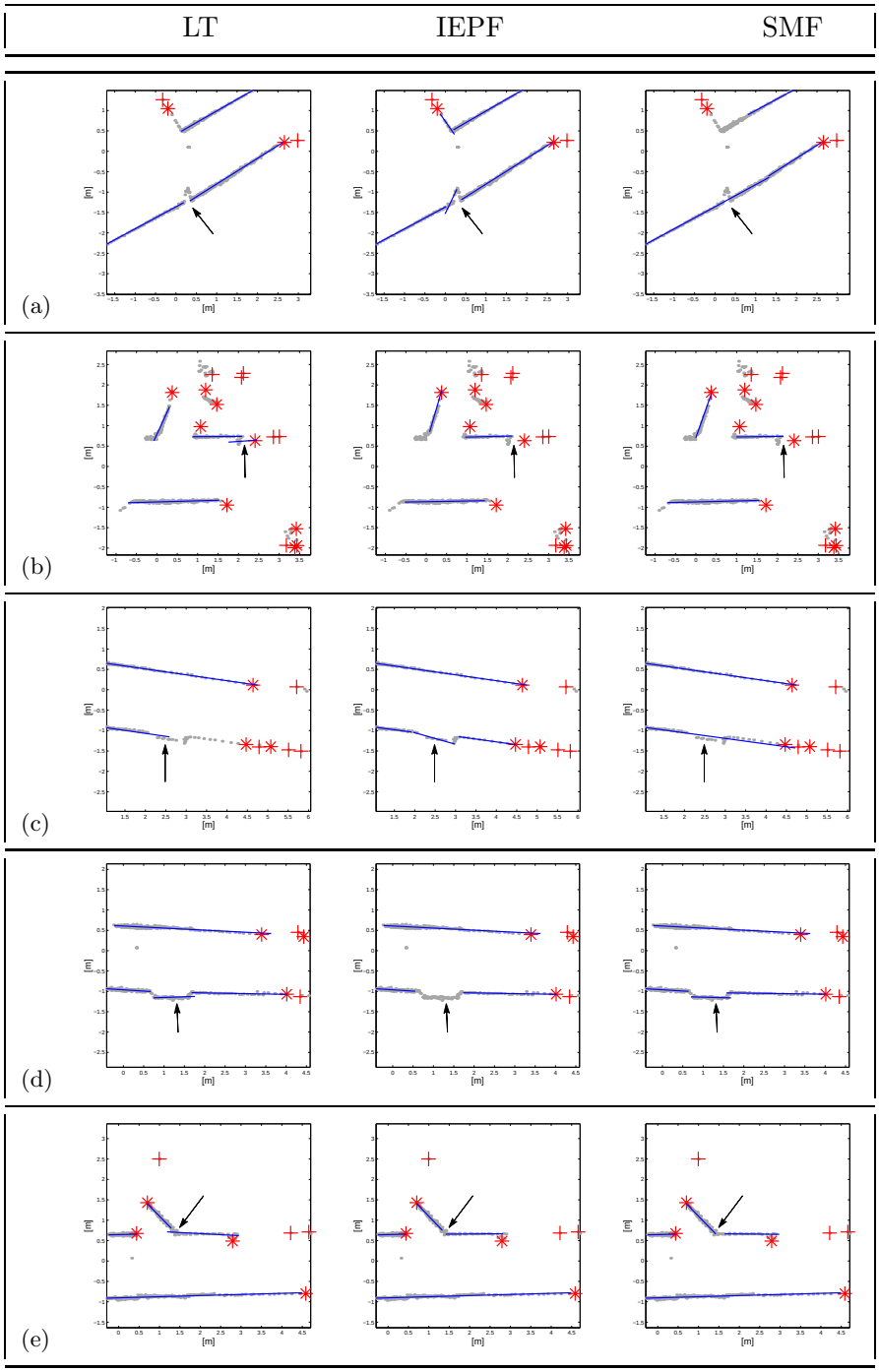
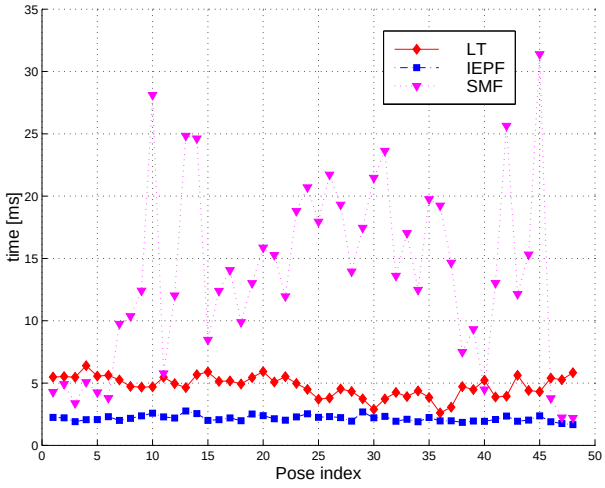
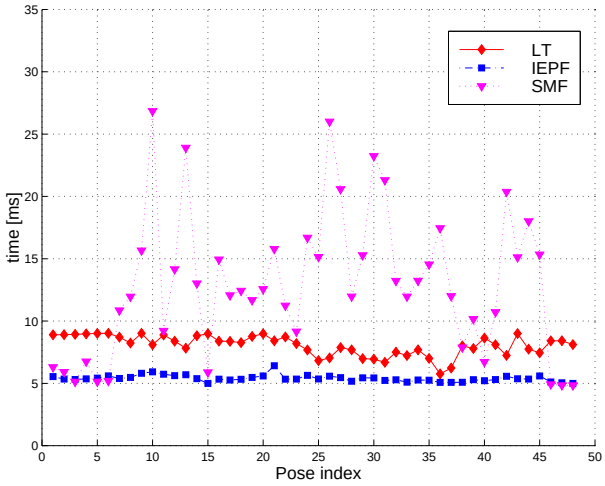


Figure 10. Experimental case-based comparison of the different line extraction kernels.

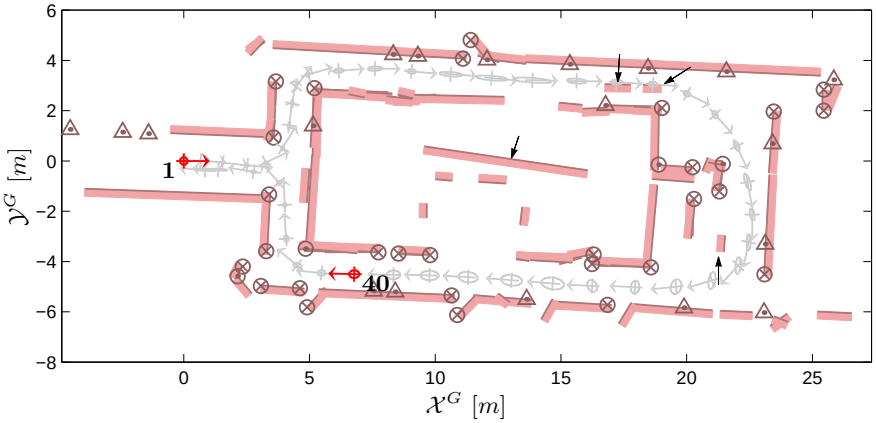


(a)

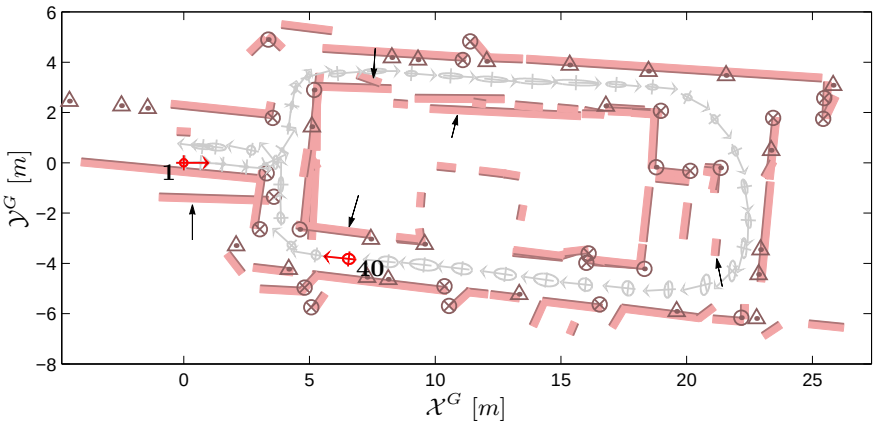


(b)

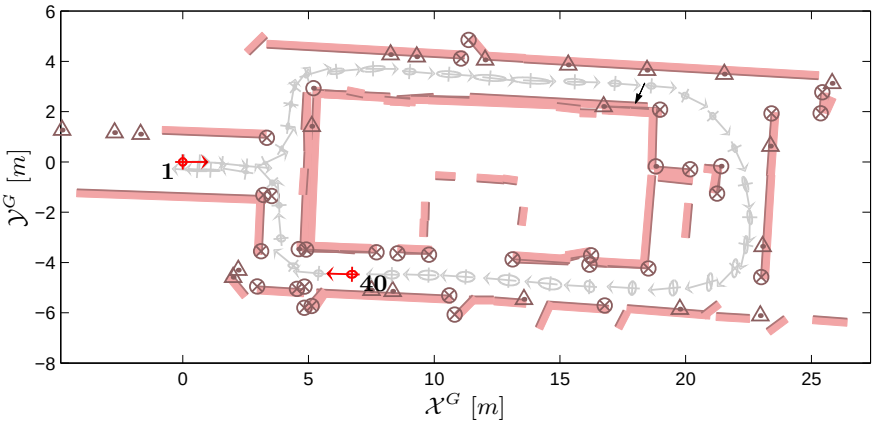
Figure 11. Computing time of the line extraction system using the different extraction kernels with (a) the adaptive breakpoint detector and (b) the KF-based breakpoint detector.



(a) Map built using LT for line extraction.



(a) Map built using IEPF for line extraction.



(a) Map built using SMF for line extraction.

Figure 12. Map building experimental results.