Time Distance: A Novel Collision Prediction and Path Planning Method

Ali Analooee, Shahram Azadi, Reza Kazemi

Department of Mechanical Engineering, K. N. Toosi University of Technology, Tehran, Iran

Abstract

Motion planning is an active field of research in robot navigation and autonomous driving. There are plenty of classical and heuristic motion planning methods applicable to mobile robots and ground vehicles. This paper is dedicated to introducing a novel method for collision prediction and path planning. The method is called Time Distance (TD), and its basis returns to the swept volume idea. However, there are considerable differences between the TD method and existing methods associated with the swept volume concept. In this method, time is obtained as a dependent variable in TD functions. TD functions are functions of location, velocity, and geometry of objects, determining the TD of objects with respect to any location. Known as a relative concept, TD is defined as the time interval that must be spent in order for an object to reach a certain location. It is firstly defined for the one-dimensional case and then generalized to 2D space. The collision prediction algorithm consists of obtaining the TD of different points of an object (the vehicle) with respect to all objects of the environment using an explicit function which is a function of TD functions. The path planning algorithm uses TD functions and two other functions called Z-Infinity and Route Function to create the collision-free path in a dynamic environment. Both the collision prediction and the path planning algorithms are evaluated in simulations. Comparisons indicate the capability of the method to generate length optimal paths as the most effective methods do.
Key words: autonomous vehicles, collision avoidance, collision prediction, mobile robots, path planning, swept volume

1. Introduction

Path planning is an innovation-demanding field in control engineering and robotics. Research in this area has yielded plenty of remarkable methods. These methods are divided into classical and heuristic algorithms. Classical algorithms include cell decomposition, potential field, roadmap, subgoal network and sampling-based methods.

In cell decomposition methods, the path is obtained by connecting a set of simple cells which decompose the free space of the robot's configuration space. A detailed explanation of these methods is represented in [1-3].

As a powerful method widely used in robotics, artificial potential field (APF) [4] has been developed by many researchers thus far. This method has also been utilized for path planning of autonomous ground vehicles. As they are non-scenario based, APF methods are advantageous in comparison to most methods associated with path planning for ground vehicles. The works mentioned in [5-12] are some instances of this type.

In sampling-based motion planning methods, the configuration space is sampled by some random or quasi-random points, and a search algorithm is used to find a path by connecting the sample points. In spite of being fast for complex problems, these algorithms are suboptimal. The most popular sampling-based methods are probabilistic roadmap (PRM) and rapidly exploring random trees (RRT). A

comprehensive study of sampling-based methods is available in [13,14].

In roadmap methods, the free configuration space is mapped into a set of one-dimensional lines [1]. Other names of this approach are retraction, highway, and skeleton. The two most important roadmap methods include visibility graph and Voronoi diagram. To study these methods, [1,3] are recommended. In [1], the subgoal method is considered as a kind of roadmap method. In this method, a set of accessible configurations from the primary configuration is obtained, instead of representing configuration obstacles explicitly. Then, if the goal configuration is achievable, the motion planning problem will be solved [1]. Applications of this method are presented in [2].

Heuristic algorithms are increasingly becoming popular and desirable. Most of these algorithms are inspired by biological behaviors. The three well-known, nature-inspired algorithms are genetic algorithm (GA), particle swarm optimization (PSO), and ant colony optimization (ACO). A comprehensive review of neural network, fuzzy logic, nature-inspired methods and hybrid algorithms is presented in [15].

1.1. Swept Volume

The early attempts to calculate swept volumes belong to 1960s [16]. The formulation introduced by Wang and Wang [17], the differential equation based formulation of Blackmore and Leu [18], envelope theory [19], implicit modelling techniques by Schroeder et al. [20] and singularity theory or the jacobian rank deficiency method [21,22] are the most remarkable methods in this area. As a good literature on the mathematical formulation of swept volume (SV) calculation, we suggest the readers see [16].

Although mathematical formulation of SVs has been developed well, their numerical computation for different purposes such as real-time collision prediction and path planning has remained a challenging problem. Based on various SV calculation methods, many algorithms for collision detection and path planning have been introduced thus far.

Cameron [23] extrudes objects from space into space-time environment and defines the collision condition as the intersection of objects' extrusions. In this study, constructive solid geometry is used to describe three-dimensional shapes.

Redon et al. [24] presented a continuous collision detection algorithm for polyhedral rigid bodies. Compared to its previous researches, this algorithm has the advantage of being continuous, which results in computing the first time of collision, and fast, which is comparable to its contemporary discrete algorithms.

A five-stage method to approximate the outer boundary of the SV of complex polyhedrons is presented in [25]. In this study, a novel marching front propagation algorithm is also presented. This marching front propagation algorithm, together with iso-surface reconstruction and topological refinement is used to classify grind points as inside or outside the surfaces.

An algorithm for collision detection between a moving avatar and its virtual environment is presented in [26]. In this algorithm, the SV of every link is obtained by interpolating between the link's discrete positions. Finally, they use a graphic hardware based computation to detect collisions.

A three-stage algorithm for continuous collision detection between a moving articulated model and the simulated environment is presented in [27]. The three stages consist of dynamic bounding-volume hierarchy culling, dynamic SV culling, and exact contact computation. To construct the dynamic bounding-volume hierarchy and perform the dynamic SV culling, interval arithmetic and a graphics hardware accelerated algorithm are used respectively.

Benitez et al. [28] obtained an algorithm for collision detection between general polygonal models. In this algorithm objects are represented by the sphere tree hierarchical representation method.

Peternell et al. [29] presented an algorithm for approximating the boundary surface of the SV using envelope theory. In this algorithm, trimming is based on a marching algorithm in which the time step is not constant.

Choi et al. [30] obtained a new method for collision detection between two elliptic discs. In spite of

other methods that use a discretization of the time interval of motion, and check the intersection of SVs in sampling time instances, they use an algebraic condition to check the collision of two ellipses. This algebraic condition is to detect real roots of a univariate equation which is the discriminant of the characteristic polynomial of the two ellipses. Thereafter, Choi et al. [31] presented another algebraic method to detect collision between ellipsoids.

As another non-graphic-based method, Hui and Hanjun [32] turned the problem of collision detection between two convex polyhedra into calculating the minimum distance between them using a fast algorithm.

Ilies [33] presented a new approach for continuous collision detection between a pair of arbitrary complex objects. In this method, no envelope computation is required, and the problem is reacted in terms of parallel set membership classification tests against the original geometric representations. Furthermore, in this method, the relative motion of objects as a simplifying point has been taken into account.

Bernabeu [34] presented a collision detection method for mobile robots which does not require any space or time discretization. This algorithm also obtains a set of speeds for straight-line motion of the robot which prevents it from colliding with obstacles. Furthermore, if the robot's speed ends in a collision with any obstacle, a set of desired accelerations to avoid collision is obtained.

Huang et al. [35] use Hermit interpolation to approximate the envelope curves of non-convex SVs. In this study, SVs are used to generate collision-free trajectories for the end effector of manipulators in the presence of static obstacles.

Täubig et al. [36] present a self-collision detection algorithm for humanoids and industrial robots. They represent SVs as convex hulls extended by a buffer radius. To detect collisions, first, the SV of all bodies is computed. Then, all pairs of SVs are checked for collision.

Schwesinger et al.[37] use workspace-time space to check the candidate trajectories of a sampling-based motion planner for collision with dynamic obstacles. In this study, which benefits from bounding volume hierarchy data structures, axis-aligned bounding boxes are used to detect collisions.

Ragaglia et al. [38] propose a novel trajectory generation algorithm to take into account the safety requirements for an industrial robot involved in a human-robot collaboration. They use SVs to model the space occupied by the human worker's body in a limited time predicted by a sensor fusion strategy. Then, the trajectory generation algorithm modifies the pre-programmed trajectory to prevent collisions between the robot and the worker considering the SVs as obstacles.

Chiang et al. [39] use SV calculations to improve the performance of sampling-based methods. In this study, size of the SV between sampled configurations has been considered as the distance metric in rapidly exploring random trees and probabilistic roadmap methods. They train deep neural networks to estimate the size of the SV between any pair of sampled configurations.

Gaschler et al. [40] present a collision detection algorithm by defining bounded geometric predicates for collision and inclusion. They propose a set of algorithms called bounding mesh algorithm, bounded convex decomposition, and SV generation which are compatible with the definition of predicates and are used to evaluate them. In this study, to generate SVs, first, the path is discretized. Then, the SV is generated from subsequent convex hulls which are computed by applying a forward kinematics function.

1.2. Contributions

In this paper, a novel method for collision prediction and path planning in a dynamic 2D environment has been developed. The method was shortly introduced in our previous work [41], and will be presented comprehensively in this paper. The method is called Time Distance (TD) and its basis returns to the SV concept. However, there are considerable differences between this method and other existing methods related to the SV idea. SVs are only used as a representation for Time Distance (TD) functions. TD functions are functions that determine the TD of objects with respect to coordinate references or other objects. TD is defined as the time remaining for an object to reach a specific location. Considering TD as

the third dimension of the workspace extends 2D objects into 3D SVs.

In all of the methods and algorithms introduced in the literature, either SVs are obtained implicitly or time is an independent variable in explicit functions. Therefore, in many of the collision detection algorithms, the position of objects must be determined in discrete time instances to approximate envelops, and check if they have intersections with other objects. Then, if an intersection is detected, an interpolation is done within the approximate envelops to obtain a more accurate value for the time of collision. Different algorithms have been suggested for increasing the speed of such calculations. Algebraic methods such as [30] also require fast algorithms to implement their approaches' instructions.

On the contrary, time is obtained as the dependent variable in SV calculations of the present study. We define TD functions in which TD is a function of objects' location, velocity, and geometry. To obtain a vehicle's time to collision with any moving or stationary surrounding obstacle, it is sufficient to substitute all obstacles' location, velocity, and geometric features into an explicit function which is a function of TD functions.

Our path planning approach also has several aspects of novelty. In this method, TD functions are used to help avoid collision with obstacles. We also define two functions called Route Function (RF) and Z-Infinity function. RF helps navigate the robot to the goal or the vehicle between the road lane markers while avoiding collision with stationary or mobile obstacles. Z-Infinity functions are functions with the value of zero inside a geometric shape and infinity outside it. They also help the obstacle avoidance procedure together with TD functions. In static environments, the use of TD functions is not necessary, and path planning can be done using only Z-Infinity and RF. However, TD functions can help improve the optimality of the path in terms of its length. Note that the path planning method and the collision prediction method are two independent methods. In other words, to perform the path planning method, no collision prediction is required. However, since the collision prediction method is more accurate, they can be used together.

The rest of the paper is composed of 6 sections. In Section 2, TD is defined and the TD function for one-dimensional motion is obtained. In Section 3, TD definition is generalized to 2D space, and TD functions for polygonal and circular objects are developed. In Section 4, collision prediction is explained. Path planning instructions are expressed in Section 5. Finally, the conclusions are placed in Section 6.

2. Time Distance Definition

Time Distance (TD) is firstly defined for the one-dimensional case for two mobile points moving on an axis and will be generalized to higher levels later. The TD between two mobile points, or the TD of two mobile points with respect to each other, is the time spent while the locations of these two mobile points coincide. If these two mobile points never touch each other, the value of TD between them is $+\infty$. This is a definition that expresses our expectations of a TD function. The reasons for this definition will be specified in later sections.

To obtain a function that satisfies the definition of TD between two mobile points, consider the one-dimensional travel of two mobile points (A and B) on an axis (y). Two kinematic equations (1) and (2) denote the location of these two mobile points as a function of time. Subtracting these two equations, the difference between the location of A and B is obtained (Eq. (3)). In a moment that the locations of these two points equal, the left side of Eq. (3) vanishes, and Eq. (4) is obtained. Assuming $v_{B/A}$ (the relative velocity of B with respect to A) to be constant, Eq. (5) is feasible. Solving Eq. (5) for Δt, an approximation for the TD between A and B is obtained (Eq. (6)). Equation (6) does not satisfy the definition of TD because in situations that $v_{B/A}$ is zero, Δt is $\pm\infty$ (not exclusively $+\infty$), and in situations that these two points are getting far from each other, Δt has a finite negative value. Equation (7) modifies Eq. (6) and presents a function which satisfies the definition of TD. This function is named TD function of B with respect to A.

$$y_A(t) = y_A + \int v_A dt \tag{1}$$

$$y_B(t) = y_B + \int v_B dt \tag{2}$$

$$y_B(t) - y_A(t) = y_B - y_A + \int (v_B - v_A)dt \tag{3}$$

$$y_B - y_A = -\int v_{B/A} dt \tag{4}$$

$$y_B - y_A = -v_{B/A}\Delta t \tag{5}$$

$$\Delta t = \frac{y_A - y_B}{v_{B/A}} \tag{6}$$

$$TD_{B/A} = 2\left(sign\left(\frac{y_A - y_B}{v_{B/A}}\right) + 1\right)^{-1}\left|\frac{y_A - y_B}{v_{B/A}}\right| \tag{7}$$

Now, if we generalize the location of point A to any arbitrary point on the axis y, Eq. (8) is obtained. In this equation, $TD_{B/y}(y)$ is the TD of point B with respect to any location on the axis y (it means that how much time will be passed until point B reaches any location on the axis y, with the assumption that $v_{B/y}$ is constant during the whole period). Plotting the diagram of Eq. (8) in a coordinate reference with the horizontal axis as y, and the vertical axis as TD is a helpful representation of the TD of point B with respect to different points of the axis y. Fig. 1 illustrates the TD diagram of 4 mobile points O_1- O_4 (moving on the axis y) with respect to the axis y. For each of the points O_1- O_4, TD diagram is a line with constant slope $\frac{1}{v_{O_i/y}}$. Every line starts from the point y_{O_i}, goes to $+\infty$, and never possesses negative values. The black polygon on the diagram is the TD diagram of the set of all 4 points O_1- O_4 with respect to the axis y; that is the time remaining for any location on the axis y to be occupied by any mobile point. The approach to obtain the TD of a set of mobile points with respect to a coordinate reference is to intersect between the TD diagrams of all mobile points. Intersecting between TD diagrams is similar to that between fuzzy sets by "min" T-Norm. If G is a set of n mobile points (Eq. (9)), the TD of this set with respect to any point on the axis y is defined by Eq. (10). Furthermore, considering $O_1 - O_4$ as obstacles, y_P is the safest location on the axis y since $TD_{G/y}(y_P)$ is the maximum value of $TD_{G/y}(y)$ over the axis y. Determining y_P is the basis for path planning in later sections. The approach to determine this point is also similar to "height method" of defuzzification. t_P is the TD of the point y_P with respect to the obstacle set G. The expression to obtain t_P is again similar to "maximum of minimum" fuzzy inference method (Eq. (11)). In order to obtain y_P, t_P must be obtained first. Since t_P is the value of $TD_{G/y}(y_P)$, the operator TD^{-1} is defined to obtain y_p (Eq. (12)).

$$TD_{B/y}(y) = 2\left(sign\left(\frac{y - y_B}{v_{B/y}}\right) + 1\right)^{-1}\left|\frac{y - y_B}{v_{B/y}}\right| \tag{8}$$

$$G = \{O_k | k = 1, \dots, n\} \tag{9}$$

$$TD_{G/y}(y) = \min_{k=1}^{n}\left(TD_{O_k/y}(y)\right) \tag{10}$$

$$t_P = \max_{y=-5}^{5}\left(TD_{G/y}(y)\right) = \max_{y=-5}^{5}\left(\min_{k=1}^{n}\left(TD_{O_k/y}(y)\right)\right) \tag{11}$$

$$y_P = TD^{-1}(t_P) = TD^{-1}\left(\max_{y=-5}^{5}\left(\min_{k=1}^{n}\left(TD_{O_k/y}(y)\right)\right)\right) \tag{12}$$
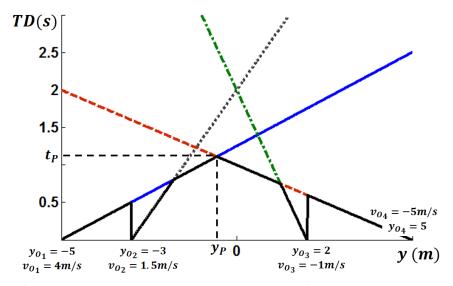
Fig. 1. TD diagram of 4 mobile points O_1-O_4 with respect to the axis y (solid blue, dotted gray, dash-dot green, and dashed red) and intersection of them (solid black) which illustrates the TD of the set of the 4 mobile points O_1-O_4 with respect to the axis y. y_p is the safest location of the axis y because its TD with respect to the obstacle set G is maximum. Note that the axis y is limited to [-5,5].

3. Generalizing the TD Definition to 2D Space

TD definition is generalized to 2D space to obtain the TD of obstacles with respect to coordinate references. To generalize the definition of TD to 2D space, consider the point B moving in the direction of axis x_1 in a 2D space with coordinate reference x-y (Fig. 2). If we use Eq. (8) to express the TD of point B with respect to this 2D space, Eq. (13) is obtained. However, since Eq. (8) was obtained for one-dimensional space, Eq. (13) only works for the points that are on the path of travel of point B. Figs. 3 and 4 illustrate the diagram of Eq. (13) and its contour plot in the x-y plane respectively. The whole points on these two diagrams do not show the TD of point B with respect to the x-y plane. In Fig. 3, only the points on top of point B's path of travel belong to the TD diagram of point B. Similarly, in Fig. 4, only the points on the intersection of contour lines and point B's path of travel show the TD of point B correctly. Therefore, Eq. (13) must be modified to be compatible with the definition of TD. In other words, the value of the TD of point B with respect to the points that are not on point B's path of travel must be $+\infty$. However, obstacles are not points in practice. Consequently, obtaining the TD of a point with respect to a 2D space is not beneficial. Obstacles can be modeled by polygons, circles, etc. Therefore, the contour lines of Fig. 4 can illustrate the TD of the edges of polygons if their length is restricted. Fig. 5 illustrates a line segment ($q_{Ri}q_{Li}$) with length l_i and center point O_i moving in the positive direction of axis x_i' with axis x_i'' normal to it. Equations (14 – 17) indicate the relation between the coordinate reference x-y and these two coordinates. In Eq. (15), $v_{iy/xy}$ and $v_{ix/xy}$ are the velocity components of point O_i with respect to the coordinate reference x-y, in the y and x directions respectively. Equations (18 and 19) denote the components of point O_i's initial location ($x_{O_i}^0$ and $y_{O_i}^0$), and Eq. (20) determines the length of the line segment $q_{Ri}q_{Li}$.

$$TD_{B/xy}(x,y) = 2\left(sign\left(\frac{x_1-x_{1B}}{v_{B/xy}}\right)+1\right)^{-1}\left|\frac{x_1-x_{1B}}{v_{B/xy}}\right| \tag{13}$$

$$\begin{bmatrix} x_i' \\ y_i' \end{bmatrix} = \begin{bmatrix} cos\theta_i' & sin\theta_i' \\ -sin\theta_i' & cos\theta_i' \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} \tag{14}$$

$$\theta_i' = \tan^{-1}\left(\frac{v_{iy/xy}}{v_{ix/xy}}\right) - \frac{\pi}{2}\left(sign(v_{ix/xy}) - 1\right)sign(v_{ix/xy}) \tag{15}$$

$$\begin{bmatrix} x_i'' \\ y_i'' \end{bmatrix} = \begin{bmatrix} cos\theta_i'' & sin\theta_i'' \\ -sin\theta_i'' & cos\theta_i'' \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} \tag{16}$$

$$\theta_i'' = \tan^{-1}\left(\frac{y_{q_{Ri}} - y_{q_{Li}}}{x_{q_{Ri}} - x_{q_{Li}}}\right) + \frac{\pi}{2} \tag{17}$$

$$x_{O_i}^0 = \frac{x_{q_{Ri}} + x_{q_{Li}}}{2} \tag{18}$$

$$y_{O_i}^0 = \frac{y_{q_{Ri}} + y_{q_{Li}}}{2} \tag{19}$$

$$l_i = \sqrt{\left(x_{q_{Ri}} - x_{q_{Li}}\right)^2 + \left(y_{q_{Ri}} - y_{q_{Li}}\right)^2} \tag{20}$$

Fig. 2. Point B moving in a 2D space. The coordinate reference x_1-y_1 is chosen so that the axis x_1 is in the direction of the instantaneous velocity vector of point B.

Fig. 3. Diagram of the TD of point B calculated by Eq. (13). Obviously, TD diagram of a point must be a line, not a plane. Therefore, Eq. (13) must be modified in order to illustrate the TD of a point in a 2D space.
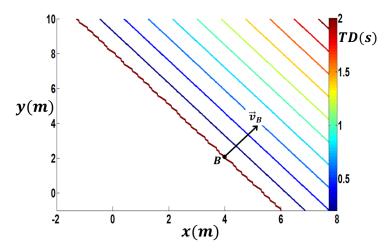
Fig. 4. Contour plot of the diagram of the TD of point B calculated by Eq. (13). Only the points that are on the intersection of point B's path of travel and the contour lines illustrate the TD of this point with respect to the 2D space.
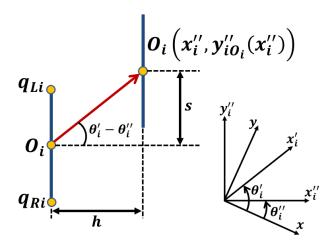

Fig. 5. Line segment $q_{Ri}q_{Li}$ with center point O_i moving in the positive direction of axis x_i' with axis x_i'' normal to it.

The contour lines of the TD diagram of the line segment $q_{Ri}q_{Li}$ are a set of parallel line segments with length l_i and center point $O_i\left(x_i'', y_{iO_i}''(x_i'')\right)$ perpendicular to the x_i'' axis. Note that since the point O_i moves in the definite direction of the axis x_i', the components x_i'' and y_i'' of this point depend on each other. In other words, $y_{iO_i}''(x_i'')$ is a function that determines y_{iO_i}'' for every value of x_i''. To obtain this function, according to Fig. 5 we have:

$$y_{iO_i}''(x_i'') = y_{iO_i}''^0 + s$$
$$s = h\tan(\theta_i' - \theta_i'')$$
$$h = x_i'' - x_{iO_i}''^0$$

Hence:

$$y_{iO_i}''(x_i'') = y_{iO_i}''^0 + \left(x_i'' - x_{iO_i}''^0\right)\tan(\theta_i' - \theta_i'') \tag{21}$$

where $x_{iO_i}''^0$ and $y_{iO_i}''^0$ are the components of the initial location of point O_i. Using this function, the set of the mentioned parallel line segments with length l_i and center point $O_i\left(x_i'', y_{iO_i}''(x_i'')\right)$ perpendicular to

the axis x_i'' is defined by Eq. (22). In other words, the set e_i is the domain of the TD function of this line segment. Consequently, the TD of the line segment $q_{Ri}q_{Li}$ is represented by Eq. (23). In this equation, $v_{x_i''O_i/xy}$ is the velocity of point O_i with respect to the coordinate reference x-y, in the x_i'' direction ($v_{x_i''O_i/xy} = v_{ix/xy}\cos\theta_i'' + v_{iy/xy}\sin\theta_i''$).

$$e_i = \left\{(x_i'', y_i'') \middle| |y_i'' - y_{iO_i}''(x_i'')| \le \frac{l_i}{2}\right\} \tag{22}$$

$$TD_{q_{Li}q_{Ri}/xy}(x,y) = 2\left(sign\left(\frac{x_i'' - x_{iO_i}''^0}{v_{x_i''O_i/xy}}\right) + 1\right)^{-1}\left|\frac{x_i'' - x_{iO_i}''^0}{v_{x_i''O_i/xy}}\right| , \quad (x_i'', y_i'') \in e_i \tag{23}$$

To integrate Eqs. (22) and (23) in order to obtain an explicit equation for the function $TD_{q_{Li}q_{Ri}/xy}(x,y)$, the multiplier Q_i is defined with the following feature:

$$\begin{cases} Q_i = 1 \Leftrightarrow (x_i'', y_i'') \in e_i \\ Q_i = 0 \Leftrightarrow (x_i'', y_i'') \notin e_i \end{cases} \tag{24}$$

An expression for Q_i is:

$$Q_i = sign\left(sign\left(\frac{l_i}{2} - |y_i'' - y_{iO_i}''(x_i'')|\right) + 1\right) \tag{25}$$

Representing the line segment $q_{Ri}q_{Li}$ with its center point O_i (for conciseness), finally, the TD of the line segment O_i with respect to the coordinate reference x-y is expressed by Eq. (26).

$$TD_{O_i/xy}(x,y) = 2Q_i^{-1}\left(sign\left(\frac{x_i'' - x_{iO_i}''^0}{v_{x_i''O_i/xy}}\right) + 1\right)^{-1}\left|\frac{x_i'' - x_{iO_i}''^0}{v_{x_i''O_i/xy}}\right| \tag{26}$$

Defining a polygon as a set of line segments (edges), the TD of every edge of the polygon is obtained from Eq. (26), and similar to Eq. (10), Eq. (27) expresses the TD of the polygon (with m edges) with respect to the coordinate reference x-y. Note that the x_i'' axis of every edge of the polygon is perpendicular to the edge. As a result, x_i'' axes of parallel edges can be the same. (As it is explained in later sections, for Z-Infinity functions, x_i'' axes of parallel edges cannot be the same.)

Similar to Section 2 in which Eqs. (8) and (10) were used to plot the TD diagram of points and a set of points, Eqs. (26) and (27) can be used to plot the TD diagram of line segments and polygons. Figs. 6 and 7 illustrate the TD diagram of a rectangle and a triangle, and its contour plot, with respect to the coordinate reference x-y respectively. The TD diagram of Fig. 6 also represents outermost boundaries of SVs (swept volumes) for these two polygons. Note that the value of the TD function inside a polygon's border is not used in collision prediction. Therefore, it does not matter what this value is in such areas. However, in path planning, the value of TD functions inside the polygons and circles must be zero. Consequently, in the path planning section, Z-Infinity functions are defined to make the value of TD functions in such areas equal to zero.

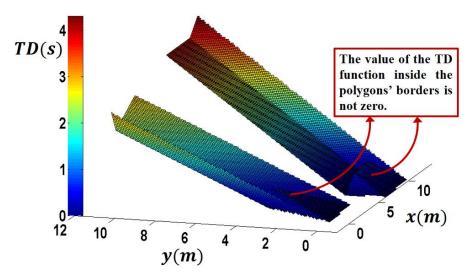$$TD_{Ply/xy}(x,y) = \min_{i=1}^{m}\left(TD_{O_i/xy}(x,y)\right) \tag{27}$$

Fig. 6. TD diagram of a rectangle and a triangle. This diagram can also be interpreted as the diagram of outermost boundaries of SVs. Logically, the value of TD inside the polygons' borders must be zero. However, to avoid redundant calculations, Z-Infinity functions are not used to make the value of TD in these areas equal to zero.
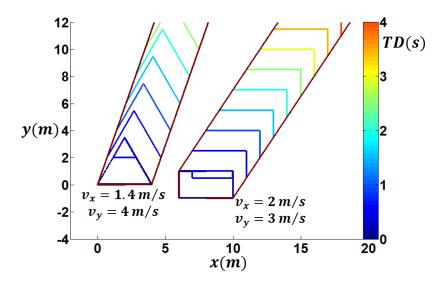

Fig. 7. Contour plot of the TD diagram of Fig. 6.

Similarly, the TD of a circle with center point O_j and radius R_j moving in the positive direction of an axis x'_j, with respect to the coordinate reference x-y, is represented by Eq. (28):

$$TD_{O_j/xy}(x,y) = \frac{2}{Q_j}\left(sign\left(\frac{x'_j - x'^0_{jO_j} - Q_j\sqrt{R_j^2 - \left(y'_j - y'^0_{jO_j}\right)^2}}{v_{O_j/xy}}\right) + 1\right)^{-1}\left|\frac{x'_j - x'^0_{jO_j} - Q_j\sqrt{R_j^2 - \left(y'_j - y'^0_{jO_j}\right)^2}}{v_{O_j/xy}}\right| \quad (28)$$

In this equation, Q_j can be one of the following expressions:

$$Q_j = 1 - \left|sign\left(Im\left(\sqrt{R_j^2 - \left(y'_j - y'^0_{jO_j}\right)^2}\right)\right)\right| \quad (29)$$

Or:

$$Q_j = sign\left(sign\left(R_j - \left|y'_j - y'^0_{jO_j}\right|\right) + 1\right) \tag{30}$$

In Eq. (29), $Im\left(\sqrt{R_j^2 - \left(y'_j - y'^0_{jO_j}\right)^2}\right)$ returns the imaginary part of $\sqrt{R_j^2 - \left(y'_j - y'^0_{jO_j}\right)^2}$. $x'^0_{jO_j}$ and $y'^0_{jO_j}$ (in Eq. (28)) are the components of the initial location of point O_j, and $v_{O_j/xy}$ is the magnitude of the circle's velocity with respect to the coordinate reference x-y (Eq. (31)). Substituting j with i in Eqs. (14) and (15), the relation between the coordinate references x-y and x'_j-y'_j is obtained. Fig. 8 demonstrates the contour plot of the TD diagram of two circles with respect to the coordinate reference x-y.

$$v_{O_j/xy} = \sqrt{v_{jx/xy}^2 + v_{jy/xy}^2} \tag{31}$$

Fig. 8. Contour plot of the TD diagram of two circles

Finally, the TD of a point $P(x_P, y_P)$ with respect to a set of geometric objects G (defined by Eq. (9)) is obtained from Eq. (32).

$$TD_{P/G} = TD_{G/xy}(x_P, y_P) = \min_{k=1}^{n}\left(TD_{O_k/xy}(x_P, y_P)\right) \tag{32}$$

4. Collision Prediction

To predict collision and the time to collision of a vehicle with its surrounding obstacles, the TD of the points on the vehicle's border with respect to the set of all surrounding obstacles must be obtained. To carry this out, the location and velocity of all obstacles must be expressed in a coordinate reference with a velocity vector equal to the vehicle's velocity vector. Consequently, the vehicle's velocity in this coordinate set is zero, and the obstacles' velocities are their relative velocities with respect to the vehicle. Then, Eq. (32) is used to determine the TD of every point P on the border of the vehicle with respect to the set of the surrounding obstacles. A finite number of points for this purpose suffices; for example, four corner points of a rectangular shape robot.

The minimum value of the TD of the points on the vehicle's border with respect to the surrounding obstacles is the time remaining for the vehicle to be involved in a collision (time to collision). If the value of this minimum is infinity, no collision will occur between the vehicle and its surrounding obstacles. Note that it is also possible to determine the obstacle with which the vehicle will collide by obtaining the mentioned minimum value for every obstacle separately.

Fig. 9 illustrates a vehicle and its surrounding obstacles. Diagram of the TD of the points on the vehicle's border with respect to the surrounding obstacles is shown in Fig. 10. The time to collision is obtained 3.5s. Therefore, if the velocities of all obstacles and the vehicle do not change, the vehicle will collide with at least one of the obstacles in 3.5 seconds. To validate this prediction, all obstacles are indicated 3.5 seconds later in Fig. 11. Note that the coordinate reference used for illustration in these figures is not the coordinate reference mentioned to calculate the TDs.

Fig. 9. A scenario for collision prediction.

Fig. 10. Diagram of the TD of the points on the vehicle's border with respect to the surrounding obstacles
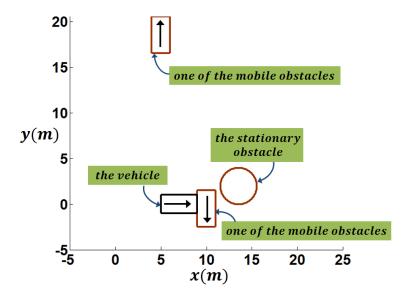
Fig. 11. Location of the obstacles and the vehicle after 3.5 seconds

5. Path Planning

When the imminence of a collision is predicted, a safe path is required to evade the collision. Generally, the collision prediction step is not necessary since the collision-free path is already prepared by the path planning module. However, since the TD collision prediction method is more accurate than the TD path planning method, one may use the collision prediction step along with the path planning step.

In order to plan a path navigating a robot to its goal or a vehicle between the road lane markers, while avoiding collision with obstacles, two other functions are required in addition to exploiting TD functions. These two functions are called Route Function (RF) and Z-Infinity function which will be defined subsequently. Before defining these functions, the Principal and Global coordinates required to express the definitions and equations and to illustrate the scenarios are defined.

5.1. Principal and Global Coordinates

The Principal Coordinate reference (PC) x-y is a coordinate reference with its origin at the vehicle's geometric center point, the x axis toward the vehicle's heading and normal to the vehicle's front edge, and the y axis normal to the left edge. The vehicle's velocity in this coordinate set is zero, and obstacles' velocities in this coordinate reference are their relative velocities with respect to the vehicle. All the equations and definitions used in path planning (consisting of TD functions introduced in previous sections, and RF and Z_∞ which will be defined subsequently) are expressed in this coordinate reference.

The Global Coordinate reference (GC) X-Y is a stationary coordinate reference with no preferred specific directions for its axes. This coordinate set is used in the figures to demonstrate the location and orientation of the vehicle and other components of the path planning scenarios.

5.2. The Path's Equation

As mentioned in Section 2, determining the safest location on the axis y of Fig. 1 is the basis of path planning. In Fig. 1, the safest location on the axis y is a point the TD of which is maximum with respect to the obstacle set G. Considering the space between the robot and the goal in PC to be sectioned by lines perpendicular to the axis x, for every value of x there is a line similar to the axis y of Fig. 1 on which

there is a safest location. The path is formed by connecting these safest points. However, since the safest location in a section is not guaranteed to be safe for the vehicle, the parameter T_s is defined. If the TD of the safest location in a section is equal or larger than T_s, the safest location is also safe for the vehicle (lower bound of T_s is restricted by the road and vehicle's parameters). Consequently, to prevent the vehicle from being led to dangerous locations, if the TD of the safest location in a section is less than T_s, this point will be the end point of the path, and there will be no path at the subsequent sections for the vehicle. As a result, the path is obtained from Eq. (33) in which, similar to Eq. (12), $T_P(x)$ is the TD of the safest location in section x:

$$y_P(x) = TD^{-1}(T_P(x)), T_P(x) \geq T_s \tag{33}$$

5.3. Z-Infinity Functions

As mentioned in Section 3, the value of TD functions inside the geometric shapes is not important in collision prediction. In contrast, the value of TD functions in such areas is important in path planning and must be zero. Therefore, Zero-Infinity (Z-Infinity or Z_∞) functions are defined, so that the value of TD will be zero inside the obstacle borders and infinity outside them.

Similar to the TD function, the Z_∞ function for a polygon is composed of the Z_∞ functions of the polygon's edges. The Z_∞ function of an edge is zero on one of its sides and infinity on the other. Equation (34) expresses the Z_∞ function for the line segment of Fig. 5:

$$Z_{\infty O_i}(\mathrm{x,y}) = \left(sign\left(sign\left(x_{iO_i}''^0 - x_i''\right) + 1\right)\right)^{-1} - 1 \tag{34}$$

Where $x_{iO_i}''^0$ and x_i'' are already defined in Section 3. However, in contrast to the TD function, the direction of x_i'' for Z_∞ differs for parallel edges of a polygon; its positive direction is pointing outwards the polygon (this definition does not interrupt the TD function). Therefore, θ_i'' is redefined. The vertex points q_{Ri} and q_{Li} of every edge are defined so that if we stand on the point O_i facing toward the positive direction of the x_i'' axis, the point q_{Ri} will be located at our right side and q_{Li} at the left side. Consequently, θ_i'' of every edge is expressed by Eq. (35):

$$\theta_i'' = \tan^{-1}\left(\frac{y_{q_{Ri}} - y_{q_{Li}}}{x_{q_{Ri}} - x_{q_{Li}}}\right) + \frac{\pi}{2}sign\left(sign\left(x_{q_{Ri}} - x_{q_{Li}}\right) + 0.5\right) \tag{35}$$

Finally, Eq. (36) expresses the Z_∞ function for a convex polygon with m edges:

$$Z_{\infty Ply}(\mathrm{x,y}) = \max_{i=1}^{m}\left(Z_{\infty O_i}(x,y)\right) \tag{36}$$

For rectangular and circular shape obstacles there are simpler Z_∞ functions. Equations (37) and (38) express the Z_∞ function for a rectangle and a circle respectively:

$$Z_{\infty Rec} = \left(sign\left(\left(sign\left(\frac{l}{2} - |x_w'' - x_{w,C}''|\right) + 1\right) \times \left(sign\left(\frac{w}{2} - |x_l'' - x_{l,C}''|\right) + 1\right)\right)\right)^{-1} - 1 \tag{37}$$

$$Z_{\infty Crcl} = \left(sign(sign(R^2 - (x - x_C)^2 - (y - y_C)^2) + 1)\right)^{-1} - 1 \tag{38}$$

In Eq. (37), w and l are the rectangle's width and length, and x_w'' and x_l'' are axes perpendicular to them respectively. $x_{w,C}''$ and $x_{l,C}''$ are the components of the location of the rectangle's center on the mentioned axes. In Eq. (38), x_C and y_C are the components of the circle's center, and R is the circle's radius.

5.4. Route Functions

RFs (Route functions) play a leading role in path planning. They guide the vehicle toward the goal or between the road lane markers. They also place the path close to the line connecting the vehicle's geometric center to the goal, as much as possible, causing the path to be optimal or near optimal.

To derive an RF for a mobile robot, consider Fig. 12 in which a mobile robot, GC, PC and the goal are illustrated. A variable required to build an RF for the robot is one which expresses the distance of the

environment's points with respect to the line connecting the vehicle's geometric center to the goal. This variable is called y_g. For a better illustration of this variable another coordinate set is chosen by drawing an axis from the vehicle's geometric center to the goal (x_g), and an axis perpendicular to it in the vehicle's geometric center (y_g). This coordinate reference is a rotated coordinate reference with respect to PC. Therefore, y_g is obtained from Eq. (39):

$$y_g(x, y) = y \cos \delta - x \sin \delta \tag{39}$$

Fig. 12. Illustration of GC, PC and the coordinate set $x_g - y_g$ for a mobile robot

The aim of RF is to make the path near optimal and safe for the vehicle. Since Eq. (33) is subjected to the condition $T_P(x) \geq T_s$, to make the path safe for the vehicle, the value of RF itself must be more than T_s. To make the path near optimal , RF must be such that the maximum of which lies on the axis x_g and decreases slightly as $|y_g|$ increases. Fig. 13 demonstrates such a function.
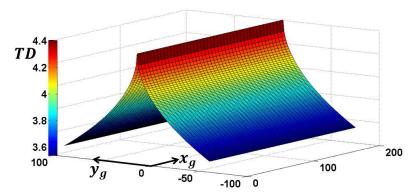

Fig. 13. Desired shape for an RF

Equation (40) presents an RF with the mentioned features for mobile robots. In this function $\alpha, \beta, and\ \gamma$ are constant positive values $(\beta, \gamma < 1 < \alpha)$. It is obvious that $RF_R(x, y)$ will be less than T_s for high enough values of $|y_g(x, y)|$. However, in a finite space with determined maximum and minimum values for y $(y_{min}\ and\ y_{max})$, it is possible to determine the parameters $\alpha, \beta, and\ \gamma$ so that $RF_R(x, y) \geq T_s$ for every x and y.

$$RF_R(x, y) = \alpha T_s - \beta |y_g(x, y)|^\gamma \tag{40}$$

5.5. Path Planning In Static Environments

For an environment with static obstacles, RF and Z_∞ are sufficient for path planning, and it is not necessary to use TD functions. However, as it will be revealed later, using TD functions improves the

optimality of the path.

Equation (41) gives the $T_P(x)$ using only RF and Z_∞, and the path is obtained from Eq. (33). Note that since TD functions are not used in Eq. (41), the condition $T_P(x) \geq T_s$ is changed to $T_P(x) > 0$. We call this path planning method, the "Static TD" method. Equation (41) is similar to Eq. (11). In this equation, there is a Z_∞ function for every configuration obstacle $(Z_{\infty\,obs_1}(x,y), Z_{\infty\,obs_2}(x,y), ...)$, but there is only one RF. Note that since the vehicle is considered as a point robot in the configuration space, the dimensions of obstacles (length of edges of polygons and radius of circles) are extended by the vehicle's diameter to obtain configuration obstacles. Then, Z_∞ functions are calculated for configuration obstacles.

For the point to be illustrated, consider Fig. 14 in which a 100mm×80mm mobile robot is supposed to reach the goal while avoiding two stationary obstacles. The path is obtained from Eqs. (33) and (41) in which RF is RF_R in Eq. (40) with $T_s = 4s, \alpha = 1.1, \beta = 0.1, \gamma = 0.1$. Fig. 15 illustrates the diagram of $\min\left(RF_R(x,y), Z_{\infty\,obs_1}(x,y), Z_{\infty\,obs_2}(x,y)\right)$ for this scenario. Although the environment is static, the path must be modified in every moment to get closer to the optimal state. For example, consider that the vehicle has reached the top of the rectangle shape obstacle (Fig. 16). At this moment, the desired path is modified as shown in Fig. 16. The final path for this scenario is illustrated and compared with the paths obtained from RRT [42], Bidirectional RRT (BRRT) [43], PRM [44], A* algorithm [45] and GA [46] in Fig. 17. The comparison indicates that the path obtained from the static TD method lies among the shortest paths obtained from PRM and A* algorithms. To illustrate the performance of this method in cluttered environments, consider the same robot in the scenario shown in Fig. 18. The robot is supposed to go from the origin of the GC to the goal at the top right corner of the map. The planned path for this scenario is illustrated on the map, and indicates the capability of this path planning method to function in cluttered environments. Next, we show that TD functions get the path closer to the optimal path.

$$T_P(x) = \max_{y_{min}}^{y_{max}} \left(\min\left(RF(x,y); Z_{\infty\,obs_1}(x,y), Z_{\infty\,obs_2}(x,y), ...\right) \right) \tag{41}$$
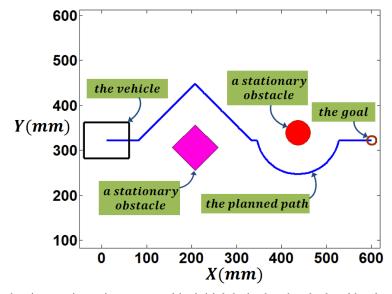

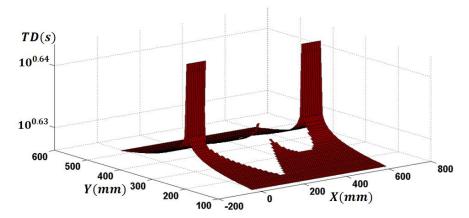**Fig. 14.** A mobile robot in a static environment and its initial desired path calculated by the static TD method.

Fig. 15. Diagram of $\min\left(RF_R(x,y), Z_{\infty\,obs_1}(x,y), Z_{\infty\,obs_2}(x,y)\right)$ for the scenario of Fig. 14 at the first moment. Although it seems that the obstacles are intersected after extending their dimensions because of shrinking the robot, they do not intersect in practice. This has occurred because of the rough grid chose for this figure.
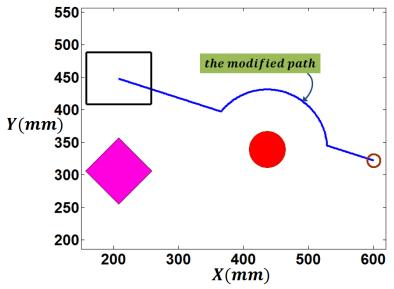

Fig. 16. The path is modified in every sequence of path planning to improve the optimality of it. Comparing this figure with Figs. 14 and 17 reveals the influence of this modification on the optimality of the path.

Fig. 17. Diagram of the final path obtained from the Static TD method (length=700mm) for the scenario of Fig. 14 and comparison of it with A* algorithm (679mm), PRM (704mm), RRT (810mm), BRRT (842mm) and GA (1019mm). After A*, the path obtained from the static TD method is the shortest path. Subsequently, the Dynamic TD method will be used for this scenario which will result in a path with shorter length due to using TD functions in its formulation (Fig. 26).
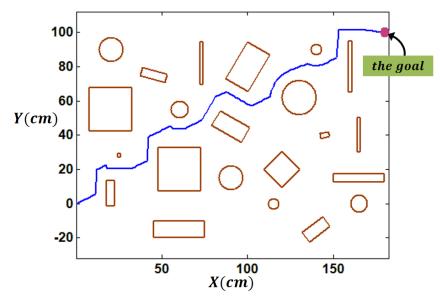

Fig. 18. Path planning in a cluttered environment using the Static TD method

By adding TD functions to Eq. (41), Eq. (42) is obtained. In this equation, the term $TD_{O_k/PC}$ gives the TD of obstacles with respect to PC and is obtained from Eq. (26) or Eq. (28) depending on the geometry of every obstacle. The path is obtained from Eqs. (33) and (42). We call this path planning method, the "Dynamic TD" method. Since the orientation of the vehicle's velocity vector affects the TD of obstacles with respect to the vehicle, the vehicle's orientation affects the path. Therefore, the vehicle's orientation

must be changed slightly while tracking the path. Consequently, a look-ahead point and a path smoothing (trajectory planning) control layer are required to determine the vehicle's trajectory from the vehicle's geometric center to the look-ahead point to smooth the path between the path segments. For this step, a 5-degree polynomial trajectory planning module is applied.

The complete version of the Dynamic TD method of path planning, which covers dynamic environments, is explained in the next section. To clarify how TD functions affect the optimality of the path, consider again the scenario of Fig. 14. The robot is supposed to reach the goal with a constant forward velocity of 15mm/s. Diagram of $\min\limits_{k=1}^{n} \left(TD_{O_k/PC}; RF; Z_{\infty\,obs_1}, Z_{\infty\,obs_2} \right)$ and its contour plot at the first moment of this scenario are shown in Figs. 19 and 20 respectively.

The initial path planned for the vehicle and its corresponding trajectory are illustrated in Fig. 21. Given that the vehicle can exactly track the trajectory, the motion of the vehicle is simulated. Note that when 10% of a trajectory is tracked, the path and trajectory planning steps are repeated. The location and orientation of the vehicle, the planned path and the desired trajectory, and contour plot of the diagram of $\min\limits_{k=1}^{n} \left(TD_{O_k/PC}; RF; Z_{\infty\,obs_1}, Z_{\infty\,obs_2} \right)$ in some other sequences are illustrated in Figs. 22-25. Variations of the path and the way TD functions affect the path during the vehicle's motion are observed in these figures. The whole vehicle's tracked trajectory is illustrated in Fig. 26. The length of the trajectory is 680mm, which is 20mm less than the path obtained in Fig. 17 using Eq. (41), and reveals the effect of TD functions on the optimality of the path. Note that the velocity of the vehicle and T_s are two parameters that can affect the optimality of the path in the Dynamic TD method.

$$T_P(x) = \max_{y_{min}}^{y_{max}} \left(\min_{k=1}^{n} \left(TD_{O_k/PC}(x,y); RF(x,y); Z_{\infty\,obs_1}(x,y), Z_{\infty\,obs_2}(x,y), \dots \right) \right) \tag{42}$$
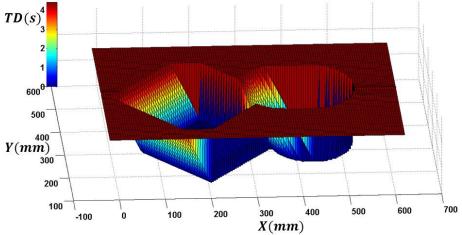

Fig. 19. Diagram of $\min\limits_{k=1}^{n} \left(TD_{O_k/PC}; RF; Z_{\infty\,obs_1}, Z_{\infty\,obs_2} \right)$ for the scenario of Fig. 14 at the first moment.
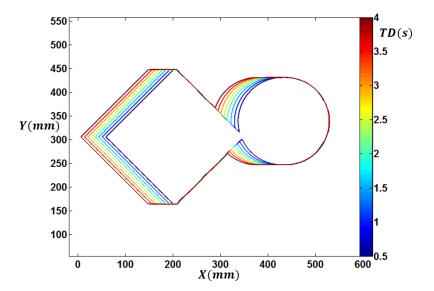
Fig. 20. Contour plot of the diagram of $\min_{k=1}^{n}\left(TD_{O_k/PC}; RF; Z_{\infty\,obs_1}, Z_{\infty\,obs_2}\right)$ for the scenario of Fig. 14 at the first moment.
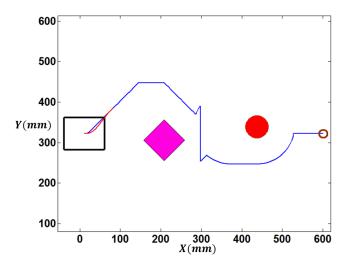

Fig. 21. Diagram of the initial path planned for the vehicle (blue) and its corresponding trajectory (red) for the scenario of Fig. 14.
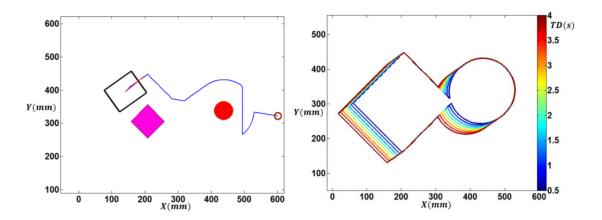
Fig. 22. Diagram of the planned path (blue) and trajectory (red) at a sequence of the vehicle's motion and contour plot of $\min_{k=1}^{n}\left(TD_{O_{k}/PC}; RF; Z_{\infty_{obs_1}}, Z_{\infty_{obs_2}}\right)$ at this sequence.
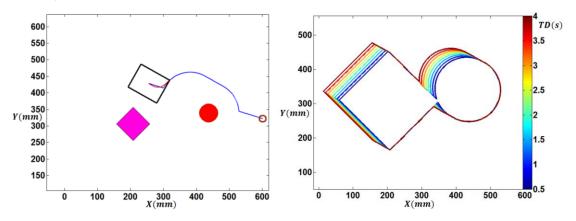

Fig. 23. Diagram of the planned path (blue) and trajectory (red) at a sequence of the vehicle's motion and contour plot of $\min_{k=1}^{n}\left(TD_{O_{k}/PC}; RF; Z_{\infty_{obs_1}}, Z_{\infty_{obs_2}}\right)$ at this sequence.
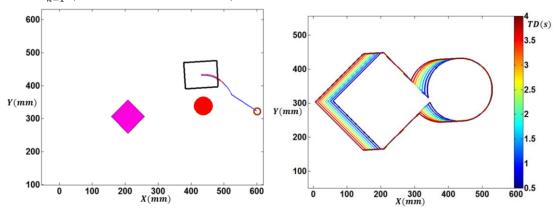

Fig. 24. Diagram of the planned path (blue) and trajectory (red) at a sequence of the vehicle's motion and contour plot of $\min_{k=1}^{n}\left(TD_{O_{k}/PC}; RF; Z_{\infty_{obs_1}}, Z_{\infty_{obs_2}}\right)$ at this sequence.
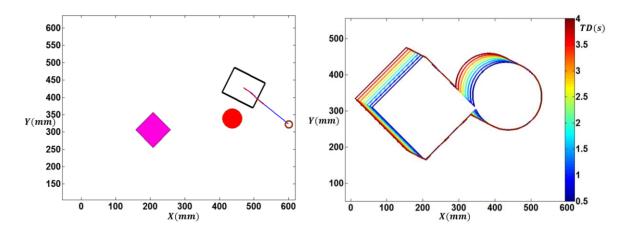
Fig. 25. Diagram of the planned path (blue) and trajectory (red) at a sequence of the vehicle's motion and contour plot of $\min_{k=1}^{n}\left(TD_{O_{k/PC}}; RF; Z_{\infty_{obs_1}}, Z_{\infty_{obs_2}}\right)$ at this sequence.
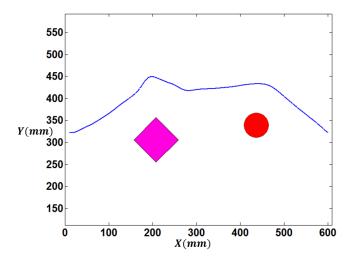

Fig. 26. Diagram of the whole vehicle's tracked trajectory for the scenario of Fig. 14 using the Dynamic TD method (path length= 680mm, vehicle's speed=15mm/s, vehicle's size=100mm×80mm, look-ahead distance=70mm from the vehicle's center point). The path length is approximately equal to the length of the path obtained from A* method (679mm, Fig. 17), and 20mm shorter than the path obtained from the Static TD method demonstrating the effect of using TD functions in the path planning formulation on the optimality of the path.

5.6. Path Planning in Dynamic Environments

Since there are mobile obstacles in dynamic environments, it is required to predict the future state of such environments.

In dynamic environments, after the obstacle borders are extended to reach configuration obstacles, their future relative location and geometry with respect to the vehicle must be determined. By the future relative geometry we mean that since the TD of the vehicle with respect to different points of an obstacle is different, a relative geometry is obtained for obstacles' future location with respect to the vehicle. To this end, first, the future relative location of vertices must be obtained. The TD to calculate the future relative location of every vertex point is the TD of the vertex point with respect to the axis y. Equation (43) expresses the TD of the point q_{R_i} with respect to the axis y. Components of the future relative location of this point are obtained from Eq. (44). In this equation, $v_{ix/GC}$ and $v_{iy/GC}$ are the components

of the velocity vector of the line segment O_i with respect to GC (i.e. the absolute velocity vector) in the x and y (not X and Y) directions respectively. By changing the indices q_{R_i} to q_{L_i} in Eqs. (43) and (44), similar expressions for calculating the future relative location of the vertex point q_{L_i} are obtained. Next, the future relative location of the center point O_i is obtained from Eqs. (18) and (19), and the new length and orientation of every line segment are obtained from Eqs. (20) and (35) respectively. Note that since the future relative geometry of circular objects is elliptical, and no equation for elliptical obstacles is obtained yet, in dynamic environments, obstacles can only be modeled by polygons. It is worth emphasizing that Z_∞ and TD functions are only applied in the future relative location and geometry of configuration obstacles.

$$TD_{qR_i/y} = 2\left(1 - sign\left(\frac{x_{qR_i}}{v_{ix/PC}}\right)\right)^{-1} \left|\frac{x_{qR_i}}{v_{ix/PC}}\right| \tag{43}$$

$$\begin{cases} x_{qR_i}^F = x_{qR_i} + v_{ix/GC} \times TD_{qR_i/y} \\ y_{qR_i}^F = y_{qR_i} + v_{iy/GC} \times TD_{qR_i/y} \end{cases} \tag{44}$$

After the obstacles are extended and transmitted to their future relative location and geometry, the collision-free path is obtained from Eq. (33) where $T_P(x)$ is obtained from Eq. (42).

Fig. 27 illustrates a scenario in which the robot must reach the goal while avoiding collision with the mobile obstacle. In this figure, the future relative location and geometry of the mobile obstacle is also illustrated.
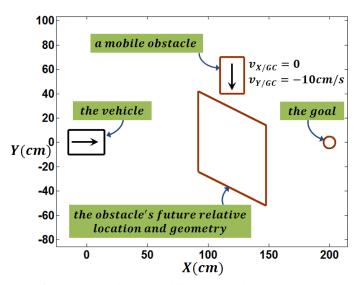

Fig. 27. A dynamic scenario for path planning: a mobile robot and its goal, and a mobile obstacle and its future relative location and geometry.

The vehicle is a 30cm×20cm mobile robot with the maximum forward velocity of 20cm/s. The path is obtained from Eqs. (33) and (42) in which RF is RF_R in Eq. (40) with $T_s = 4s, \alpha = 1.1, \beta = 0.1, \gamma = 0.1$. The look-ahead point is in a 22.5cm distance ahead of the vehicle's geometric center point. The smooth trajectory is a 5-degree polynomial curve connecting the vehicle's geometric center to the look-ahead point. The maximum lateral acceleration of the vehicle is considered to be $a_{n_{max}} = 13.3cm/s^2$. Therefore, the minimum possible radius of curvature for the vehicle at the maximum speed is :

$$\rho_{min} = \frac{v_{max}^2}{a_{n_{max}}} = \frac{400}{13.3} = 30cm \tag{45}$$

In the points where the radius of curvature of the trajectory ρ is less than 30cm, the vehicle's desired speed is obtained from (46):

$$v_d = \sqrt{a_{n_{max}}\rho} \qquad\qquad (46)$$

Figures 28 and 29 illustrate the diagram of $\min_{k=1}^{n}\left(TD_{O_k/PC}, Z_{\infty\,obs}, RF_R\right)$ and its contour plot at the first moment of this scenario respectively. The initial path planned for the vehicle and its corresponding trajectory are illustrated in Fig. 30. Similar to the previous section, considering an ideal control, the vehicle's motion is simulated, and the path and trajectory planning steps are repeated after 7-10% of the trajectory is tracked. Figs. 31 and 32 illustrate two other sequences of path and trajectory planning and contour plot of their corresponding diagram of $\min_{k=1}^{n}\left(TD_{O_k/PC}, Z_{\infty\,obs}, RF_R\right)$. Figure 33 illustrates the whole trajectory from the start point to the goal, and Fig. 34 illustrates the diagram of the desired velocity for the vehicle according to (46). The length of the whole trajectory, passed by the vehicle in 11.3 seconds, is 218cm.
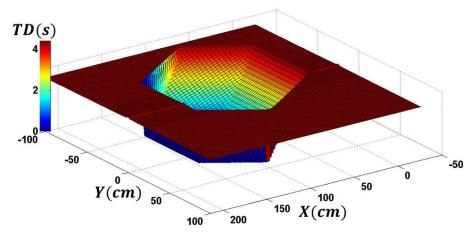

Fig. 28. Diagram of $\min_{k=1}^{n}\left(TD_{O_k/PC}, Z_{\infty\,obs}, RF\right)$ for the scenario of Fig. 27 at the first moment.
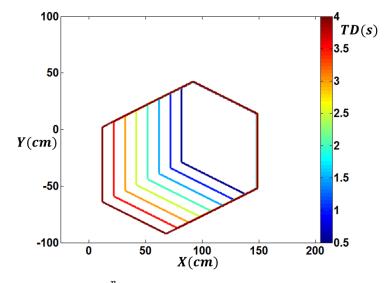

Fig. 29. Contour plot of the diagram of $\min_{k=1}^{n}\left(TD_{O_k/PC}, Z_{\infty\,obs}, RF\right)$ for the scenario of Fig. 27 at the first moment.
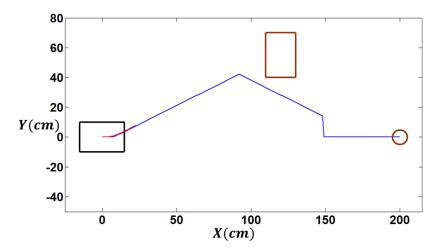
Fig. 30. The initial path (blue) and its corresponding trajectory (red) generated for the scenario of Fig. 27.

Fig. 31. Diagram of the planned path (blue) and trajectory (red) at a sequence of the vehicle's motion and contour plot of $\min_{k=1}^{n}\left(TD_{O_k/PC}; RF; Z_{\infty_{obs}}\right)$ at this sequence.
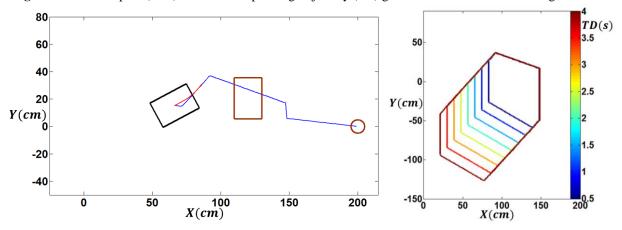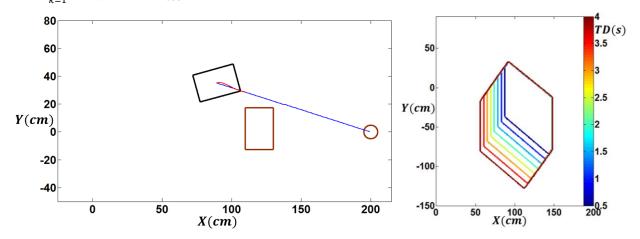

Fig. 32. Diagram of the planned path (blue) and trajectory (red) at a sequence of the vehicle's motion and contour plot of $\min_{k=1}^{n}\left(TD_{O_k/PC}; RF; Z_{\infty_{obs}}\right)$ at this sequence.
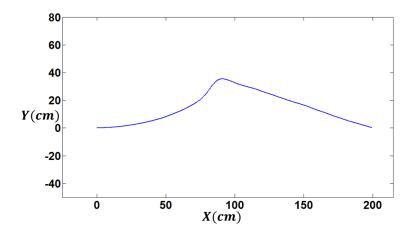
Fig. 33. Diagram of the whole vehicle's tracked trajectory for the scenario of Fig. 27 using the Dynamic TD method (path length= 218cm, time duration=11.3s, vehicle's size=30cm×20cm, look-ahead distance=22.5cm from the vehicle's center point).
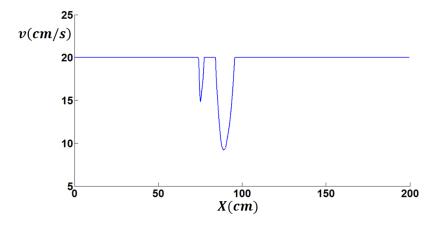

Fig. 34. Diagram of the desired (tracked) velocity of the vehicle for the scenario of Fig. 27.

In addition to facilitating its motion, the vehicle's change in the forward velocity illustrated in this example indicates the adaptability of the Dynamic TD method to variations in the vehicle and environment's states. Further illustrations of this kind will be available in our future works. Extending the methodology to multi-agent systems is also a future plan for this work. To this end, one strategy is to define priorities. In the first increment of path planning, a path is determined for every agent without considering the effect of other agents. In the Nth increment, first, the path of the prior agent is determined without considering the effect of other agents. Then, the path of the agent with second priority is determined by considering the first agent as a mobile obstacle. In this case, the first agent's path in (N-1)th increment is used to determine the future relative location and geometry of this agent with respect to the second agent in the Nth increment. Similarly, the path of the third agent is determined by considering the first two agents as mobile obstacles, and their path in (N-1)th increment is used to determine the future relative location and geometry of them with respect to the third agent in Nth increment. This strategy keeps the path planning procedure of different agents to be parallel processes in conjunction with preventing them to collide with each other.

6. Conclusions and Future Work

In this paper, a novel formulation for collision prediction and path planning based on the SV (swept volume) idea was introduced. SV calculations were developed in TD functions which are explicit functions with time as the dependent variable.

The most important point regarding this method is to obtain time as the dependent variable in TD functions which enhances the real-time quality of collision prediction and path planning. The current methods related to the SV idea are mostly called *collision detection*, not collision prediction. The difference stems from the fact that in collision detection methods associated with the SV idea, time is not the dependent variable in SV calculations, and interpolation algorithms are required to detect collisions. In contrast, the TD method uses an explicit formula to obtain the time to collision which is faster and more accurate.

The TD path planning method also has a unique formulation which benefits from SV calculations. Z-Infinity and TD functions conduct an explicit representation of obstacles which is not achieved in many other methods. For example, in sampling-based, cell decomposition and roadmap methods, for every sampling or grid point, it is checked whether the configuration lies in the configuration obstacles region or not. Specifically, TD functions represent a spectrum of low-risk to high-risk regions in the configuration space which leads to escape the robot from collision risky areas. In addition, in the TD method, the collision-free path is obtained by an explicit formula which is advantageous in comparison to other methods which require hierarchical operations such as sampling, collision checking, searching, etc. Furthermore, the path obtained by the TD method is well optimized in terms of path length. This fact was illustrated in a comparative example. In static environments, TD functions are not necessary for path planning. However, they can help improve the optimality of the path in terms of its length. Accordingly, the TD path planning method is divided into two approaches called Static and Dynamic TD methods.

The TD method is geometrically similar to the APF (Artificial Potential Field) method. APF works with the gradient of the scalar potential functions. In contrast, TD works directly with the scalar TD, Z-Infinity and RF functions. In the APF method, the robot is inclined to go to valleys of the potential function. In contrast, in the TD method, the robot has the tendency to go to the ridges of the scalar function $\min_{k=1}^{n}\left(TD_{O_k/PC}(x,y); RF(x,y); Z_{\infty_{obs_1}}(x,y), Z_{\infty_{obs_2}}(x,y), \dots\right)$. In the APF method, scalar functions are added to each other and their gradient vector is used to generate control inputs which leads to some problems such as local minima, no passage between closely spaced obstacles, and oscillations and unstable motion in narrow passages and in the presence of obstacles. In contrast, using the max and min operators to obtain path segments in the TD method, provides the ability to generate paths among closely spaced obstacles and leads to more stable motions. In addition, the TD method is less likely to be trapped in deadlock situations. Compared to APF in which may be various local minima points, in the TD method only special situations such as U shaped objects may result in similar failures. Furthermore, the dimension of the scalar functions in the TD method is time, and the method benefits from an inherent prediction based on TD which results in a better performance in collision avoidance.

There are abundant future works associated with the TD method. Obtaining the TD function for elliptical shape objects and implementing the future relative location and geometry for circular and elliptical objects is also of importance. Since most works done for path planning of ground vehicles are scenario based methods, the TD path planning method can be helpful in this scope because of being non-scenario based. In our future works, RF for ground vehicles, which navigates the vehicle between the road lane markers, will be introduced, and the TD path planning method will be implemented to this category. Furthermore, determining T_s based on the vehicle's velocity and maneuvering capabilities and the optimality expected in path planning will be a beneficial research. Considering rotational motion of the vehicle and obstacles to have a better collision prediction is also a useful future work. Finally, implementing the TD method to multi-agent systems is one of our future plans.

References

1. Hwang, Y.K., Ahuja, N.: Gross motion planning—a survey. ACM Computing Surveys (CSUR) **24**(3), 219-291 (1992).
2. Atyabi, A., Powers, D.M.: Review of classical and heuristic-based navigation and path planning approaches. International Journal of Advancements in Computing Technology **5**(14), 1 (2013).
3. Šeda, M.: Roadmap methods vs. cell decomposition in robot motion planning. In: Proceedings of the 6th WSEAS International Conference on Signal Processing, Robotics and Automation 2007, pp. 127-132. World Scientific and Engineering Academy and Society (WSEAS)
4. Khatib, O.: Real-time obstacle avoidance for manipulators and mobile robots. In: Autonomous robot vehicles. pp. 396-404. Springer, (1986)
5. Zhenhai, F.A.G., Liyong, S.B.J.: Optimal preview trajectory decision model of lane-keeping system with driver behavior simulation and artificial potential field. In: Intelligent vehicles symposium, 2009 ieee 2009, pp. 797-801. IEEE
6. Kala, R., Warwick, K.: Planning autonomous vehicles in the absence of speed lanes using an elastic strip. IEEE Transactions on Intelligent Transportation Systems **14**(4), 1743-1752 (2013).
7. Xie, S., Wu, P., Peng, Y., Luo, J., Qu, D., Li, Q., Gu, J.: The obstacle avoidance planning of USV based on improved artificial potential field. In: Information and Automation (ICIA), 2014 IEEE International Conference on 2014, pp. 746-751. IEEE
8. Mohammadi, S.S., Khaloozadeh, H.: Optimal motion planning of unmanned ground vehicle using SDRE controller in the presence of obstacles. In: Control, Instrumentation, and Automation (ICCIA), 2016 4th International Conference on 2016, pp. 167-171. IEEE
9. Jiang, H., Wang, Z., Chen, Q., Zhu, J.: Obstacle avoidance of autonomous vehicles with CQP-based model predictive control. In: Systems, Man, and Cybernetics (SMC), 2016 IEEE International Conference on 2016, pp. 001668-001673. IEEE
10. Rasekhipour, Y., Khajepour, A., Chen, S.-K., Litkouhi, B.: A potential field-based model predictive path-planning controller for autonomous road vehicles. IEEE Transactions on Intelligent Transportation Systems **18**(5), 1255-1267 (2017).
11. Rasekhipour, Y., Fadakar, I., Khajepour, A.: Autonomous driving motion planning with obstacles prioritization using lexicographic optimization. Control Engineering Practice **77**, 235-246 (2018).
12. Ji, J., Khajepour, A., Melek, W.W., Huang, Y.: Path planning and tracking for vehicle collision avoidance based on model predictive control with multiconstraints. IEEE Transactions on Vehicular Technology **66**(2), 952-964 (2017).
13. Elbanhawi, M., Simic, M.: Sampling-based robot motion planning: A review. Ieee access **2**, 56-77 (2014).
14. LaValle, S.M.: Planning algorithms. Cambridge university press, (2006)
15. Mac, T.T., Copot, C., Tran, D.T., De Keyser, R.: Heuristic approaches in robot path planning: A survey. Robotics and Autonomous Systems **86**, 13-28 (2016).
16. Abdel-Malek, K., Yang, J., Blackmore, D., JOY, K.: Swept volumes: fundation, perspectives, and applications. International Journal of Shape Modeling **12**(01), 87-127 (2006).
17. Wang, W., Wang, K.: Real-time verification of multiaxis NC programs with raster graphics. In: Robotics and Automation. Proceedings. 1986 IEEE International Conference on 1986, pp. 166-171. IEEE
18. Blackmore, D., Leu, M.C.: A differential equation approach to swept volumes. In: Computer Integrated Manufacturing, 1990., Proceedings of Rensselaer's Second International Conference on 1990, pp. 143-149. IEEE
19. Martin, R.R., Stephenson, P.: Sweeping of three-dimensional objects. Computer-Aided Design **22**(4), 223-234 (1990).
20. Schroeder, W.J., Lorensen, W.E., Linthicum, S.: Implicit modeling of swept surfaces and volumes. In: Proceedings of the conference on Visualization'94 1994, pp. 40-45. IEEE Computer Society Press
21. Abdel-Malek, K., Yeh, H.-J.: On the determination of starting points for parametric surface intersections. Computer-Aided Design **29**(1), 21-35 (1997).
22. Abdel-Malek, K., Othman, S.: Multiple sweeping using the Denavit–Hartenberg representation method. Computer-Aided Design **31**(9), 567-583 (1999).
23. Cameron, S.: Collision detection by four-dimensional intersection testing. IEEE Transactions on Robotics and Automation **6**(3), 291-302 (1990).
24. Redon, S., Kheddar, A., Coquillart, S.: Fast continuous collision detection between rigid bodies. In: Computer graphics forum 2002, vol. 3, pp. 279-287. Wiley Online Library

25. Kim, Y.J., Varadhan, G., Lin, M.C., Manocha, D.: Fast swept volume approximation of complex polyhedral models. Computer-Aided Design **36**(11), 1013-1027 (2004).
26. Redon, S., Kim, Y.J., Lin, M.C., Manocha, D., Templeman, J.: Interactive and continuous collision detection for avatars in virtual environments. In: Virtual Reality, 2004. Proceedings. IEEE 2004, pp. 117-283. IEEE
27. Redon, S., Lin, M.C., Manocha, D., Kim, Y.J.: Fast continuous collision detection for articulated models. Journal of Computing and Information Science in Engineering **5**(2), 126-137 (2005).
28. Benitez, A., del Carmen Ramírez, M., Vallejo, D.: Collision detection using sphere-tree construction. In: Electronics, Communications and Computers, 2005. CONIELECOMP 2005. Proceedings. 15th International Conference on 2005, pp. 286-291. IEEE
29. Peternell, M., Pottmann, H., Steiner, T., Zhao, H.: Swept volumes. Computer-Aided Design and Applications **2**(5), 599-608 (2005).
30. Choi, Y.-K., Wang, W., Liu, Y., Kim, M.-S.: Continuous collision detection for two moving elliptic disks. IEEE Transactions on Robotics **22**(2), 213-224 (2006).
31. Choi, Y.-K., Chang, J.-W., Wang, W., Kim, M.-S., Elber, G.: Continuous collision detection for ellipsoids. IEEE Transactions on visualization and Computer Graphics **15**(2), 311-325 (2009).
32. Hui, L., Hanjun, J.: An algorithm for computing the minimum distance between two convex polyhedra in three-dimensional space. In: Knowledge Acquisition and Modeling, 2008. KAM'08. International Symposium on 2008, pp. 313-317. IEEE
33. Ilieş, H.T.: Continuous collision and interference detection for 3D geometric models. Journal of Computing and Information Science in Engineering **9**(2), 021007 (2009).
34. Bernabeu, E.J.: Fast generation of multiple collision-free and linear trajectories in dynamic environments. IEEE Transactions on Robotics **25**(4), 967-975 (2009).
35. Huang, T.J., Wu, J.T., Huang, C.T.: A swept volume approach based on hermite interpolation for optimization in collision-free motion planning. Journal of the Chinese Institute of Engineers **33**(7), 945-953 (2010).
36. Täubig, H., Bäuml, B., Frese, U.: Real-time swept volume and distance computation for self collision detection. In: Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on 2011, pp. 1585-1592. IEEE
37. Schwesinger, U., Siegwart, R., Furgale, P.T.: Fast collision detection through bounding volume hierarchies in workspace-time space for sampling-based motion planners. In: ICRA 2015, pp. 63-68
38. Ragaglia, M., Zanchettin, A.M., Rocco, P.: Trajectory generation algorithm for safe human-robot collaboration based on multiple depth sensor measurements. Mechatronics (2018).
39. Chiang, H.-T.L., Faust, A., Tapia, L.: Deep Neural Networks for Swept Volume Prediction Between Configurations. arXiv preprint arXiv:1805.11597 (2018).
40. Gaschler, A., Petrick, R.P., Khatib, O., Knoll, A.: KABouM: Knowledge-Level Action and Bounding Geometry Motion Planner. Journal of Artificial Intelligence Research **61**, 323-362 (2018).
41. Cadkhodajafarian, A., Analooee, A., Azadi, S., Kazemi, R.: Collision-free navigation and control for autonomous vehicle in complex urban environments. Modares Mechanical Engineering **17**(11), 277-288 (2018).
42. Kala, R.: Code for Robot Path Planning using Rapidly-exploring Random Trees. In. Indian Institute of Information Technology, Allahabad, (2014)
43. Kala, R.: Code for Robot Path Planning using Bidirectional Rapidly-exploring Random Trees. In. Indian Institute of Information Technology, Allahabad, (2014)
44. Kala, R.: Code for Robot Path Planning using Probabilistic Roadmap. In. Indian Institute of Information Technology, Allahabad, (2014)
45. Kala, R.: Code for Robot Path Planning using A* algorithm. In. Indian Institute of Information Technology, Allahabad, (2014)
46. Kala, R.: Code for Robot Path Planning using Genetic Algorithms. In. Indian Institute of Information Technology, Allahabad, (2014)