Online Detection and Tracking of 2D Geometric Obstacles from LRF Data

Mateusz Przybyła*

Poznań University of Technology
mateusz.przybyla@put.poznan.pl

February 21, 2017

Abstract

This work proposes a method for detection and tracking of local geometrical obstacles from sequences of two-dimensional range scans. Detected obstacles are represented with linear or circular models. Circular obstacles are subject to tracking algorithm based on Kalman filter. Solutions to both the correspondence and the update problem of the tracking system are provided. The occurence of obstacles fusion or fission is defined and addressed. A by-product of the system is the information on tracked obstacles velocity. The algorithm is dedicated to wheeled mobile robots with mounted planar laser range finders.

1. Introduction

Environment, in which the mobile robots work, may be very irregular and changing with time. Perception of its geometry is one of the key features needed to provide autonomy to mobile robots. With the help of laser range finders (LRF), sensing of surrounding material objects became simple and reliable. However, using raw data provided by such devices is often not suitable for well established algorithms of motion with obstacle avoidance or path planning [6, 21, 3, 15]. Extraction of more concise information from acquired data is an essential step between sensing and actuation.

There are two mainstream approaches of spatial data representation: grid-based and vector-based forms. Both of them can be mutually used to represent the environment in detail [13, 8]. This work centres on extraction and tracking of 2D vector-based geometric objects, from data provided by horizontal-working LRFs. Such devices can provide only local, spatial information due to their working principles (e.g. occlusion). Still, locally detected objects can be exploited in such problems as: target detection and tracking, real-time reactive obstacle avoidance, local path planning, environment visualization, removal of moving objects for SLAM.

The obstacles related to the operations performed by mobile robots can have many forms. The following list distinguishes several genres of obstacles and provides examples:

- Obstacles distinguished by geometry: sparse (e.g. wire fence, foliage), dense (e.g. wall, tree trunk), convex (e.g. furniture, car), concave (e.g. hole in the ground, cage).
- Obstacles distinguished by color/opacity: opaque (e.g. wooden doors, rock), reflective (e.g. mirror, polished metals), transparent (e.g. glass doors, plexiglass wall).
- Obstacles distinguished by corporeality: material (e.g. solid box, textile curtain), virtual (e.g. programmed restrictions, mechanical constraints).

*Faculty of Computing, Chair of Control and Systems Engineering, ul. Piotrowo 3A, 60-965 Poznań, Poland, (+48 61) 665-29-87

- Obstacles distinguished by variability in time: stationary (e.g. sofa, tree), movable (e.g. chair, trash can), mobile (e.g. person, robot).

Since the method proposed in this work relies on data provided by planar LRFs, it is dedicated to dense, convex, opaque, material objects of any variability in time, situated in the local vicinity of the robot.

Presented method is composed of two parts. The first part, described in Sec. 2, focuses solely on the obstacle extraction problem. The obstacles are extracted in an asynchronous manner directly from the spatial data provided by the sensors. This step has a non-deliberative form, meaning that the pure obstacle extraction does not involve the history of extracted objects.

Taking history of obstacles into consideration is the objective of the second part - the obstacle tracking, which is described in Sec. 3. The obstacle tracking process works in a synchronous manner with a constant sampling time. The main goal of this step is to sustain the existence of extracted obstacles in the case of sudden disappearance (e.g. as a result of obstacle occlusion) and to enhance their quality by proper filtration.

The problem of LRF-based detection and tracking of moving objects (DATMO) has been well studied in the past years [7, 1, 10]. In this work, we propose a new approach based on obstacles fusion and fission during correspondence evaluation. The key value of presented method lays in its simplicity, satisfactory effects, and an intuitive process of parameters tunning. The main contribution of this work is the novel heuristic for extracting circular objects from the LRF data as well as the non-trivial solution to the correspondence problem between obstacles sets.

Detected obstacles are only approximations of the true objects existing in the real world. However, throughout the article, we will use terms 'detected obstacle' and 'obstacle' interchangeably.

2. Obstacle extraction

The goal of obstacle extraction is to provide a set of structures $\mathbb{O}$ representing geometric objects existing in the robot workspace. Note that the intention of obstacle detection is not to find an optimal model of workspace geometry but to find a rough approximation of its material contents. Since the method is dedicated to data provided by LRFs, let's define several fundamental terms.

Definition 1. ***Laser scan*** *(or scan) is a data packet consisting of a set of N_R subsequent geometric points described in polar coordinates system, where the angular position for each consecutive point value is incremented by a* ***constant*** *angle.*

Definition 2. ***Angular-ordered point cloud*** *(AO-PCL) is a data packet consisting of a set of N_A subsequent geometric points, provided in an ordered manner, where the order is dictated by the growing angle of position vector.*

Definition 3. ***Obstacle from laser scan*** *is a set of N_P geometric points, representing a single body existing in the workspace.*

The difference between a laser scan and an AO-PCL is that the restriction on constant angular step between consecutive points in the laser scan is repealed in AO-PCL. In this sense, the AO-PCL data type is more general and the laser scan is its special case. For this reason in the rest of the document we will refer to input data as AO-PCL.

From the perspective of motion planning or closed-loop control with colision avoidance a set of points is usually not a suitable data (because of its numerousness and quality). Hence, in this study, we propose to approximate the obstacles from laser scan with one of two geometric models:

- **line segment** l represented with two extreme points:

$$l \triangleq \{\vec{p}_1, \vec{p}_2\} = \{(x_1, y_1), (x_2, y_2)\}, \quad (1)$$

- **circle** c represented with a radius and a central point:

$$c \triangleq \{r, \vec{p}_0\} = \{r, (x_0, y_0)\}. \quad (2)$$

Approximation of point sets with such geometric models provides two major advantages. Firstly, it simplifies data structures because instead of finite number of discrete points we obtain a short representation of continuous objects. Secondly, it reduces the measurement noise due to the averaging effect while computing model state. Similar geometric models (with the addition of arc segments) were used in work [19], in which the authors focus on more accurate determination of the environment geometry. The mentioned work provides a comprehensive survey on methods for detecting objects from LRFs.

We define the resulting set of obstacles $\mathbb{O}$ as:

$$\mathbb{O} \triangleq \{\mathbb{L}, \mathbb{C}\}, \quad (3)$$

where $\mathbb{L}$ denotes a set of N_L segment-type obstacles and $\mathbb{C}$ denotes a set of N_C circular obstacles. The following subsections describe the details of extraction detection process.

2.1. Grouping

The input AO-PCL is subject to a grouping procedure, where the groups are distinguished by examining distances between consecutive points. The aim of grouping is to provide a collection of point subsets S_j ($j \in \{1, ..., N_\mathrm{S}\}$) representing possibly separate objects. The point $\vec{p}_i$ ($i \in \{2, ..., N_\mathrm{A}\}$) is assigned to the group of point $\vec{p}_{i-1}$ if the following distance criterion is satisfied:

$$d^i_{i-1} < d_\mathrm{group} + R_i d_\mathrm{p}, \quad (4)$$

where d^i_{i-1} represents the euclidean distance between points $\vec{p}_i$ and $\vec{p}_{i-1}$, R_i represents the range of point $\vec{p}_i$ (Fig. 1), d_group is the user-defined threshold for grouping and d_p is the user-defined distance proportion which loosens the criterion for distant points. If the criterion (4) is not satisfied then a new group is created. Resulting groups constitute point subsets which are further subject to splitting procedure.

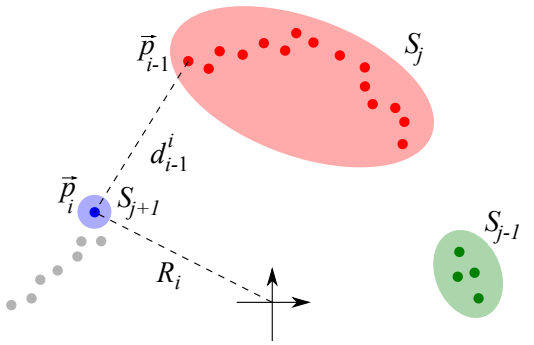

Figure 1: *Graphic representation of variables involved in synthetic points grouping.*

2.2. Splitting

Each point subset S_j is examined for possible splitting into two separate subsets. For simplicity, we do not process groups consisting of less than N_min points. The procedure of splitting relies on the Iterative End Point Fit algorithm, which builds a leading line upon two extreme points of the group and seeks the point of the group that lays farthest apart from this line (Fig. 2). The criterion, which satisfaction splits the set into two (Fig. 3), is defined as:

$$\bar{d}_j > d_\mathrm{split} + R_j d_\mathrm{p}, \quad (5)$$

where $\bar{d}_j$ represents the distance between the leading line and the farthest point of set S_j, d_split is a user-defined threshold for splitting, and R_j is the range of the farthest point of set S_j.

In the case of splitting, the point of division is assigned to both of the new subsets. The procedure is repeated recursively for each new subset until no more splitting occurs. After splitting, the new number of subsets N^+_S can be smaller, the same or greater than previous number N_S.

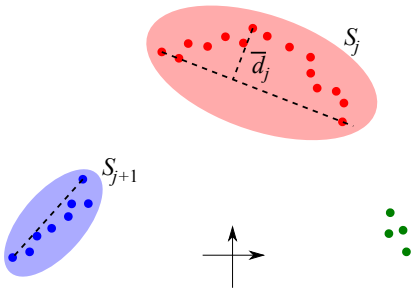

Figure 2: *Graphic representation of Iterative End Point Fit applied to synthetic data: seeking most distant point.*

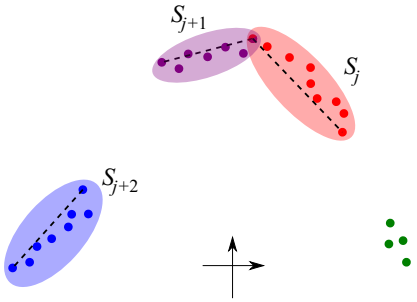

Figure 3: *Graphic representation of Iterative End Point Fit applied to synthetic data: splitting the group.*

2.3. Segmentation

Every subset S_m ($m \in \{1, ..., N_S^+\}$) is approximated with a segment model l_m. The approximation procedure consists of two parts. First, a leading line is build upon all of the N_m points belonging to the subset. The leading line is modeled with the general form equation:

$$A_m x + B_m y + C_m = 0, \tag{6}$$

where A_m, B_m, C_m are parameters, and x, y are variables. We use total least squares regression to compute the parameters. Because there are infinitely many solutions for this problem one can set arbitrary value of $C_m \neq 0$ and solve the problem only for A_m and B_m:

$$\begin{bmatrix} A_m \\ B_m \end{bmatrix} = \begin{bmatrix} x_1 & y_1 \\ \vdots & \vdots \\ x_{N_m} & y_{N_m} \end{bmatrix}^{\dagger} \begin{bmatrix} -C_m \\ \vdots \\ -C_m \end{bmatrix}, \tag{7}$$

where symbol † represents the Moore-Penrose matrix pseudoinverse. Secondly, the two extreme points of the subset are projected onto the leading line to form a final segment l_m (Fig. 4). A drawback of such extreme points projection is the possibility of inclusion of mixed pixels, which can lead to erratic changes in segment length. A possible solution (not tested here) would be to check if the distance from extreme points to the segment fall into single standard deviation. In case they do not, the algorithm should pick next points, respectively. Each point subset S_m is saved in memory for further processing.

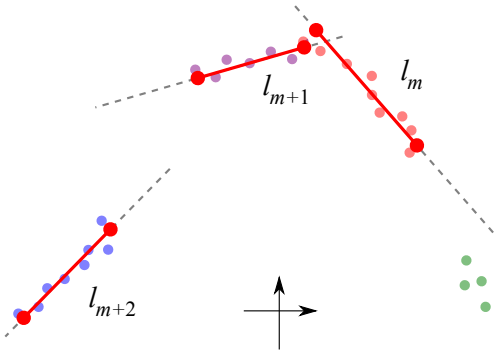

Figure 4: *Graphic representation of segments obtained from synthetic data.*

There are many methods for extraction of segments from sets of 2D points. A good survey and comparison of several of them can be found in work [9].

2.4. Segments merge

Next, the collection of extracted segments is subject to a merging procedure. The goal of this step is to find pairs of segments that potentially comprise a single object and recreate it based on original point subsets. The criterion dictating whether two segments should be merged is composed of two parts. First, called the connectivity test, checks if both segments are close to each other:

$$d_0 < d_{\text{merge}}, \tag{8}$$

where d_0 denotes distance between neighboring points of both segments and d_{merge} is a user-defined threshold for connectivity (Fig. 5). If condition (8) is satisfied, a new leading line is build upon sum of point subsets comprising both segments. Second part, called spread test, checks if both segments are collinear:

$$\max\left(d_1, d_2, d_3, d_4\right) < d_{\text{spread}}, \tag{9}$$

where d_{1-4} are distances between segments extreme points and the new leading line, and d_{spread} is a user-defined threshold for spread (Fig. 6). If both conditions are satisfied, two extreme points of point subsets are projected onto the leading line and a new segment replaces the original pair.

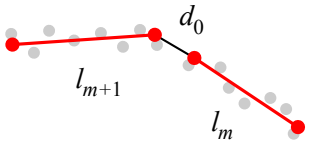

Figure 5: *Graphic representation of segments merging criterion: connectivity test.*

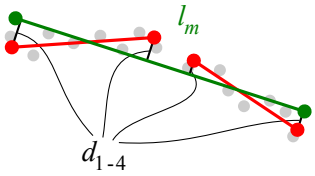

Figure 6: *Graphic representation of segments merging criterion: spread test.*

Although the segments are provided in an angular-ordered fashion (as a consequence of ordered input data), we check any segment with any for possible merging. The reasoning behind this is that the merging can occur not only between the neighboring segments but between any two in the collection. The extracted segments constitute set $\mathbb{L}$.

2.5. Circles extraction

It is often convenient to represent detected obstacles with their circular outline (e.g. for motion planning using stream functions [16], hydrodynamic potential [14] or obstacle avoidance using potential based methods [4]). Note that not many objects in true world scenario, especially in man-made constructions, can be well approximated with circles. Hence, the intention of circles extraction is not to find an optimally fitting figure which represents the group of points but rather to assert that the resulting figure encloses the points and does not cover much of the free-space area. For those reasons, we propose a heuristic procedure for obtaining circular obstacles directly from segments.

For each segment l_n ($n \in \{1, ..., N_{\text{L}}\}$) in the set $\mathbb{L}$ we construct a conceived equilateral triangle with its base coinciding with the segment and its central point placed away from the origin. Next, on the basis of the triangle we construct an escribed circle c_n (Fig. 7) with radius:

$$r_n = \frac{\sqrt{3}}{3}\bar{l}_n, \tag{10}$$

where $\bar{l}_n$ denotes segment length, and with central point:

$$\vec{p}_{0n} = \frac{1}{2}\left(\vec{p}_{1n} + \vec{p}_{2n} - r_n\vec{n}_n\right), \tag{11}$$

where $\vec{n}_n$ denotes segment normal vector pointing towards origin (the orientation of segment can be easily obtained with the use of cross product). The radius is then enlarged by a user-defined margin r_{d}. If the resulting radius is not greater than a user-defined threshold r_{max}, the circle is added to the set $\mathbb{C}$. Otherwise it is discarded. The reason for maximal allowable radius of the circle is the fact, that extracted segments can be very long (e.g. corridor walls), and extraction of circle from such segment would result in object covering a vast area of the workspace.

Note that the points of segments should be described with respect to a local (sensor) coordinate frame. If the segments were described with respect to other (e.g. global) coordinate frame, the orientation of normal vector $\vec{n}_n$ could be opposite and the resulting circle would cover much of free-space in front of the

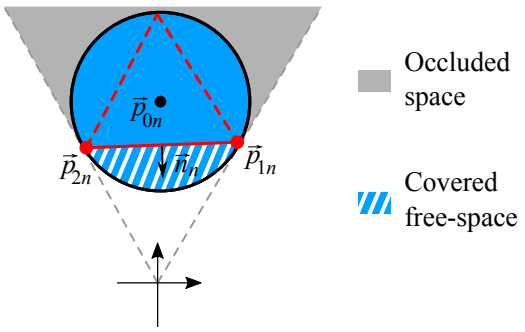

Figure 7: *Extracting circle from a segment.*

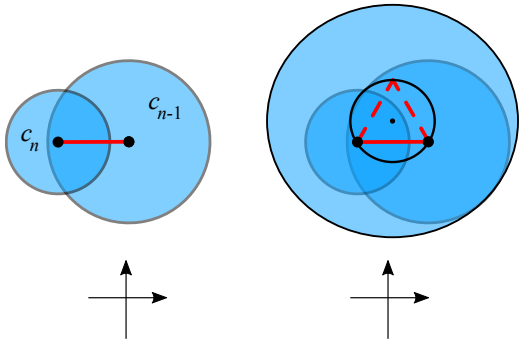

Figure 8: *Graphic representation of circles merging.*

obstacle (as seen from the range finder point of view).

2.6. Circles merge

The resulting set of circles is also subject to a merging procedure. If a circular obstacle is located inside other circular obstacle it is discarded. If two circular obstacles intersect, a new segment is created upon their central points and a new circle is created upon this segment. Next, to assert that the new obstacle encloses all of the points constituing previous obstacles, the radius of a newly created circle is enlarged by the radius of the larger obstacle (Fig. 8). If the resulting radius is not greater than the previously defined threshold $r_{\max}$, the new circle is added to the set and the two previous obstacles are removed. An interesting read about 2D obstacles clustering can be found in [2]. The authors of the mentioned work dedicate the method for polygon-type objects but the method could be easily adopted to circular ones. Because the method presented in this work is dedicated to local vicinity of the robot and the laser range finders are restricted to detect only the closest objects (due to occlusion) we do not use clustering on the final set of obstacles.

3. Obstacle tracking

Obstacle tracking allows us to improve the quality of extracted obstacles. Information on history of samples for each obstacle enables velocity estimation, super-sampling or filtering. In this work the obstacle tracking is applied only to circular obstacles. Hence, in the rest of the work terms 'circular obstacle' and 'obstacle' will be used interchangeably. The obstacle tracking problem is composed of two subproblems: the correspondence problem and the update problem.

3.1. Correspondence problem

The number of obstacles extracted from two consecutive AO-PCLs may vary. Moreover, the order of their occurence in the list can also differ from sample to sample. The correspondence problem consist in figuring out which obstacle extracted from AO-PCL taken at sample $k-1$ corresponds to which obstacle extracted from AO-PCL taken at sample k. For convenience, we will refer to obstacles extracted in sample $k-1$ as **old obstacles** and to obstacles extracted in sample k as **new obstacles**.

In the literature, this problem is also known as an assignment problem. One of the methods to solve this problem, which was used in work [17], is called Hungarian method. The authors of mentioned work exploit this method for finding a correspondence between points obtained from ultrasonic sonars to form the obstacles in outdoor environment.

Let N_{k-1} and N_k represent the number of old and new obstacles, respectively. For any two circular obstacles c_i ($i \in \{1, ..., N_k\}$) and c_j ($j \in \{1, ..., N_{k-1}\}$) we define a scalar cost function c_{ij}, which determines the difference

between them:

$$c_{ij} \triangleq \sqrt{(x_{0i} - x_{0j})^2 + (y_{0i} - y_{0j})^2 + (r_i - r_j)^2}. \tag{12}$$

Next, we construct a cost matrix $C \in \mathbb{R}^{N_k \times N_{k-1}}$ representing the disparity between obstacle sets from both samples:

$$C = \begin{bmatrix} c_{11} & \cdots & c_{1N_{k-1}} \\ \vdots & \ddots & \vdots \\ c_{N_k 1} & \cdots & c_{N_k N_{k-1}} \end{bmatrix}. \tag{13}$$

In the cost matrix (13) the rows represent new obstacles and the columns represent old obstacles.

Let us define function $\min \operatorname{idx}(A)$ as a function which returns indices of elements with lowest value for each of the rows of matrix A. If the minimal value in n-th row is greater than a user defined value c_{corr}, the function returns value -1 at n-th position.

From the cost matrix we construct two vectors:

- vector of minimal indices by rows:

$$\vec{n}_{\text{rows}} \triangleq \left(n_{r1}, ..., n_{rN_k}\right)^\top = \min \operatorname{idx}(C),$$

- vector of minimal indices by columns:

$$\vec{n}_{\text{cols}} \triangleq \left(n_{c1}, ..., n_{cN_{k-1}}\right)^\top = \min \operatorname{idx}\left(C^\top\right).$$

The vector of minimal indices by rows (columns) stores the indices of old (new) obstacles that have the minimal cost with each of new (old) obstacles.

By examining $\vec{n}_{\text{rows}}$ and $\vec{n}_{\text{cols}}$ we can distinguish several correspondence situations:

- i-th new obstacle has no corresponding old obstacle ($n_{ri} = -1$),
- i-th new obstacle has one and only one corresponding old obstacle j ($n_{ri} = j$),
- i-th new obstacle has two or more corresponding old obstacles ($n_{cj} = i$ and $n_{ck} = i$ where $j \neq k$),
- i-th old obstacle has two or more corresponding new obstacles ($n_{rj} = i$ and $n_{rj} = i$ where $j \neq k$).

Determination of the correspondence type for currently examined obstacle provides vital information for the update problem. In the first case execution of update step is impossible because of lacking data. We mark such obstacle as untracked. In the second case, execution of update step is explicit. We mark such obstacle as tracked. Let us define the third and fourth cases as obstacles fusion and fission.

Definition 4. ***Obstacles fusion****: if two or more old obstacles connect to comprise a single new obstacle: we call it a* ***fusion*** *(Fig. 9).*

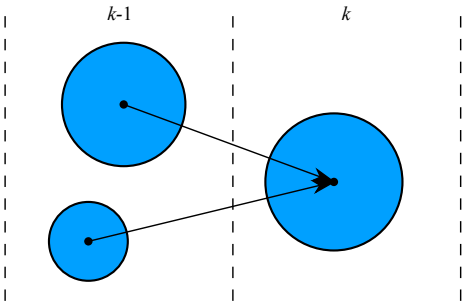

Figure 9: *Graphical representation of obstacles fusion.*

Definition 5. ***Obstacle fission****: if a single old obstacle splits into two or more new obstacles: we call it a* ***fission*** *(Fig. 10).*

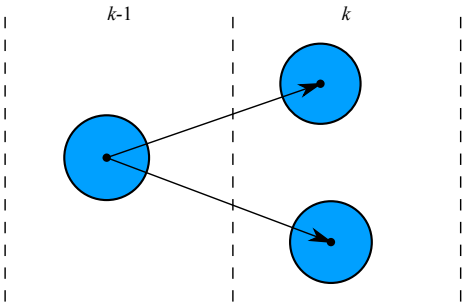

Figure 10: *Graphical representation of obstacle fission.*

Fusion or fission of obstacles happen often when there are moving elements in the environment. A very good example of such phenomena are human legs while walking. The obstacles fusion (fission) can be detected by examining $\vec{n}_{\text{cols}}$ ($\vec{n}_{\text{rows}}$) for duplicate indices.

If there are two or more old (new) obstacles corresponding to the same new (old) obstacle, we consider it as fused (fissured).

In the case of obstacles fusion, a new (fused) obstacle has to be updated based on all of the constituing old obstacles. In the case of obstacle fission, any new (fissured) obstacle has to be updated based on the original obstacle. Any fused or fissed obstacles are marked as tracked.

3.2. Update problem

The obstacle tracking system works in a synchronous manner with a constant sampling time T_P. For each obstacle marked as tracked we create a separate Kalman filter and a counter. Both the prediction and the correction step of the filter as well as decrementation of the counter is executed with every sample of time. The measurement update occurs only after new set of extracted obstacles was obtained and the correspondence was resolved. In such case the counter is restarted. This helps to sustain the existence of tracked obstacles even in the absence of many measurement samples. If the counter reaches zero, the obstacle is removed from the list.

Let us define a state vector for tracked obstacle as:

$$\vec{q} \triangleq \left(x_0, \dot{x}_0, y_0, \dot{y}_0, r, \dot{r}\right)^\top. \tag{14}$$

For convenience we omit the indices of obstacles in the equations. The state includes the radius of obstacle as it is an important variable to be updated. The state is also extended to include the rates of change for each variable. Although we cannot measure them, we can estimate them as disturbances. With such definition of state vector we propose the following model of the discrete state equation:

$$\vec{q}_k = \begin{bmatrix} J & 0 & 0 \\ 0 & J & 0 \\ 0 & 0 & J \end{bmatrix} \vec{q}_{k-1} \quad \text{where} \quad J = \begin{bmatrix} 1 & T_\mathrm{P} \\ 0 & 1 \end{bmatrix}. \tag{15}$$

The model of obstacle motion assumes constant velocity and is autonomous because we do not have any information on obstacles input.

For the update (if the correspondence was resolved) we obtain the information only about new position and radius of the obstacle, hence the output equation is simply:

$$\vec{y}_k = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \vec{q}_k. \tag{16}$$

If the correspondence problem was resolved explicitly (the new obstacle had only one corresponding old obstacle), we update it with the new information set. If the fussion occured, we create a new obstacle with state equal to the average of states of constituing obstacles, and then we update it with the new information set. If the fission occured, we copy the original obstacle for each new one, and update them based on new information sets.

4. Experimental results

The following sections describe experiments conducted in order to verify the effectiveness of the proposed method.

4.1. Experimental setup

The setup used for conducted experiments consisted of two Hokuyo URG-04LX-UG01 laser scanners mounted back-to-back on a metal plate fixed to Kuka youBot (Fig. 11). The range finders work with a framerate of 10 Hz. Both devices provide 240° angular range with 0.352° angular resolution and 5.6 m linear range with 0.03 m linear resolution. An in-depth characteristics of LRFs of the same product family was provided in work [5]. Before feeding into the algorithm, laser scans obtained from both scanners were firstly merged into a single 360° AO-PCL with its origin situated between the devices.

During motion, the platform was tracked with the OptiTrack motion capture system with a framerate of 100 Hz. The obstacles detected during experiments were hollow, yellow cylinders with diameter of 27 cm. The algorithm was implemented in ROS (version Jade) on a regular personal computer with processor Intel

Figure 11: *Mobile robot with mounted laser scanners.*

Core i5-2300, 2.8 GHz and 4 GB of RAM. For implementation of any matrix algebra we used Armadillo C++ library [11].

4.2. Parameters tuning

Values of parameters required by the method were chosen empirically, based on intuition and through trial and error. Most of them have a direct physical meaning and hence the process of tuning does not impose a significant burden on the user. The list of parameters and their values is shown in Tab. 1.

Table 1: *Values of parameters used for the experiments*

Symbol	Name	Value
d_{group}	grouping thr.	5 cm
d_{p}	distance proportion	6.14 mrad
d_{split}	splitting thr.	6 cm
N_{min}	min. points num.	5
d_{merge}	connectivity thr.	15 cm
d_{spread}	spread thr.	7 cm
r_{d}	radius margin	0 cm
r_{max}	radius thr.	30 cm
c_{corr}	correspondence thr.	30 cm
T_{P}	sampling time	0.01 s

Tunning of Kalman filter [18] involves providing measurement and process covariances R and Q $\left(\text{here } R \in \mathbb{R}^{3\times3} \text{ and } Q \in \mathbb{R}^{6\times6}\right)$. Because neither of them can be fine-estimated (measurement covariance depends on number of points comprising an obstacle and its geometry), they are provided based on intuition. Small process covariance means more *trust* put into state prediction. Because we want to smoothen out the measurement noise, we set the measurement covariance to $R = \text{diag}(1, ..., 1)$ and the process covariance to $Q = \text{diag}(0.005, ..., 0.005)$. The intuition was supported by visual examination of the output of tracking system with random (relatively slow) motion of obstacles. The examination consisted in comparing pure obstacles detected from first part of the method with tracked obstacles. Too much *inertia* applied to the tracking system by small process covariance resulted in a phase lag and an improper correspondence solutions (objects moving too fast were not recognized as the same from sample to sample). While tunning, one must find a reasonable ratio between noise reduction and motion tracking. A low sensor update frequency (here 10 Hz) is one of the main factors restricting noise attenuation.

4.3. Pure obstacle extraction

To present the effectiveness of the pure obstacle extraction, several experiments were performed with stationary obstacles placed at various positions in the workspace. Each experiment consisted in detection of a single circular obstacle of radius 135 mm and collection of 1000 samples of its parameters. Tab. 2 presents statistical data obtained from the experiments. For each obstacle it provides average number of points $\bar{n}$ comprising an obstacle, average distance from the origin $\bar{d}$ [m] , variance of center position $\sigma^2_{p_0}$ [mm^2] (calculated as a l^2-norm of variances of its both coordinates: $\sigma^2_{y_0}$ and $\sigma^2_{y_0}$), average radius $\bar{r}$ [m] as well as variance of radius σ^2_r [mm^2] .

The number of points comprising an obstacle is proportional to the inverse of squared range (as a consequence of spreading laser beams). Based on this fact, one could expect that the variance of obstacle parameters will increase with increasing distance from the LRF. From the experimental data it can be seen that, surprisingly, this is not always the case. It turns

Table 2: *Statistical data obtained from experiments with pure obstacle detector*

N	$\bar{n}$	$\bar{d}$	$\sigma^2_{p_0}$	$\bar{r}$	σ^2_r
1	86	0.498	1.565	0.131	1.254
2	59	0.714	1.574	0.134	0.877
3	42	1.008	2.613	0.136	1.187
4	38	1.123	1.847	0.139	1.003
5	28	1.508	9.316	0.139	11.473
6	27	1.590	2.812	0.142	1.898
7	21	2.055	6.878	0.144	7.677
8	20	2.005	3.645	0.132	0.898
9	15	2.504	17.250	0.120	18.788

out that the major factor increasing the variances of estimated variables is the random existence of the mixed pixels effect mentioned in Subsec. 2.3. A blinking point at the end of the set can result in unwanted erratic changes of estimated parameters. Such effect can be seen in Fig. 12, which depicts an exemplary plot of detected obstacle radius against time. The removal of mixed pixels for laser range finders was proposed e.g. in works [20] and [12]. The erratic motion of obstacles can be also suppressed with the obstacle tracking system.

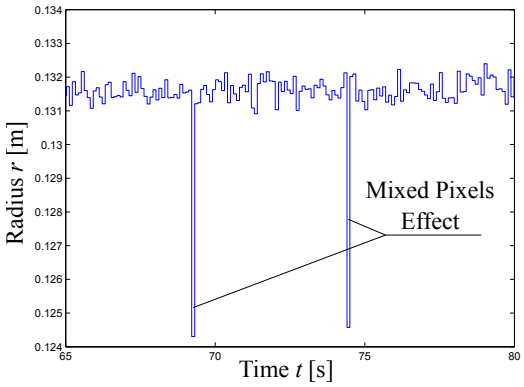

Figure 12: *Estimated radius of obstacle no. 7 showing spikes due to mixed pixels.*

4.4. Obstacle detection with tracking

To show the effectiveness of proposed obstacle tracking system, experiments from previous section were repeated. The experimental setup did not change. Tab. 3 presents the statistical data for collected samples. Because the obstacle tracking system works with ten times greater framerate than the obstacle detector (as a consequence of 10 Hz framerate of exploited range finders), we collected 10 000 samples for each obstacle.

Table 3: *Statistical data obtained from experiments with obstacle tracking*

No.	$\bar{n}$	$\bar{d}$	$\sigma^2_{p_0}$	$\bar{r}$	σ^2_r
1	86	0.498	0.859	0.131	0.417
2	59	0.714	0.939	0.134	0.320
3	42	1.008	1.491	0.136	0.429
4	38	1.123	0.785	0.139	0.357
5	28	1.508	3.938	0.138	3.940
6	27	1.590	1.235	0.142	0.685
7	21	2.055	2.531	0.144	2.681
8	20	2.005	2.587	0.132	0.333
9	15	2.504	6.321	0.120	6.921

The variance of estimated obstacle radius is in average about three times smaller than previously. The erratic motion due to the mixed pixels effect has been suppressed. It can be seen by comparing two exemplary plots of estimated obstacle radius: one for pure obstacle extraction and one for detection with tracking (Fig. 13).

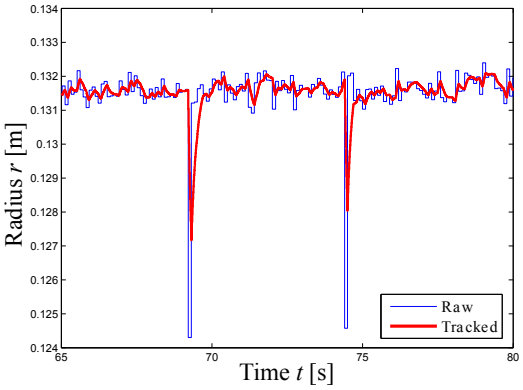

Figure 13: *Comparison of estimated value of radius with and without tracking system.*

4.5. Estimation of obstacles velocity

A by-product of using Kalman filter with extended state vector (14) is the estimate of obstacle velocity. Information on obstacles velocity can be beneficial in many scenarios (e.g. object following, dynamic reaction to environment changes). In order to verify the correctness of obstacles velocity estimation, the following experiment was performed.

A single obstacle was situated in the workspace. The robot with mounted laser scanners moved back and forth in the direction of the obstacle (the motion was applied by human operator via manual controller). Based on the pose of the robot (which was accurately measured with the OptiTrack system) we calculated its velocity. For the experiment, the position of obstacle was described with respect to local reference frame (base of the robot). The relative motion of robot-obstacle system emulated obstacle motion. Proper transformations allowed for comparison of true velocity (based on OptiTrack measurement) and estimated velocity (based on Kalman filtering) of the obstacle (Fig. 14).

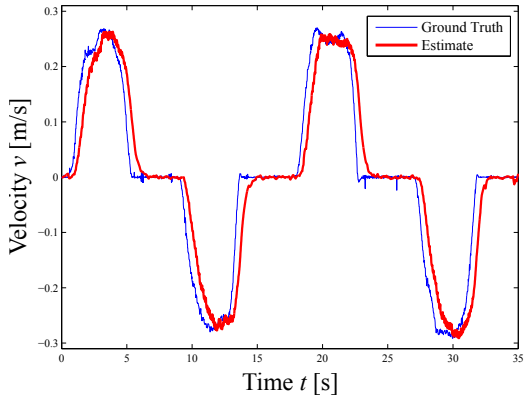

Figure 14: *Comparison of estimated velocity with ground truth.*

The results obtained for a robot moving with the speed of 25 cm/s showed, that parameters tunned for static experiments in Subsec. 4.2 are not efficient for moving obstacles. Results shown in Fig. 14 were obtained with measurement covariance $R = \mathrm{diag}(1, ..., 1)$ and process covariance $Q = \mathrm{diag}(0.01, 0.1, 0.01, 0.1, 0.01, 0.1)$. One can note, that variances applied to prediction of rate variables (i.e. $\dot{x}_0, \dot{y}_0, \dot{r}$) are ten times greater than the rest of variables. This implies less trust into prediction of velocities (which, from the model (15), assumes no change in time). The experiment provided vital information for tunning process. For the mobile or movable objects (if one can classify them as such) one should resign from noise attenuation for the sake of better velocity estimation.

4.6. Use of velocity in correspondence problem

Estimated obstacle velocity can improve the results of correspondence problem. Taking direction of obstacle motion into consideration, when calculating cost between two obstacles, can eliminate incorrect matching while both obstacles are close to each other. Consider an old, tracked obstacle c_i with estimated velocity $\vec{v}_i$ and a new obstacle c_j which velocity we yet do not know (Fig. 15(a)). We assume that the estimated velocity $\vec{v}_i$ is good.

First, we calculate the standard cost value c_{ij} from (12). Next, we transform the coordinates of obstacles center points to a new coordinate frame F_o, with X_o direction aligned with the direction of $\vec{v}_i$ (Fig. 15(b)). Then we evaluate a penalty function defined as:

$$f_{ij} \triangleq \exp\left(-\frac{1}{2}\left(\vec{p}^{\,o}_{0j} - \vec{\mu}^{\,o}_i\right)^\top \Sigma_i \left(\vec{p}^{\,o}_{0j} - \vec{\mu}^{\,o}_i\right)\right), \tag{17}$$

where $\vec{p}^{\,o}_{0j}$ is the center of new obstacle described in the F_o coordinate frame, $\vec{\mu}^{\,o}_i = \vec{p}^{\,o}_{0i} + T_\mathrm{P}\vec{v}^{\,o}_i$ is the predicted position of old, tracked obstacle for the current sample of time, also described in the F_o coordinate frame, and:

$$\Sigma_i \triangleq \begin{bmatrix} \sigma^2 + (T_\mathrm{P}|\vec{v}_i|)^2 & 0 \\ 0 & \sigma^2 \end{bmatrix}$$

is a covariance-like matrix, describing ditribution of penalty function (Fig. 15(c)). User-defined parameter σ (e.g. 0.15 m) describes the standard deviation of position. One can note, that the distribution of penalty function situates the old obstacle in one of the focal points

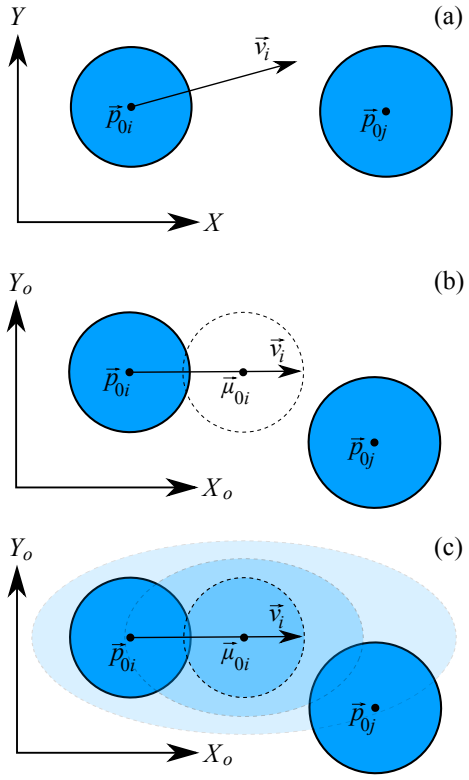

Figure 15: *Graphical representation of inclusion of velocity into evaluation of correspondence cost.*

of the standard deviation ellipse and evaluates to $f_{ij} = 1$ if the new obstacle is situated in the predicted position of old obstacle $\vec{\mu}_i^o$ for the current time sample.

Finally, we evaluate a new cost function defined as:

$$c'_{ij} = \frac{c_{ij}}{f_{ij}} \tag{18}$$

and input it into cost matrix (13). The penalty function increases the cost of correspondence for obstacles situated far from the distribution. However, it has the drawback on nonintuitive tunning for correspondence threshold c_{corr}.

5. Conclusions

Proposed method provides a solution to the problem of detection and tracking of geometrical obstacles from data provided by LRFs. The pure obstacle detection can work separately (without tracking) and gives good results in detection of both segment-type and circular obstacles (the latter are based on the former, Fig. 16). The tracking system additionally

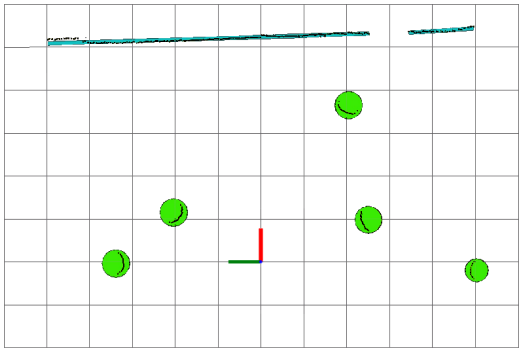

Figure 16: *Exemplary result of obstacle detection for real scanner data. Circular obstacles are represented by disks.*

enhances estimated state of circular obstacles and temporarily sustains their existence even in the case of sudden disappearance from the workspace. Presented results show a problem of mixed pixels influence on obtained values. The quality of velocity estimation was also presented and the directions for tunning of KF were provided. Classification of obstacles into stationary, movable or mobile could provide a vital information for KF tunning.

Potential improvement for proposed method can be obtained by:

- reduction of mixed pixel effect,
- providing tracking system for segment-type obstacles,
- classification of obstacles into stationary, movable or mobile and application of appropriate KF parameters.

Author would like to note, that the implementation of proposed algorithm can be found at online repository: `www.github.com/tysik/obstacle_detector`.

References

[1] A. Azim and O. Aycard, Detection, classification and tracking of moving objects in

a 3D environment. *Proceedings of IEEE Intelligent Vehicles Symposium*, 2012, pp. 802–807.

[2] J.L. Chen and J.S. Liu and W.C. Lee, A recursive algorithm of obstacles clustering for reducing complexity of collision detection in 2D environment. *Proceedings of IEEE International Conference on Robotics and Automation*, 2001, Vol. 4, pp. 3795–3800.

[3] F. Kamil, S. H. Tang, W. Khaksar, N. Zulkifli, S. A. Ahmad, A Review on Motion Planning and Obstacle Avoidance Approaches in Dynamic Environments. *Advances in Robotics and Automation*, 2015, Vol. 4(2).

[4] O. Khatib, Real-time obstacle avoidance for manipulators and mobile robots. *Proceedings of IEEE International Conference on Robotics and Automation*, 1985, Vol. 2, pp. 500–505.

[5] L. Kneip and F. Tache and G. Caprari and R. Siegwart, Characterization of the compact Hokuyo URG-04LX 2D laser range scanner. *Proceedings of IEEE International Conference on Robotics and Automation*, 2009, pp. 1447–1454.

[6] V. Kunchev, L. Jain, V. Ivancevic, A. Finn, Path Planning and Obstacle Avoidance for Autonomous Mobile Robots: A Review. *Proceedings of International Conference on Knowledge-Based Intelligent Information and Engineering Systems*, 2006, pp. 537–544.

[7] A. Mendes, L. C. Bento and U. Nunes, Multi-target detection and tracking with a laserscanner. *Proceedings of IEEE Intelligent Vehicles Symposium*, 2004, pp. 796–801.

[8] L. Montesano, J. Minguez and L. Montano, Modeling the static and the dynamic parts of the environment to improve sensor-based navigation. *Proceedings of IEEE International Conference on Robotics and Automation*, 2005, pp. 4556–4562.

[9] V. Nguyen and A. Martinelli and N. Tomatis and R. Siegwart, A comparison of line extraction algorithms using 2D laser rangefinder for indoor mobile robotics. *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005, pp. 1929–1934.

[10] A. H. A. Rahman, H. Zamzuri, S. A. Mazlan and M. A. A. Rahman, Model-based detection and tracking of single moving object using laser range finder. *Proceedings of International Conference on Intelligent Systems, Modelling and Simulation*, 2014, pp. 556–561.

[11] C. Sanderson and R. Curtin, Armadillo: a template-based C++ library for linear algebra. *Journal of Open Source Software*, 2016, Vol. 1, pp. 26.

[12] P. Skrzypczynski, How to recognize and remove qualitative errors in time-of-flight laser range measurements. *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008, pp. 2958–2963.

[13] P. Skrzypczynski and P. Drapikowski, Environment modelling in a multi-agent mobile system. *Proceedings of Third European Workshop onAdvanced Mobile Robots*, 1999, pp. 41–48.

[14] S. Sugiyama, J. Yamada and T. Yoshikawa, Path planning of a mobile robot for avoiding moving obstacles with improved velocity control by using the hydrodynamic potential. *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 1421–1426.

[15] S. H. Tang, W. Khaksar, N. B. Ismail and M. K. A. Ariffin, A review on robot motion planning approaches. *Pertanika Journal of Science & Technology*, 2012, Vol. 20(1), pp. 15–29.

[16] S. Waydo and R. M. Murray, Vehicle Motion Planning Using Stream Functions. *Proceedings of IEEE International Conference*

on Robotics and Automation, 2003, Vol. 2, pp. 2484–2491.

[17] C. Wei and Y. Zeng and T. Wu, Obstacle detection based on a 2D large range sonar model. *Proceedings of International Congress on Image and Signal Processing*, 2013, Vol. 2, pp. 1127–1131.

[18] G. Welch and G. Bishop, An Introduction to the Kalman Filter. ACM SIGGRAPH `http://web.archive.org/web/20080207010024/http://www.808multimedia.com/winnt/kernel.htm`, 2001, Course 8.

[19] J. Xavier and M. Pacheco and D. Castro and A. Ruano and U. Nunes, Fast Line, Arc/Circle and Leg Detection from Laser Scan Data in a Player Driver. *Proceedings of IEEE International Conference on Robotics and Automation*, 2005, pp. 3930–3935.

[20] C. Ye, Mixed pixels removal of a Laser Rangefinder for mobile robot 3-D terrain mapping. *Proceedings of International Conference on Information and Automation*, 2008, pp. 1153–1158.

[21] M. Zohaib, M. Pasha, R. A. Riaz, N.Javaid, M. Ilahi, R. D. Khan, Control Strategies for Mobile Robot With Obstacle Avoidance. *Journal of Basic and Applied Scientific Research*, 2013, Vol. 3(4), pp. 1027–1036.