Real-Time Avoidance Strategy of Dynamic Obstacles via Half Model-Free Detection and Tracking With 2D Lidar for Mobile Robots

Huixu Dong, *Member, IEEE*, Ching-Yen Weng, Chuangqiang Guo, Haoyong Yu, *Senior Member, IEEE*, and I-Ming Chen, *Fellow, IEEE*

***Abstract*—Avoidance is a necessary capability for a mobile robot to perform tasks, such as delivering objects in household or industrial scenarios. The existing avoidance strategy based on timed elastic band local planner and cost-map provided by robotics operating system cannot realize the excellent performance when a robot and an obstacle both move. In this article, we present a real-time, simple, and reliable approach to detecting and tracking obstacles via a two-dimensional lidar in dynamic scenarios where the mobile robot and the obstacle are moving. Obstacles are represented by a set of points against their outlines and the information of obstacles is initialized and updated via the raw laser measurement. First, the obstacle is detected by three main steps: preprocessing, segmentation, and merging, classification of consequent measurements. Second, we use a hierarchical method to realize data associations for figuring out the corresponding matches among obstacles with the consecutive time. Last, after doing the data association, we need to estimate the motion of the dynamic obstacle for being tracked by the Kalman filter. Extensive experiments performed in the simulation and practical scenarios indicate that the proposed method enables a mobile robot to perform dynamic avoidances efficiently [Real-time Avoidance Strategy of Dynamic Obstacles via Half Model-free Detection and Tracking (T-MECH)].**

***Index Terms*—Dynamic avoidance, mobile robot, obstacle detection, obstacle tracking, two-dimensional (2D) lidar.**

Manuscript received January 29, 2020; revised June 5, 2020 and July 19, 2020; accepted October 27, 2020. Date of publication October 30, 2020; date of current version August 13, 2021. Recommended by Technical Editor X. Zhang and Senior Editor X. Chen. This work was supported by the State Key Laboratory of Robotics and System (HIT) Open Cooperation under Grant SKLRS-2019-KF-02. *(Corresponding author: Chuangqiang Guo.)*

Huixu Dong, Ching-Yen Weng, and I-Ming Chen are with the Robotics Research Centre, Nanyang Technological University, Singapore 639798, Singapore (e-mail: dong0076@e.ntu.edu.sg; weng0025@e.ntu.edu.sg; michen@ieee.org).

Chuangqiang Guo is with the Robotics Institute, Harbin Institute of Technology, Harbin 15000, China (e-mail: chuangqiang.guo@hit.edu.cn).

Haoyong Yu is with the Bio-robotics Laboratory, National University of Singapore, Singapore 119077, Singapore (e-mail: bieyhy@nus.edu.sg).

This article has supplementary downloadable material available at https://ieeexplore.ieee.org, provided by the authors.

Color versions of one or more of the figures in this article are available online at https://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMECH.2020.3034982

I. Introduction

AVOIDING obstacles, a key function of mobile navigation, poses a typical challenge for mobile robots, for which the real-time perception is considered as the key bottleneck because there is a large amount of the environment information that needs to be processed. The detection and tracking of participants such as pedestrians and robots play a crucial role in a safe mobile navigation. A common scenario involves the autonomous navigation of a mobile robot, in which a robot is required to detect obstacles and predict the motion of moving obstacles in unstructured environments, as shown in Fig. 1. The perception efficiency for dynamic avoidance yields the limitations of the capability of robots making decisions such as avoidance actions, especially in typical human-involved environments with considerable clutter. Moreover, few of the perception strategies can run in real time due to a large number of relevant vectors.

In terms of robotic applications, a perception system interprets the surroundings using visual cameras and/or laser scanners [1]–[6]. Although vision-based perception approaches have advantages especially in semantic understanding [7], their major limitations result in high sensitivity to illumination. Here we consider the 2D point-cloud measurement from a 2D lidar which is with high resolution, insensitive to lighting conditions, and just costs less computational resources. There is a classical avoidance strategy including the timed elastic band (TEB) local planner [8]–[10] and the cost-map converter in the robotics operating system (ROS) [11]. Specifically, the TEB problem is formulated as a weighted multiobjective optimization framework by dynamic constraints of the motions. Moreover, the TEB local planner utilizes a cost-map converter [8]–[10] to transform occupied cells of 2D map to a set of convex geometric primitives (points, lines, polygons) that represent obstacles in the map by clustering. By integrating the geometric primitives representing obstacles during the navigation period, the TEB local planner can avoid obstacles. However, all the obstacles detected by a mobile robot are also divided into a large number of geometric primitives considered as obstacles by TEB local planners, which lead to a considerable computational resource. Thus, a mobile robot with the TEB local planner has unsatisfactory performances in avoiding dynamic obstacles.

The target of this article is to achieve safe navigation for a mobile robot in indoor environments with some pedestrians. We

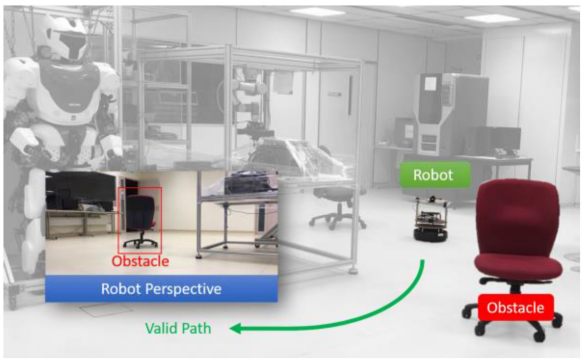


Fig. 1. Robot avoiding an obstacle based on the proposed algorithm.

propose a principled framework for the detection and tracking of dynamic obstacles by means of one single 2D lidar regardless of obstacle classes and shapes, integrating the TEB local planner to realize obstacle avoidance. The circles with infinite height and the line are used to be representatives for modeling the obstacles in the environment. To realize a collision-free navigation, we approximate the real underlying model of the obstacle which is deemed acceptable for yielding a dynamic local map since the obstacle recognition is not our objective. The motivation of the line and the circle representing obstacle models is threefold. First, a large obstacle such as a long wall is represented by a line with just two endpoints. Second, smaller obstacles such as desk legs or moving persons in practical scenarios are, in general, modeled as cylinder-like shapes. Third, the circle model with three parameters in the 2D space is inexpensive to compute comparing to occupancy grids. Third, enlarging the dimension of the cylinder can generate a safe distance between an obstacle and a moveable robot. As for the pre-processing, we first use the median filter to discard the noise and segment the 2D point cloud into independent laser-point cloud blocks collected by the 2D laser. Then, using geometric primitive representation, the segmentation is performed on laser-point cloud blocks, followed by merging (clustering) laser-point cloud blocks with small distances. Further, all the detected obstacles are modeled as lines and circles. Despite a higher computation resource, we indicate that some online calculations can still be completed. Finally, we utilize a hierarchical data association strategy based on Kalman Filter (KF) to estimate the motions.

We aim to achieve a better avoidance strategy by the proposed obstacle detection and tracking algorithm when the mobile robot and the obstacle are moving. However, robotic navigation is a systemic work, not only combining the perception system but also integrating the robotic control [12], planners [13], and localization [14]. In this work, the simple designed controller based on a differential kinematic model is used for driving a two-wheel mobile robot. The global planner in ROS is integrated into the robotic navigation system for planning a global path and a particle filter-based adaptive Monte Carlo localization is adopted for tracking the robot's pose in a known map extracted from 2D data.

The rest of this article is organized as follows. After reviewing existing methods in Section II, we introduce the core concept of detecting and tracking dynamic obstacles in detail in Section III. The performance of the proposed avoidance strategy is evaluated by abundant experiments in Section IV. Finally, Section V concludes this article.

II. Related Work

There exists a vast amount of research works regarding the problem of detection and tracking of multiple dynamic obstacles.

In terms of describing the surrounding environment, these existing lidar-based approaches can mainly be classified into two categories in detail: 1) grid-based [15], [16]: and 2) vector-based [3] methods. Most approaches apply the 2D or 3D occupancy grid (grid-based) to represent environments [17]. In particular, the occupancy grid in a map is divided into spaced cells and then complex geometries are used for representing obstacles for detecting and tracking obstacles. For example, the grid trajectories features were applied to detecting dynamic obstacles in [18]. Another commonly used representation of the environment, the vector-based method, incorporates simple and higher-level predefined geometric features such as line segments, circles, ellipses and rectangles, boxes to directly model obstacles in [19]–[21]. Therefore, obstacles described by geometric features can be expressed via the pose (position and orientation) of the geometric feature. In comparison to the first category, the vector-based approach with the compact representation of the surrounding environment is particularly suitable for describing a sparse scenario due to the low memory consumption. Our work concentrates on the obstacle detection and tracking by 2D vector-based representatives.

The dynamic obstacle avoidance system involved by the perception prediction and path planning is modifying and estimating the pose of a moving robot in real time such that the robot is able to avoid collisions with moving obstacles found on its path. There are several classical methods such as virtual force field (VFF) [22], [23], vector field histogram (VFH) [24], dynamic window approach (DWA) [25]and TEB approaches [8], [9], [26]. In terms of VFF [22], [23], the histogram grids are applied to represent the area through which the robot can pass. However, such a method is not available when the mobile robot and moving obstacle close to each other generate the repellent effect. To address such an issue in VFF, Borenstein and Koren [24] presented the VFH method that employs the 2D histogram grid to describe the surroundings for obtaining a 1D polar histogram constructed around the momentary location of the robot. In contrast to VFF and VFH methods, DWA was developed to avoid obstacles by means of defined cost functions based on the constraints of mobile robots [25] on the kinematics and dynamics. The elastic band based method can adapt to dynamic path changes by adjusting the path to generate a new once a new obstacle is detected [26].

III. Methodology

In the section, we first introduce how to detect obstacles in Section III-A. Then, the details of tracking obstacles are described in Section III-B.

A. Obstacle Detection

1) Preprocessing: A set $\boldsymbol{p}$ with N measurement points is from a lidar scan. We represent each raw point p_i in $\boldsymbol{p}$ in the form of Polar coordinates (R_i, θ_i) as follows:

$$\boldsymbol{p} \triangleq \{\, p_i = (R_i, \theta_i) \,\}, i \in [1,\ N]\,. \tag{1}$$

The accuracy of measurements is disturbed by the noise. The outliers may lead to such a situation where some measurement points fail to represent an obstacle as they may be unstable or noise points in practical environments. A range filter is integrated into the proposed algorithm to remove useless points with invalid values represented by NaN or beyond the range. A 3×3 median filtering matrix is applied to smoothing the 2D point data. The distance of a point i from a laser scan is considered as $R_{(t,\ i)}$ at the current time t. Thus, the 3×3 median matrix used for filtering out some points is given as follows:

$$\begin{bmatrix} R_{(t-1,\ i-1)} & R_{(t-1,\ i)} & R_{(t-1,\ i+1)} \\ R_{(t,\ i-1)} & R_{(t,\ i)} & R_{(t,\ i+1)} \\ R_{(t+1,\ i-1)} & R_{(t+1,\ i)} & R_{(t+1,\ i+1)} \end{bmatrix}. \tag{2}$$

We use the median of the nine elements in the median filter as the distance between the laser point i and the laser mounted on the robot at the time t for discarding noise points in consideration of the space and time, reducing the complexity and increase accuracy of the obstacle classification.

After the noise points are filtered out, the large laser-point cloud is segmented into independent small laser-point cloud blocks in order. For each laser-point cloud block, the starting point index satisfies $R_{i-1} = 0$ and $R_i \neq 0$. Moreover, the ending point index satisfies $R_i \neq 0$ and $R_{i+1} = 0$, as shown in Fig. 2(a).

2) Segmentation and Merging: All laser points are segmented into point-cloud blocks that have strong correlations based on practical properties. There exist two main segmentation types for the point cloud measured via a lidar such as the complete segmentation and partial segmentation. The complete segmentation means one obstacle is considered as one single segment. In terms of partial segmentation, one obstacle can be characterized by two or more segments (e.g., human legs). In this article, we work on partial segmentation. In practical environments, some large segments, such as walls, are usually interrupted by small intervals, which results in a wrong case that one obstacle is represented by more than one segment. It is obvious that the density of measurements gradually decreases in the lidar data and the layout of points becomes sparse as the measured distance increases [see Fig. 2(a)]. Here we use the distance of two endpoints of the small interval to determine the segmentation.

For segmenting the data after filtering, we first determine the distance d between the two sequential measurement points i and $i+1$ defined in the polar coordinate

$$d\ (R_i, R_{i+1}) = \sqrt{{R_i}^2 + {R_{i+1}}^2 - 2R_iR_{i+1}\cos\Delta\alpha} \tag{3}$$

where R_i and R_{i+1} represent the measurement distances of the points i and $i+1$. $\Delta\alpha$ denotes the resolution angle of the lidar. If d is bigger than kW, the laser-point cloud block is segmented into two small blocks, as shown in Fig. 2(b). W denotes the width of the robot and k is the setting dynamic amplification factor as follows:

$$\begin{cases} k = \frac{W \cdot R_i \cdot R_{i+1}}{100(R_i + R_{i+1})}, & \frac{R_i + R_{i+1}}{2} > T_d \\ k = 0.15, & 0 < \frac{R_i + R_{i+1}}{2} \leq T_d \end{cases} \tag{4}$$

where T_d($T_d = 10$) is a threshold of the measured distance. We define the dynamic amplification factor k to adjust the distance threshold between the two adjacent points with the width of the robot and the measured distance changing. For example, as the measured distance increases, the distance between two adjacent lidar points also becomes larger. We increase the threshold to reduce the number of segments. Moreover, if the robot width is large, the small gap with a small distance between the two adjacent points tends to merge together so that the robot cannot go through the gap and reduce the number of segments in the long distance.

The merging processing needs to be done for some laser-point cloud blocks. As shown in Fig. 2(c), L is the space distance between the termination point of the first point set with the index value p and the starting point of the last point set with the index q. This connection can be done if the following condition is met.

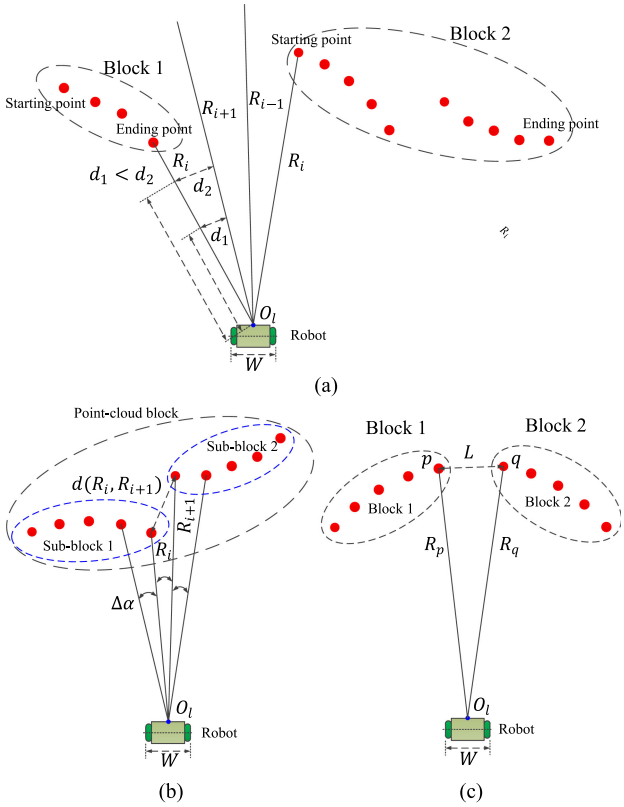


Fig. 2. (a) Segmenting the laser-point cloud into independent laser-point cloud blocks, (b) segmenting one block into two small blocks, and (c) merging two blocks into one blocks. d_1 or d_2 is the difference of two points, on the two laser rays, with equal distance to the laser origin O_l; R_i represents the distance of the point i; $d(R_i, R_{i+1})$ denotes the distance between the two sequential measurement points; L indicates the distance between the termination points of two adjacent laser-point cloud blocks; W represents the width of the robot; the red dots are the laser points.

Algorithm 1: Splitting Segments Into Subsets.

Data:
$N \leftarrow$ *The number of measurement points in the point-cloud block* $B \triangleq \{p_i\}, i \in [1, N]$;
$T_n \leftarrow$ *The threshold;*
$D_m \leftarrow$ *The maximum value of the distances between each point and the line fit by two endpoints;*
$S \leftarrow$ *the distance between two endpoints;*
$V_{result} \leftarrow$ *the vector storing the point-cloud block with N less than* T_n;
$V_c \leftarrow$ *the vector storing the point-cloud block with N more than* T_n;
$N_c \leftarrow$ *the number of elements in* V_c;
$N_{c0} \leftarrow$ *the number of points in* $V_c[0]$;

Put B into V_c
While $(N_c \neq 0)$ ***do***
  If $(N_{c0} > T_n)$
    Calculate D_m *of* $V_c[0]$ *and get* p_k *that corresponds to* D_m
    If $(D_{max} > 0.2|S|)$
      Split $V_c[0]$ *into* $B_1(B_1 \triangleq \{p_1, p_2, \ldots, p_k\})$ *and* $B_2(B_2 \triangleq \{p_k, p_{k+1}, \ldots, p_{N_{c0}}\})$
      If $(k > T_n)$
        Put B_1 *into* V_c
      Else
        Put B_1 *into* V_{result}
      If $(n - k + 1 > T_n)$
        Put B_2 *into* V_c
      Else
        Put B_2 *into* V_{result}
      Delete $V_c[0]$
    Else
      Delete $V_c[0]$
  Else
    Delete $V_c[0]$ *in* V_c
    Put $V_c[0]$ *into* V_{result}
End

Output: V_{result}

The robot cannot pass the narrow gap if L is smaller than W: $L < W$. The enhancement of the algorithm involves the combination of two segments separated by a small interval. In this case, these two laser-point cloud blocks are merged into one laser-point cloud block, as shown in Fig. 2(c). Also, we compensate the corresponding values to those laser points between p and q for making the two combined point sets spatially continuous. In order to determine the number N of the compensated points, we calculate the average distance d_{aver} of the adjacent points of two point-cloud blocks and then $N = \frac{L}{d_{\text{aver}}}$. The distances of 2D points p and q is R_p and R_q, respectively, thus the distance of the compensated point i is represented as follows:

$$R_i = R_p + i \cdot \frac{R_q - R_p}{N}. \tag{5}$$

3) Classification: By segmenting and merging, the point cloud coming from a laser consists of independent point sets (point-cloud blocks). The basic idea of classification is to describe any obstacle via a set of simple shapes [27]. The point-cloud block can be formulated as two basic shapes such as the circle and liner segment. The approximation of the point-cloud blocks based on these geometric models has two major advantages. First, a number of discrete points used for representing any obstacle are replaced by just inexpensive parameters such as position and radius, which simplifies computing. Second, such averaging effect for describing the model can be robust to cope with the measurement noise. Although such an approximation may result in unnecessary loss of motion space of the mobile robot. Indeed, as our main target is to realize safe dynamic navigation rather than obstacle recognition, the real underlying model of the obstacle can be approximated as deemed acceptable.

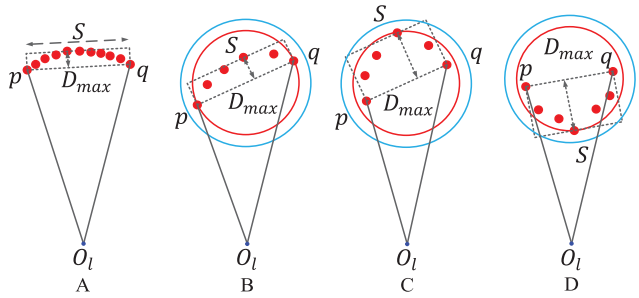


Fig. 3. Classification of segments. The two points with the starting index p and the ending index q in a laser-point cloud block; S denotes the distance between p and q; $D_{\max}$ represents the maximum one among distances from points on the laser-point block to the line segment formed by p and q; O_l is the laser origin; the dashed rectangles cover all the laser points and the red dots are the laser points.

To store the obstacles for further processing, we build a set of obstacles as $\mathbb{O}$, $\mathbb{O} \triangleq \{\mathbb{L}, \mathbb{C}\}$. $\mathbb{L}$ denotes a set of line-type obstacles and $\mathbb{C}$ represents a set of circle-type obstacles.

1) If the number of points of a point set is more than a predefined threshold T_N($T_N = 10$), we first use a line segment to connect the starting point p and the ending point q and then calculate the maximum value $D_{\max}$ of distances from each point to the line segment, as illustrated in Fig. 3. If $D_{\max} < 0.2|S|$, this point-cloud block will fit to a line segment with two endpoints p and q [see Fig. 3(a)].

If $D_{\max} > 0.2|S|$, the point-cloud block can be split further. The corresponding algorithm is shown in Algorithm 1. We recursively repeat this procedure for each new subset. When the number of points in a new subset is less than T_N, the splitting stops.

2) When the number of points in the point-cloud block is less than T_N, we utilize a circle to represent this point-cloud block with a relatively small size. There are two main reasons. First, if the size of point-cloud segments is very large (e.g., corridor walls), the circle extracted from such a segment would cause that an obstacle covers a vast area of the workspace. The circle representative is suitable for a point-cloud block with a small size. Second, indeed, the size of a dynamic obstacle (e.g., humans) is far less than that of static obstacles (e.g., walls) in indoor environments.

To determine the radius of the circle extracted from a point-cloud block, we first use the method stated above to fit the point set. When the point-cloud block satisfies $D_{\max} < 0.2|S|$, the midpoint of line segment connected by two endpoints is set as the center and the distance between the center and the maximum-distance point is the circle's radius [see Fig. 3(b)]. Otherwise, we first need to determine the convexity or concavity of the

point-cloud block since the parameters of the circle extracted from a point-cloud block depend on the convexity or concavity.

Here we propose a highly efficient algorithm to determine the convexity or concavity of the point-cloud blocks based on the vector cross product, as shown in Algorithm 2. We first take the two endpoints of the segment and then obtain the midpoint of them, considering the midpoint as the center of the estimated shape. Further, the distance from the center to each point of the segment is calculated so that the maximum one among these distances is extracted. Finally, this maximum distance serves as the circle's radius.

In order to realize a free collision between a robot and an obstacle, the radius of the circle is enlarged by a margin value defined as a safe distance, as shown in Fig. 3 (blue circles). If this segment is concave, the zone of the triangle formed by two endpoints of the segment and the origin of the lidar is free for the mobile robot [see Fig. 3(c)]. While if the segment is convex, the zone between the origin of lidar and the nearest point from the lidar to the segment is safe. The robot can move at the safe zone [see Fig. 3(d)]. All the point-cloud blocks represented by circles are added to the set $\mathbb{C}$.

Algorithm 2: Determine the Convexity or Concavity of The Point-Cloud Block.

Data:
$o_l \leftarrow$ *The origin of the lidar;*
$T_{io} \leftarrow$ *The threshold;*
$N \leftarrow$ *The number of measurement points in the point-cloud block*
$B \triangleq \{p_i\}, i \in [1, N]$;
$V_i \leftarrow$*the vector storing* N_i *points inside* $\Delta o_l p_1 p_N$;
$V_o \leftarrow$*the vector storing* N_o *points outside* $\Delta o_l p_1 p_N$;
$r_{io} \leftarrow$*the ratio of the numbers of points inside* $\Delta o_l p_1 p_N$ *to outside* $\Delta o_l p_1 p_N$;

Suppose that the order of three points consisting of the triangle $\Delta o_l p_1 p_N$ *is anticlockwise such as* o_l, p_1, p_N .
True represents B is convex relative to the origin of the lidar;
False denotes B is concave relative to the origin of the lidar.

While $(i \neq N)$ ***do***
  Calculate three cross products as follows,
  $T_1 = p_i o_l \otimes p_i p_1$;
  $T_2 = p_i p_1 \otimes p_i p_N$;
  $T_3 = p_i p_N \otimes p_i o_l$;
  If $(T_1 > 0 \;\&\&\; T_2 > 0 \;\&\&\; T_2 > 0)$
    $\|\; (T_1 < 0 \;\&\&\; T_2 < 0 \;\&\&\; T_2 < 0)$
    Put p_i *into* V_i
  Else
    Put p_i *into* V_o
  End
  $r_{io} = \frac{N_i}{N_o}$;
  If $(r_{io} > T_{io})$
    True;
  End
    False.

Output: True *or* ***False***

B. Obstacle Tracking

We implement a simple KF [28] into the proposed system, allowing for the tracking of circular obstacles. The detailed tracking algorithm is described in this section. Specifically, we introduce how any dynamic obstacles with any shape be modeled such as to be tracked. In terms of tracking new dynamic obstacles, the state initialization is nontrivial due to the motion arbitrariness and independent of each other. New dynamic obstacles can be initialized by the system and put into new track containers while static obstacles need to be merged into the static background for achieving dynamic tracks. After initializing a new track, we put it into the set of existing dynamic tracks if it is continuously detected more than predefined times; otherwise, this track will be discarded.

1) Hierarchical Data Association:

a) Coarse-level data association: After classing the measurements from a lidar scan, the point-cloud segment (point set) is then assigned to the static background or a dynamic obstacle recursively. The target of the data association is to figure out which detected obstacle taken at the last time interval $t-1$ corresponds to which detected obstacle captured at the current time interval t (Note that $t-1$ represents the last time interval rather than the time before 1 second). Specifically, to track obstacles, two-point sets P and Q detected at different time intervals are aligned. The global nearest-neighbor search algorithm is applied to finding the "best" fits for the existing tracks. If no correspondences are found for some point sets, then these point sets are considered as new tracks and are initiated by the KF. The new point set P is associated with its nearest neighbor Q if the Euler distances of their positions and radii are within two thresholds (5 and 3), respectively; otherwise, it is considered as a new point set to be tracked. First, obstacles detected at $t-1$ are considered as old obstacles, and obstacles detected at t are regarded as new obstacles. Then, we obtain the distances between each old obstacle and each new obstacle regarding the positions and the sizes by a traversing algorithm. Third, for each old/new obstacle, we ascendingly sort its distances with new/old obstacles. Thus, corresponding new/old obstacles with minimum distance to each old/new obstacle are extracted. Based on the pairing information, two preliminary correspondence maps including $M_{n\to o}$ and $M_{o\to n}$ are generated. Finally, a preset distance threshold $T_{\max}$ $(T_{\max} = 5)$ is used in $M_{n\to o}$ and $M_{o\to n}$ for removing the correspondences with distances larger than $T_{\max}$. We update these correspondences to the map M_r.

b) Fine-level data association: Given M_r, the fine level data association can further take into account the following correlations among observations. Normally, each detected new obstacle corresponds to only one old obstacle in the environment. However, in this resulted correspondence map, there might be that one new/old obstacle is associated with multiple old/new obstacles. A case example is human's legs while walking. These two cases can be detected by exploring the duplicate correspondences in the map M_r. Determination of the correspondence type for currently examined obstacles provides crucial information in terms of the state update. The state of a new obstacle is updated, relying on old obstacles. Any new obstacle is updated depending on the original obstacle in the second case. All obstacles involved in these two cases are marked as tracked ones.

It is possible that some new obstacles are not associated with any old obstacle after the data association. These unmatched

point sets are then regarded as new obstacles in the proposed detection and tracking system. These new obstacles are put into the dynamic map such as to be initiated and tracked by KFs. If an obstacle is deemed dynamic, its state is estimated by KF.

2) Kalman Filter: For realizing dynamic avoidance, the perception system needs to estimate the motion parameters of each moving obstacle. For tracking obstacles, we utilize KF [28] to estimate the states of detected obstacles. Each newly detected obstacle marked as tracked will be initialized by a separate filter. For each sample time, the used filter updates the prediction and the correction steps. The position and size of obstacles and the rates of change of each variable describing the obstacle's state are estimated for realizing the safe navigation of a mobile robot in the 2D environment. We denote a vector including detected obstacles as $\mathcal{A}_t$ at the time instant t. This vector is provided as follows:

$$\mathcal{A}_t = \left\{\mathcal{A}_t^i\right\} \ , i \in [1, n] \, , \ n \geq 1 \tag{6}$$

where

$$\mathcal{A}_t^i = \left[x_t^i, \dot{x}_t^i, y_t^i, \dot{y}_t^i, \ r_t^i, \dot{r}_t^i, I_t^i \ \right]^T$$

and n represents the number of obstacles in an environment. Equation 6 illustrates the state of the ith obstacle, with x_t^i and y_t^i being the position of the obstacle, r_t^i the circle radius, $\dot{x}_t^i$ and $\dot{y}_t^i$ the linear velocities, and I_t^i the index of the obstacle. Due to the invariance of the index of the obstacle, we omit the parameter in the equations. With such a definition of the state vector, we present the track state transition modeled

$$\mathcal{A}_{t+1}^i = \ F\mathcal{A}_t^i + Gw_t. \tag{7}$$

F is expressed as follows:

$$F \ = \begin{bmatrix} A & 0 & 0 \\ 0 & A & 0 \\ 0 & 0 & A \end{bmatrix} \text{with } A \ = \begin{bmatrix} 1 & s \\ 0 & 1 \end{bmatrix}. \tag{8}$$

s is the length of a sampling period. G represents the noise gain matrix, which is expressed as follows:

$$G \ = \begin{bmatrix} B & 0 & 0 \\ 0 & B & 0 \\ 0 & 0 & B \end{bmatrix} \text{with } B \ = \begin{bmatrix} \frac{1}{2}s^2 \\ s \end{bmatrix}. \tag{9}$$

The measurement equation can be formulated as follows:

$$m_{t+1} = \ H\mathcal{A}_{t+1}^i + v_{t+1} \tag{10}$$

where H is the measurement model, expressed as

$$H \ = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}. \tag{11}$$

According to the chosen kinematics model, the process noise w_t and measurement noise v_{t+1}satisfy zero-mean white Gaussian noise distributions

$$p\left(w\right) \sim N\left(0, \ Q\right) p\left(v\right) \sim N\left(0, \ R\right). \tag{12}$$

Thus, covariance kernels

$$E\left\{w_{t_i} {w_{t_j}}^T\right\} = Q_{t_i} \, \delta_{ij} E\left\{v_{t_i} {v_{t_j}}^T\right\} \ = R_{t_i} \, \delta_{ij} \tag{13}$$

where δ_{ij} is Kronecker delta function. When all of the parameters are known, the Kalman filtering equations are as follows:

$$\hat{\boldsymbol{A}}_{(tt)}^i = \hat{\mathcal{A}}_{(tt-1)}^i \ + K_t \left[m_t - H\hat{\mathcal{A}}_{(tt-1)}^i\right] \tag{14}$$

$$\hat{\mathcal{A}}_{(t+1t)}^i = \ F\hat{\mathcal{A}}_{(tt)}^i \tag{15}$$

where $\hat{\mathcal{A}}_{(tt)}^i$ is the estimate of $\mathcal{A}_t^i$ based on the measurements $\{m_0, \ \ldots, m_t\}$ and $\hat{\mathcal{A}}_{(t+1t)}^i$ is the prediction of $\mathcal{A}_{t+1}^i$ with the measurements $\{m_0, \ \ldots, m_t\}$. The matrix K_t is the "KF gain" and is determined by

$$K_t = P_{(tt-1)} \ H^T \left[HP_{(tt-1)}H^T + R_t\right]^{-1} \tag{16}$$

where the covariance matrix of the error in predicting $\mathcal{A}_t^i$ is provided as

$$P_{(tt-1)} = \ E\left\{\left[\mathcal{A}_t^i - \hat{\mathcal{A}}_{(tt-1)}^i\right]\left[\mathcal{A}_t^i - \hat{\mathcal{A}}_{(tt-1)}^i\right]^T\right\} \tag{17}$$

and $P_{(tt-1)}$ is computed using

$$P_{(t+1t)} = \ FP_{(tt)}F^T + GQG^T \tag{18}$$

where

$$P_{(tt)} = P_{(tt-1)} \ - K_t H P_{(tt-1)}. \tag{19}$$

Generally, one new tracked obstacle corresponds to just one old obstacle. If a new obstacle maps more than one old obstacles, we use the average of states of old obstacles to be that of the new obstacle. When an old obstacle has more than one corresponding new obstacles, the original obstacle is copied for each new detected one and their states are updated in new information sets.

The convexity–concavity characteristics of the circle-shape model are invariant with respect to the distance. However, the size of the circle is adjusting with the measurement distance changing. Tracking the obstacle aims to achieve the state $\mathcal{A}_k$ of the circle representing an obstacle in real time for realizing the avoidance. The state includes the position, the radius, the velocities, the changing rate of the radius, and the obstacle identification. All the parameters in the state are updated at the same time. Hence, in parallel with the assigned obstacle, the parameters of the circle are conveyed to the avoidance system.

IV. Experiments and Discussions

Here, we evaluate the proposed avoidance strategy quantitatively and qualitatively. The performance of the proposed method is compared against one benchmarking obstacle avoidance approach that is a standard solution (TEB local planner + cost-map converter) in ROS in simulated scenarios (Section IV-A); the case of robotic avoidance experiments in real-word environments (Section IV-B).

The simulations and real experiments of avoiding obstacles are implemented on a Turtlebot-2 mobile robot with two wheels and a Hokuyo 2D laser scanner. The measurement distance is up to 20 meters and the sampling rate is 100 ms and the angular resolution is 0.36° for the used laser scanner. Indeed, the thresholds T_d, T_N, and $T_{\max}$ are determined experimentally according to the specific environment and the characteristic parameters of the

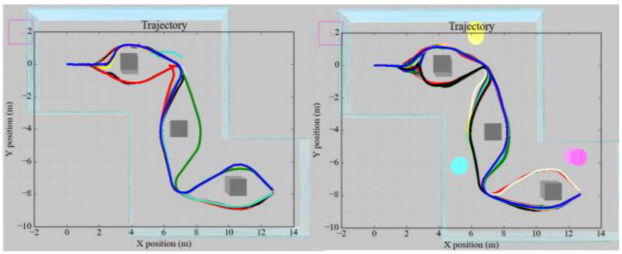


Fig. 4. Simulation tests in the z-shape area with different placements of 3 (left side) and 6 (right side) obstacles, respectively.

2D laser scanner. For instance, if the surrounding environment is small, we need to make $T_{\max}$ smaller.

A. Simulation Experiments

Simulation serves to create highly repeatable and consistent test conditions for evaluating a variety of avoidance implementations. To verify that the proposed algorithm is robust with dynamic layouts and obstacles, we conduct a lot of simulations with the open-source simulation platform-Gazebo. We locate obstacles along the path defined by three setting goals for (1) and one setting goal for (2). Before the simulation, we first run the SLAM algorithm [29] around the built environment to construct a global map. For every instantiated scene, we initialize the navigation pipeline with the detection-tracking algorithm and TEB local planner. The objective of each task for the robot is to simply move from the initial position to the destination without crashing.

1) Avoiding New Static and Dynamic Obstacles:

a) The first group of navigation tests is conducted in a simulated environment that involves positioning obstacles in a z-shape area. The robot's goal is to travel 22 m along the z-shape. The number of obstacles spawned is set to three and six. For each quantity of obstacles, 50 trails are created. The two test environments are depicted in Fig. 4, one sparsely populated scene and one more densely populated scene with more diverse obstacle categories. The trajectory visualizations show in detail that the robot completes the avoidances successfully without collisions. Fig. 4 records the dynamic trajectories of the mobile robot when it is operating around many obstacles. The participants traverse these areas with similar topology during each navigation route almost.

b) The practical relevance of the complex trials is to enable the navigation of avoiding obstacles in crowded environments, where new obstacles are unpredicted. There are 50 trial scenes with randomly spawned obstacles. To increase the difficulty of the scenario, we add more new obstacles in a fixed spawn zone on the original map than before, thereby increasing the density of the "obstacle forest." For each area (top, middle and bottom areas), we put six new obstacles randomly. Fig. 5 shows some examples of a robot avoiding a variety of obstacles spawn randomly in the z-shape area. The proposed system can enable a robot to achieve 96% success rate of avoiding obstacles in such complex scenarios.

c) As shown in Fig. 6, these simulations are intended to evaluate the system's ability to avoid the dynamic obstacle in complex environments. The robot confronts one dynamic obstacle in the form of one person stepping into the way of the robot and several new obstacles (standing persons) that do not belong to the known map in the course. This is important as any realistic scenario always contains some forms of dynamic obstacles. We set the velocities of the robot and the walking person as 0.75 and 0.5 m/s, respectively.

Most of the trajectories of the robot include moderate serrations since there are new obstacles on the pathways. We conduct 50 simulations. The robot reaches the destination successfully and correctly responds to the turns and new dynamic or static obstacles, except three cases that the robot stops before the walking person and has collisions with the walking person.

Here we present an experimental case. For instance, a series of snapshots in Fig. 6 shows that the robot involved in this simulation can realize the dynamic avoiding obstacles via the proposed avoidance algorithm. The robot bypasses the walking person safely and also avoids several standing persons successfully on the robot.

2) Performance Comparison Against the Baseline System in Dynamic Scenarios: Fig. 7 illustrates several simulation scenarios where a mobile robot needs to avoid a moving obstacle and attempts to arrive at the destination by a baseline avoidance strategy (TEB local planner + cost-map converter) in Fig. 7(d)–(f) and the proposed strategy (TEB local planner + detection-tracking algorithm) with different obstacle's speeds from 0.4 to 1.2 m/s with the 0.1 m/s interval in Fig. 7(a)–(c). Initially, the obstacle locates on the right side and the mobile robot stands at the left side in the simulation. Additionally, the goal is set at the right side of the obstacle so that the obstacle is located between the initial and final positions of the robot. Then, the moving obstacle moves toward the left direction while the robot moves toward the goal. Sequentially as it approaches the obstacle, the avoidance performance is executed. We evaluate the results with cases where the avoidance is successful by allowing the robot to execute its motion to the destination within the setting time (30 s). In simulation environments, if the robot has a collision with the obstacle, the execution time will be beyond the defined time. We declare failure if the robot does not arrive at the destination or use more than the setting time to complete. For each method with each speed of the obstacle, we simulate 200-time experiments of dynamic avoidances.

The avoidance process is clearly illustrated in Fig. 7. The circles represent the footages of both the mobile robot and the moving obstacle, respectively, at the constant interval 0.2 s in this figure. The distance of the centers of two successive circles indicates the velocity scale. That is, if this distance is big, the velocity is big, vice versa. From Fig. 7, we can see that the velocity of the mobile robot decreases significantly when it avoids the heading obstacle. It can be seen that the trajectories of Fig. 7(a)–(c) are smoother than the cases of Fig. 7(d)–(f) in terms of the same velocity.

As for the cases of Fig. 7(a)–(c), their linear, angular velocities, and orientations are shown in Fig. 8(a)–(c) for the proposed avoidance system, respectively. Similarly, Fig. 8(d)–(f) also shows the linear, angular velocities, and orientations of the cases of Fig. 7(d)–(f). Although the linear velocity has drastic impulses during the avoiding period, the motion remains almost

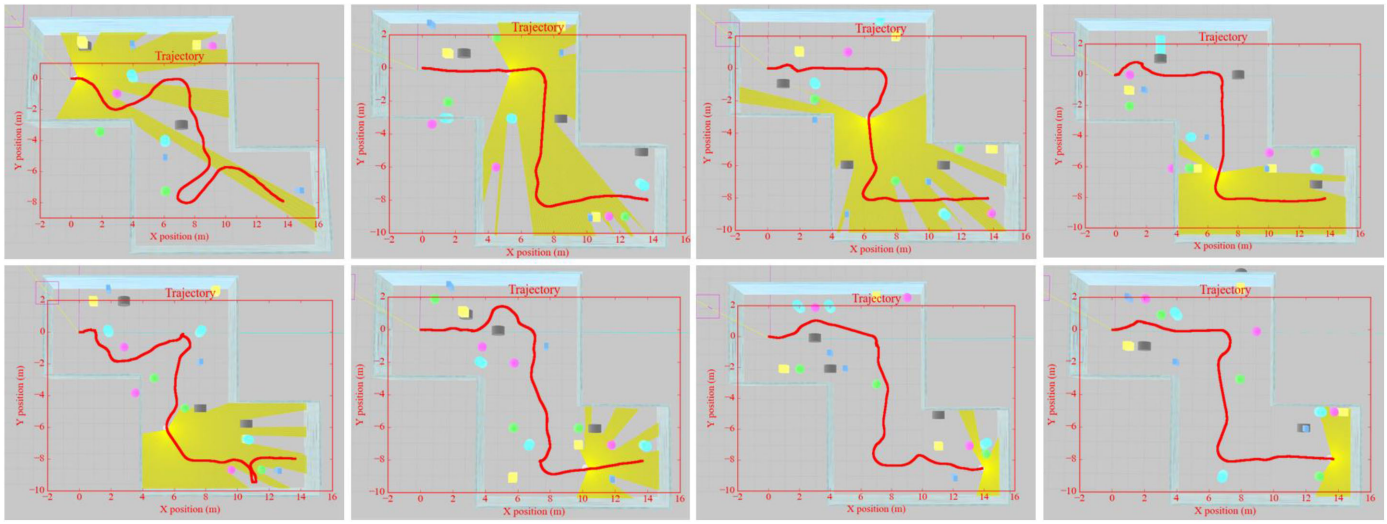


Fig. 5. Eight different cases of robotic avoiding many obstacles spawn in random locations. The thick red curves represent the trajectories of the mobile robot avoiding obstacles deployed by the system randomly; spheres, cubes, and cylinders with different colors denote obstacles; the yellow areas are detected by the robot's lidar.

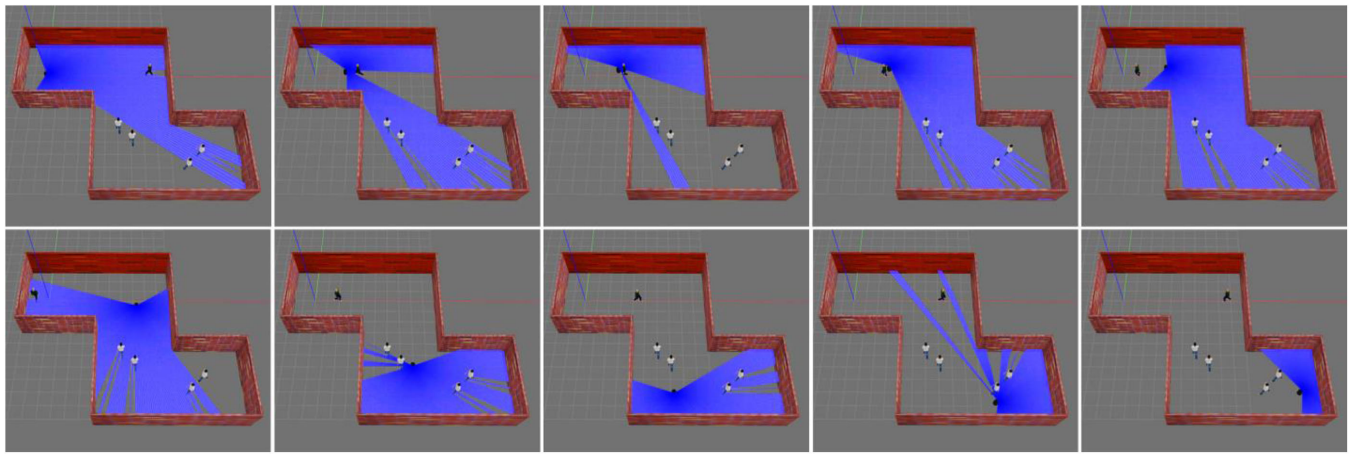


Fig. 6. Snapshots of robotic avoiding the dynamic walking person and static persons.

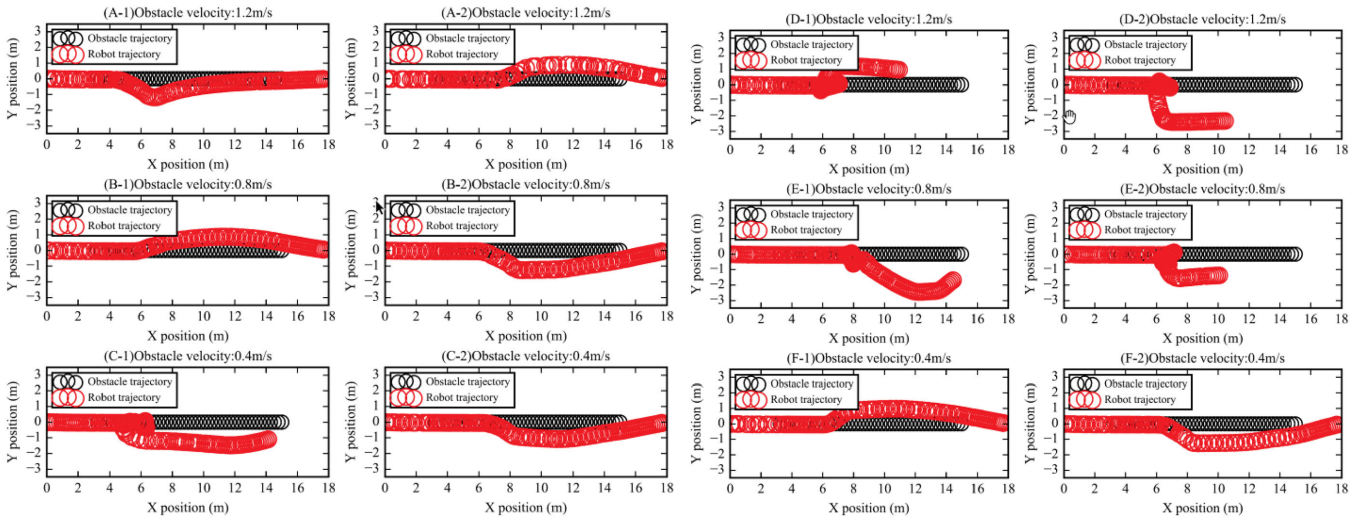


Fig. 7. (a), (b), (c) Motion trajectories with the proposed approach and (d), (e), (f) the standard method. The velocity of the robot is 0.7 m/s while the moving obstacle velocity has three values 0.4, 0.8, and 1.2 m/s.

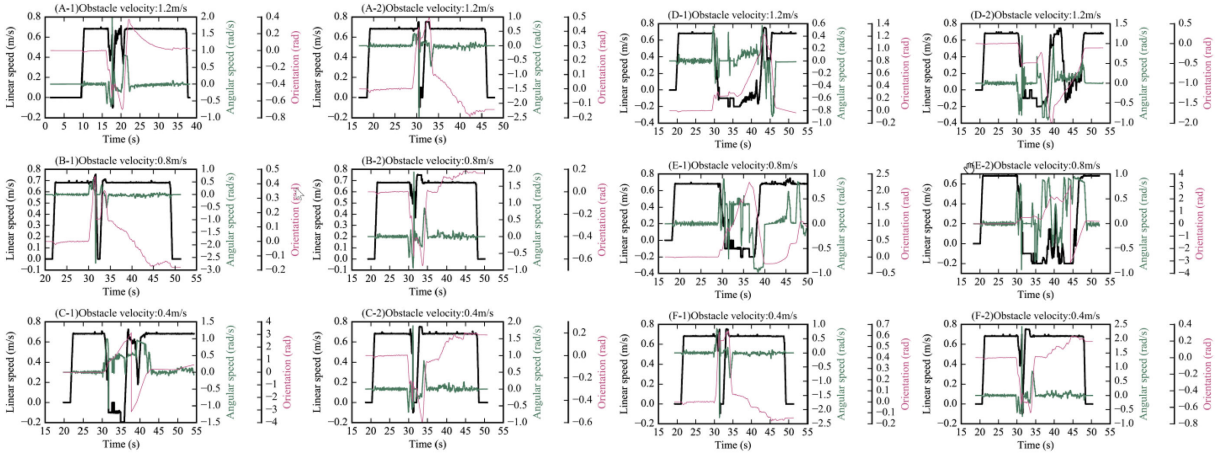


Fig. 8. (a), (b), (c) Linear, angular speeds, and the orientations with the proposed approach and (d), (e), (f) the standard method for the cases of Fig. 7.

smooth for the proposed avoidance system. With the linear speed of the obstacle increasing, the velocity of the mobile robot generally maintains the original value for successful avoidances. It indicates that the proposed method is robust with the speed of the obstacle lower than 1.2m/s. Moreover, the angular velocity for the successful case is relatively smooth. In Fig. 7(c-1)–(e-2), the robot and the obstacle have some collisions when they meet each other. The curves as shown in Fig. 8(a)–(c) have few sharp points at all the time. However, the curves as shown in Fig. 8(d)–(f) have a lot of sharp points. In terms of robot's orientations, the robot performs many turning actions if the robot and the obstacle have a collision.

The results of the two systems are generated by varying the speed of the moving obstacle from 0.4 to 1.2 m/s. The proposed avoidance strategy realizes 91.12% successful rate of avoiding obstacles. The proposed method outperforms the baseline system by a significant margin (40.5%). This is somewhat expected since the baseline avoidance method always cost much computational resource to recognize the obstacles and is not working in real time. The proposed method allocates each track's frame of reference to be attached rigidly to the obstacle. Dynamic obstacles are approximated to cylinders and handled differently to static ones. As a result, as for the dynamic avoidance, the effectiveness of the proposed method is better than that of the baseline avoiding strategy in ROS. This illustrates that the proposed avoidance strategy works very well in such environments.

B. Real Experiments

Finally, we implement the proposed avoidance system on a mobile robot called Turtlebot-2 equipped with a 2D laser scanner. Given a goal and a known starting position, the robot tends to navigate through obstacles. A chair pushed by a person is moving forward as one dynamic obstacle. Fig. 9 mainly shows the spatial evolution of detected and tracked obstacles in the human-robot involved scenario over a period of time. The robot first moves along the global path and only detects static obstacles. After a few seconds, a chair pushed by a person appears in the view of 2D lidar and is heading to the robot. The tracking algorithm based on KF reasonably estimates the obstacle's state by the iterations with measurements. The person continues moving toward the robot at around 0.5 m/s. The robot changes its direction to another side and then passes through this chair successfully.

Fig. 9. Snapshots of dynamic avoidance.

C. Discussions

The baseline (TEB local planner + cost-map converter) which does directly represent obstacles as geometric primitives including points, lines, polygons is widely used for robotic navigation. However, once a large number of geometric primitives

are involved, the baseline algorithm is generally considered of higher computational demand since huge parameters need to be addressed. Allowing for the limited computation power and real-time requirement, such a strategy is not available to handle fast-evolving scenarios where a mobile robot and moving obstacles "co-exist." In contrast, the proposed approach can realize a better avoidance performance. The proposed method processes the laser-point cloud for obtaining line segments and circles as consistent and meaningful representations of detected obstacles, which enormously reduces the computational resource. The corresponding reasons are twofold. First, in the geometric complexity, the lines and circles used for modeling the detected obstacles are just represented by four parameters and three parameters, respectively. However, the compared baseline usually considers a polygon with at least six parameters as an obstacle. Second, in quantitative terms of geometric primitives, a line segment with two points and a circle with one three parameters replace a lot of points, polygons to represent obstacles. Moreover, the laser scan is first processed with the segmenting, merging, and classifying algorithm described above, which significantly reduces the parameters to be solved. The proposed hierarchical data association integrated with an independent KF can improve the obstacle tracking accuracy by greedily assigning each observed geometric model. Integrating TEB local planner into the proposed detection-tracking algorithm, we enable the robot to realize dynamic avoidances. However, there is a challenge for the proposed avoidance perception strategy in understanding critical semantic information from a sliced sample of the world captured by a 2D scanner.

The dynamic avoidance is very complicated. The good dynamic avoidance system does not only require a robust perception system based on visual or laser sensors but also needs suitable robotic control, localization, local, and global path planners. Thus, a robustly dynamic avoidance requires all the subsystems have good performances.

To evaluate the dynamic avoidance performance of the proposed strategy, we set relatively simple real experiments in cluttered real environments. Indeed, in practical scenarios, the dynamic avoidance strategy costs much computational resources, and thus, the performance is also limited by the computing power of a personal computer (PC), especially in cluttered surrounding environments. To solve this issue, we use a local PC to process the algorithm and send the commands to the robot by wireless; however, with a long distance between the local PC and the robot, the communication leads to a delay as well. It is well known that the ROS structure cannot be in real time, which causes that we have to use a highly efficient perception system to relieve the communication delay in the ROS structure.

Moreover, there are two potentially complex dynamic avoidance experiments. In particular, a person walks through the line of the robotic heading direction, most cases are that the robot suddenly stops and then continues moving without changing the direction after the person goes away. If the person stops before the robot, the robot stops and then turns to continue moving. But actually, these two examples cannot show the performance of dynamic avoidance because, generally, the robot does not need to configure the dynamic avoidance strategy and still can stop before an obstacle and update to a new trajectory. The spirits behind the set experiment with both a person and a robot face-to-face walking in the same line are twofold. First, if the robot cannot precisely track the dynamic person to obtain the position using the proposed perception algorithm, the robot would stop or have a collision with the person. Equipped with the proposed dynamic avoidance strategy, the robot keeps moving to the target point while tracking by updating the paths in real time. Second, due to face-to-face moving in the same line, the robot must turn a more angle to avoid the dynamic person. Otherwise, the robot just needs to rotate a small angle to avoid the obstacle when they are in different lines, in this case, it cannot illustrate an obvious avoidance performance.

V. Conclusion

In this article, we presented a novel avoiding strategy combined with the TEB local planner for realizing dynamic obstacle avoidance. We performed simulations and real experiments to demonstrate the capabilities of the proposed avoidance system on the detection and tracking of obstacles in real time. Also, these experiments illustrated that the proposed system is robust and responsive to clutter environments. In the future, we will combine the 3D perception solution to recognize the obstacles represented directly by full 3D models to improve the capability of the robot navigation.

References

[1] A. Asvadi, C. Premebida, P. Peixoto, and U. Nunes, "3D Lidar-based static and moving obstacle detection in driving environments: An approach based on voxels and multi-region ground planes," *Robot. Auton. Syst.*, vol. 83, pp. 299–311, 2016.

[2] J. Park and Y. Kim, "Collision avoidance for quadrotor using stereo vision depth maps," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 51, no. 4, pp. 3226–3241, Oct. 2015.

[3] N. Gageik, P. Benz, and S. Montenegro, "Obstacle detection and collision avoidance for a UAV with complementary low-cost sensors," *IEEE Access*, vol. 3, pp. 599–609, 2015.

[4] H. Dong, E. Asadi, G. Sun, D. K. Prasad, and I.-M. Chen, "Real-time robotic manipulation of cylindrical objects in dynamic scenarios through elliptic shape primitives," *IEEE Trans. Robot.*, vol. 35, no. 1, pp. 95–113, Feb. 2019.

[5] H. Dong, D. K. Prasad, and I.-M. Chen, "Object pose estimation via pruned hough forest with combined split schemes for robotic grasp," *IEEE Trans. Automat. Sci. Eng.*, early access, Sep. 17, 2020 doi: 10.1109/TASE.2020.3021119.

[6] H. Dong, D. K. Prasad, Q. Yuan, J. Zhou, E. Asadi, and I.-M. Chen, "Efficient pose estimation from single RGB-D image via hough forest with auto-context," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 7201–7206.

[7] H. Dong, G. Sun, W.-C. Pang, E. Asadi, D. K. Prasad, and I.-M. Chen, "Fast ellipse detection via gradient information for robotic manipulation of cylindrical objects," *IEEE Robot. Automat. Lett.*, vol. 3, no. 4, pp. 2754–2761, Oct. 2018.

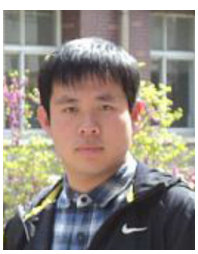
[8] C. Rösmann, W. Feiten, T. Wösch, F. Hoffmann, and T. Bertram, "Efficient trajectory optimization using a sparse model," in *Proc. IEEE Eur. Conf. Mobile Robots*, 2013, pp. 138–143.

[9] C. Rösmann, W. Feiten, T. Wösch, F. Hoffmann, and T. Bertram, "Trajectory modification considering dynamic constraints of autonomous robots," in *Proc. 7th German Conf. Robot.*, 2012, pp. 1–6.

[10] C. Rösmann, F. Hoffmann, and T. Bertram, "Online trajectory planning in ROS under kinodynamic constraints with timed-elastic-bands," in *Robot Operating System (ROS)*. New York, NY, USA: Springer, 2017, pp. 231–261.

[11] M. Quigley *et al.*, "ROS: An open-source Robot Operating System," in *Proc. ICRA Workshop Open Source Softw.*, vol. 3, no. 3/2, p. 5, 2009.

[12] C.-L. Hwang and H.-H. Huang, "Experimental validation of a car-like automated guided vehicle with trajectory tracking, obstacle avoidance, and target approach," in *Proc. IEEE 43rd Annu. Conf. IEEE Ind. Electron. Soc.*, 2017, pp. 2858–2863.
[13] M. Keller, F. Hoffmann, C. Hass, T. Bertram, and A. Seewald, "Planning of optimal collision avoidance trajectories with timed elastic bands," *IFAC Proc. Vol.*, vol. 47, no. 3, pp. 9822–9827, 2014.
[14] M. Thuy and F. P. Leon, "Non-linear, shape independent object tracking based on 2D Lidar data," in *Proc. IEEE Intell. Veh. Symp.*, 2009, pp. 532–537.
[15] T. Weiss, B. Schiele, and K. Dietmayer, "Robust driving path detection in urban and highway scenarios using a laser scanner and online occupancy grids," in *Proc. IEEE Intell. Veh. Symp.*, 2007, pp. 184–189.
[16] M. S. Darms, P. E. Rybski, C. Baker, and C. Urmson, "Obstacle detection and tracking for the urban challenge," *IEEE Trans. Intell. Transp. Syst.*, vol. 10, no. 3, pp. 475–485, Sep. 2009.
[17] C.-C. Wang, C. Thorpe, S. Thrun, M. Hebert, and H. Durrant-Whyte, "Simultaneous localization, mapping and moving object tracking," *Int. J. Robot. Res.*, vol. 26, no. 9, pp. 889–916, 2007.
[18] T. Mori, T. Sato, H. Noguchi, M. Shimosaka, R. Fukui, and T. Sato, "Moving objects detection and classification based on trajectories of LRF scan data on a grid map," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2010, pp. 2606–2611.
[19] M. Przybyła, "Detection and tracking of 2d geometric obstacles from LRF data," in *Proc. IEEE 11th Int. Workshop Robot Motion Control*, 2017, pp. 135–141.
[20] C. Mertz *et al.*, "Moving object detection with laser scanners," *J. Field Robot.*, vol. 30, no. 1, pp. 17–43, 2013.
[21] M. Dekan, D. František, B. Andrej, R. Jozef, R. Dávid, and M. Josip, "Moving obstacles detection based on laser range finder measurements," *Int. J. Adv. Rob. Syst.*, vol. 15, no. 1, pp. 17298814–17748132, 2018.
[22] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Autonomous Robot Vehicles*. New York, NY, USA: Springer, 1986, pp. 396–404.
[23] J. Borenstein and Y. Koren, "Real-time obstacle avoidance for fast mobile robots," *IEEE Trans. Syst., Man, Cybern.*, vol. 19, no. 5, pp. 1179–1187, Sep./Oct. 1989.
[24] J. Borenstein and Y. Koren, "The vector field histogram-fast obstacle avoidance for mobile robots," *IEEE Trans. Robot. Automat.*, vol. 7, no. 3, pp. 278–288, Jun. 1991.
[25] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robot. Automat. Mag.*, vol. 4, no. 1, pp. 23–33, Mar. 1997.
[26] C. Rösmann, F. Hoffmann, and T. Bertram, "Integrated online trajectory planning and optimization in distinctive topologies," *Robot. Auton. Syst.*, vol. 88, pp. 142–153, 2017.
[27] H. Dong, I.-M. Chen, and D. K. Prasad, "Robust ellipse detection via arc segmentation and classification," in *Proc. IEEE Int. Conf. Image Process.*, 2017, pp. 66–70.
[28] G. Bishop and G. Welch, "An introduction to the kalman filter," *Proc. SIGGRAPH, Course*, vol. 8, no. 27599–23175, p. 41, 2001.
[29] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (SLAM): Part II," *IEEE Robot. Automat. Mag.*, vol. 13, no. 3, pp. 108–117, Sep. 2006.
[30] Accessed: Nov. 21, 2020. [Online]. Available: https://www.youtube.com/watch?v=A2Ad8rNQqYw



Huixu Dong (Member, IEEE) received the B.Sc. degree in mechatronics engineering from the Harbin Institute of Technology, Harbin, China, in 2013, and the Ph.D. degree from the Robotics Research Centre of Nanyang Technological University, Singapore, 2018.

He was a Postdoctoral Fellow with the Robotics Institute of Carnegie Mellon University and is currently is a Research Fellow with the Bio-Robotics Lab of National University of Singapore. His current research interests include robotic perception and grasp in unstructured environments, robot-oriented image processing, computer vision and robot-oriented artificial intelligence, the navigation of mobile robot, and optimal design of robotic gripper.

Ching-Yen Weng received the B.S. degree in electrical engineering from National Tsing Hua University, Taiwan, in 2013, and the M.S. degree in mechanics from Tsinghua University, Beijing, China, in 2016. He is currently working toward the Ph.D. degree in robotics from Nanyang Technological University, Singapore.

His research interests include factory automation, system performance analysis, robotic manipulation, and human-robot collaboration.

Chuangqiang Guo received the B.S. and M.S. degrees from the Harbin University of Science and Technology, Harbin, China, in 2005 and 2008, respectively, and Ph.D. degree from Harbin Institute of Technology, Harbin, China, in 2012, all in mechanical engineering.

He is currently an Associate Research Fellow of mechanical engineering with the State Key Laboratory of Robotics and System, Harbin Institute of Technology. His current research interests include the design and control technologies of AC motor drive and robotic system.

Haoyong Yu (Senior Member, IEEE) received the B.S. and M.S. degrees from Shanghai Jiao Tong University, Shanghai, China, in 1988 and 1991, respectively, and the Ph.D. degree from the Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, in 2002, all in mechanical engineering.

He was a Principal Member of Technical Staff with DSO National Laboratories, Singapore, until 2010. Currently, he is an Associate Professor with the Advanced Robotic Centre and the Department of Biomedical Engineering at the National University of Singapore. His current research interests include biomedical robotics and devices, rehabilitation engineering and assistive technology, biologically inspired robotics, intelligent control, and machine learning.

I-Ming Chen (Fellow IEEE) received the B.S. degree in mechatronics from National Taiwan University, Taipei, Taiwan, in 1986, and M.S. and Ph.D. degrees in robotics from California Institute of Technology, Pasadena, CA, USA, in 1989 and 1994, respectively.

He is currently a Full Professor with the School of Mechanical and Aerospace Engineering, former directions of Robotics Research Centre and Intelligent System Centre in Nanyang Technological University, Singapore. He also acts as the Deputy Program Manager of A∗STAR SERC Industrial Robotics Program to coordinate project and activities under this multi-institutional program involving NTU, NUS, SIMTech, A∗STAR I2R, and SUTD. His research interests include different topics in robotics, such as mechanism, actuator, human-robot interaction, perception and grasp, and industrial automation.

Dr. Chen is a Fellow of the ASME, Fellow of Academy of Engineering(Singapore), General Chairman of 2017 IEEE International Conference on Robotics and Automation. He is also a Senior Editor of IEEE TRANSACTION ON ROBOTICS and the Editor-in-Chief of IEEE TRANSACTIONS ON MECHATRONICS.