Clutter-resilient autonomous mobile robot navigation with computationally efficient free-space features

Rômulo T. Rodrigues[1], Nikolaos Tsiogkas[1] Nico Huebel[2], and Herman Bruyninckx[1,3]

[1] Department of Mechanical Engineering, KU Leuven, Leuven, Belgium, FlandersMake Core Lab ROB, Belgium,
{romulo.rodrigues,nikolaos.tsiogkas,herman.bruyninckx}@kuleuven.be
[2] University of Antwerp, Antwerp, Belgium, FlandersMake Core Lab Cosys, Belgium
nico.huebel@uantwerpen.be
[3] Department of Mechanical Engineering, TU of Eindhoven, Eindhoven, the Netherlands

Abstract. This paper proposes free-space motion tubes, a motion primitive for the local navigation of mobile robots equipped with range sensors. The geometry of a candidate motion tube captures the free-space required such that the robot may execute a maneuver without colliding with obstacles. Computational efficiency is achieved by selecting meaningful samples of the tube and evaluating them at run-time in the sensor space. Increasing the sensor resolution or the number of obstacles do not have any impact in the computational cost. Experimental results with a mobile platform show that free-space motion tubes are well-suited for navigating in cluttered environments and narrow passages.

Keywords: free space, local navigation, motion tube, obstacle avoidance

1 Introduction

Local navigation is a basic skill of autonomous mobile robots. It is the ability to move in their environment, which can be possibly dynamic and only partially known, while avoiding collisions. Two versatile (and intrinsically very similar) families of methods for local navigation, the Dynamic Window Approach (DWA) [1] and Model Predictive Control (MPC) [2], formulate the problem as finding the trajectory that within a limited time horizon

1. respects the kinodynamics constraints of the vehicle;
2. does not collide with obstacles in the vicinity of the robot;
3. minimizes a task-centric cost function.

The Benchmark Autonomous Robot Navigation 2022 (BARN Challenge) [3] invited researchers all over the world to design a navigation method well-suited for highly constrained environments. With only one team being able to accomplish the task at low speed (and considerable parameter tuning for the specific scene), the competition has shown that autonomous robot navigation in cluttered environments is an open problem.

Most of the approaches found in the literature are relying on having a map to check for trajectory collisions. This involves transforming *all* sensor readings to the map space and doing the collision check there. This can have a large computational cost, reducing the applicability of many of the methods to embedded systems.

Dynamic Window Approach [1] simulates arc trajectories, projects the kinodynamically feasible ones on the map space, and based on a cost function selects the optimal one. In [4], the authors proposed elastic band (e-band), an efficient method for local path planning and control. The initial path is the one given by a high-level path planner that may contain obstacles not present in the global map. It is then accordingly deformed to avoid any local obstacles, and that can be smoothly followed by a robot. A recent work comparing a DWA based local planner with an elastic band Reeds-Shepp curve based planner is presented in [5]. The work focuses in narrow passages for agricultural robots. From the results the e-band based planner succeeded in 18 out of 20 trials while DWA only in 6 out of 20.

Some methods try to overcome narrow passages and cluttered spaces by minimising the effects of noise. For example, Vector Field Histograms (VFH) [6] (Vand its variation) builds a one-dimensional polar cell histogram from an occupancy-grid map. High values on the histograms are related to a higher likelihood of collision. Thus, local minimum (valleys) in the histogram are candidate directions that may provide a path through a narrow passage. [7] is using "semantic grid" and topological map for global navigation. To be able to navigate through narrow spaces, extra points are generated before and after the narrow passage. This is done in hope that the quality of the map will be improved and allow a trajectory to be found and followed. The work of [8] neither requires a map nor global localization. A passage detection method, based on the discontinuity of measurements, is used to find potential spots to drive through. These passage points are tracked using an extended Kalman filter, which aims at coping with sensor noise. Finally a smooth control policy is used. Given the reliance on continuously detecting and tracking points of interest, the computational complexity of the method can be high. Unfortunately, the authors do not provide any metrics on that topic.

An efficient approach that can handle dynamic obstacles is presented in [9]. This approach relies on MPC combined with non-euclidean special rotation groups that improve the performance of the motion of non-holonomic vehicles. [10] uses parametrized splines in an optimization-based approach to compute collision-free trajectories in uncertain environments with moving obstacles. The spline properties are exploited to solve the problem fast enough for an online application in a receding horizon control. In [11] a passivity-based MPC approach is presented. It uses the concept of energy shaping with a navigation cost function to generate feasible trajectories. Different local optimization algorithms are compared. [12] introduces a harmonic potential function without local minima for global path planning and then uses a model predictive control (MPC) for the local planning taking into account robot contraints. [13] shows better performance in simulation compared to potential field methods. It uses tightening polytopic constraints in an MPC design for robot navigation in order to reach a goal while staying away from the boundary of the defined polytopic working zone.

In [14], authors propose an efficient collision detection module for evaluating the transversability of edges in a Probabilistic Roadmap (PRM). In short, a PRM is a graph where nodes are randomly sampled from the configuration space and edges connecting them correspond to collision-free paths. The authors in [14] represent the edges

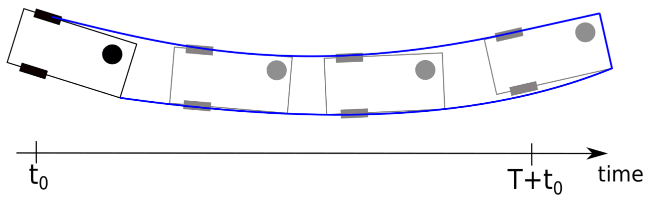


Fig. 1: An example of a free-space motion tube (painted in solid blue lines). The left-most vehicle corresponds to the current position of the platform.

of the PRM by the swept volume of the robot. For that, the space around the robot is discretized into depth pixels. Efficiency is achieved by designing a collision detection circuit, which is embbeded in a FPGA. The results are promising, with very low computational time. In contrast to our method, it requires building a map and a PRM for each local environment, and an available path may not be found due to the probabilistic nature of the method.

In addition to the above methods some learning based approaches have appeared recently in the literature. The work of [15] presents a Deep Reinforcement Learning method for passing through narrow gaps. The authors report a success rate of 93% in simulation and 73% in the real-world, while other methods developed by the same group were failing. In [16] a dynamic movement primitives-based obstacle avoidance approach is presented. Multiple cost functions are analysed in an obstacle avoidance task.

To address the computational complexity that is introduced by the above methods, this paper presents the primitive of a *free space motion tube* for local navigation, as seen in Fig. 1. The geometrical parameters representing such a tube form the configuration space of an MPC method, where each sample of these parameters corresponds to one particular tube. Therefore, the proposed method can also be seen as an MPC method, which generates a set of motion tubes and efficiently verifies their feasibility directly in sensor space. Then, among the feasible tubes, selection is based on a specified criterion/cost function as in [1].

This approach provides significant computational efficiencies, as it maps the control output to a small sample of possible motions around the trajectory that can be expected from the current control signal, and their collision-freeness can be checked directly in sensor space. The presented approach improves computational efficiency in *three complementary ways*:

– it avoids the transformation of *all* sensor readings to map space and doing the collision checking there, as only the sensor readings closer than the motion template are to be taken into account.
– samples of the tube template can be associated *offline* to a sensor beam index. Since the samples are always computed with respect to the same local frame of the robot, there is no need to recompute them unless the sensor or the robot configuration changes at runtime.
  The cost to evaluate a free space template, performed online, for each scan reading, is $\mathcal{O}(n)$, where n is the number of samples.

– the tube represents guaranteed free space available to the robot some seconds in the future, hence the motion generation computations can be done at much lower rate than the motion control and/or sensor reading rates.

The computational gains described above have a "cost" too, namely their dependence on a set of "magic numbers" that have to be set "right" for each particular application context.

The rest of the paper is organised as follows: In Section 2 the problem formulation is presented, while Section 3 presents the proposed method. Section 4 gives an overview of the experimental setup, along with detailed results. Finally, Section 5 concludes the paper, discussing the results, and proposes fruitful future directions for further research.

2 Problem formulation

A differential drive platform with a LiDAR sensor has coordinate frames as in Fig. 2:

- $\{B\}$ body-fixed frame attached to the mid-point of the wheel axle of the platform. The x-axis of $\{B\}$ is aligned with the forward direction of the platform and its y-axis points towards the left side of the vehicle.
- $\{L\}$ LiDAR-fixed frame attached to the origin of the range sensor. The x-axis of $\{L\}$ is aligned with the central beam of the sensor.
- $\{R\}$ reference frame when evaluating the motion tube. It is inertial during the time interval $[t_0, t_0 + T]$, and at time t_0 it coincides with $\{B\}$.

The orientation of the platform with respect to $\{R\}$ at time t is denoted as $\psi(t)$. The control inputs of the platform are its forward speed (v) and angular rate (w), both described in $\{B\}$. The 2D LiDAR onboard the platform provides a set of m measurements $\{(\alpha_i, r_i)\}_{i=1}^{m}$, where r_i is the range distance to an object w.r.t. the origin of $\{L\}$ measured by the i-th ray of the sensor and α_i is the angle of the i-th ray w.r.t. the x-axis of $\{L\}$.

Problem 1. Consider a differential drive platform with specified dimensions equipped with a 2D range sensor. Compute the set of control inputs (v, w) that allows for a collision-free trajectory from time t_0 to $t_0 + T$.

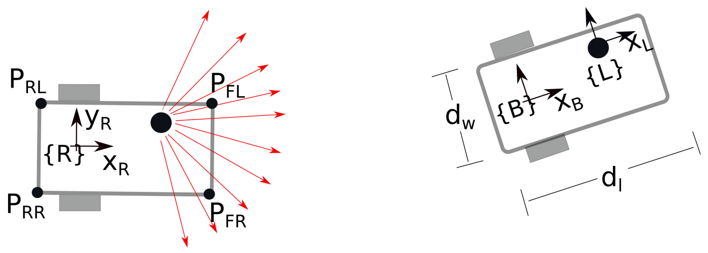


Fig. 2: Platform, coordinate frames, and critical points.

Notation

For the rest of the paper the following notation is used: scalars are written in lower case letter and points are typed in upper case letter, e.g, $P_i = [x_i, y_i]$. Vectors are typed in lower case bold letter, e.g, $\mathbf{p}_{ij} = P_i - Pj$.

3 Proposed solution

The proposed method relies on templates that capture the trajectory of the vehicle when excited by a control input pair (v, w). Each template is a combination of continuous geometric shapes, specifically, circular arcs and line segments, for the simplest possible, i.e., constant, motion specification. Evaluating the occupancy of the trajectory is performed by "sweeping" the area inside a template with a range sensor. Brute force sweeping, that is, inferring the occupancy of the trajectory for each beam of the sensor that intersects with the template is typically inefficient due to the large amount of data collected by modern sensors. Thus, the proposed method considers only a subset of the measurements which corresponds to "meaningful" samples of the template. That is, these samples are as far apart as possible, while still guaranteeing detection of obstacles with dimensions greater than a pre-specified value.

3.1 Assumptions and knowledge

Our method relies on some assumptions and domain specific knowledge. Assumptions are simplifications of real-life, which were used for designing the algorithm. They have to be monitored such that violating an assumption triggers a proper behaviour, e.g., correcting the trajectory or halting the platform. In contrast, domain specific knowledge holds in the situation in which the robot operates. It is provided by the system designer or acquired online by the robot.

1. (assumption) during the time interval $[t_0, t_0 + T]$ the vehicle moves at constant forward speed and angular rate.
2. (domain knowledge) the structure of the platform is within the area delimited by the closed polygonal chain $\mathcal{P} = \{P_{rl}, P_{rr}, P_{fl}, P_{fl}\}$, which is defined by the vertices P_{rl} (rear-left), P_{rr} (rear-right), P_{fl} (front-left), P_{fr} (front-right). The line segments $\overline{P_{rl}P_{rr}}$ and $\overline{P_{fl}P_{fr}}$ are parallel to the axle of the vehicle, while the line segments $\overline{P_{rl}P_{fl}}$ and $\overline{P_{rr}P_{fr}}$ are perpendicular to it.
3. (domain knowledge) the range sensor is within the polygon $\mathcal{P}$.
4. (domain knowledge) the smallest dimension of an object (width and length) is greater or equal to d_{min}.
5. (domain knowledge) obstacles to be avoided area detectable by the 2D slice of the range sensor.
6. (domain knowledge) Position ($P_L = [x_L, y_L]^T$) and orientation of the sensor in the platform w.r.t. $\{R\}$ described in $\{R\}$ is known.

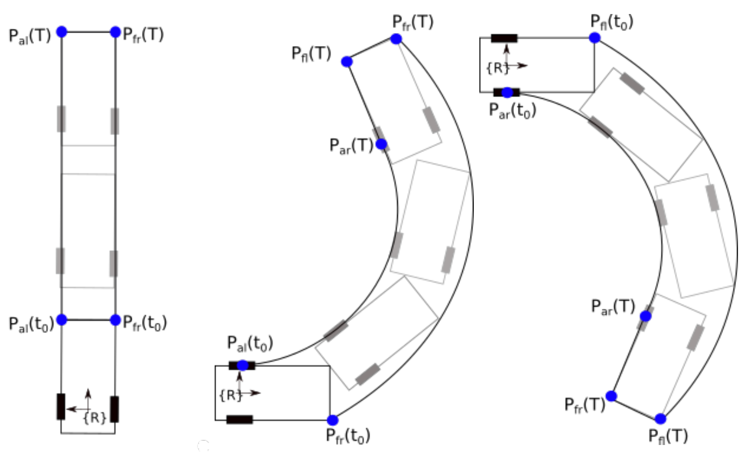
In addition to $P_{rl}, P_{rr}, P_{fl}, P_{fl}$, we define the points P_{ar} (axle-right) and P_{al} (axle-left), which belong to the polygon $\mathcal{P}$ and lie in the wheels axle of the vehicle, as shown in Fig. 3b and Fig. 3c. For the sake of simplicity, we will also add the following domain knowledege: for a circular arc trajectory, either P_{fl} or P_{fr} (front points) experiences the smallest curvature, i.e.,$\|P_{fl} - P_{al}\| > \|P_{rl} - P_{al}\|$ and $\|P_{fr} - P_{ar}\| > \|P_{rr} - P_{ar}\|$. If that is not the case, in Sec. 3.4 the trajectories of the rear points would also have to be considered .

3.2 Designing motion tubes

The proposed template is a geometric shape that delimits the area that must be available in order for the vehicle to perform a specific maneuver without colliding with an object in the scene. For example, Fig. 3 shows the trajectory of the vehicle when driven by control inputs $v > 0$ and $w = 0$ (Fig. 3a), $w > 0$ (Fig. 3b) or $w < 0$ (Fig. 3c).

The simplest controller for a mobile platform follows the Reeds-Shepp method [17], so that the geometric shape of the motion tube is a composition of circular arcs (due to constant steering direction) and line segments spanned by the critical points $\mathcal{P}$, that is, the vertices of the convex hull of the robot platform. The second simplest control approach uses clothoid curves, with linearly changing *steering* direction. The particular selection of critical points depends on the direction of rotation, which leads to the definition of three free space templates: $\mathcal{T}_{MS}$ (MS: move straight), $\mathcal{T}_{SL}$ (SL: steer left), and $\mathcal{T}_{SR}$ (SR: steer right). The geometry of each template is as follows:

- $\mathcal{T}_{MS}$, free space template *move straight* ($w = 0$):
  - line segments $\overline{P_{fr}(t_0), P_{fr}(t_0+T)}$, $\overline{P_{fr}(t_0+T)P_{fl}(t_0+T)}$, and $\overline{P_{fl}(t_0+T)P_{fl}(t_0)}$
- $\mathcal{T}_{SL}$, free space template *steer left* ($w > 0$):
  - circular arcs $\int_{t_0}^{t_0+T} P_{fr}(\tau)d\tau$ and $\int_{t_0}^{t_0+T} P_{al}(\tau)d\tau$.



(a) $\mathcal{T}_{MS}$ (b) $\mathcal{T}_{SL}$ (c) $\mathcal{T}_{SR}$

Fig. 3: Free space templates generated by assuming constant linear/angular velocity : move straight (a), steer left (b), and steer right (c).

    - line segments $\overline{P_{fr}(t_0+T)P_{fl}(t_0+T)}$ and $\overline{P_{fl}(t_0+T)P_{al}(t_0+T)}$.
- $\mathcal{T}_{SR}$, free space template *steer right* ($w<0$):
    - circular arcs $\int_{t_0}^{t_0+T} P_{fl}(\tau)d\tau$ and $\int_{t_0}^{t_0+T} P_{ar}(\tau)d\tau$.
    - line segments $\overline{P_{fl}(t_0+T)P_{fr}(t_0+T)}$ and $\overline{P_{fr}(t_0+T)P_{ar}(t_0+T)}$.

3.3 Sweeping to detect collision

For introducing the sweeping process, consider Fig. 4. The goal is to check whether the area surrounded by the template (in dashed blue lines) is free space.

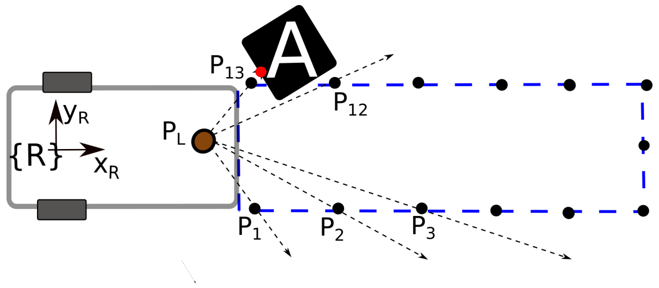


Fig. 4: Sweeping a free space template by selecting samples on the edge of the template (naive approach). Solid circles are samples drawn from the template using the naive sampling policy. A solid red circle represents the intersection between a sensor beam and an obstacle.

First, we show a naive approach with the aid of the following proposition:

Proposition 1. *The largest dimensions of a square object that can be inscribed in a triangle must be smaller than the smallest edge of the triangle.*

Let the area of the template be contained inside the area covered by the union of triangles $\triangle(P_L, P_i, P_{i+1})$, where P_L is the origin of the sensor and $\{P_i = [x_i, y_i]^T\}_{i=1}^n$ is a set of n samples drawn from the edge of template. Therefore, if all the triangles are in free space, so is the area of the template.

For the sake of simplicity, assume that there is a ray of the sensor that points towards P_i, and let the range distance measured by this ray be denoted as $r(P_i)$. If $r(P_i) > \|P_i - P_L\|$ and $r(P_{i+1}) > \|P_{i+1} - P_L\|$, then the triangle $\triangle(P_L, P_i, P_{i+1})$ cannot contain a square object with dimensions greater than $\|P_i - P_{i+1}\|$. This follows from Proposition 1: the smallest edge of $\triangle(P_L, P_i, P_{i+1})$ is smaller or equal to $\|P_i - P_{i+1}\|$. Consequently, the dimensions of an inscribed square is also upper bounded by $\|P_i - P_{i+1}\|$.

By sampling the template at a constant sampling interval d_{min}, one may verify that there is no object greater than d_{min} inside the template. However, as shown in Fig. 4, this process may fail to detect an object greater than d_{min} that *partially* intersects with the template (see 'obstacle A' between P_{12} and P_{13}). The problem is further analyzed in Fig. 5a. The image shows that halving the sampling interval (at the expense of computational effort) does not solve the problem. This is because an object may always fit in-between two samples regardless of how close by they are.

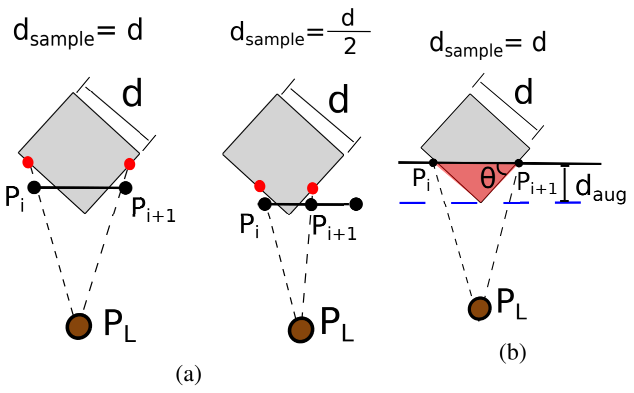


Fig. 5: (a) shows that increasing the sampling rate does not provide guarantees that an object will be detected. (b) illustrates how to overcome the problem by introducing a gap d_{aug} between the template and samples.

The problem is addressed by sampling not the edge of the template itself, but a distance d_{aug} from it. We claim that if the gap between the sample and the template is greater than $d_{sample}/2$, then any object greater than d_{sample} that partially intersects with the inner template is detected.

Proof. Consider an object $\mathcal{O}$ with dimensions greater than d_{sample} and two parallel lines $\mathcal{L}_{inner}$ and $\mathcal{L}_{outer}$, which are d_{aug} apart. Let the length of the intersection of $\mathcal{O}$ and $\mathcal{L}_{outer}$ be denoted as h. If $h < d_{sample}$, then the object may go through two consecutive samples of $\mathcal{L}_{inner}$ that are d_{sample} apart. In that case, choose d_{aug} such that the intersection between the object and the two lines is a triangle of base h and height d_{aug} (red area in Fig. 5b). We are interested in solving the problem

$$d^*_{aug} = \underset{h \in [0, d_{sample})}{\arg\max} \; d_{aug} \tag{1}$$

which corresponds to the minimum fixed distance that the two lines must be apart such the intersection of $\mathcal{O}$ and $\mathcal{L}_{inner}$ is either empty or a point (worst case). For a solution, first compute the area of the triangle with base h and height d_{aug}:

$$\frac{1}{2} h d_{aug} \text{ or } \frac{1}{2} h^2 \cos(\theta) \sin(\theta), \tag{2}$$

where θ is one of the acute angles of the triangle. Equating the two formulae, yields

$$d_{aug} = h \frac{\sin(2\theta)}{2}. \tag{3}$$

It is trivial to see that under the constraints of the problem ($h < d_{sample}$), the maximum value of d_{aug} converges to

$$d^*_{aug} = \frac{1}{2} d_{sample}, \tag{4}$$

which is obtained as $h \to d^-_{sample}$ and $\theta = \pi/4$.

The actual sweeping process is shown in Fig. 6. The inner geometric shape in dash blue lines corresponds to a free space template and the black circles are samples P_i drawn at a constant spatial sampling interval d_{sample}. The distance between a sample and the closest point in the template is d_{aug}, except at the front corners when it is zero. The sweeping process result shall be interpreted as

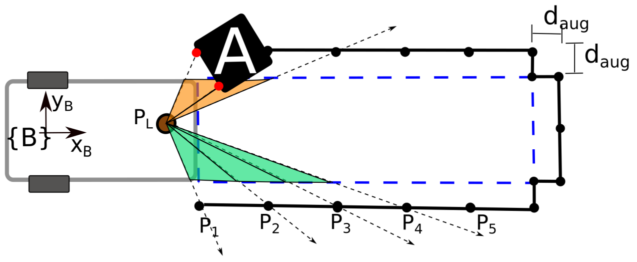


Fig. 6: Proposed sweeping process: samples are drawn at a distance d_{aug} from the template.

- If $d_{aug} \geq \frac{d_{sample}}{2}$, then asserting an area as free space guarantees that there is no obstacle with dimensions greater than d_{sample} inside or partially colliding with the area of the template;
- If $0 \leq d_{aug} < \frac{d_{sample}}{2}$, then asserting an area as free space means that there is no obstacle with dimensions greater than d_{sample} inside the free space template, but there might might be an obstacle with dimensions greater than $2d_{aug}$ partially colliding with the template.

3.4 Sampling

Let $P(t) = [x(t), y(t)]^T$ be the position of a point attached to the platform. With the notation introduced in Section 2, the movement of this point w.r.t to $\{R\}$ can be described as

$$\begin{aligned} x(t) &= x_0 \cos\psi(t) - y_0 \sin\psi(t) + \frac{v}{w}\sin\psi(t) \\ y(t) &= x_0 \sin\psi(t) + y_0 \cos\psi(t) + \frac{v}{w}(1 - \cos\psi(t)) \end{aligned}, \tag{5}$$

where $\psi(t) = (t - t_0)w$ and $P(t_0) = [x_0, y_0]^T$. In particular, if $w = 0$, then one may compute $P(t)$ w.r.t to $\{R\}$ as following:

$$\lim_{w \to 0} P(t) = \begin{cases} x(t) & = x_0 + v(t - t_0) \\ y(t) & = y_0 \end{cases}, \tag{6}$$

For sampling (5) and (6), one has to convert the spatial sampling interval d_{sample} to a sampling time Δt along those trajectories. We first compute r_p, the radius of the trajectory of point P:

$$r_p = \sqrt{x_0^2 + (\frac{v}{w} - y_0)^2}. \tag{7}$$

We want to find the sampling time Δt that corresponds to the spatial sample d_{sample}. From the arc length formula, we have that

$$r_p w \Delta t = d_{sample} \Rightarrow \Delta t = \frac{d_{sample}}{w r_p}. \tag{8}$$

If $w = 0$, then the sampling time is given by

$$\Delta t = \lim_{w \to 0} \frac{d_{sample}}{w r_p} = \frac{d_{sample}}{v} \tag{9}$$

Given the sampling time, one may compute the number of samples n by considering the time horizon T:

$$n = \left\lceil \frac{T}{\Delta t} \right\rceil + 1, \tag{10}$$

and refine the sampling time to ensure that the endpoints are considered:

$$\Delta t = \frac{T}{n - 1}. \tag{11}$$

3.5 Template samples to sensor space

Next, the samples are linked to the sensor space. For that, the sensor configurations are required, specifically, the number of beams (m), the angular resolution ($\Delta\alpha$), the minimum angle (α_1), the minimum and maximum ranges (r_{min} and r_{max}). Given the sensor position and orientation w.r.t. $\{R\}$, one may transform the sample P_i to the sensor coordinate frame, which is denoted as $P_{i,L} = [x_{i,L}, y_{i,L}]^T$. The index i of the beam of the sensor that observes the i-sample of the template can be obtained as:

$$\theta_i = tan^{-1}\left(\frac{y_{i,L}}{x_{i,L}}\right) \tag{12}$$

$$k = \left\lfloor \frac{\theta_i - \alpha_1}{\Delta\alpha} \right\rfloor, \tag{13}$$

where θ_i is the direction of the $i\text{-}th$ sample w.r.t. the sensor and α_k is the direction of the beam actually chosen due to the discrete angular resolution of the sensor. The following verifications needs to be performed for all the samples P_i of a template to ensure that given the sensor configuration, the motion tube can be properly evaluated:

- within valid range: $r_{min} \leq \|P_{i,L}\| \leq r_{max}$.
- within field of view: $\alpha_1 \leq \alpha_k \leq \alpha_m$.
- high enough angular resolution: $\|\theta_{i+1} - \theta_i\| \geq \Delta\alpha$.

Algorithm 1 summarizes the key step for sampling a circular arc template and mapping the samples into sensor beam indices. The points to be considered are the ones shown in Fig. 7, which corresponds to the critical points augmented by the distance d_{aug} in the direction of a possible collision.

Algorithm 1 Samples to sensor beam indices

```
**signature**: $P_1, \ldots, P_n \leftarrow$ sample_circular_arc($\mathbf{p}_0$, $d_{sample}$, $v$, $w$, $T$)
1: Compute sampling rate ($\Delta t$): (8) or (9)
2: Compute number of samples ($n$): (10)
3: Compute refined sampling rate ($\Delta t$): (11)
**for** $i \leftarrow 1, n$ **do**
    4: Compute sample $P_i$: (5) or (6)
    5: Compute direction of the sample $\theta_i$: (12)
    6: Compute the corresponing index of the sensor beam $k$: (13)
**end for**
```

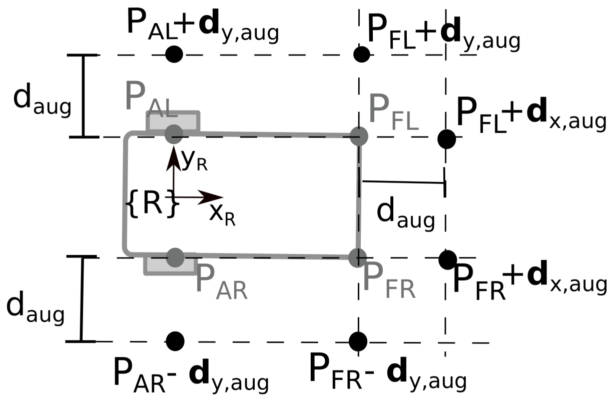


Fig. 7: The critical points for generating the samples of the templates are composed with $\mathbf{d}_{x,aug} = [d_{aug}, 0]^T$ and $\mathbf{d}_{y,aug} = [0, d_{aug}]^T$.

4 Experimental validation

The proposed method was evaluated using the experimental platform shown in Fig. 8. The robot measures 0.4×0.3 m and its main hardware components are a KELO drive

100 with two active wheels, a Hokuyo-URG04 LiDAR sensor, an ODROID-XU4 on-board computer, and a LiPO battery pack. The LiDAR provides 720 measurements (240° field of view) at 10 Hz. The computations are entirely performed by the onboard computer.

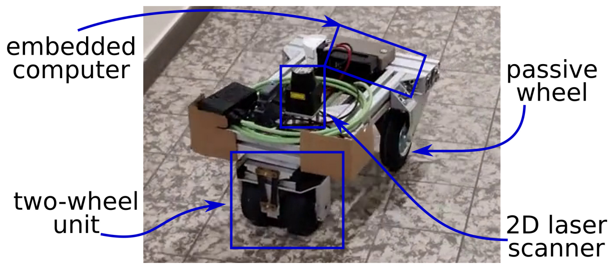


Fig. 8: Mobile platform employed for experimental validation

The tests were performed in a cluttered office environment, shown in Fig. 9a. The goal is to explore the office, going through narrow passages and avoiding obstacles such as cabinets, boxes, and table legs. Local navigation is addressed using the free-space motion tubes. For that, four time horizons were considered, specifically, $T = 1, 2, 3, 4$ s. To ensure that the platform is able to stop before a collision, the nominal velocity of the primitives are proportional to the time horizon, with $v = 0.4$ m/s for $T = 4$ s, linearly decreasing to $v = 0.25$ m/s for $T = 1$ s. For each time horizon and nominal velocity pair, the angular rate was selected by equally sampling the interval $[-\pi/2, \pi/2]$ rad/s. Therefore, for the same angular rate, the lower the time horizon (and forward speed) the higher is the curvature. The motion tubes were generated at starting time using Algorithm 1, and then evaluated online using the latest range scan measurements.

The results are illustrated in Fig. 10, and a video of the experiment is available as a multimedia material. The map was generated offline for visualization purposes using the algorithm in [18] Since this paper does not aim at global navigation, we ensured that there is a valid path for exploring the office. However, in a full application, a global

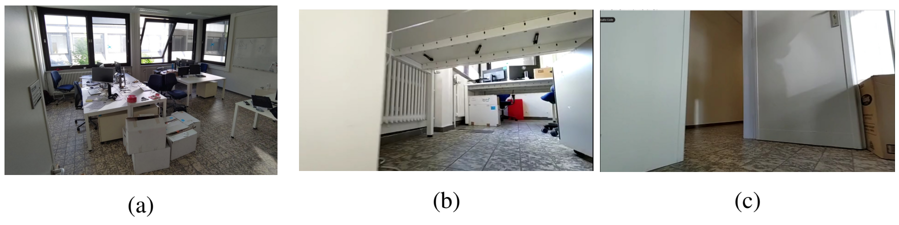


Fig. 9: Office room where experiments were performed (a), cluttered workstation inside the room (b), and door that connects the room to the corridor of the building (c). (b) and (c) show the point of the view of the robot during the experiments.

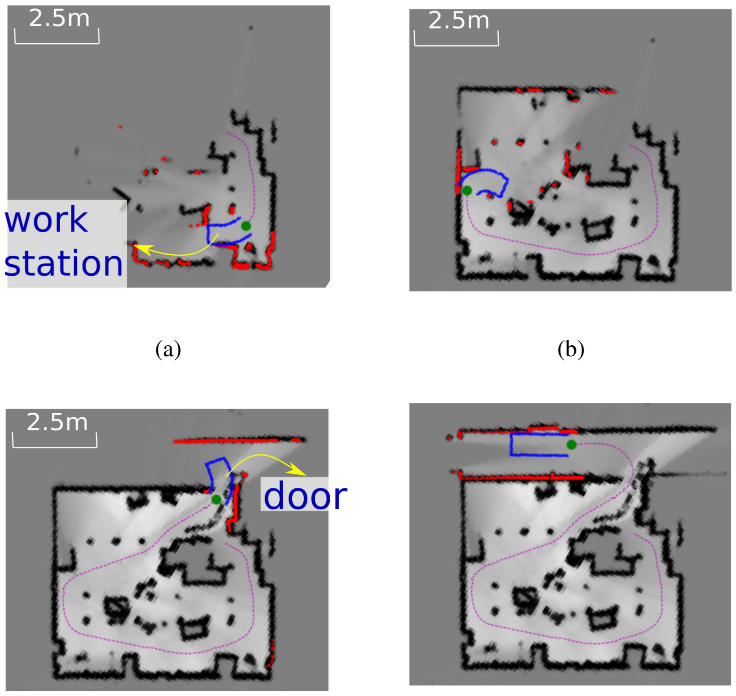
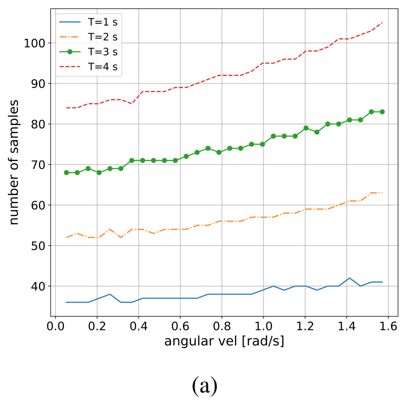


Fig. 10: Local navigation in a cluttered environment using free-space motion tubes. The map is shown in grayscale, where dark pixels correspond to obstacles and white pixels to free space. The position of the robot is shown in a solid green circle, the trajectory in a dashed magenta line, and the selected free-space template in a blue tube.

planner would be required to avoid getting trapped in a local minima. The robot starts at the top right side of the room. The local navigation activity selects the available free-space motion tube with the longest time-horizon (tube painted in blue). In Fig. 10a, the platform is below a cluttered workstation, which corresponds to the camera view shown in Fig. 9b. In Fig. 10b the robot performs a sharp turn to go through a narrow passage that leads to the center of the room. In Fig. 10c, the platform leaves the office through a door (depicted in Fig. 9c). In these three snapshots, local navigation chooses the template with $T = 3$ s and $T = 2$ s. This is due to: 1) the limited visibility of the sensor (occluded by nearby objects) and 2) higher curvature motion tubes are available at low speed and shorter time horizon. Finally, in Fig. 10d the robot has already explored the entire rooom and leaves to the corridor, where the template with the largest horizon is chosen. The platform model (tricycle) does not match the kinematic model (differential drive) of the motion tubes, but the method is robust enough to cope with it. The resulting trajectory of the vehicle, illustrated by a dashed magenta line, is smooth.



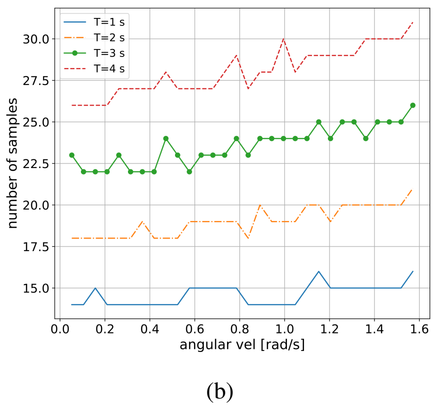


Fig. 11: Number of samples per free-space motion tube as a function of the angular velocity and time horizon ($v = 0.4$ m/s is fixed). The sampling distances are 0.05 m and 0.2 m in Fig. 11a and Fig. 11b, respectivly.

Next, we evaluate the computational time to generate samples of the free-space motion tube in the sensor space and to evalute them online using range scan measurements. As a baseline, we consider the brute-force method for the Hokuyo-URG-04LX sensor (720 angular steps). Brute-force requires computing and evaluating all the beams of the sensor, while the proposed method samples the model such that an obstacle with dimensions greater than a pre-specified value is detected. Fig. 11 shows the number of samples per template as a function of the time horizon and the angular velocity. The templates in Fig. 11a were obtained for $d_{sample} = 0.05$ m, while in Fig. 11b with $d_{sample} = 0.2$ m. The first would be appropriate for a cluttered office-like environment, while the latter would be appropriate in a warehouse (where no small objects are expected). The number of samples per template increases proportionally with the the time horizon. The angular velocity also has an impact. As the angular rate increases, the arc-length corresponding to the inner side of template becomes shorter while the outer side of tube becomes larger. For the two configurations considered here, the motion tubes with the most samples have 104 units ($d_{sample} = 0.05$ m) and 31 units ($d_{sample} = 0.2$ m). Since computational time increases with the number of samples, we take these two worst-case scenarios into account. Table 1 shows the computational time using the Odroid-XU4.

Table 1: Computational time to generate and evaluate different configurations of free-space motion tubes. Results were obtained using an Odroid-XU4.

# samples	Template generation [μs]	Template evaluation [μs]
31	26	0.37
104	82	0.96
720 (brute-force)	540	5.73

For the particular sensor configuration and a template with 31 samples, sampling the motion tube is 20 times faster than brute-force and evaluating the template is 15 times faster. In terms of absolute values, the selected onboard computer is capable of evaluating 2702 free-space motion tubes with 31 samples in 1 ms. The computational time increses almost linearly with the number of samples. For instance, with 104 samples, the proposed method is 6 times faster than brute-force for generating the template samples and 9.5 times faster for evaluating them. For the selected hardware, it would be possible to evaluate 1041 motion tubes with 104 samples in 1 ms. The computational efficiency of the proposed method allows to evaluate a large variety of motion tubes if a finer grained motion is required. It would also be possible to evaluate concatenations of motion tubes in larger spaces with longer visibility ranges for an increased prediction horizon.

5 Conclusions and discussion

This paper presented an efficient method for the local navigation for autonomous mobile robots. Free-space motion tubes capture the spatial region in the environment required by the robot when driving with a particular control input. By selecting meaningful samples and projecting them into the sensor space, the proposed method outperforms brute-force methods. In constrast to map based methods, increasing the sensor resolution does not lead to an increase of computational cost, but it does improve the accuracy of the method and allows to detect smaller objects further away from the sensor. Experimental results show that the free-space motion tubes are well-suited for local navigation in cluttered and non-cluttered environments. The method relies on instaneous sensor readings, which could be problematic when measurements are strongly corrupted by outliers and noise. Fortunately, modern sensors are relatively accurate. Also, during our experiments in a cluttered enviroment the perceived free space was not perturbed by sensor errors. The computational cost is associated with the robot geometry, dimensions of expected obstacles in the environment, and time-horizon. Increasing the sensor resolution improves the accuracy of the method, but it does not have an impact on the computational efficiency. Therefore, we believe that a similar approach will be advantageous in 3D range sensing, where the amount of data is considerably larger.

Acknowledgment. This work was supported by Flanders Make project *HySLAM*, financed by the Flemish government through VLAIO, *RobMoSys* (H2020-732410, *Composable Models and Software for Robotics Systems-of-Systems*), and *ROPOD* (H2020-731848, *Ultra-flat, ultra-flexible, cost-effective robotic pods for handling legacy in logistics*).

References

1. D. Fox, W. Burgard, and S. Thrun, “The dynamic window approach to collision avoidance,” *IEEE Robotics Autom. Mag.*, vol. 4, pp. 23–33, 1997. 1, 2, 3
2. J. Ji, A. Khajepour, W. W. Melek, and Y. Huang, “Path planning and tracking for vehicle collision avoidance based on model predictive control with multiconstraints,” *IEEE Transactions on Vehicular Technology*, vol. 66, no. 2, pp. 952–964, 2017. 1

3. X. Xiao, Z. Xu, Z. Wang, Y. Song, G. Warnell, P. Stone, T. Zhang, S. Ravi, G. Wang, H. Karnan, J. Biswas, N. Mohammad, L. Bramblett, R. Peddi, N. Bezzo, Z. Xie, and P. Dames, “Autonomous ground navigation in highly constrained spaces: Lessons learned from the barn challenge at icra 2022,” 2022. 1
4. S. Quinlan and O. Khatib, “Elastic bands: Connecting path planning and control,” in *[1993] Proceedings IEEE International Conference on Robotics and Automation*, pp. 802–807, IEEE, 1993. 2
5. A. Choudhary, Y. Kobayashi, F. J. Arjonilla, S. Nagasaka, and M. Koike, “Evaluation of mapping and path planning for non-holonomic mobile robot navigation in narrow pathway for agricultural application,” in *2021 IEEE/SICE International Symposium on System Integration (SII)*, pp. 17–22, IEEE, 2021. 2
6. J. Borenstein and Y. Koren, “The vector field histogram-fast obstacle avoidance for mobile robots,” *IEEE Trans. Robotics Autom.*, vol. 7, pp. 278–288, 1991. 2
7. J. Zhang, W. Wang, X. Qi, and Z. Liao, “Social and robust navigation for indoor robots based on object semantic grid and topological map,” *Applied Sciences*, vol. 10, no. 24, p. 8991, 2020. 2
8. G. M. Maciel, M. F. Pinto, I. Júnior, A. L. Marcato, *et al.*, “Methodology for autonomous crossing narrow passages applied on assistive mobile robots,” *Journal of Control, Automation and Electrical Systems*, vol. 30, no. 6, pp. 943–953, 2019. 2
9. C. Rösmann, A. Makarow, and T. Bertram, “Online motion planning based on nonlinear model predictive control with non-euclidean rotation groups,” in *2021 European Control Conference (ECC)*, pp. 1583–1590, IEEE, 2021. 2
10. T. Mercy, R. Van Parys, and G. Pipeleers, “Spline-based motion planning for autonomous guided vehicles in a dynamic environment,” *IEEE Transactions on Control Systems Technology*, vol. 26, no. 6, pp. 2182–2189, 2017. 2
11. A. Tahirovic, G. Magnani, and P. Rocco, “Mobile robot navigation using passivity-based mpc,” in *2010 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pp. 483–488, 2010. 2
12. X. Liu, H. Chen, C. Wang, F. Hu, and X. Yang, “Mpc control and path planning of omnidirectional mobile robot with potential field method,” in *Intelligent Robotics and Applications* (Z. Chen, A. Mendes, Y. Yan, and S. Chen, eds.), (Cham), pp. 170–181, Springer International Publishing, 2018. 2
13. N. T. Nguyen and G. Schildbach, “Tightening polytopic constraint in mpc designs for mobile robot navigation,” in *2021 25th International Conference on System Theory, Control and Computing (ICSTCC)*, pp. 407–412, 2021. 2
14. S. Murray, W. Floyd-Jones, Y. Qi, D. Sorin, and G. Konidaris, “Robot motion planning on a chip,” 06 2016. 2
15. B. Tidd, A. Cosgun, J. Leitner, and N. Hudson, “Passing through narrow gaps with deep reinforcement learning,” in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3492–3498, IEEE, 2021. 3
16. M. Ginesi, D. Meli, A. Roberti, N. Sansonetto, and P. Fiorini, “Dynamic movement primitives: Volumetric obstacle avoidance using dynamic potential functions,” *Journal of Intelligent & Robotic Systems*, vol. 101, no. 4, pp. 1–20, 2021. 3
17. J. A. Reeds and L. A. Shepp, “Optimal paths for a car that goes both forwards and backwards,” *Pacific J. of Mathematics*, vol. 145, no. 2, pp. 367–393, 1990. 6
18. R. T. Rodrigues, N. Tsiogkas, A. Pascoal, and A. P. Aguiar, “Online range-based slam using b-spline surfaces,” *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1958–1965, 2021. 12