Efficient Geometrical Obstacle Reshaping for Inevitable Collision States Avoidance

Hamed Bastani
School of Engineering and Science
Jacobs University Bremen
Campus Ring 1, Bremen 28759, Germany
h.bastani@jacobs-university.de

Hamid MirMohamad Sadeghi
Information and Communication Technology Institute
Isfahan University of Technology
Isfahan 84154, Iran
sadegi_m@cc.iut.ac.ir

Abstract—Classical motion planners and most of navigation routines discard dynamics of the system and therefore fail in some special conditions. If a robot is moving towards an obstacle with a high acceleration and speed, collision is inevitable because the obstacles are taken into account only either once they are in range, or a physical collision is reported by detection routines. In this paper, we have introduced a new approach named T-Bone Algorithm, that considers kinematics of an extended case of a Dubins car moving in a static environment, in order to avoid potential collisions, however it inherently can be extended for more complex systems. This method deforms the perception of the collision sensors – which traditionally discard dynamics of the system and only consider physical appearance of the obstacles. Deformed state space is then fed into the planner and navigation unit in order to move the vehicle, safely. Bounded complexity of the algorithm makes it proper to be used in realtime applications. Implementation results of the algorithm and some typical test runs are presented in the paper.

Index Terms—Mobile robotics navigation, inevitable collision state, obstacle avoidance.

INTRODUCTION

One of the most challenging aspects of planning strategies can be visualized in terms of the Region of Inevitable Collision, or in other words Inevitable Collision Obstacle, denoted by ICO. An inevitable collision state for a system can be defined as a state for which, independent from what the next control input is, a collision with an obstacle eventually happens. An inevitable collision state routine needs to take into account the dynamics of both vehicle and obstacles, fixed or moving. In classical motion planning, the planner only considers the actual position of obstacles in the state space, and the system changes the path only once the collision detector senses an obstacle (figure 1-a). It is problematic if the vehicle suffers from not having sensors with long range and/or a wide field of view, its speed/acceleration is too high, or if it is interacting in a non-static environment. Generally speaking about the configuration space \((C_s)\), at low enough speeds \(C_{ico}\) and \(C_{obstacle}\) are approximately the same; however, \(C_{ico}\) grows dramatically as the speed increases or some other variables change.

The concept of ICO is very general and can be useful both for navigation and motion planning purposes in robotics1. For its own safety, a robotic system should never find itself in an inevitable collision state. The main contribution of this paper is to lay down and explore a new concept for geometrically calculating the inevitable collision state, and insert it in the motion planning chain of the system while getting the state and control inputs into consideration (refer to figure 1-b). The formulated approach – called the T-Bone Algorithm, is defined for a system with nonholonomic kinematics, while its differential equations can be extended and modified for similar cases. This solution considers that static obstacles are represented by a set of intersecting linear segments. The strong point of view this algorithm is using the concept of approximation to calculate the ICO. Even though it returns the precise solution, using approximation reduces the price of calculations and gives much faster solution than the point-by-point analysis. It returns the solution for the marginal

1Inevitable collision obstacle principles are to some extent related to the danger zone concept that can be found in the Air Traffic Control literature [2][5].
boundaries as the worst case scenario, such that surely covers the unsafe area.

Without loss of generality, after modifying the main parameters used for the T-Bone definition, this algorithm can be extended for other dynamical models as well. Once T-Bone parameters are calculated, it only takes the control input, \( u \), as well as obstacles’ linear boundary segments to be substituted with the respective ICO. Similar to classical views, a map sketch of the work space is needed for motion planning purposes. This map can be fetched from a formerly built database or can be generated while the vehicle is moving – using techniques such as SLAM (Simultaneously Localization And Mapping) [15]. Unfortunately most of the works in SLAM suppose that the environment is static, but still other alternatives like SLAMMOT [13], standing for SLAM-and Moving Object Tracking addresses the dynamic configuration space.

I. RELATED WORK

In some literatures braking prisms is used instead of ICO. They are subsets of the configuration space associated with a given state of a robotic system and known to contain the braking trajectory corresponding to a given braking policy. There are discussions and proposed solutions on how to embed these policies into the control loop of the system [7]. A configuration without any braking prism included in the free configuration space must be avoided because it would yield a collision no matter which braking policy is used. [9] describes a trajectory planning scheme for a robotic system with a limited field of view. It characterizes robust states as states for which the robotic system can safely stop simply by braking even when placed in an environment with moving obstacles (basically, the field of view is shrunk according to the moving obstacles maximum possible speed). Also, inevitable collision states are to some extent related to the danger zone concept that can be found in the air traffic control literature. Outside such danger zones, evasive maneuvers are provably safe [5]. When it comes to motion planning, the inevitable collision state concept is also very useful. Consider the problem of planning motions for a robotic system moving in a partially known environment. The system is subject to sensing constraints (limited field of view) and moves among unknown obstacles. Imagine a surveillance robot, it has a map of the building it must patrol, but does not know a priori position of the furniture or if people are moving around. Based on a priori information available, a nominal trajectory for the robotic system can be computed and locally modified based on partial changes in the state space. Reactive sensor-based approaches that control the movements of the robot based on its sensory input are developed in the same perspective. Typical members of this class are the curvature-velocity method [20], the dynamic window approach [17], the vector-field histogram technique [16] and the potential field representation [18]. There are developed motion planning algorithms based on dynamic programming and Markov Decision Process (MDP) such as in [8]. With similar complexity, it is feasible to use a probabilistic model such as Bayesian for obstacle avoidance ([3]) which is known to have high mathematical and therefore time complexity. Last two cases are basically proper for slow movements. On the other hand, there is no urgent need for including the ICO concepts in such cases, because in low speeds, it is approximately equal to the obstacle itself. Regarding the kinematics of the system, there are plenty of works done to reinforce the idea by means of the control theory tools. For example in [14], they analyse the motion along circular reference paths with radius greater than or equal to the car’s minimal turning radius. [10] proposes a sliding-mode feedback taking into account the curvature constraint to stabilize a Dubins car along an unknown reference path, and [21] follows the same principle as well.

II. HYPOTHESIS

A. Problem Statement

When referring to configuration space of a robotic system, the focus is on the geometric aspects of motion planning (no collision between the system and the fixed obstacles of the workspace). This is the appropriate framework to address path planning problems. On the other hand, the state space, is more appropriate when it comes to address trajectory planning problems where the dynamics of the system is taken into account. In the configuration space, the notion of forbidden or collision configuration (ie. configurations yielding a collision), is well-known and so is the notion of configuration obstacles (ie. the set of configurations yielding a collision between the system and a particular obstacle) [4]. Transposing these notions in the state space, it is straightforward to define collision states and state obstacles. However, the interest of extending these notions is to take into account the dynamics of the system by introducing the concept of inevitable collision states (ICS). As a simple example, consider a point mass \( P \) with only 1 degree of freedom, in a one dimensional work space. A state of \( P \) is characterized by its position on \( x \) axis as well as its speed, \( v \). Its state can be written by \( X_p = (x,v) \). If this system faces an obstacle (obs in fig.2) in the work space, the states whose positions correspond to the obstacles are obviously collision states. Assuming that it takes \( P \) a certain distance \( d(v) \) to decelerate and stop, the states corresponding to the obstacle and the states located at a distance less than \( d(v) \) of it are such that, when \( P \) is in there, no matter what it does, in the future a collision will occur. These states are inevitable collision states for \( P \). Clearly, for \( P \)’s own safety, when it is moving at speed \( v \), it never should get closer than \( d(v) \) to any obstacle in order to certainly avoid any collision. The shape of the ICS region generally depends on the dimensions

616
of the work space, form of the obstacles in $C$, dynamics and variables of the system, and the number of degrees of freedom of the vehicle. Assuming that $d(v)$ varies linearly with $v$, the complete set of ICS is shown in figure 2. In a 3D workspace, the ICS will grow in volume as a function of $v$.

B. Dynamical Constraints

Car like models are the most common ones to represent variety of vehicles and mobile robots mechanical movement structures. They include a 3 DoF representation for the system in a 2D plane. The three variables of the state space for this model are basically $(x, y, \theta)$. One of the most common categories in the class are Dubins cars [1]. It is by definition called a Dubins car, if a car like vehicle (a vehicle with 4 wheels, which has the front pair for steering and at least two for driving – similar to figure 3) can only go forward with a fixed speed of $v = 1$. In a very similar view, if $v = \pm 1$, it is referred to as a Reeds-Shepp car [12]. Implicitly it means the latter also can move reversely unlike the former case. The system considered in this paper is defined to be a Dubins car but with variable speed $v$, and controllable steering angle $\phi$. These parameters are included in the control input of the system; let the scalars $a = \dot{v}$ and $\phi$, respectively denote the acceleration of the car and the steering angle. In other words, the control input for this system is defined as $u_s = (a, \phi)$. This control input is bounded because in reality the speed can not change instantly, i.e. never $\dot{v} = \infty$ so always $|a| < a_{\text{max}}$ holds in real situations. Additionally most of the steering vehicles have a limited maximum steering angle, $\phi_{\text{max}}$, such that $0 < \phi_{\text{max}} < \frac{\pi}{2}$. A maximum steering angle implies a minimum turning radius which is equal to $r = \frac{L}{\tan \phi}$, $L$ giving the distance between the front and the rear axle. State of such a system (ref. to figure 3) can be represented by $X = (x, y, v, \theta)$ and its kinematic therefore can be driven by $\dot{X} = F(X)$:

$$
\begin{pmatrix}
\dot{x} \\
\dot{y} \\
\dot{v} \\
\dot{\theta}
\end{pmatrix}
= 
\begin{pmatrix}
v \sin \theta & v \cos \theta & a & \frac{v}{L} \tan \phi
\end{pmatrix}
$$

III. T-BONE ALGORITHM

Inevitable collision obstacle calculation depends on system’s dynamics, that is dynamics of the mobile robotic system and dynamics of the obstacles, if they move. The proposed solution assumes that the obstacles are stationary. Without any limitation, the idea given in [3] which uses the field of view of the obstacle detector sensors, plus a priori knowledge about obstacles (their position in last time interval), the T-Bone method can be adjusted\(^2\). The principle of this algorithm is based on approximation to determine the ICO for the boundaries of the system’s constraints, both physically and mathematically. It means estimating the most enlarged obstacles according to the control input sequence and state of the system. Two dimensional projection of ICO is never smaller than the original obstacle, therefore the deformation of obstacle is always enlargement. To study the boundaries of the ICO approximation, three cases are checked. In all cases it should be noted that if the vehicle is accelerating and if the continuation of the calculated trajectory has intersection with any obstacle, the robot will inevitably collide. On the other hand, if the robot is decelerating, after some distance it is likely to stop. Length of the traversed distance, no matter linear or curvature, is a function of the speed and deceleration values. Furthermore, trajectory of movement is dependant on the steering angle.

• **Straight forward movement** ($\phi = 0$)

Constraints to obtain approximation boundaries are $\phi = 0$; $v = \text{arbitrary value in range}$ and directionally orthogonal to the obstacle; $a = \text{deceleration and, } \theta = \text{arbitrary value for the heading angle}$. In this situation, the system will stop after traversing a linear distance of $r = \frac{v^2}{2a}$. Direction of the movement is given by the velocity vector $v$. Figure 4-Left, illustrates this case.

• **Turning with steering angle $\phi < 0$**

Constraints to obtain approximation boundaries are: $\phi = -\phi_{\text{max}}$; $v = \text{arbitrary value in range}; a = \text{deceleration and, } \theta = \text{arbitrary value for the heading angle}$. Here, the vehicle will stop after traversing a curvature distance of length $\frac{v^2}{2a}$. This

\(^2\)In our simulations, the boundaries of rectangular workspace are also treated as intersecting half-plane obstacles. The same concept can be applied to the boundaries of the FoV of the obstacle detector sensor instead of the boundaries of the workspace, which may reduce the size of $C$ but will essentially depend on the accurate availability of the sensor’s pose information.
curve is an arc of the circle \( c2 \) in figure 4-Right, starting from point B and ending on point D. Direction of movement is given by the rotated velocity vector \( v \), while it originally makes angle \( \phi \) with the circle \( c2 \) on the tangent point B. The radius of the imaginary circle which gives the trajectory for this case is equal to \( r = \frac{L}{|\tan \phi|} \).

- Turning with steering angle \( \phi > 0 \)

Constraints to obtain approximation boundaries are: \( \phi = +\phi_{max} \), \( v = \) arbitrary value in range; \( a = \) deceleration and, \( \theta = \) arbitrary value for heading angle. Again, the vehicle will stop after traversing a curvature distance of length \( \frac{aL}{\tan \phi} \), an arc of the circle \( c1 \) starting from point B and ending on point C in figure 4-Right. Direction of movement is given by the rotated velocity vector \( v \), when makes angle \( -\phi \) with the circle \( c1 \) on the tangent point B. The radius of the imaginary circle that gives the trajectory for this case is again equal to \( r = \frac{L}{|\tan \phi|} \).

Intersection of the cases above, is an area bounded by three arcs. Area of this surface depends on the dynamics of the system, velocity and acceleration. In this surface, we embed a bone looking like the letter T, where having three known points B, C and D are sufficient to represent it. These 3 points are chosen out of the solutions of 3 pairs of polynomial second order equation systems (from each intersection subset of \( \{c1, c0\}, \{c2, c0\}, \{c1, c2\} \}, only one valid points are chosen). After all, a normal unity vector indicated by \( \vec{n} \) is associated to the bone (figure 5-Left).

![Diagram](image)

**Fig. 4.** Left: The vehicle is moving directly towards an infinite line obstacle \( v \) orthogonal to the obstacle. B is the middle of the front axel of the vehicle. Right: The robot is turning with the steering angle \( \phi < 0 \). Here \( \theta = 90 \) in the given coordinate system.

![Diagram](image)

**Fig. 5.** Left: Original T-Bone. Vector \( \vec{n} \) will rotate according to \( \theta \). Right: The edge displacement is a scaler value \( H = H(a, v, \phi, \theta, L) \).

T-Bone sweeps all obstacles in order to determine the ICO region. Enlargement of the obstacles depend on the size of T-Bone and the angle it faces each segment of a polygonal obstacle. Direction of \( \vec{n} \) depends on \( a \), i.e. \( m \sim sgn(a) \) and \( |m| = \frac{v}{a} \). It is important to confirm if the inner product of \( \vec{m}, \vec{n} < 0 \) where \( \vec{n} \) is the unity normal vector of the faced segment of the obstacle. Direction of this vector is pointing outwards from the obstacle segments.

Figure 6 shows an example to illustrate how a linear segment of the boundary of an obstacle is deformed by means of a sweeping T-Bone. The enlarged region for a half plane obstacle is represented by an infinite line parallel to the half plane in distance \( H \), where this distance is a function of the system state and the control input; \( H = H(a, v, \phi, \theta, L) \). It can be seen that \( a = \frac{v^2 \tan \phi}{4aL} \) if the length of arcs BC and BD are both equal to \( \frac{aL}{\tan \phi} \). In the triangle OBC in figure 5-Right, the length of the edge BC is equal to

\[
k = \frac{2L}{\tan \phi} \sin \left( \frac{\alpha}{2} \right)
\]

(2)

For the case that \( \vec{m}, \vec{n} < 0 \), the distance between the linear segment of the obstacle and the new infinite line which is drawn by the ending tail of the T-Bone will be \( k \sin(\theta \pm \beta) \). In other words, H can respectively be calculated for points B and C, whereas with no limit we can consider \( \beta \approx \phi \). Hence

\[
H = \frac{2L}{\tan \phi} \sin \left( \frac{v^2 \tan \phi}{4aL} \right) \sin(\theta \pm \beta)
\]

(3)

It needs to be remarked that the vector \( \vec{n} \) is defined only for the deceleration case. Also \( \vec{m}, \vec{n} < 0 \) has to hold to assure that the T-Bone is facing a linear segment and is going toward the obstacle. It is assumed that the obstacles inside the configuration space are all convex. If not, they will be diffracted into convex sub-obstacles which each two adjacent ones have at most one segment in common. Figure 7 gives principles of how to diffract a non convex obstacle, as well as segment normal unitary vector (\( \vec{n} \)) assignment.

![Diagram](image)

**Fig. 6.** Segment deformation: two typical sweeping T-Bones influencing half plane obstacles. Left for \( \theta = 90 \) and right for \( \theta = 45 \). Generally speaking, the edge displacement value, \( H \), is an scaler function of 5 variables; \( H(a, v, \phi, \theta, L) \).

![Diagram](image)

**Fig. 7.** Non-convex polygonal obstacle decomposition and normal unity vector assignment to linear segments of each convex sub-obstacle.

Any polygon can be represented with intersection of half planes. Number of facets on the polygon are at most equal to the number of half planes. If not overlapping, each two intersecting half planes will have only one common intersection.
point. This intersection represents one corner of the polygonal obstacle. For accomplishing the approximated boundaries the algorithm is run for the intersection points, only. In other words, in the order of \# of obstacles’ corners the T-Bone Algorithm gives the most secure ICO. Applying the T-Bone to linear segments for deformation, returns linear results indeed. Therefore geometrical representation for original obstacles are obtained directly by either displacement or replacement operators.

The algorithm consists of 10 steps to be done for each obstacle in the work space, separately. Then the results are merged together, i.e. the union of all sub results makes the overall output for any control input and applies the DeMorgan’s theorem to the seminal results. In the sequence of the algorithm, the current obstacle is named \( O \). In there, the configuration space is \( C_s \), \( C_O \) denotes the current obstacle in the \( C_s \), \( S = \{ S_i \} \) is the set of segments making the boundary of the obstacle \( O \). \( P = \{ P_i \} \) contains the set of start and end points of all segments in \( S \), in other words it is the set of all corners of the obstacle \( O \). Also, consider points \( B, C \) and \( D \) as the known parameters of the T-Bone. Steps below target each obstacle of the \( C_s \) separately:

**Algorithm: T-Bone (\( C_s \))**

1. **Point \( C \)** sweeps all points in \( P \), if relative position of \( B \) with respect to the current corner point \( P_i \) is \( \in \{ \{ C_s \} \backslash \{ C_o \} \} \), add this relative position into \( P \).
2. **OC** ← Convex Hull \( (P) \).
3. **Update \( S \)**. In **OC**, name ex-edge the segments in \( S \) which have only one node belonging to \( O \). Replace ex-edges with translated and rotated arc BC. Rotation matrix is given by \( R = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix} \) and translation depends on the position of the current corner point \( P_i \).
4. **Exclude newly added points from \( P \)**.
5. **D** sweeps all points in \( P \), if relative position of \( B \) with respect to the current corner point \( P_i \) is \( \in \{ \{ C_s \} \backslash \{ C_o \} \} \), add this relative position into \( P \).
6. **OD** ← Convex Hull \( (P) \).
7. **Update \( S \)**. In **OD**, replace ex-edges with transformed and rotated arc BD similar to step 3.
8. ** ICO \(_J\)** ← \( \{ OC \cap OD \} \).
9. **Repeat steps above for all obstacles**.
10. **ICS** ← \( \cup \{ ICO \_J \} \).

Results are segments of polygons assigned with the original obstacles, while some of them are replaced with curvature (a very simple example can be seen in figure 8-c). The true hypothesis is that the calculated ICO with this algorithm is geometrical, because the curvature segments (substitution of the ex-edges) are given by known circle-equations with start and end angular constraints, and the linear segments are represented by line-equations. These lines are parallel to the segments of the original obstacles meanwhile keeping distance of \( H = H(a, v, \theta, \phi, L) \). More geometrically, if we calculate transformation of all \( P \_i \)s according to the scaler \( H \) – while having the slope of the original lines, we implicitly know the new line equations.

**IV. Evaluation and Conclusion**

In this paper we introduced and investigated the T-Bone Algorithm as a new low-complexity approach for calculating Inevitable Collision Obstacle regions. The highlight of this approach is returning geometrical results to represent the ICO. Generating the geometrical or so to say mathematical model for deformed (or always enlarged) obstacle representation, is built on the concept of approximation which reduces the burden of calculations meanwhile keeping the certainty of the validity of the solution still positive. Therefore, the overall performance is increased in comparison with the classical solutions. Besides, having this formulated representation in hand to apply it into the control unit or navigation/motion-commands-generator of a car like vehicle, makes the calculations much faster and easier, because there is no need to process all possible cases of the configuration space, point by point. This fact makes the solution suitable for real time applications.

A simulation of the given algorithm on a fixed work space of 200 by 200 units, while having a changing \( C_s \), shows that the time complexity of the algorithm is linear and only depends on the number of obstacles’s corners. In our test cases, the two variables of T-Bone are set randomly to be \( 1 < v < 10 \) and \( 10 < \theta < 170 \). Number of obstacles are increased, one per each time-step (up to 100 obstacles). They are in a square shape, with a random edge size of less than 8 units, such that each new generated obstacle has no overlap with the available ones (finally the coverage percentage reached \( \%4.45 \) for 100 obstacles). The boundaries of the workspace are also considered as obstacles during the simulations. Figure 9 indicates that largeness of the obstacles’ area has no influence on the performance of the algorithm.

![Fig. 8. A very simple example of a rectangular obstacle when deformed by the algorithm:](image)
Fig. 9. X-axis shows number of obstacle with random number of vertices of 3 or 4 and a random edge size of smaller than 8 units, incrementally added into a work space of size 200 by 200 units. Obstacles are places such that avoid any overlap with formerly fixed ones. The Dubins car is of size 1 by 2 units. The time performance of the algorithm (dark trend) behaves linearly while the covered area by the obstacles are increasing in size (1780.57 units for 100 obstacles of variant surface size. Dotted graph shows the area covered by the last added obstacle. The whole covered part of work space is squared units is equal to the integral of the light dotted graph.

APPENDIX

There are some typical examples after being deformed by the T-Bone algorithm. Simple cases are presented to compare the contribution of different elements of the function $H$ in deforming a simple workspace of 3 obstacles.

ACKNOWLEDGMENT

Authors would like to be thankful to the insights on the problem driven by Prof. Stefano Carpin, as well as constructive discussions and feedbacks of Gorkem Erinc and Dr. Kaustubh Pathak.

REFERENCES