sensors

MDPI

Article

ITC: Infused Tangential Curves for Smooth 2D and 3D Navigation of Mobile Robots †

Abhijeet Ravankar [1,*,‡], Ankit A. Ravankar [2,‡], Arpit Rawankar [3], Yohei Hoshino [1] and Yukinori Kobayashi [2]

1 School of Regional Innovation and Social Design Engineering, Faculty of Engineering, Kitami Institute of Technology, Kitami, Hokkaido 090-8507, Japan; hoshinoy@mail.kitami-it.ac.jp
2 Division of Human Mechanical Systems and Design, Faculty of Engg., Hokkaido University, Sapporo, Hokkaido 060-8628, Japan; ankit@eng.hokudai.ac.jp (A.A.R.); kobay@eng.hokudai.ac.jp (Y.K.)
3 Department of Electronics and Telecommunication, Vidyalankar Institute of Technology, Mumbai 400037, India; arpit.rawankar@vit.edu.in
* Correspondence: aravankar@mail.kitami-it.ac.jp
† This paper is an expanded version of Ravankar, A.; Ravankar, A.A.; Kobayashi, Y.; Emaru, T. Real-Time Path Smoothing for Mobile Robots in 2D and 3D Environments. In Proceedings of the 2018 JSME Annual Conference on Robotics and Mechatronics (Robomec), Kitakyushu, Japan, 2–5 June 2018.
‡ These authors contributed equally to this work.

Received: 27 August 2019; Accepted: 7 October 2019; Published: 10 October 2019

Abstract: Navigation is an indispensable component of ground and aerial mobile robots. Although there is a plethora of path planning algorithms, most of them generate paths that are not smooth and have angular turns. In many cases, it is not feasible for the robots to execute these sharp turns, and a smooth trajectory is desired. We present 'ITC: Infused Tangential Curves' which can generate smooth trajectories for mobile robots. The main characteristics of the proposed ITC algorithm are: (1) The curves are tangential to the path, thus maintaining G^1 continuity, (2) The curves are infused in the original global path to smooth out the turns, (3) The straight segments of the global path are kept straight and only the sharp turns are smoothed, (4) Safety is embedded in the ITC trajectories and robots are guaranteed to maintain a safe distance from the obstacles, (5) The curvature of ITC curves can easily be controlled and smooth trajectories can be generated in real-time, (6) The ITC algorithm smooths the global path on a part-by-part basis thus local smoothing at one point does not affect the global path. We compare the proposed ITC algorithm with traditional interpolation based trajectory smoothing algorithms. Results show that, in case of mobile navigation in narrow corridors, ITC paths maintain a safe distance from both walls, and are easy to generate in real-time. We test the algorithm in complex scenarios to generate curves of different curvatures, while maintaining different safety thresholds from obstacles in vicinity. We mathematically discuss smooth trajectory generation for both 2D navigation of ground robots, and 3D navigation of aerial robots. We also test the algorithm in real environments with actual robots in a complex scenario of multi-robot collision avoidance. Results show that the ITC algorithm can be generated quickly and is suitable for real-world scenarios of collision avoidance in narrow corridors.

Keywords: robot path smoothing; robot navigation; safe navigation; multi-robot navigation; collision avoidance

1. Introduction

It is well understood that, in the near future, mobile robots will replace many works currently done manually by people. These is mostly dull (moving stuff in warehouse), dangerous (handling hazardous materials), and demanding (lifting heavy items) work. In order ro set the scene for the

paper, we take an example of a mobile item dispatch robot whose task is to carry items from one location to another location in an indoor environment. The environment has a lot of obstacles like furniture, walls, moving people, and other robots. The robot has a map of the environment in which the static obstacles and free navigational space are indicated. The robot must avoid collision with these static and dynamic obstacles. For this purpose, mobile robots have a path planning module to generate collision free trajectories from the start to the goal location.

Path planning is a two-step process [1]. The item dispatch robot would first plan a global path from the start to the goal location. At this stage, total path length is a dominating factor, and only the static obstacles are considered while making the path. Once the global path is generated, the item dispatch robot starts navigating on it. When the robot finds a dynamic obstacle, it should change its trajectory to avoid collision. This is done by the second stage of local planning. The local planner alters the trajectory to avoid collision with obstacles.

However, many path planners have a drawback that the path generated has sharp turns. This is undesired as it might be kinematically infeasible for the robot to execute these sharp turns without stopping or considerably reducing the speed. In addition, the robot's maneuver on the paths with sharp turns is not natural for people in the vicinity, and they may fail to anticipate the robot's path, which could lead to collision. A person walking behind the robot may dash into it if the robot suddenly stops to execute a sharp turn. Therefore, smooth paths are desired for robot motion. A robot can navigate the smooth paths at a constant velocity without completely stopping. The smooth paths are natural, and they are safe for the carried payload. Hence, path smoothing is important for safe robot navigation. Path smoothing is even more useful for applications like autonomous robotic wheelchairs to carry patients and disabled people.

We present an algorithm that can smooth the sharp turns of any traditional path planners while ensuring navigational safety. Our approach induces smooth curves thay are tangential to the original path. This tangentiality is important to enable robot navigation at a constant speed. It avoids any jerks or kinks in the path. We optimize the algorithm to generate these smooth induced curves for real-time applications. We mathematically formulate the problem and provide solutions for both 2D and 3D cases. The 2D case applies to UGVs (Unmanned Ground Vehicles), and the 3D case applies to UAVs (Unmanned Aerial Vehicles). We consider a complex distribution of obstacles and complicated maneuvers while discussing the proposed smoothing algorithm. In addition, we verify the accuracy and applicability of the proposed algorithm by providing results in both simulation and actual multi-robots in real environments, and comparing our results with the traditional algorithms.

Researchers have timely tested the existing path planning algorithms in various scenarios. Moreover, researchers have optimized these algorithms to meet the time constraints of real world applications. Therefore, the proposed work is not meant to replace already existing path planning algorithms. Instead, we propose the algorithm as a 'smoothing extension' for already existing algorithms to smooth out the sharp turns.

This is an extension of our previous work [2]. We thoroughly improve the previous work through mathematical formations, proofs, safe navigation, generating trajectories of specific curvatures, more experiments in both simulation and real world scenarios, and a detailed analysis. We summarize the main characteristics and novel contributions of this work as follows:

1. The proposed ITC algorithm generates smooth curves that are tangential to the original path. Thus, G^1 continuity is always guaranteed.
2. The smooth curves are infused in the original global path to smooth out the turns. Thus, the original global path is kept intact.
3. The straight segments of the global path are kept straight and only the sharp turns are smoothed. This is advantageous to keep a safe distance from the walls while navigating a narrow corridor.
4. Safety is embedded in the ITC trajectories, and robots are guaranteed to maintain a safe distance from the obstacles.

5. The curvature of ITC curves can easily be controlled and smooth trajectories can be generated fast in real-time.
6. The ITC algorithm smooths the global path on a part-by-part basis thus local smoothing at one point does not affect the global path.
7. The path smoothing is possible for both 2D navigation of ground vehicles, and 3D navigation of aerial robots.
8. We present ITC as a smoothing extension that can work in conjunction with any of the traditional path planners.

Related Works

The literature is full of global and local planning of mobile robots. The most widely used algorithms for global planning are: A* [3], D* [4,5], potential fields [6], Probabilistic Roadmap Planner (PRM) [7], rapidly exploring random tree (RRT) [8–10], and Dijkstra's algorithm [11–13], among many others. For local planning, the TEB (Timed-Elastic Band) planner [14] is worth mentioning. Many algorithms for cooperative multi-robot path planning have also been proposed [15,16]. Obstacle avoidance is an integral part of path planning and various approaches using visibility binary trees [17], inter-robot communication [18,19], path sharing [20], caching [21], and bio-inspired algorithms [22,23] have been proposed.

In recent years, with the proliferation of mobile robots and middle-ware robotics software like ROS (Robot Operating System), path smoothing has gained attention. Over the years, researchers have proposed to use geometric curves for smoothing paths. This includes approaches using clothoids [24,25], circle [26–29], splines (B-spline [30], intrinsic splines [31], quintic G^2 splines [32], non-uniform rational B-spline [33–35]), Bezier curve [36–38], hypocycloid [39,40], and other geometric curves. Path smoothing using interpolation [41] is also popular. Other researchers have used optimization techniques [42–47] to smooth robots' paths. Among optimization based planners, 'Time Elastic Band' planner (TEB Planner) [14,48–51] is widely used. For an extensive review of various path smoothing algorithms, we direct the readers to a review of previous works related to path smoothing [1] that extensively compares the merits, demerits, and challenges of the various path smoothing approaches.

A map is a prerequisite for path planning. A map is generated by using any of the SLAM (Simultaneous Localization and Mapping) algorithms available in the literature (see [52–54]). A SLAM module basically builds a map of the environment, and simultaneously localizes the robot in it. However, if the map has already been generated, it can be reused and the robot just needs to localize itself in it by matching sensor data using different algorithms [52,55]. Most commonly used sensors are Lidar, camera, and inertial sensors like IMU (Inertial Measurement Units). In this paper, it is assumed that there are sensors attached to the robot to avoid obstacles, and localize the robot in the map that has been generated already.

2. Induced Tangential Curves: 2D Path Smoothing Case

This section discusses the path smoothing for the 2D case. The 3D path smoothing case is discussed in Section 5.

Most of the traditional path planning algorithms generate a path which has many sharp and angular turns. Let us assume that the path ABCD (shown in black in Figure 1) is a section of the robot's path with a sharp turn at point B, and point C. The coordinates of the point A (x_1, y_1), B (x_2, y_2), C (x_3, y_3), and D (x_4, y_4) are also shown in Figure 1a. The obstacle is shown in gray color. The sharp turns at points B and C needs to be smoothed out.

First, we find two points P_1 and P_2 such that the line joining the two points $\overline{P_1P_2}$ is at a safe threshold distance (δ_{thresh}) from the obstacle. To find these points, we find intermediate points P_1 on the line $\overline{BA}$ at a fixed small distance from the point B. We fix another intermediate point P_2 on the line $\overline{BC}$ at the same fixed distance from the point B, i.e., $\overline{BP_1} = \overline{BP_2}$. We define this process of finding

intermediate points from the turn-point as 'diffusion'. The turn point B is diffused into points P_1 and P_2 along the directions $\overline{BA}$ and $\overline{BC}$, respectively.

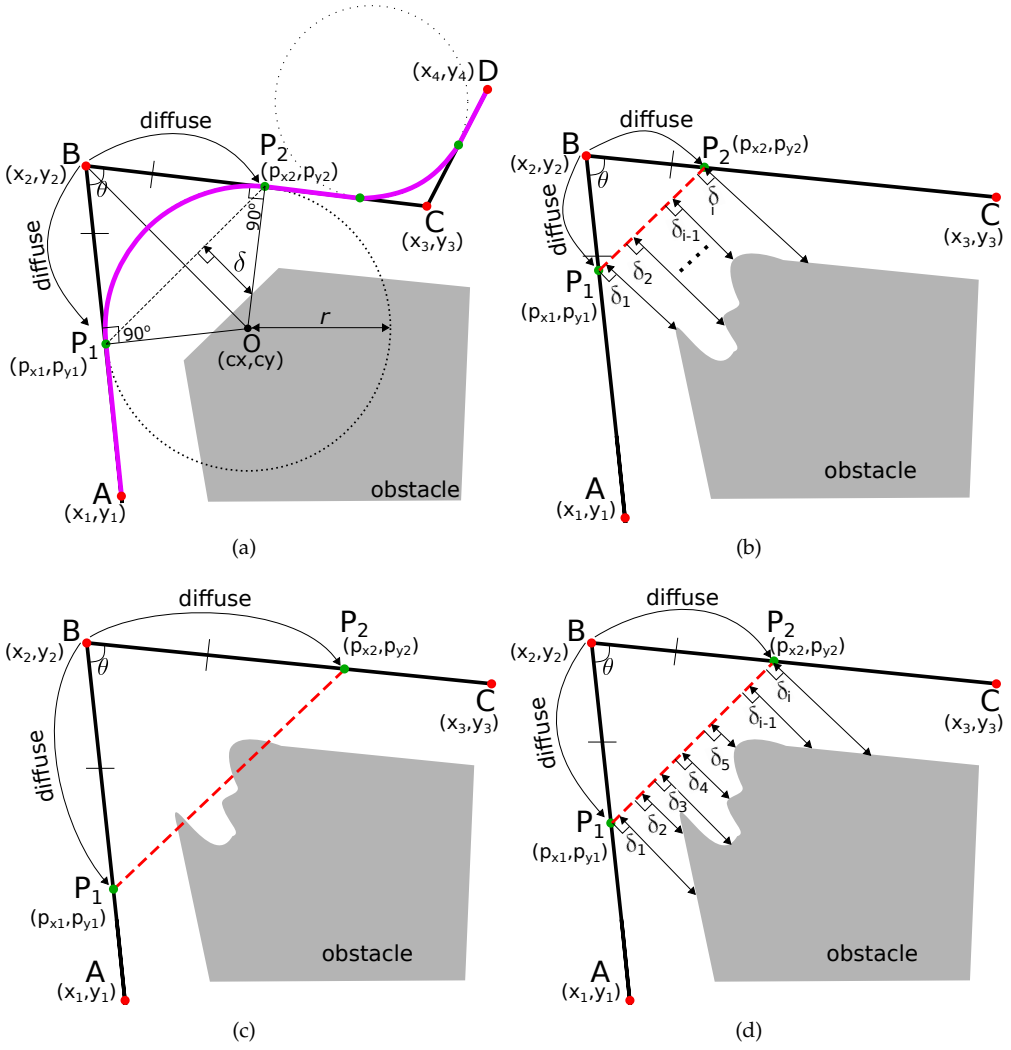


Figure 1. Diffusion in ITC (Infused Tangential Curves) based smoothing. (**a**) ITC curve $\overset{\frown}{P_1P_2}$ is induced in the original path and tangential at points P_1 and P_2; (**b**) under diffusion of point B; (**c**) over diffusion of point B; (**d**) appropriate diffusion of point B.

The turn point B is diffused into points P_1 and P_2 along the directions $\overline{BA}$ and $\overline{BC}$, such that the line joining the two points $\overline{P_1P_2}$ is at a safe threshold distance (δ_{thresh}) from the obstacle. If $\chi = \{\delta_1, \delta_2, \cdots, \delta_n\}$ is a set of perpendicular distances from line P_1P_2 to the obstacle at different points on the line, then there are three cases for such diffusion.

1. **Under Diffusion**: This case is shown in Figure 1b. The line $\overline{P_1P_2}$ is too far from the obstacles and the minimum perpendicular distance from the line is greater than the minimum safety threshold distance, or $\min_{\delta_i \in \chi} > \delta_{\text{thresh}}$.
2. **Over Diffusion**: This case is shown in Figure 1c. The line $\overline{P_1P_2}$ cuts through the obstacle.
3. **Appropriate Diffusion**: This case is shown in Figure 1d. The line $\overline{P_1P_2}$ is at an appropriate distance δ_{thresh} from the obstacle. To facilitate programming, we define an ϵ

$$\min_{\delta_i \in \chi} = \delta_{\text{thresh}} \pm \epsilon. \tag{1}$$

The parameter ϵ enables adjusting the threshold distance considering the width of the robot.

To find the minimum distance of the line $\overline{P_1P_2}$ from the obstacle ($\min_{\delta_i \in \chi}$), we discretize the line $\overline{P_1P_2}$ into several intermediate points (x_t, y_t) separated by a small distance Δ, as shown in Figure 2. We then estimate the distance of the obstacle from each intermediate point (x_t^i, y_t^i). The total points are

$$n_{\text{total\_pts}} = \frac{\sqrt{(p_{x2} - p_{x1})^2 + (p_{y2} - p_{y1})^2}}{\Delta}. \tag{2}$$

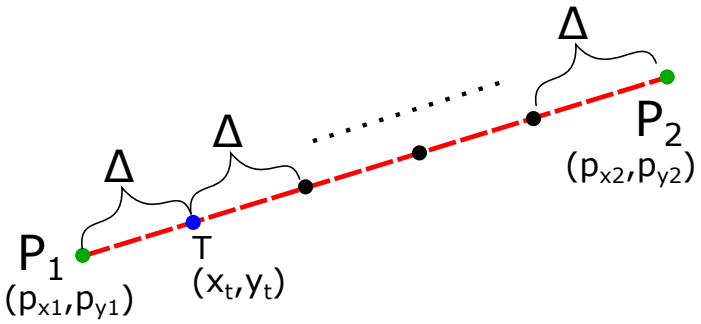


Figure 2. Discretizing the line $\overline{P_1P_2}$ into points x_t^i, y_t^i separated by Δ distance.

To find the intermediate point 'T' (x_t^i, y_t^i) at the distance Δ from P_1, we first find the unit vector $(\hat{\mathbf{u}})$ from the point P_1 to P_2. The unit vector is

$$\hat{\mathbf{u}} = \frac{p_{x2} - p_{x1}}{\sqrt{(p_{x2} - p_{x1})^2 + (p_{y2} - p_{y1})^2}}\hat{\mathbf{x}} + \frac{p_{y2} - p_{y1}}{\sqrt{(p_{x2} - p_{x1})^2 + (p_{y2} - p_{y1})^2}}\hat{\mathbf{y}}, \tag{3}$$

where $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ are the unit vectors in the x and y directions. Point $T(x_t^i, y_t^i)$ at a distance Δ from the point P_1 along the line $\overline{P_1P_2}$ is

$$\overrightarrow{T} = \overrightarrow{P_1} + d\hat{\mathbf{u}}. \tag{4}$$

Splitting up the respective x and y components gives

$$\begin{aligned} x_t &= p_{x1} + \frac{\Delta}{\sqrt{(p_{x2} - p_{x1})^2 + (p_{y2} - p_{y1})^2}}(p_{x2} - p_{x1}), \\ t_t &= p_{y1} + \frac{\Delta}{\sqrt{(p_{x2} - p_{x1})^2 + (p_{y2} - p_{y1})^2}}(p_{y2} - p_{y1}). \end{aligned} \tag{5}$$

Thus, the intermediate points are calculated as

$$\begin{aligned} x_t^i &= p_{x1} + \frac{\Delta \cdot i}{\sqrt{(p_{x2} - p_{x1})^2 + (p_{y2} - p_{y1})^2}}(p_{x2} - p_{x1}), \quad i \in \{1, 2, \cdots, n\} \text{ and } x_t^i \leq p_{x2}, \\ y_t^i &= p_{y1} + \frac{\Delta \cdot i}{\sqrt{(p_{x2} - p_{x1})^2 + (p_{y2} - p_{y1})^2}}(p_{y2} - p_{y1}), \quad i \in \{1, 2, \cdots, n\} \text{ and } y_t^i \leq p_{y2}. \end{aligned} \tag{6}$$

We briefly explain the calculation of the minimum distance $\min_{\delta_i \in \chi}$ from each intermediate point (x_t^i, y_t^i). The slope of the line $\overline{P_1P_2}$ is

$$m_{p_1p_2} = \frac{p_{y2} - p_{y1}}{p_{x2} - p_{x1}}. \tag{7}$$

The slope of line perpendicular to the line $\overline{P_1P_2}$ is

$$m_{\perp} = \frac{-1}{m_{p_1p_2}}. \tag{8}$$

Using Equation (6), we generate points on the line from each point (x_t^i, y_t^i) with slope $m_{\perp}$ in small units until the point touches the obstacle. Since the process is the same as explained above, we omit the explanation as it is straightforward.

Notice that Equation (6) is also used to diffuse the turn point B into intermediate points P_1 and P_2 along the directions $\overline{BA}$ and $\overline{BC}$ by the same distance Δ. Diffusion is stopped when appropriate points are found within the safe distance from the obstacle.

The two diffused points acts are points of contact of an induced smooth curve that is tangential at these points. In order to find the curve, we first find the circle and its radius whose arc will replace the angular path. The center of this circle is the point of intersection of the two perpendicular segments from these diffused points P_1 and P_2. The steps are explained below.

The slope of line $\overline{AB}$ in Figure 1a is

$$m_{AB} = \frac{y_2 - y_1}{x_2 - x_1}, \tag{9}$$

and the slope of line $\overline{BC}$ in Figure 1a is

$$m_{BC} = \frac{y_2 - y_3}{x_2 - x_3}. \tag{10}$$

In Figure 1a, the line $\overline{P_1O}$ is perpendicular to the line $\overline{AB}$ from point $P_1(p_{x1}, p_{y1})$. Hence, the slope of line $\overline{P_1O}$ is

$$m_{P_1O} = \frac{-1}{m_{AB}}. \tag{11}$$

Similarly, line $\overline{P_2O}$ is perpendicular to the line $\overline{BC}$ from point $P_2(p_{x2}, p_{y2})$, and the slope of line $\overline{P_2O}$ is

$$m_{P_2O} = \frac{-1}{m_{BC}}. \tag{12}$$

The point of intersection of lines $\overline{P_1O}$ and $\overline{P_2O}$ (point $O(c_x, c_y)$ in Figure 1a) is the center of the circle whose arc will define the smoothed path.

The general equation of a line of slope m passing through a point (x_1, y_1) is

$$\begin{aligned} y - y_1 &= m(x - x_1) \\ \implies y &= mx - mx_1 + y_1. \end{aligned} \tag{13}$$

Since the lines $\overline{P_1O}$ and $\overline{P_2O}$ intersect at $O(c_x, c_y)$,

$$\begin{aligned} c_y &= m_{P_1O}(c_x - p_{x1}) + p_{y1} = m_{P_2O}(c_x - p_{x2}) + p_{y2}, \\ &\implies m_{P_1O} \cdot c_x - m_{P_1O} \cdot p_{x1} + p_{y1} = m_{P_2O} \cdot c_x - m_{P_2O} \cdot p_{x2} + p_{y2} \\ &\implies m_{P_1O} \cdot c_x - m_{P_2O} \cdot c_x = m_{P_1O} \cdot p_{x1} - m_{P_2O} \cdot p_{x2} + p_{y2} - p_{y1} \\ &\implies c_x \cdot (m_{P_1O} - m_{P_2O}) = m_{P_1O} \cdot p_{x1} - m_{P_2O} \cdot p_{x2} + p_{y2} - p_{y1}. \end{aligned}$$

Thus, the x-coordinate of the center of the circle is

$$c_x = \frac{m_{P_1O} \cdot p_{x1} - m_{P_2O} \cdot p_{x2} + p_{y2} - p_{y1}}{(m_{P_1O} - m_{P_2O})}. \tag{14}$$

The y-coordinate of the center of the circle can be obtained by plugging the value of c_x in the equation of line $\overline{P_1O}$ or $\overline{P_2O}$,

$$c_y = m_{P_1O} \cdot c_x - m_{P_1O} \cdot p_{x1} + p_{y1}. \tag{15}$$

The radius of the circle r is

$$r = \sqrt{(c_x - p_{x1})^2 + (c_y - p_{y1})^2}, \quad \text{or} \quad r = \sqrt{(c_x - p_{x2})^2 + (c_y - p_{y2})^2}. \tag{16}$$

The circle with center (c_x, c_y) and radius r is shown in Figure 1a. We take the arc $\widehat{P_1P_2}$ of the circle between points P_1 and P_2 shown in magenta in Figure 1a. This curve $\widehat{P_1P_2}$ has the following properties:

- Curve $\widehat{P_1P_2}$ is tangential to the robot's original path. Hence, G^1 geometric continuity is guaranteed.
- Curve $\widehat{P_1P_2}$ is smoother to traverse compared to the original path of the robot. In Figure 1a, the original path of the robot is ABC with sharp turn at the point B. The smooth path is $\widehat{AP_1P_2C}$.
- Curve $\widehat{P_1P_2}$ is 'infused' inside the original path of the robot.

Due to the three properties above, $\widehat{P_1P_2}$ is called an 'Infused Tangential Curve'.

3. Accelerating ITC Path Smoothing Algorithm

The smoothing algorithm is accelerated on two fronts. First, we accelerate the diffusion of the point of sharp turn. Second, we also accelerate the algorithm to estimate the minimum distance from the obstacles. These are explained below.

3.1. Accelerating the Diffusion Algorithm

Figure 3a shows the normal approach of diffusing the node B (x_2, y_2) into points P_1 and P_2. The normal algorithm diffuses the point into small increments of Δ_{diff}. As shown in Figure 3a, if P_1 (p_{x1}, p_{y1}) and P_2 (p_{x2}, p_{y2}) are the appropriate diffused positions which maintain a safe threshold distance (δ_{thresh}) from the obstacles, the total steps n_{diff} are

$$n_{\text{diff}} = \frac{\sqrt{(p_{x2} - x_2)^2 + (p_{y2} - y_2)^2}}{\Delta_{\text{diff}}}. \tag{17}$$

Hence, normal diffusion algorithm would have a complexity of $\mathcal{O}(n_{\text{diff}})$.

We accelerate the algorithm using binary search. The idea is shown in Figure 3b, and it is used for explanation. In the binary search, the first diffusion of point B (x_2, y_2) occurs at the maximum distance at point C (x_3, y_3). This is shown by a magenta-colored arrow in Figure 3b. A check is performed if the line joining the diffused points i.e., $\overline{AC}$ maintains a safe threshold from the obstacles. As shown in Figure 3b, the line $\overline{AC}$ crosses over the obstacle, and it is a case of 'over-diffusion'. Therefore, in step 2, the diffusion distance is half the distance of the previous diffusion (i.e., $\frac{\sqrt{(x_3-x_2)^2+(y_3-y_2)^2}}{2}$). This is shown by a blue arrow in Figure 3b, and the safety threshold is checked again. This is again a case of over-diffusion, so, in step 3, the new diffusion distance is half the distance of the previous diffusion (i.e., $\frac{\sqrt{(x_3-x_2)^2+(y_3-y_2)^2}}{4}$) and shown with a green arrow. A check for safety clearance is performed, and it is determined to be a case of 'under-diffusion'. Due to this, in the next step 4, the diffusion is increased by half of the previous distance (i.e., $\frac{3\cdot\sqrt{(x_3-x_2)^2+(y_3-y_2)^2}}{8}$). This process is repeated until the appropriate diffusion point (P_2) has been found.

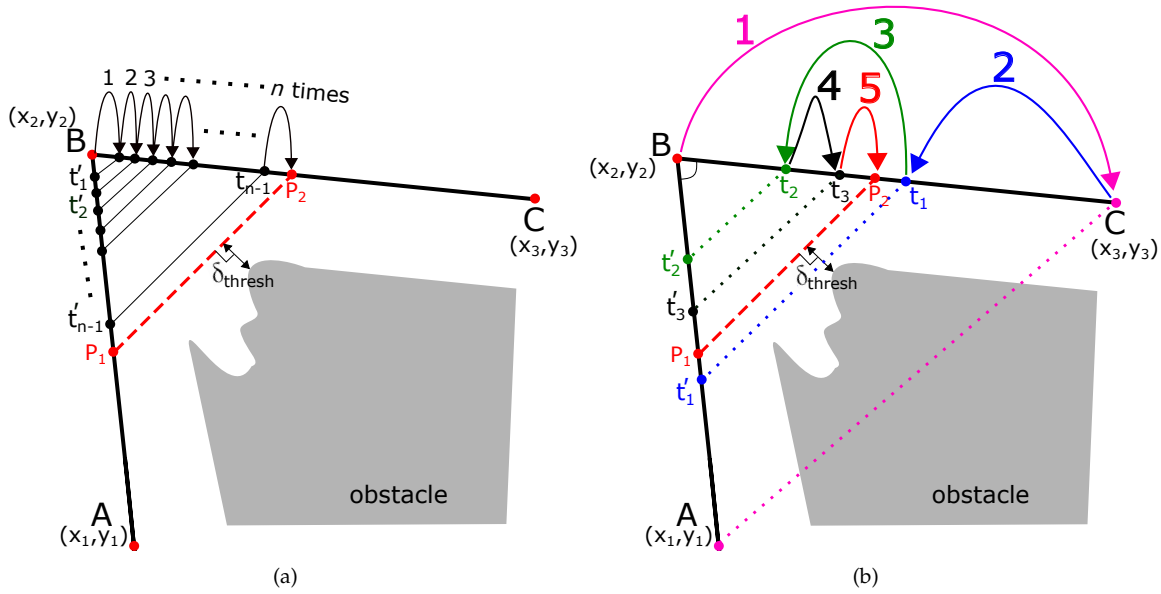


Figure 3. Normal and accelerated diffusion of point B. (**a**) normal diffusion; (**b**) accelerated diffusion based on binary search.

Compared to the normal process, the accelerated algorithm takes $\log(n_{\text{diff}})$ steps, and the complexity of the algorithm is $\mathcal{O}(\log n_{\text{diff}})$. The pseudo-code is given in Algorihtm 1.

Algorithm 1: Fast Algorithm to Find Diffusion Point

Data: Point1 (x_1, y_1), Point2 (x_2, y_2), Point3 (x_3, y_3), M : Map (Refer Figure 3b)
Result: Diffusion Point (x_d, y_d) which maintains appropriate clearance δ_{thresh} (Refer Figure 3b)

Function get_diffused_point(Point1, Point2, Point3, M)
  $x_2, y_2 \leftarrow$ Point2
  $x_3, y_3 \leftarrow$ Point3
  $x_L, y_L \leftarrow x_2, y_2$ // Left point
  $x_R, y_R \leftarrow x_3, y_3$ // Right point
  while left $\leq$ right **do**
    $\text{mid}_x \leftarrow \frac{x_L + x_R}{2}$
    $\text{mid}_y \leftarrow \frac{y_L + y_R}{2}$
    $c_d \leftarrow$ estimate_obstacle_clearance (Point1, Point2, Point3, M, mid_x, mid_y)
    if $c_d < \delta_{\text{thresh}}$ **then**
      `// Adjust the right point`
      $x_R \leftarrow \text{mid}_x - 1$
      $y_R \leftarrow \text{mid}_y - 1$
    if $c_d > \delta_{\text{thresh}}$ **then**
      `// Adjust the left point`
      $x_L \leftarrow \text{mid}_x + 1$
      $y_L \leftarrow \text{mid}_y + 1$
    if $c_d = \delta_{\text{thresh}} \pm \epsilon$ **then**
      `// Return the point`
      $x_d \leftarrow \text{mid}_x$
      $y_d \leftarrow \text{mid}_y$
      return(x_d, y_d)
  return False

3.2. Accelerating the Minimum Distance Calculation

Similarly, we also accelerate the perpendicular distance estimation using the binary search algorithm.

Figure 4a shows the normal incremental approach of finding the perpendicular distance from the obstacle on the line with starting point (x, y) and slope $m_\perp$. The normal algorithm diffuses the point into small increments of $\Delta_\perp$. As shown in Figure 4a, the total steps $n_{\text{clearance}}$ are

$$n_{\text{clearance}} = \frac{\text{Perpendicular distance to obstacle}}{\Delta_\perp}. \tag{18}$$

Hence, the algorithm has a complexity of $\mathcal{O}(n_{\text{clearance}})$.

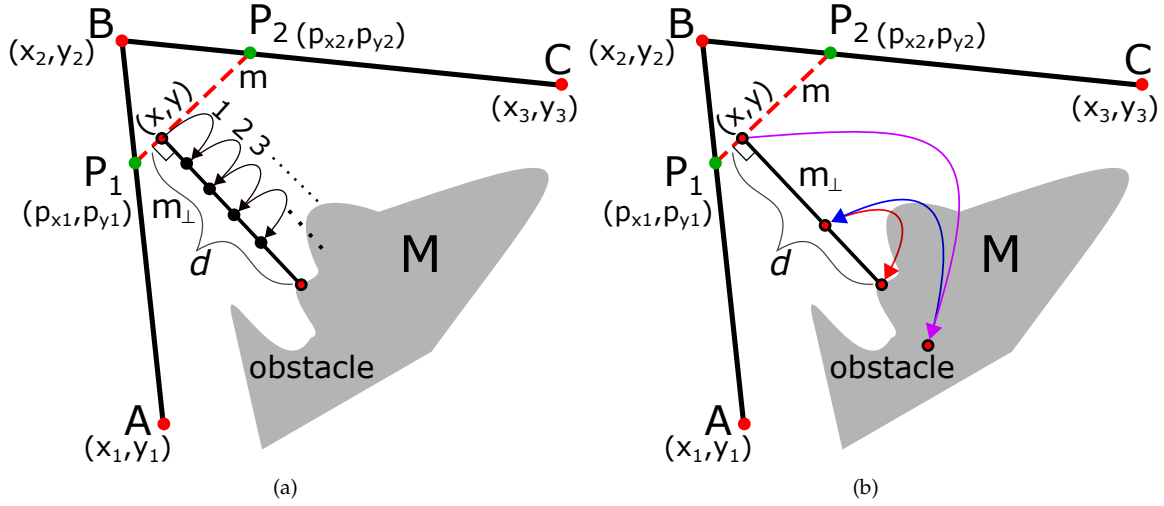


Figure 4. Normal and accelerated calculate of minimum distance from the obstacle. (**a**) normal approach; (**b**) accelerated approach.

The accelerated algorithm using binary search is explained in Figure 4. Since the idea is similar to that explained in the previous section, we omit explanation, and the complexity of the algorithm is $\mathcal{O}(\log n_{\text{clearance}})$. The pseudo-code is given in Algorithm 2.

The values of n_{diff}, $n_{\text{clearance}}$, and $n_{\text{total\_pts}}$ are given in Equations (2), (17), and (18), respectively. Compared to the incremental algorithm, the overall speedup is

$$\text{speedup} = \frac{n_{\text{diff}} \times n_{\text{clearance}} \times n_{\text{total\_pts}}}{\log\left(n_{\text{diff}}\right) \times \log(n_{\text{clearance}}) \times n_{\text{total\_pts}}}. \tag{19}$$

Algorithm 2: Estimate Obstacle Clearance

Data: Point1 (x_1, y_1), Point2 (x_2, y_2), M : Map
Result: Diffusion Point (x_d, y_d) which maintains appropriate clearance δ_{thresh}

Function estimate_obstacle_clearance (Point1, Point2, Point3, M, mid_{x1}, mid_{y1})
  $x_1, y_1 \leftarrow$ Point1
  $x_2, y_2 \leftarrow$ Point2 // Turn point 1
  $x_3, y_3 \leftarrow$ Point3
  $dist_{diffuse} \leftarrow \sqrt{(x_2 - mid_{x1})^2 + (y_2 - mid_{y1})^2}$ // Distance between Point2 and (mid_{x1}, mid_{y1}) diffused pt.
  $dist_{p2p1} \leftarrow \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$ // Distance between Point2 and Point1
  $mid_{x2} \leftarrow x_2 + \frac{dist_{diffuse}}{dist_{p2p1}}(x_2 - x_1)$ // Turn point 2
  $mid_{y2} \leftarrow y_2 + \frac{dist_{diffuse}}{dist_{p2p1}}(x_2 - x_1)$ // Turn point 2
  $d_{total} \leftarrow \sqrt{(mid_{x2} - mid_{x1})^2 + (mid_{y2} - mid_{y1})^2}$ // Distance between (mid_{x2}, mid_{y2}) and (mid_{x1}, mid_{y1}).
  $m_{l2} \leftarrow \frac{mid_{y2} - mid_{y1}}{mid_{x2} - mid_{x1}}$ // Slope of line joining mid-points
  $m_{\perp} \leftarrow \frac{-1}{m_{l2}}$ // Slope of perpendicular line
  $i \leftarrow 0$
  $d_{min} \leftarrow (-1)$
  while $i < \frac{d_{total}}{\Delta}$ **do**
    $x \leftarrow mid_{x1} + \frac{\Delta \cdot i}{d_{total}}(mid_{x1} - mid_{x2})$ // x-coordinate of Point i
    $y \leftarrow mid_{y1} + \frac{\Delta \cdot i}{d_{total}}(mid_{y1} - mid_{y2})$ // y-coordinate of Point i
    // If the point (x,y) touches obstacle, return (-1).
    // In the Grid-Map, obstacle value is 1, free-space is 0, un-defined value is 0.5
    if $M[x][y] = 1$ **then**
      return (-1)
    else
      // Estimate length of line from point (x,y) with slope $m_{\perp}$ to obstacles
      while Obstacle surface point not found **do**
        $(x_{obs}, y_{obs}) \leftarrow$ binary_search$(x, y, m_{\perp}, M)$ // Refer Figure 4b
      $d \leftarrow \sqrt{(x - x_{obs})^2 + (y - y_{obs})^2}$ **if** $d < d_{min}$ **then**
        $d_{min} \leftarrow d$
    $i \leftarrow i + 1$
  return (d_{min})

4. Robot's Proximity from Obstacles on the Smooth Path

In Figure 5, the original path ABCD of the robot with sharp turns at points B and C has been smoothed by the path $\widehat{AP_1P_2}$. Originally, the robot would make a turn at point B keeping a distance BQ from the obstacle. On the smooth path, the robot comes closer to the obstacle while traversing the smooth green curve, and the distance is B′Q. The proximity difference (shown as e in Figure 5) is $e = BB'$. Since the robot itself has some width, the overall proximity to the obstacle will be increased while traversing the smooth curve. Hence, it is important to calculate this difference in proximity.

In Figure 5, we calculate the $\angle ABC = \theta$ using the cosine law as

$$\theta = \cos^{-1}\left(\frac{a^2 + c^2 - b^2}{2ac}\right),$$
where,
$$a = \sqrt{(p_{x2} - x_2)^2 + (p_{y2} - y_2)^2}, \quad (20)$$
$$b = \sqrt{(p_{x2} - p_{x1})^2 + (p_{y2} - p_{y1})^2},$$
$$c = \sqrt{(p_{x1} - x_2)^2 + (p_{y1} - y_2)^2}.$$

In $\triangle P_1OB$ and $\triangle P_2OB$, we have

$$\begin{aligned}
&P_2B = P_1B \text{ (diffusion by same distance)}\\
&P_2O = P_1O \text{ (circle radius 'r')}\\
&BO = BO \text{ (common side)}\\
\therefore &\triangle P_1OB \cong \triangle P_2OB \text{ (Congruent triangles by SSS Congruence Theorem)}\\
\therefore &\angle P_1BO = \angle P_2BO \text{ (From property of congruent triangles)}\\
&\angle P_1BO + \angle P_2BO = \theta \text{ (Figure 5)}\\
\therefore &\angle P_1BO = \angle P_2BO = \frac{\theta}{2} \text{ (Figure 5).}
\end{aligned} \tag{21}$$

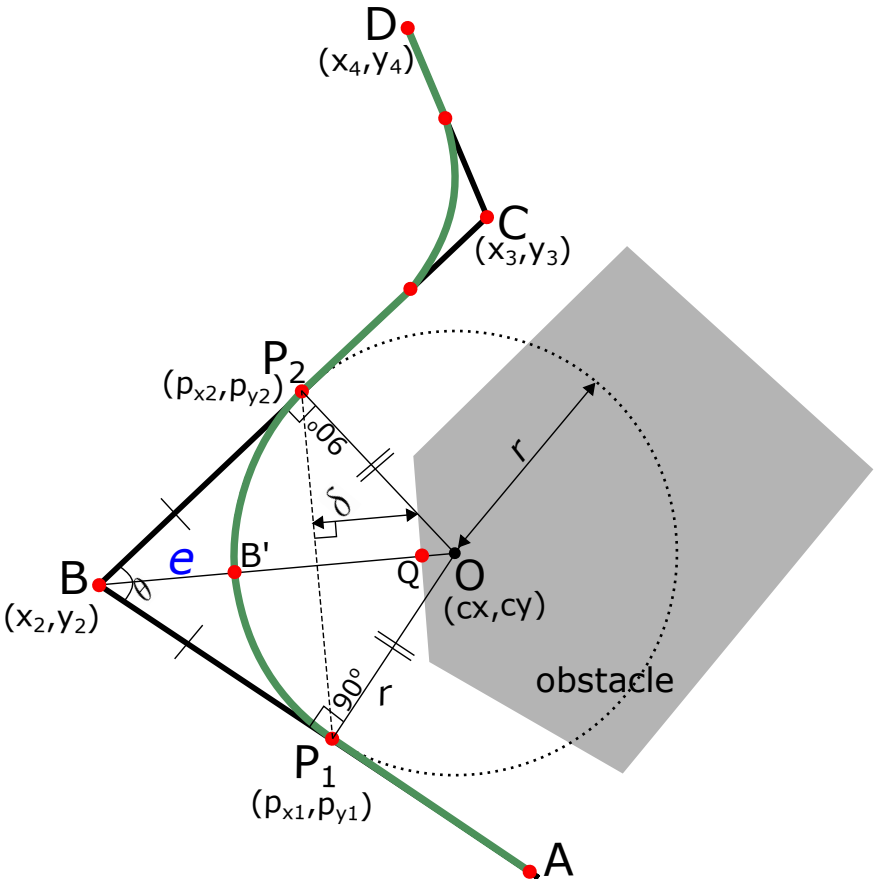


Figure 5. Calculation of robot's proximity from obstacles on the smooth path compared to the original path.

In other words, $\overline{BO}$ bisects $\angle P_1BP_2$, and $\angle P_1BO = \frac{\theta}{2}$. In $\triangle P_1OB$, point P_1 is the tangent point, therefore $\angle BP_1O = 90^\circ$. Therefore,

$$\sin\frac{\theta}{2} = \frac{P_1O}{BO} = \frac{P_1O}{BB' + B'O} = \frac{r}{e + r}. \tag{22}$$

Hence, the distance $e = \overline{BB'}$ is calculated as

$$\begin{aligned} e &= \frac{r}{\sin\frac{\theta}{2}} - r, \\ \text{or, } e &= r(\text{cosec}\frac{\theta}{2} - 1). \end{aligned} \tag{23}$$

Assuming that the robot has a width of W_r and it accurately traverses the curve, the proximity to the obstacles is increased by a distance $\frac{W_r}{2}$. If the proximity of the robot is less than the threshold distance δ_{thresh}, the diffusion points P_1 and P_2 are moved closer to the point B by a distance of $\frac{W_r}{2}$, and the curve is recomputed to ensure safety.

5. Induced Tangential Curves: 3D Path Smoothing Case

We now explain the case of 3D path smoothing for UAVs (Unmanned Aerial Vehicles) and drones. The overall idea for the 3D case is an extension of the 2D case. 'Node' or the point of turn is now a point in 3D space. First, the overall path is generated by using traditional path planners. Commonly used robot path planning algorithms like A* algorithm[3], D* algorithm[4,5], potential fields algorithm [6], Probabilistic Roadmap Planner (PRM) [7], rapidly exploring random tree algorithm (RRT) [8–10], etc. generally have 3D extensions. Hence, any of the traditional algorithms can be used to generate the overall path.

Figure 6 shows three points A′ (x_1', y_1', z_1') , B (x_2, y_2, z_2), and C′ (x_3', y_3', z_3') in 3D space. These points mark the original path of the robot. Point B is the point with sharp turn on the robot's path. As explained in the previous sections, point B has been diffused to points A and C which are equidistant from point B, i.e., $|AB| = |BC|$. The line joining the two 3D points $\overline{AC}$ is at a safe distance ($d = \delta_{\text{thresh}}$) from the obstacle in 3D space shown in green in Figure 6. The aim is to find the circle (shown in yellow in Figure 6).

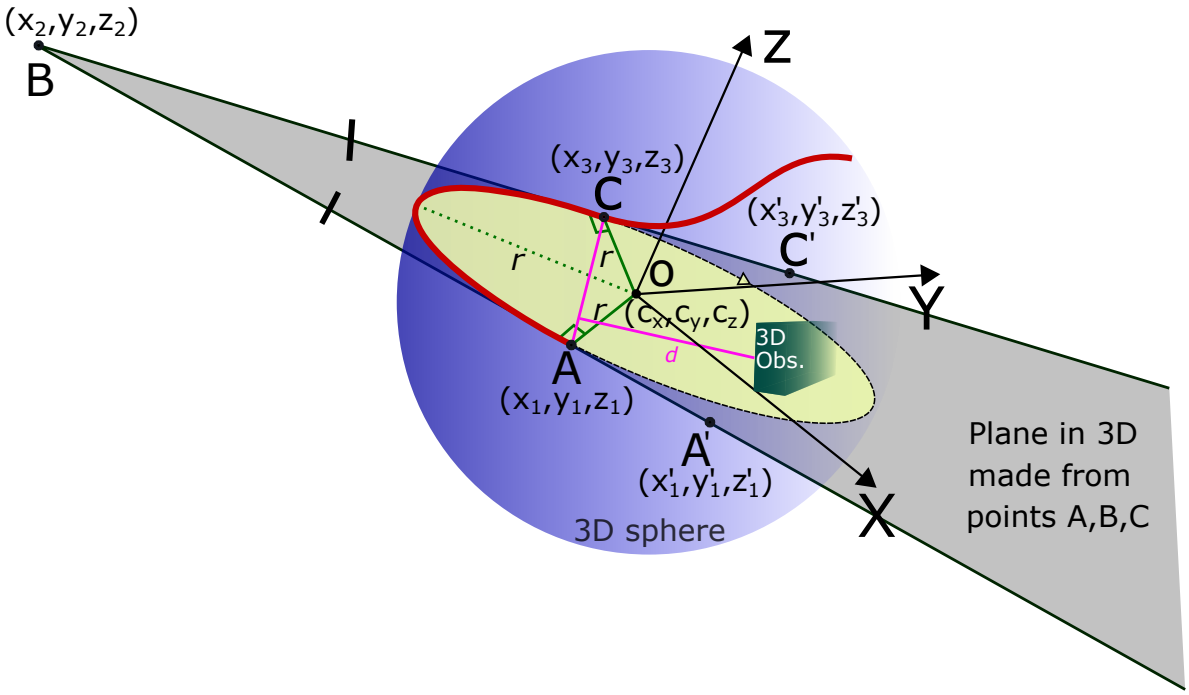


Figure 6. 3D path smoothing with ITC. The points A, B, and C are in 3D space. The intersection of the 3D sphere and the plane generated from the three points marks the arc to smooth the path.

The yellow circle in Figure 6 lies on the plane which is formed by the three points A (x_1, y_1, z_1), B (x_2, y_2, z_2), and C (x_3, y_3, z_3). The plane is shown in gray color.

The equation of the plane formed by three points is

$$\mathrm{A}x + \mathrm{B}y + \mathrm{C}z + \mathrm{D} = 0, \tag{24}$$

where

$$\mathrm{A} = \begin{bmatrix} 1 & y_1 & z_1 \\ 1 & y_2 & z_2 \\ 1 & y_3 & z_3 \end{bmatrix}, \mathrm{B} = \begin{vmatrix} x_1 & 1 & z_1 \\ x_2 & 1 & z_2 \\ x_3 & 1 & z_3 \end{vmatrix}, \mathrm{C} = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{bmatrix}$$

and

$$\mathrm{D} = \begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{bmatrix}.$$

The equation of line AB and BC is given as

$$(x_1, y_1, z_1) + t(x_2 - x_1, y_2 - y_1, z_2 - z_1), \tag{25}$$

and

$$(x_2, y_2, z_2) + t(x_3 - x_2, y_3 - y_2, z_3 - z_2), \tag{26}$$

respectively.

The normal of the plane is calculated by taking the cross product,

$$n = (\mathrm{A} - \mathrm{B}) \times (\mathrm{C} - \mathrm{B}). \tag{27}$$

n is the vector $(\mathrm{A}, \mathrm{B}, \mathrm{C})$ formed by three of the coefficients from the equation of the plane in Equation (24). The direction within the plane and perpendicular to AB is

$$v = n \times (\mathrm{A} - \mathrm{B}). \tag{28}$$

Similarly, the direction within the plane perpendicular to BC is

$$w = n \times (\mathrm{C} - \mathrm{B}). \tag{29}$$

A point on these lines can be represented as

$$\mathrm{A} + \lambda v \quad \text{and} \quad \mathrm{C} + \mu w. \tag{30}$$

The equation of a point which lies on both of them is

$$\mathrm{D} = \mathrm{A} + \lambda v = \mathrm{C} + \mu w. \tag{31}$$

We solve for λ and μ. There are actually three equations, one for each coordinate, and two variables, so the system is over-determined. We can use pseudo inverse to avoid special cases and get the center of the sphere. Once we have the center, we can calculate the radius (r) of the circle with center O (c_x, c_y, c_z),

$$r = \sqrt{(c_x - x_2)^2 + (c_y - y_2)^2 + (c_z - z_2)^2}. \tag{32}$$

As shown in Figure 6, the plane defined from the points A, B, and C cuts through the blue sphere. The boundary of the intersection of the sphere and the plane is a circle shown in yellow in Figure 6.

The circle can be described with a parametric description, for which we require two orthonormal vectors within the plane. One such vector is

$$e_1 = \frac{(\mathrm{A} - \mathrm{B})}{\|\mathrm{A} - \mathrm{B}\|}, \tag{33}$$

and the other is

$$e_2 = \frac{(n \times e_1)}{\|n \times e_1\|}. \tag{34}$$

Once the circle has been calculated, we take the arc $\overset{\frown}{\mathrm{AC}}$ to generate the smooth trajectory. The original path of the robot was A′BC′. The smooth path is $\overset{\frown}{\mathrm{A'ACC'}}$.

6. Experiments and Results

We now discuss the results of path smoothing using the proposed ITC algorithm. Section 6.1 first shows the 2D results. The 3D path smoothing results are shown in Section 6.2. Section 6.3 discusses the comparison with interpolation based smoothing algorithms. Section 6.4 shows results of smooth path generation with different curvatures. Section 6.5 shows smoothing results with actual robots in a real environment while discussing a multi-robot collision avoidance scenario.

6.1. Results of 2D Path Smoothing

We first discuss path smoothing in indoor corridor environments which are frequently navigated by mobile service robots. Such an environment is shown in Figure 7 in which the smooth paths are also shown. In such environments, robots generally navigate the center of the corridor maintaining a safe distance from the walls on both sides. On the other hand, some robots are programmed to navigate the corridors on the left or right side. We discuss these cases below.

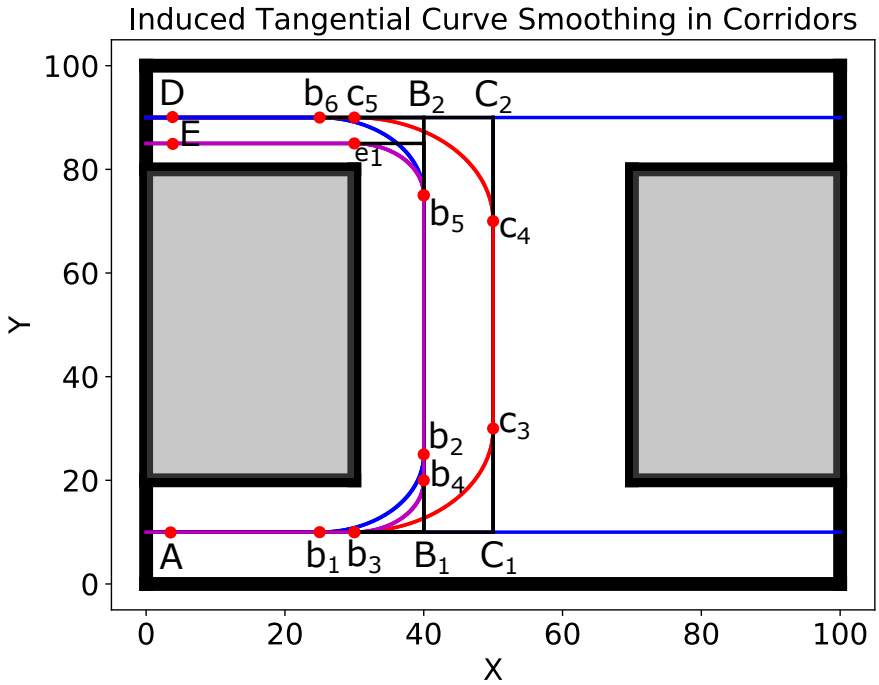


Figure 7. The case of path smoothing in corridors using ITC.

- Case I: As shown in Figure 7, if the robot is programmed to navigate the center of the corridors, the original path of the robot is $\mathrm{AC_1C_2D}$ with A as the starting point and D as the goal point.

The path has two 90-degree turns at points C_1 and C_2. Two smooth curves $\widehat{b_3c_3}$ and $\widehat{c_4c_5}$ have been induced in the original path. The smooth path is $\widehat{Ab_3c_3c_4c_5D}$.

- Case II: For 'left-traversal' robots, the original path of the robot is AB_1B_2D with A as the starting point and D as the goal point. This path also has two 90-degree turns at points B_1 and B_2. Two smooth curves $\widehat{b_1b_2}$ and $\widehat{b_5b_6}$ have been induced in the original path. The smooth path is $\widehat{Ab_1b_2b_5b_6D}$.

  Two other curves $\widehat{b_3b_4}$ and $\widehat{b_5e_1}$ have also been induced in the original path for different curvature requirements. In any case, all of the induced curves are tangential and guarantee G^1 continuity.

Figure 8a shows a complex scenario with many sharp turns of different angles. The original path of the robot is shown in black with the start point as P1 and the goal point as P14. There are sharp turns at points P2, P3, $\cdots$, P13. A major advantage of the proposed ITC algorithm is that it is possible to have a different safe threshold distance (δ_{thresh}) at different turn points based on several factors. In the simulation, we fixed different thresholds for different turn points as shown in Figure 8a. This is evident from the fact that the curvature of different induced curves ($\widehat{a_1a_2}$, $\widehat{b_1b_2}$, $\widehat{c_1c_2}$, $\cdots$, $\widehat{i_1i_2}$) shown in red in Figure 8a is different. For example, the curvature of the induced curve $\widehat{a_1a_2}$ is less than that of curve $\widehat{j_1j_2}$, as the safety threshold distance for the curve $\widehat{a_1a_2}$ ($\delta_{\text{thresh}} = 3$) was more than the safety threshold distance for the curve $\widehat{j_1j_2}$ ($\delta_{\text{thresh}} = 1$).

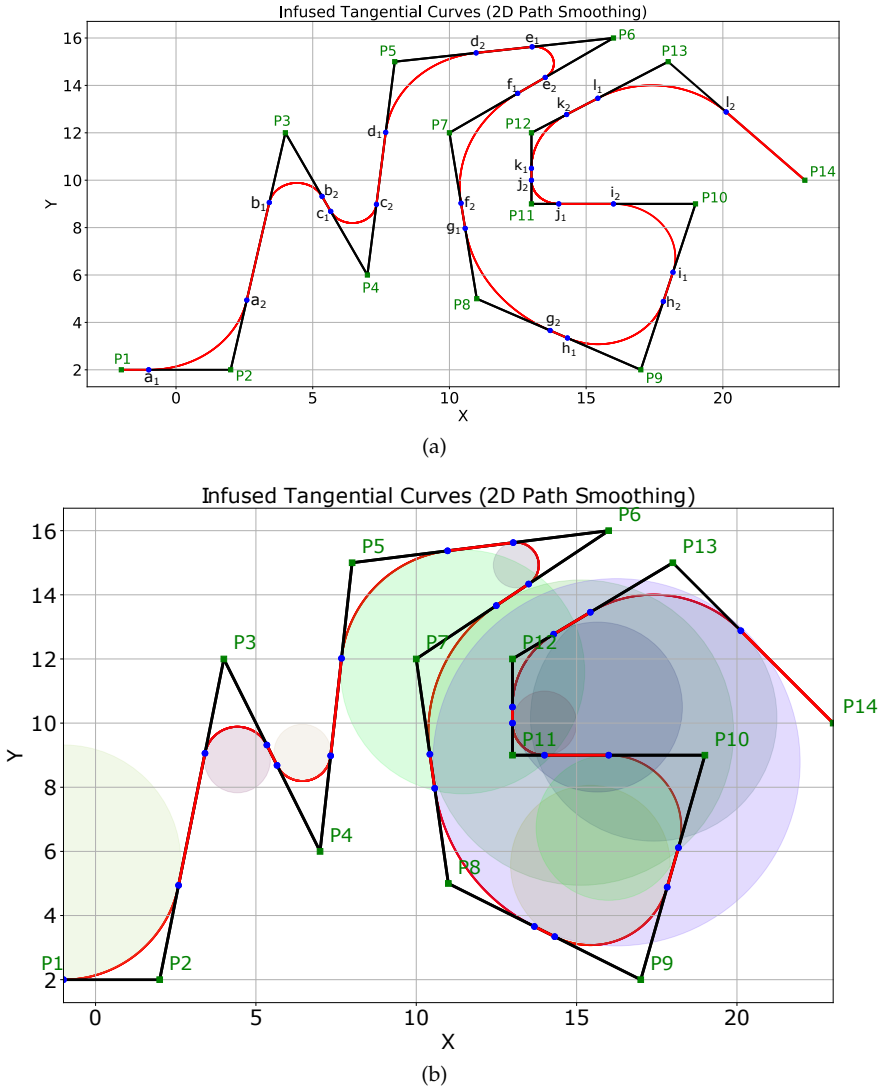
The curvatures of different curves can also be checked visually from Figure 8b, which shows the different circles whose segments are used for smoothing the turns. It is also visually evident that $\widehat{j_1j_2}$ and $\widehat{b_1b_2}$ are generated from arcs of circles with different radii, and hence different curvatures.

Table 1 summarizes the coordinates of the points and distance thresholds set for the different curves $\widehat{a_1a_2}$, $\widehat{b_1b_2}$, $\widehat{c_1c_2}$, $\cdots$, $\widehat{i_1i_2}$ shown in Figure 8a.

The threshold distances summarized in Table 1 should not be confused with the radius of the circles used to generate the tangential curves. The radii of different circles used to generate tangential curves at points P2, P3, $\cdots$, P13 of Figure 8 are shown in Figure 9a. Similarly, the curvature of the different ITC curves are shown in Figure 9b. The ITC curve at point P6 ($\widehat{e_1e_2}$) had the minimum radius of 0.7082 units and thereby the minimum curvature, whereas the ITC curve at point P8 ($\widehat{g_1g_2}$) had the maximum radius and thereby the minimum curvature.

Table 1. 2D coordinates of turn points in Figure 8a .

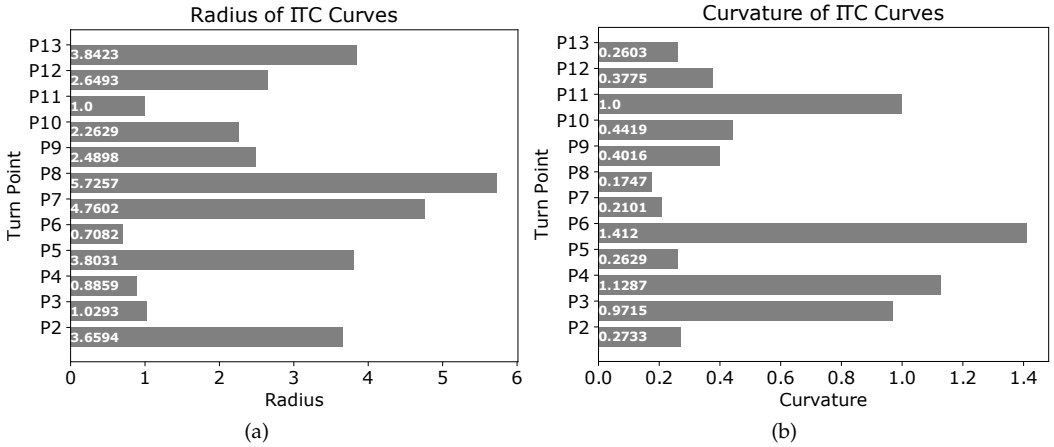
Coord	P-1	P-2	P-3	P-4	P-5	P-6	P-7	P-8	P-9	P-10	P-11	P-12	P-13	P-14
X	-2	2	4	7	8	16	10	11	17	19	13	13	18	23
Y	2	2	12	6	15	16	12	5	2	9	9	12	15	10
Threshold (δ): $\delta 1 = 3$, $\delta 2 = 3$, $\delta 3 = 3$, $\delta 4 = 3$, $\delta 5 = 3$, $\delta 6 = 3$, $\delta 7 = 3$, $\delta 8 = 3$, $\delta 9 = 3$, $\delta 10 = 1$, $\delta 11 = 1.5$, $\delta 12 = 3$														



(a)

(b)

Figure 8. 2D path smoothing using ITC in a complex scenario. (**a**) tangential curves $\widehat{a_1a_2}$, $\widehat{b_1b_2}$, $\widehat{c_1c_2}$, $\cdots$, $\widehat{i_1i_2}$ have been induced to smooth sharp turns with different δ_{thresh}; (**b**) visual representation of the different curves and their curvatures.



(a)

(b)

Figure 9. Radius and curvature of the different curves in Figure 8a. (**a**) radius of curves at different points; (**b**) curvature of curves at different points.

6.2. Results of 3D Path Smoothing

Figure 10 shows three complex scenarios of path smoothing for UAVs in 3D space. The original path of the UAV is shown in red. The smoothed paths are shown in black. The 3D points have been marked in green.

Figure 10a shows the first case of 3D path smoothing. The original path of the UAV is a closed-loop path with the same start and goal points ($x = y = z = 0$). Different values of threshold distances have been used to generate smooth curves of appropriate curvatures. The 3D coordinates of different points and different clearance thresholds have been summarized in Table 2.

Table 2. 3D coordinate of turn points in Figure 10a.

Coord	Point-1	Point-2	Point-3	Point-4	Point-5	Point-6	Point-7	Point-8	Point-9	Point-10
X	0	20	50	80	90	80	50	20	20	0
Y	0	10	20	20	50	80	80	60	40	0
Z	0	15	20	15	20	15	20	15	10	0
Threshold Distance (δ): $\delta 1 = 14$, $\delta 2 = 13$, $\delta 3 = 14$, $\delta 4 = 15$, $\delta 5 = 12$, $\delta 6 = 12$, $\delta 7 = 10$, $\delta 8 = 9$										

Similarly, Figure 10b also shows a closed path closed-loop UAV path with the same start and goal points ($x = y = z = 0$). Compared to the path in Figure 10a, the path is more complex with difficult maneuvers and sharp turns. However, the proposed method is still able to smooth the sharp and angular turns. Different values of threshold distances have been used to generate smooth curves of appropriate curvatures. The 3D coordinates of different points and different clearance thresholds have been summarized in Table 3.

Table 3. 3D coordinate of turn points in Figure 10b.

Coord	Point-1	Point-2	Point-3	Point-4	Point-5	Point-6	Point-7	Point-8	Point-9	Point-10
X	0	60	80	90	70	60	50	30	20	0
Y	0	80	70	40	60	90	50	60	20	0
Z	0	85	25	55	20	35	60	15	10	0
Threshold Distance (δ): $\delta 1 = 18$, $\delta 2 = 18$, $\delta 3 = 16$, $\delta 4 = 16$, $\delta 5 = 14$, $\delta 6 = 16$, $\delta 7 = 18$, $\delta 8 = 23$										

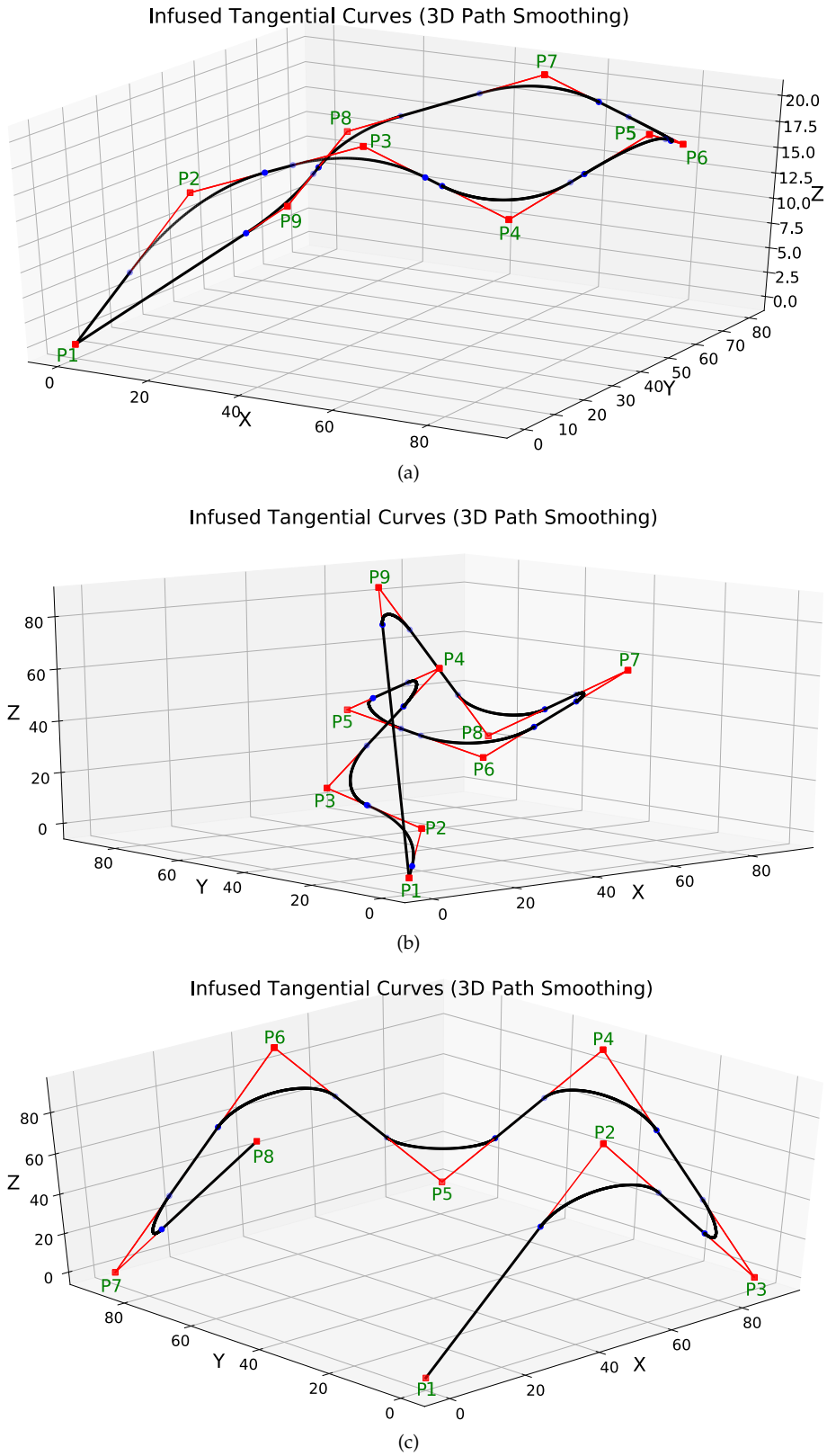


Figure 10. 3D path smoothing. (**a**) closed loop 3D path smoothing; (**b**) a complex closed loop 3D path smoothing case; (**c**) an open loop 3D path smoothing case.

Figure 10c shows an open-loop UAV path with different start (P1 : $x = y = z = 0$) and goal points (P8 : $x = 0, y = 45, x = 90$). There are six sharp turns at points P2, P3, P4, $\cdots$, $P7$. It is clear

from Figure 10c that the proposed method keeps straight paths of the UAV straight. The turns can be smoothed for different curvatures. In this case, we used the same clearance threshold distance for the different turns. Hence, the curvature of all the smooth induced curves is the same. The 3D coordinates of different points and different clearance thresholds have been summarized in Table 4.

Figure 11 shows the different 3D spheres for path smoothing in case of Figure 10c. The coordinates of the points are the same as given in Table 4. In this case, the surface of the 3D sphere is used to smooth out the sharp turns. It should be noted that the spheres in Figure 11 look like a circle as only a particular projection of the 3D space is shown.

Table 4. 3D coordinate of turn points in Figure 10c.

Coord	Point-1	Point-2	Point-3	Point-4	Point-5	Point-6	Point-7	Point-8
X	0	45	90	90	90	45	0	0
Y	0	0	0	45	90	90	90	45
Z	0	90	0	90	0	90	0	90
Threshold Distance (δ): $\delta 1 = 35$, $\delta 2 = 35$, $\delta 3 = 35$ $\delta 4 = 35$, $\delta 5 = 35$, $\delta 6 = 35$, $\delta 7 = 35$								

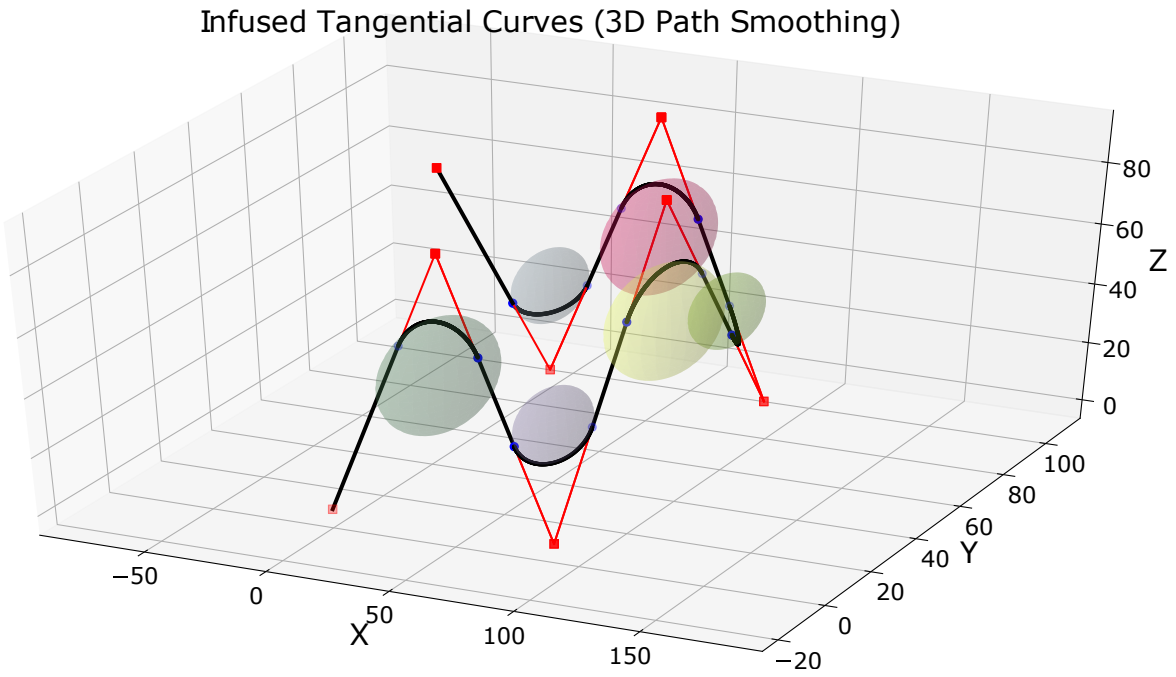


Figure 11. Visual representation of 3D path smoothing in case of Figure 10c.

6.3. Comparison with Other Works

In order to compare the strengths of our work, we compare our work with the path smoothing method proposed in the works of Huh and Chang in [56]. The method proposed in this paper uses an interpolation technique to smooth the sharp turns of the robot's path. Interpolation technique was first proposed by Warning [57,58]. Precisely, given $m+1$ pairs (x_i, y_i), the problem consists of finding a function $\phi = \phi(x)$ such that $\phi(x_i) = y_i$ for $i = 0, \cdots, m, y_i$ being some given values, and say that ϕ interpolates $\{y_i\}$ at the nodes $\{x_i\}$. We speak about polynomial interpolation if ϕ is an algebraic polynomial, trigonometric approximation if ϕ is a trigonometric polynomial, or piecewise polynomial interpolation (or spline interpolation) if ϕ is only locally a polynomial [1]. As interpolation

based methods are widely used in path smoothing algorithms [41,59,60] found in the state-of-the-art, comparison with this method can highlight the merits as well as drawbacks of the proposed method.

In the comparison, we used the same dataset of points as used in work [56]. The dataset contains total ten points in a grid of size 50×50 units. The 2D coordinates of the ten points have been summarized in Table 5. As shown in Figure 12, the ten points (P1, P2, $\cdots$, P10) are marked in green. The starting point is P1 :$x = 8, y = 5$, and the goal point is P10 :$x = 25, y = 25$ with many sharp turns at different points. In Figure 12, the original path is marked in black, whereas the smooth path is shown in red.

To smooth the paths, we used four sets of thresholds. The results of smoothing with different thresholds are shown in Figure 12a–d. The values of different thresholds along with the 2D coordinates of the points are summarized in Table 5 for the different figures.

Table 5. Coordinates of different points in the comparative work of Figure 12 for different cases.

Coord	Point-1	Point-2	Point-3	Point-4	Point-5	Point-6	Point-7	Point-8	Point-9	Point-10
X	8	7	2	11	11	45	45	25	12	25
Y	5	15	10	10	14	30	12	45	45	25
Figure 12a	Threshold Distance (δ): $\delta1 = 3,\ \delta2 = 3,\ \delta3 = 1.5,\ \delta4 = 1.5,\ \delta5 = 3,\ \delta6 = 3,\ \delta7 = 3,\ \delta8 = 3$									
Figure 12b	Threshold Distance (δ): $\delta1 = 3,\ \delta2 = 3,\ \delta3 = 1.5,\ \delta4 = 1.5,\ \delta5 = 4,\ \delta6 = 4,\ \delta7 = 4,\ \delta8 = 4$									
Figure 12c	Threshold Distance (δ): $\delta1 = 3,\ \delta2 = 3,\ \delta3 = 1.5,\ \delta4 = 1.5,\ \delta5 = 5,\ \delta6 = 5,\ \delta7 = 5,\ \delta8 = 5$									
Figure 12d	Threshold Distance (δ): $\delta1 = 3,\ \delta2 = 3,\ \delta3 = 1.5,\ \delta4 = 1.5,\ \delta5 = 6,\ \delta6 = 8,\ \delta7 = 6,\ \delta8 = 6$									

Figure 12a shows the path smoothing with a minimum threshold. The left section of the figure has been enlarged to provide a better view. We successively increased the clearance thresholds in different steps. The consequent smoothing results are shown in Figure 12b–d. Path smoothing at point P7 can be seen in different figures to notice the curvature change of the induced curve.

Figure 12e is the same as Figure 12d, but the obstacles are also shown. Notice that the path between the points P5 and P6 is a straight corridor between the walls shown in gray color. A robot traversing this path is expected to maintain a safe distance from the walls and move as straight as possible. The proposed method is able to achieve exactly that goal. The straight paths are kept straight, while only the sharp turns are smoothed out. For comparison, we direct the readers to paper [56] (page 7, Figure 11 of [56] to be exact). In work [56], since interpolation techniques are used, the path between the points P5 and P6 is not straight but dangerously close to the walls at multiple points. This also happens at other locations, and the robot's proximity to the obstacles is compromising safety. Moreover, since interpolation is used, it is difficult to control the curvature of paths, especially at the turns. On the other hand, in the proposed method, it is easy to control the curvature.

However, our proposed method has a disadvantage that G^2 continuity is not possible. G^2 continuity is important for a robot that accelerates significantly on the paths. In fact, the reason why the work in [56] brings the robot close to one of the walls is because it emphasizes achieving a G^2 continuity. Adjusting a particular point on paths using interpolation is difficult as changing one point changes the whole path. Hence, there is a possibility that adjusting one point to a safe distance brings other portions of the continuous path close to the obstacles. The proposed method only guarantees a G^1 continuity that is tangential continuity. The G^1 continuity is important so that the robot does not experience a sudden kink or bump while traversing from a straight line to a curve. G^2 continuity is important for robots traveling at high speeds. However, most of the service robots have limited speed (generally around 2 m/s) to ensure operational safety. Although ensuring a G^2 continuity is beneficial, G^1 continuity is enough for operations at lower speeds. Since the proposed method only smooths the

turns and keeps the straight paths straight, a robot can always navigate the straight segments of the path at high speeds and slow down before approaching a turn.

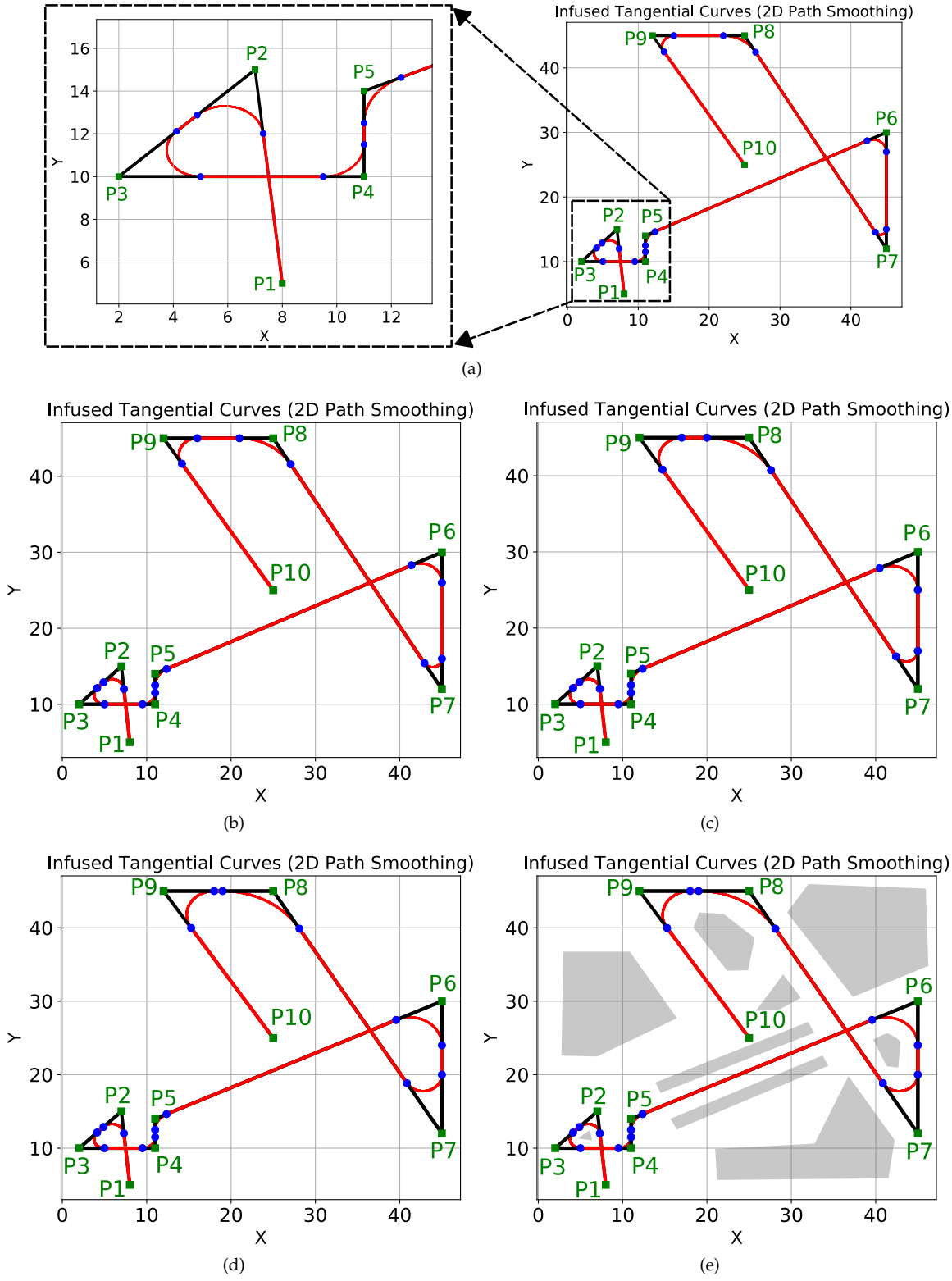


Figure 12. Comparison with interpolation based smoothing proposed in [56] using the same set of points. The various thresholds δ_{thresh} in (**a**–**e**) are given in Table 5.

6.4. Generation of Smooth Paths with Variable Curvature

A strong merit of the proposed ITC path smoothing is that the curvature of the smooth trajectories can be controlled easily. Figure 13 shows the results of path smoothing in which the ITC curves have been generated with different curvatures. The original path is $P_1P_2P_3P_4$ shown in black color with sharp turns at points P_2 and P_3. The coordinates of the four points are: $P_1 : (50, 50)$, $P_2 : (250, 250)$, $P_3 : (500, 50)$, $P_4 : (750, 250)$.

As shown in Figure 13, at the same turn point P_2, different ITC curves ($\widehat{AA'}$, $\widehat{BB'}$, $\cdots$, $\widehat{HH'}$) have been generated. The radius and curvature of the different ITC curves are shown in Figure 14a,b, respectively. Thus, depending on the kinematics of the robot and the configuration of obstacles, appropriate path smoothing can easily be achieved using the proposed method.

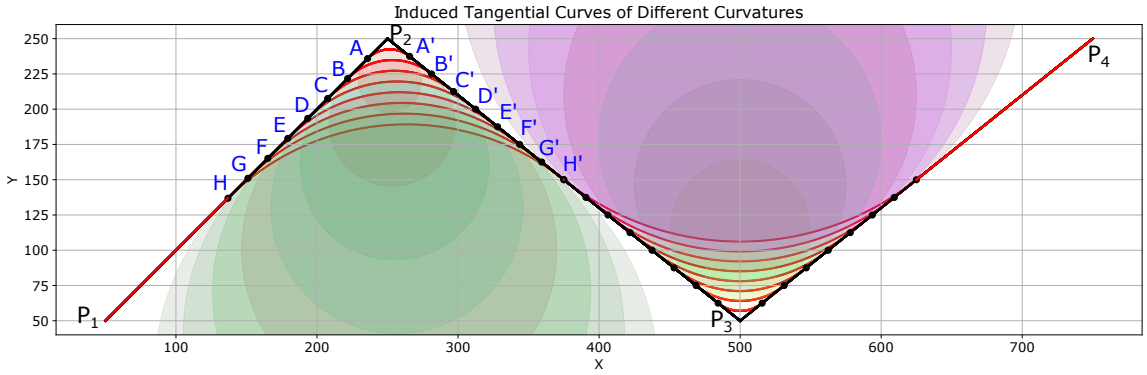


Figure 13. ITC path generation with different curvatures.

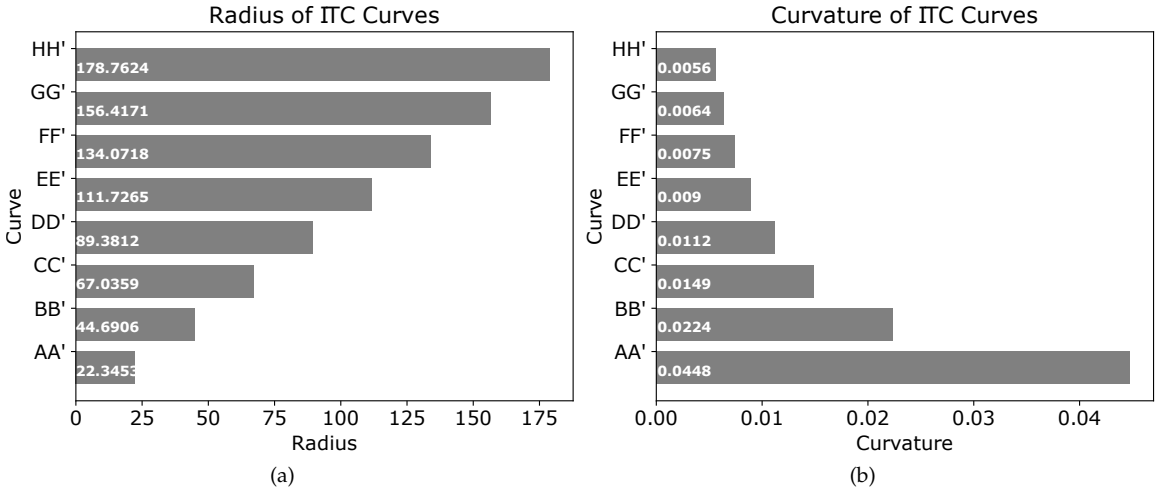


Figure 14. Radius and curvature of the different curves in Figure 13. (**a**) radius of curves at different points; (**b**) curvature of curves at different points.

6.5. Results of Smoothing in Real Environment with Actual Robots

This section discusses results in a real environment with actual robots. Figure 15 shows the robots used and its motion model. We used a Pioneer-P3DX [61] robot and Turtlebot robot [62] shown in Figure 15a,b, respectively. Both the robots were equipped with distance sensors (Microsoft Kinect [63] and UHG-08LX laser range sensor [64]) and cameras. The distance sensor is accurate within $\pm$30 mm within 1 m, and within 3% of the detected distance between 1 and 8 m. The angular resolution is approximately 0.36 degrees, and other specifications can be found in [64]. Specifications of Kinect sensors can be found in [63]. The robots were programmed in ROS [65]. Both are differential drive

robots. We adopt the motion model from our previous work [52] and briefly describe here. The distance between the left and the right wheel is W_r, and the robot state at position P, is given as $[x, y, \theta]$. From Figure 15c, turning angle β is calculated as

$$\begin{aligned} r &= \beta \cdot (R + W_r), \\ l &= \beta \cdot R, \\ \therefore \beta &= \frac{r - l}{W_r}, \end{aligned} \tag{35}$$

and the radius of turn R as

$$R = \frac{l}{\beta}, \beta \neq 0. \tag{36}$$

The coordinates of the center of rotation (C, in Figure 15c) are calculated as

$$\begin{bmatrix} Cx \\ Cy \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} - \left(R + \frac{W_r}{2} \right) \cdot \begin{bmatrix} \sin\theta \\ -\cos\theta \end{bmatrix}. \tag{37}$$

The new heading θ' is

$$\theta' = (\theta + \beta) \bmod 2\pi, \tag{38}$$

from which the coordinates of the new position P' are calculated as

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} Cx \\ Cy \end{bmatrix} - \left(R + \frac{W_r}{2} \right) \cdot \begin{bmatrix} \sin\theta' \\ -\cos\theta' \end{bmatrix}, \ \beta \neq 0 \implies r \neq l. \tag{39}$$

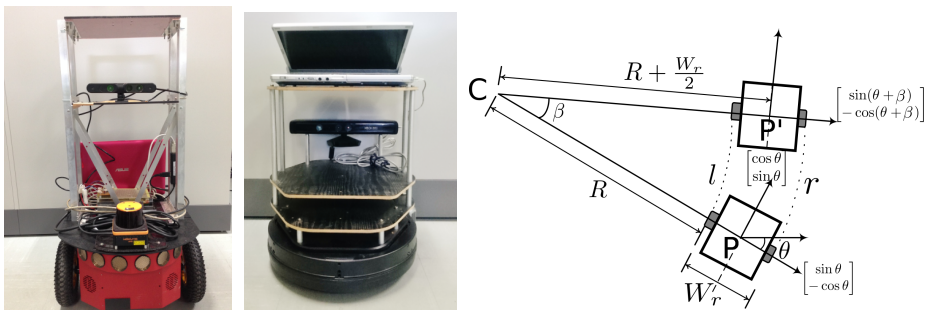
If $r = l$, i.e., if the robot motion is straight, the state parameters are given as

$$\theta' = \theta, \tag{40}$$

and

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + l \cdot \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix}, \ (l = r). \tag{41}$$

Figure 16 shows the experiment environment and its grid map. As shown in Figure 16a, the environment was conducted in a narrow corridor of our university. The corresponding grid map is shown in Figure 16b, in which the actual experiment section of the corridor is marked and shown enlarged.



(a) (b) (c)

Figure 15. Robots used in the experiments. (**a**) Pioneer P3DX; (**b**) Kobuki Turtlebot; (**c**) Motion Model.

(a) (b)

Figure 16. Experiment setup. (**a**) corridor scenario used in experiments; (**b**) grid-map of the environment. The zoomed out section shows the width of the corridor. Starting locations and directions of movement of the two robots are also shown.

6.5.1. Non-Smoothed Collision Avoidance and Navigation (Real Environment with Actual Robots)

Path smoothing in open and static environments can easily be demonstrated. However, real-time path smoothing for multiple robots in a dynamic and realistic scenarios is more challenging. We tested the proposed path smoothing method in a dynamic multi-robot collision avoidance scenario. In the experiment, the Pioneer P3DX robot and Turtlebot robot navigated towards each other in the narrow corridor and tried to avoid collision. We compared the trajectories in both traditional (non-smoothed) and the proposed (smoothed) method.

The direction of movement of both the robots is indicated in Figure 16b. Pioneer P3DX robot robot navigated from North to South direction with the starting point marked as P. The Turtlebot robot navigated in the opposite direction in the same corridor from South to North with the starting point marked as T in Figure 16b. The width of the corridor was 234 cm, and both the robots were programmed to navigate the center of the corridor (i.e., $\approx$ 127cm from either of the walls). The threshold collision avoidance distance was set to 4 m. Once an obstacle at this distance is found, the robots were programmed to avoid it using the traditional and proposed smooth algorithms.

Figure 17 shows the timely snapshots (Figure 17(ns-1), $\cdots$, Figure 17(ns-30)) of the experiment with traditional path planning and navigation. For the ease of readability, we have summarized the various actions take by the two robots at different time-steps in Figure 17 in Table 6. The flowchart of the non-smoothed collision avoidance and navigation is shown in Figure 18a.

In the traditional navigation without smoothing, the robots stopped when the frontal distance was less than the threshold distance. Then, the robots took a sharp 90-degree turn towards the left, moved towards the wall, stopped, and again took a 90-degree right turn. The robots crossed-over, and then repeated the process to come to the center of the corridor from where they continued to navigate towards their respective goals.

Readers are advised to see the attached video to see the traditional navigation and multi-robot collision avoidance. It is clear that such a robot navigation with abrupt stops, and sharp turns, is not natural, potentially hazardous for the items carried on the robot, and even dangerous for people moving in the vicinity.

Figure 17. Timely snapshots of the non-smoothed navigation and collision avoidance. The robot actions at different steps are summarized in Table 6. Please see the supplementary video.

Table 6. Description of robot actions at different steps (non-smooth case of Figure 17).

Figure	Action	Description
Figure 17(ns-1)∼(ns-4)	Move Straight	Robots navigate towards each other
Figure 17(ns-5)	Stop	Both the robots stop upon threshold distance
Figure 17(ns-6)∼(ns-9)	Move Left	Robots move left to ≈127 cm of the wall
Figure 17(ns-10)∼(ns-12)	Stop and Turn Right	Robots turn right
Figure 17(ns-13)∼(ns-20)	Crossover	Robots cross each other
Figure 17(ns-21)	Stop	Robots stop to make a turn again
Figure 17(ns-22)∼(ns-25)	Approach Center and Turn	Robots navigate to center of corridor and turn
Figure 17(ns-26)∼(ns-30)	Move Straight	Robots continue towards their goal

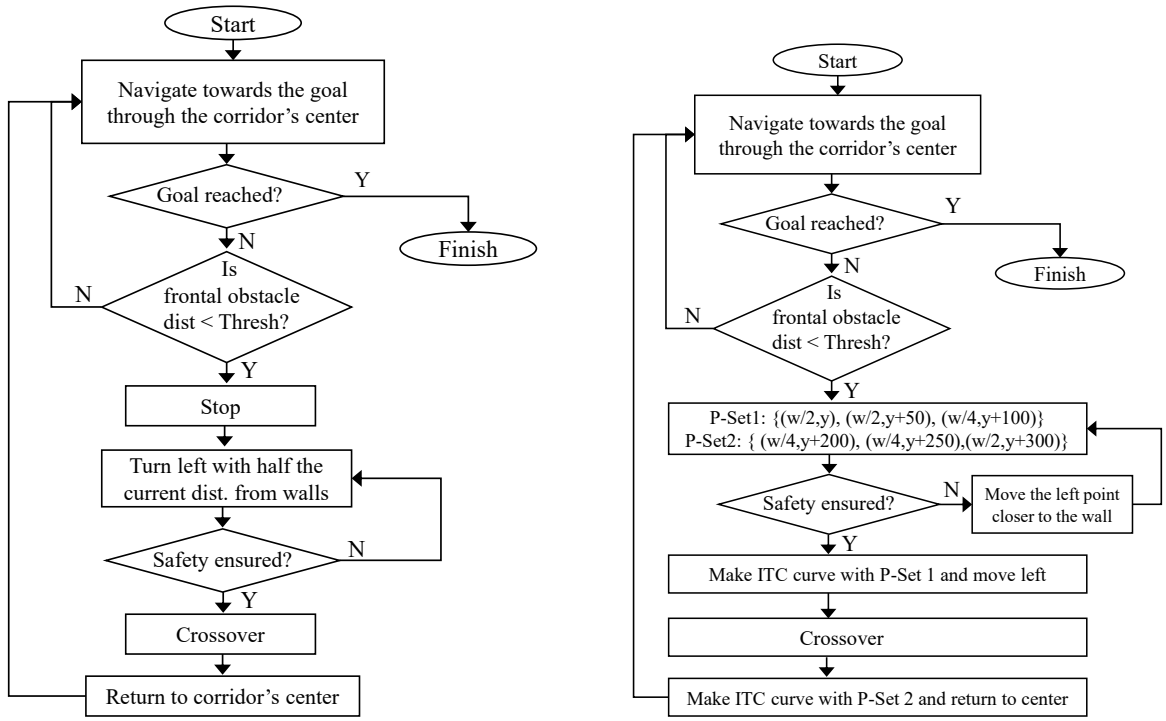


Figure 18. Flowchart of non-smoothed and ITC based smooth navigation and collision avoidance. (**a**) non-smoothed case; (**b**) ITC based smoothed case.

6.5.2. Smoothed Collision Avoidance and Navigation (Real Environment with Actual Robots)

Figure 19 shows the timely snapshots (Figure 19(sm-1), $\cdots$, Figure 19(sm-30)) of the navigation with the proposed smoothing algorithm. The experiment was conducted in the same environment with the same direction, start, and goal locations of Turtlebot and Pioneer robots. The flowchart of the smoothed collision avoidance and navigation is shown in Figure 18b.

The two robots approached each other while traversing the center of the corridor. When the frontal threshold distance was less than the threshold distance, the robots essentially generated three control points over which the tangential curve could be induced. As shown in Figure 18b, if W is the width of the corridor, the robot is traversing the corridor on a line $\{\frac{W}{2}, y\}$, $y \in \mathbb{R}_{map}$ the robot generates a set (P-Set1) of three points:

$$\text{P-Set1} = \left\{ A : \left(\frac{W}{2}, y\right), B : \left(\frac{W}{2}, y + \psi_1\right), C : \left(\frac{W}{\lambda}, y + 2\psi_2\right) \right\}. \tag{42}$$

Essentially, the point $A = (\frac{W}{2}, y)$ and $B = (\frac{W}{2}, y+\psi_1)$ lies on the straight line on the center of the corridor. Point $C = (\frac{W}{\lambda}, y + 2\psi_2)$ lies on the left side of the corridor. The parameter λ controls the distance of the trajectory from the left wall of the corridor. For collision avoidance, λ is generally set to 4, which generates a trajectory at a distance of $\frac{W}{4}$ from the left wall of the corridor. The parameter ψ_1 controls the turning point in front of the robot. The parameter ψ_2 controls the point on the frontal left side of the robot. The effect of the parameter ψ is explained later. From the point set (P-Set1), an ITC curve is generated and induced in the original trajectory.

Once the robots have shifted left, they move in a straight line and cross each other. Once the robots have crossed over, the robots need to get back to the center of the corridor again. This is done by generating a set (P-Set2) of three points:

$$\text{P-Set2} = \left\{ A : \left(\frac{W}{\lambda}, y_t\right), B : \left(\frac{W}{\lambda}, y_t + \psi_2\right), C : \left(\frac{W}{2}, y_t + 2\psi_2\right) \right\}. \tag{43}$$

An ITC is generated again from set (P-Set2), and the robots smoothly traverse it to come back to the center of the corridor and navigate towards their respective goals.

We have summarized the various actions taken by the two robots at different time-steps in Figure 19 in Table 7.

Figure 19. Timely snapshots of ITC based smoothed navigation and collision avoidance. The robot actions at different steps are summarized in Table 7. Please see the supplementary video.

Table 7. Description of robot actions at different steps (smooth case of Figure 19).

Figure	Action	Description
Figure 19(sm-1)∼(sm-7)	Move Straight	Robots navigate towards each other
Figure 19(sm-8)∼(sm-11)	Smooth Turn Left	Both the robots smoothly move to left
Figure 19(sm-12)∼(sm-20)	Crossover	Robots cross each other
Figure 19(sm-21)∼(sm-24)	Smooth Turn Right	Both the robots smoothly approach corridor's center
Figure 19(sm-25)∼(sm-30)	Move Straight	Robot navigate towards their goal

Figure 20 shows the trajectories of the two robots. In Figure 20, 'T' and 'P' show the starting positions of Turtlebot and Pioneer robots, respectively. The trajectory of Turtlebot is shown in red, whereas the trajectory of Pioneer robot is shown in blue. The original trajectories are shown in black. As shown in Figure 16, the width of the corridor was 234 cm, and both the robots initially started from the center of the corridor. The center line is shown as a dotted line in Figure 20.

In the experiment shown in Figure 20, we set the parameters $\psi_1 = \psi_2 = 50$. The Turtlebot's starting position of turn was $(x = 117, y = 0)$ and the Pioneer robot's starting position at a smooth turn was $(x = 117, y = 400)$. Setting the values in Equation (42), we get (P-Set1) as, P-Set1 $= \{A : (117, 0), B : (117, 50), C : (58.5, 100)\}$. Similarly, another set of points are generated and the smooth ITC trajectories are generated for both Turtlebot (red trajectory in Figure 20) and

Pioneer robot (blue trajectory in Figure 20). Figure 20 also shows a zoomed out section of the trajectory to visually confirm the induced G^1 tangential curves. In the experiment, the same values of both the parameters ψ_1 and ψ_2 were set for both of the robots. Hence, the generated trajectories in Figure 20 are symmetrical. The smooth ITC trajectories were generated in 39.73 ms on Ubuntu 16.04 with Core-i7 processor and 16 GB RAM using Python 3.6 language. This is fast enough for real-time applications.

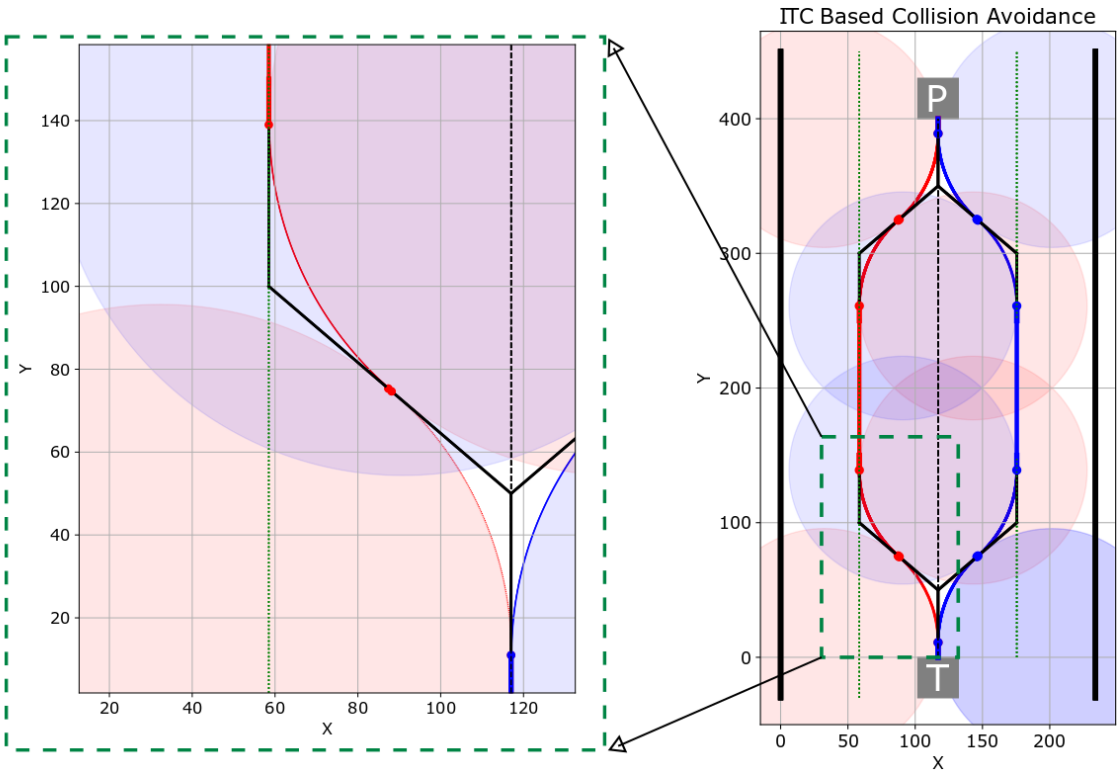


Figure 20. Trajectories of the two robots. Pioneer P3DX and Turtlebot are marked as P and T, respectively. The parameters were set as $\psi_1 = \psi_2 = 50$.

6.6. Effect of ψ on a Smooth Trajectory

We now discuss the effect of parameters ψ_1 and ψ_2 on the smoothness of ITC curves. The parameter ψ_1 controls the robot's frontal starting point of trajectory generation. Parameter ψ_2 is crucial in controlling the curvature of the smooth trajectories. This is explained using Figure 21, in which the robot is assumed to be on the center of the corridor shown as a dotted line. Point A marks the starting of the smooth trajectory generation, and the second point B is at a distance of ψ_1 from point A. The points C, C′, and C″ are generated using different values 50, 75, and 100 of ψ_2, respectively.

It should be noted that point B is common for the different ITC curves generated in Figure 21. However, the values of ψ_2 were different generating different set of points on the left side of the corridor. The blue ITC curve in Figure 21 corresponds to $\psi_2 = 50$. The green and magenta ITC curves corresponds to $\psi_2 = 75$, and $\psi_2 = 100$, respectively. The seed point set for blue ITC curve generation was P-Set1 $= \{A : (117, 0), B : (117, 50), C : (58.5, 100)\}$. The seed point set for green ITC curve generation was: P-Set1 $= \{A : (117, 0), B : (117, 50), C : (58.5, 125)\}$. Similarly, the seed point set for magenta ITC curve generation was: P-Set1 $= \{A : (117, 0), B : (117, 50), C : (58.5, 150)\}$.

The radius of the three curves seen in Figure 21 are shown in the plot of Figure 22a. Similarly, the curvatures of the three curves are given in Figure 22b. It is clear that increasing the value of ψ_2 generates an ITC curve with lesser curvature. The actual radii and curvatures of the three curves are shown in Figure 22a,b, respectively.

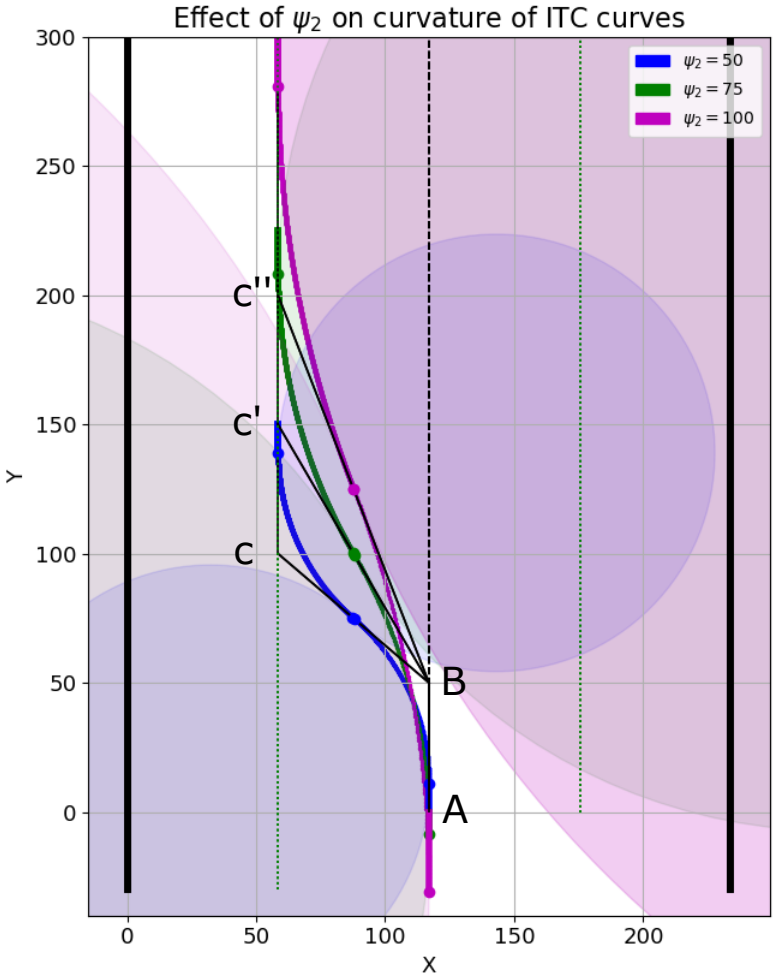


Figure 21. Effect of ψ_2 on ITC curve generation. Blue, green, and magenta curves were generated with $\psi_2 = 50$, $\psi_2 = 75$, and $\psi_2 = 100$, respectively.

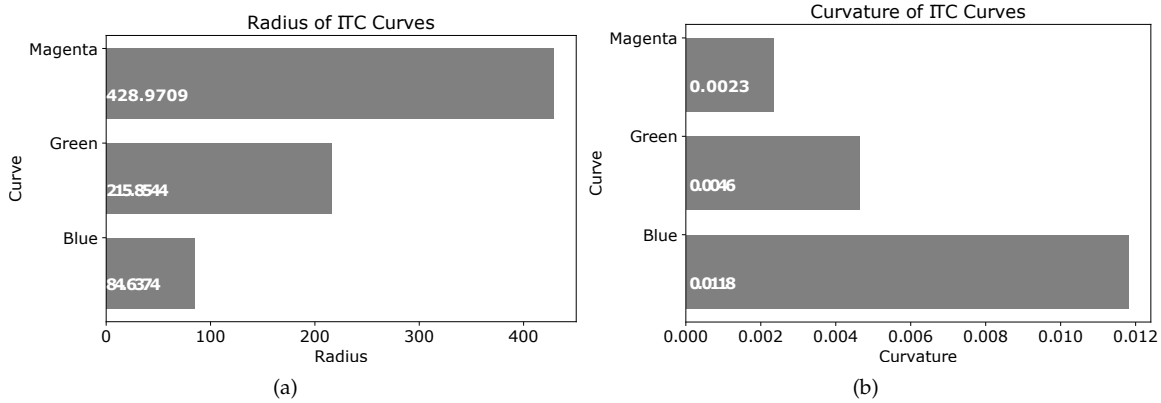


Figure 22. Radius and curvature of blue ($\psi_2 = 50$), green ($\psi_2 = 75$), and magenta ($\psi_2 = 100$) curves in Figure 21. (**a**) Radius of curves at different points. (**b**) Curvature of curves at different points.

6.7. Effect of λ on Smooth Trajectory

The parameter λ in Equation (42) controls the distance of the trajectory from the left wall of the corridor. Figure 23 shows the smooth ITC trajectory generation with different values of λ. For generating the left turn, the condition is:

$$2 < \lambda < \frac{W_{corridor}}{\frac{W_{robot}}{2} + \delta_{thresh}},$$

where W_{robot} is the width of the robot, and δ_{thresh} is the safety threshold from the corridor's left wall. Setting $\lambda = 2$ generates points on the straight line in the center of the corridor which does not require smoothing. As shown in Figure 23, the black ITC curve marked 'A' is generated with $\lambda = 2.5$, and is closest to the corridor's center. On the other hand, the magenta ITC curve marked 'F' is generated with $\lambda = 12$, and is the farthest from the corridor's center and closest to the corridor's left wall. The other curves with different values of λ are also shown. The radii and curvatures of the different curves are shown in Figure 24a,b, respectively.

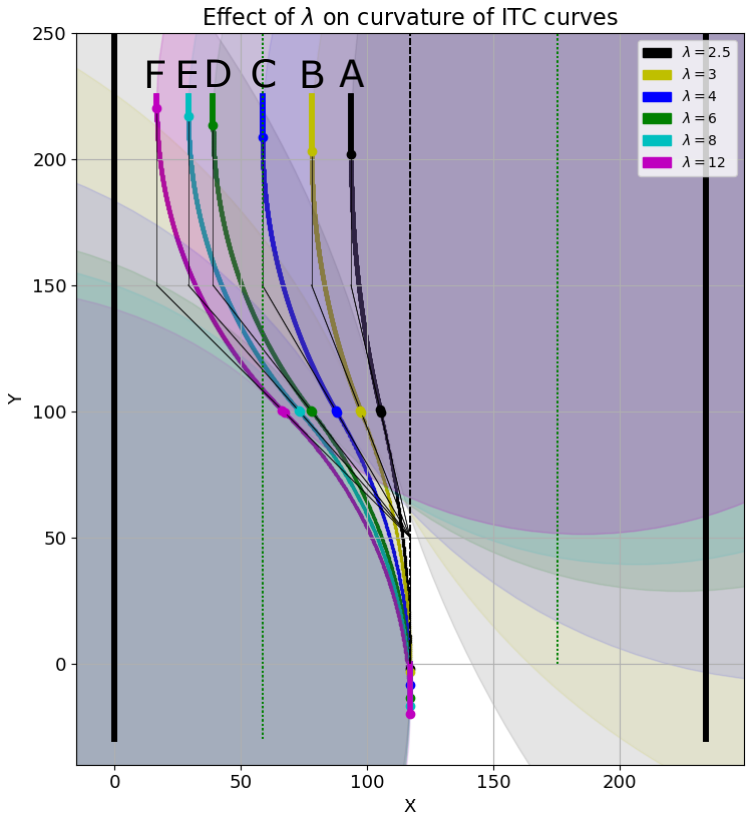


Figure 23. Effect of λ on different ITC curve generations for smooth left turns.

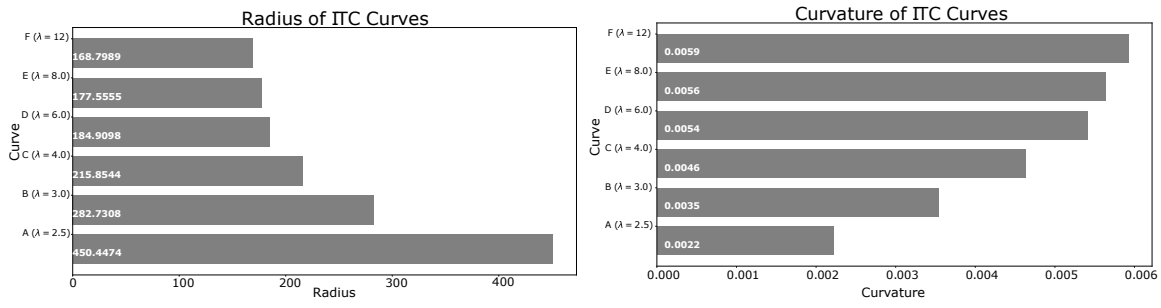


Figure 24. Radius and curvature of curves corresponding to different λ in Figure 23. (**a**) radius of curves at different points; (**b**) curvature of curves at different points.

The same parameter is also used for generating the smooth right turn of the robot. Figure 25 shows right turn trajectory generation. The value of λ for right turn generation is

$$\frac{W_{corridor}}{W_{corridor} - \left(\frac{W_{robot}}{2} + \delta_{thresh}\right)} < \lambda < 2.$$

As shown in Figure 25, the black ITC curve marked 'A' is generated with $\lambda = 1.8$, and is closest to the corridor's center. On the other hand, the magenta ITC curve marked 'F' is generated with $\lambda = 1.05$, and is the farthest from corridor's center, and closest to the corridor's right wall. Setting $\lambda = 1$ generates a trajectory touching the right wall of the corridor. The other curves with different values of λ are also shown. The radii and curvatures of the different curves are shown in Figure 26a,b, respectively.

Thus, depending on the width of the robot and the obstacle ahead, appropriate value of λ can be chosen to avoid collision, for both right and left turns.

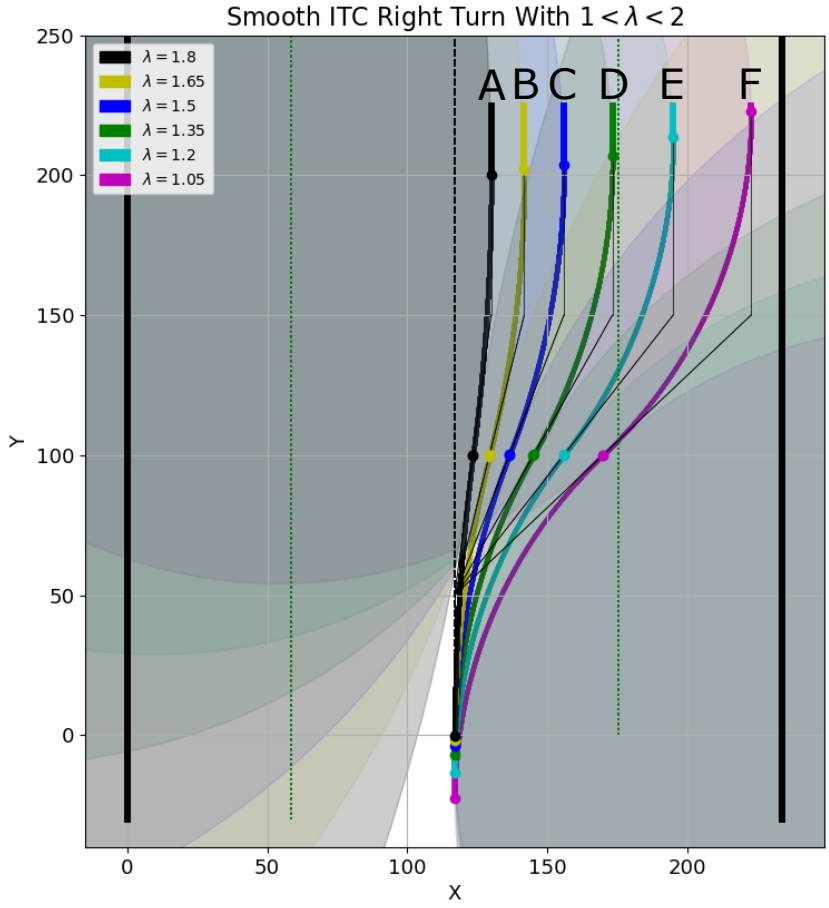


Figure 25. Effect of λ on different ITC curve generations for smooth right turns.

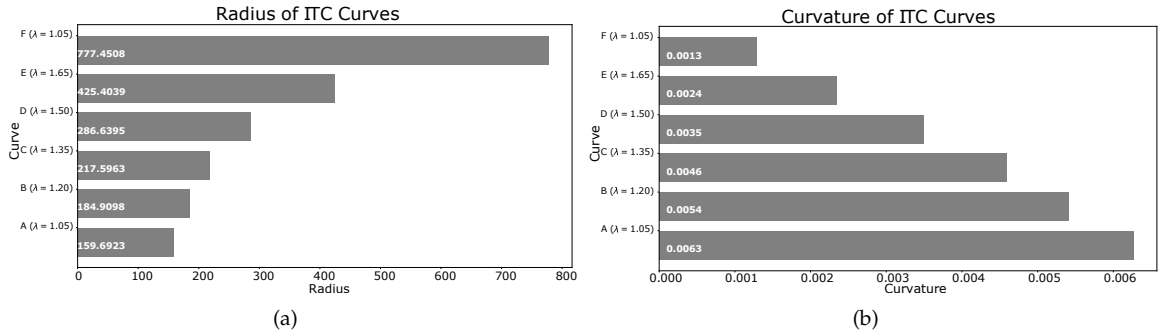


Figure 26. Radius and curvature of curves corresponding to different λ in Figure 25. (**a**) radius of curves at different points; (**b**) curvature of curves at different points.

7. ITC as a Path Smoothing Extension

An ITC path smoother is proposed to work in conjunction with traditional path planning algorithms, and not to replace them. The overall idea of ITC as an 'extension' is shown in Figure 27. The input to the global path planner is: (a) map with obstacles and free space marked, (b) start, and (c) goal location in the map. Any of the global path planners like A*, D*, PRM, or RRT path planners can be used. The output of the global path planner is the input to the proposed ITC path smoother. The ITC smoother first detects the sharp turns and then smooths only the turns while keeping the straight segments straight. It can be seen in Figure 27 that the map information is also input to the ITC block. This is because ITC trajectories are generated keeping a safe distance from the obstacles. Thus, a map which marks the location of obstacles is required. The outputs of the ITC smoother are smooth trajectories whose angular turns have been smoothed out. In this way, the proposed ITC algorithm can be used as an extension with existing planners. A major benefit of such extension is that there is no need to replace the already tested planning algorithms used with the robots. The embedded software used in robot platforms is generally tightly coupled with the hardware and replacing the existing algorithms with new algorithms is generally avoided unless absolutely necessary as additional testing and benchmarking must be performed for the new algorithm. In this regard, the proposed ITC extension will integrate easily with existing algorithms. In addition, there is a lot of scope to customize the ITC smoother as generation of smooth trajectories is done on a part-by-part basis and there is much less computational overhead.

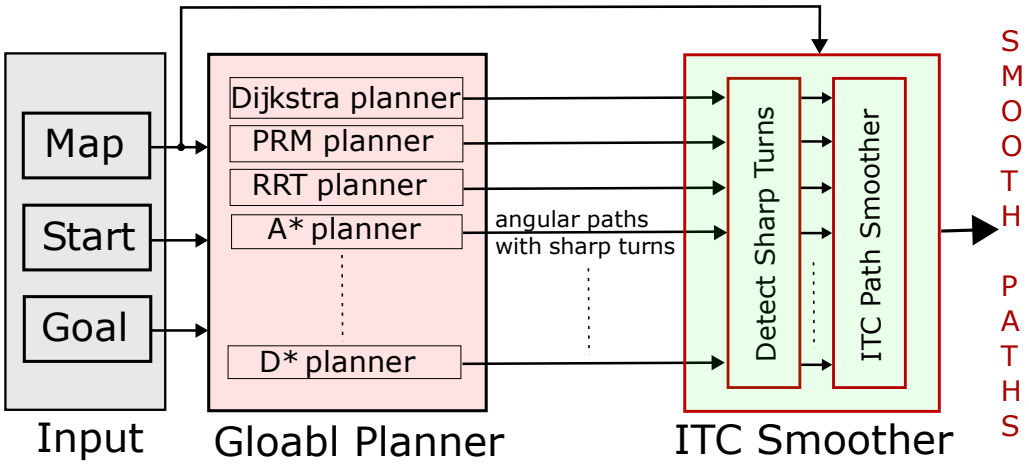


Figure 27. ITC path smoothing extension.

8. Conclusions

We presented a new algorithm called ITC for smooth trajectory generation for mobile robot robots. The algorithm can smooth out the sharp turns in the path generated by the global path planner. The trajectories generated by the algorithm are tangential to the path, thus preserving G^1 continuity. The curves can be generated fast in real-time by using only three key points on the path. Safety is embedded in an ITC algorithm, and it is guaranteed that the robot maintains a safe threshold distance from the obstacles, which is a crucial feature of mobile robot navigation. An essential feature of the algorithm is that only the turns are smoothed out, while the straight paths are kept straight. This feature is highly desired in case of mobile robot navigation in narrow corridors. We discussed ITC curve generation for both 2D path smoothing for UGVs, and 3D path smoothing for UAVs.

We compared the proposed ITC algorithm with interpolation based approaches. The comparison shows that ITC paths maintain a safe distance from the walls of the corridors and enables the robot to move in the center. Unlike interpolation algorithms, it is easier to define the control points in ITC algorithm. Moreover, unlike interpolation based methods, changing one point does not alter the whole path. This is another merit of the proposed ITC algorithm—that it is easy to define control points, and smoothing is done on a part-by-part basis for each turn. Thus, smoothing one section of the path with sharp turns does not affect other paths. It is difficult to follow such an approach in interpolation based algorithms as smoothing has to be done on a global basis, otherwise discontinuities or kinks get introduced in the overall path. The proposed ITC algorithm has a disadvantage that G^2 continuity is not guaranteed. This may be a limitation for mobile robots moving at high speed. However, for lower to medium speed robot navigation, this is not a problem. In fact, robots can traverse the straight sections of the ITC path with high speed, and slow down at the turns while executing a smooth turn, which is normally the case seen in robot's navigation to ensure safety due to a change in view, and the sudden appearance of moving obstacles at turn points. We showed how ITC curves can be generated for different curvatures for smooth left and right turns by easily defining the parameters. Finally, we showed a complex real world scenario of collision avoidance with real robots. We compared the traditional navigation of the robots for collision avoidance with the proposed ITC based navigation and collision avoidance. It was clear that traditional navigation required the robot to stop and execute sharp turns. However, the ITC based navigation was smooth, natural, and robots could avoid collision by computing the smooth trajectories in real-time. Our current work presented a real-world 2D navigation of mobile robots and complex scenarios of collision avoidance. In the future, we will test the algorithm with multiple UAVs. A ground robot's 2D navigation and collision avoidance will be compared to other algorithms with actual robots. In addition, apart from collision avoidance, we also consider trajectory smoothing of robotic arm manipulators as future work.

Supplementary Materials: Experiment video is available online at http://www.mdpi.com/1424-8220/19/20/4384/s1, Video S1: Video of Traditional Non-Smooth vs ITC Based Smooth Collision Avoidance of Multiple Robots.

Author Contributions: A.R. (Abhijeet Ravankar) and A.A.R. conceived the idea, designed, and performed the experiments; A.R. (Arpit Rawankar) helped with proof checking and visualizations; Y.H. and Y.K. made valuable suggestions to analyze the data and improve the work. The manuscript was written by A.R. (Abhijeet Ravankar).

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ravankar, A.; Ravankar, A.A.; Kobayashi, Y.; Hoshino, Y.; Peng, C.C. Path Smoothing Techniques in Robot Navigation: State-of-the-Art, Current and Future Challenges. *Sensors* **2018**, *18*, 3170. [CrossRef] [PubMed]
2. Ravankar, A.; Ravankar, A.A.; Kobayashi, Y.; Emaru, T. Real-Time Path Smoothing for Mobile Robots in 2D and 3D Environments. In Proceedings of the 2018 JSME Annual Conference on Robotics and Mechatronics (Robomec), Kitakyushu, Japan, 2–5 June 2018; pp. 1A1–J03. [CrossRef]

3. Hart, P.; Nilsson, N.; Raphael, B. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Trans. Syst. Sci. Cybern.* **1968**, *4*, 100–107. [CrossRef]
4. Stentz, A.; Mellon, I.C. Optimal and Efficient Path Planning for Unknown and Dynamic Environments. *Int. J. Robot. Autom.* **1993**, *10*, 89–100.
5. Stentz, A. The Focussed D* Algorithm for Real-Time Replanning. In Proceedings of the International Joint Conference on Artificial Intelligence, Montreal, QC, Canada, 20–25 August 1995; pp. 1652–1659.
6. Hwang, Y.; Ahuja, N. A potential field approach to path planning. *IEEE Trans. Robot. Autom.* **1992**, *8*, 23–32. [CrossRef]
7. Kavraki, L.; Svestka, P.; Latombe, J.C.; Overmars, M. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. Autom.* **1996**, *12*, 566–580. [CrossRef]
8. Lavalle, S.M. *Rapidly-Exploring Random Trees: A New Tool for Path Planning*; Technical Report; Iowa State University: Ames, IA, USA, 1998.
9. LaValle, S.M.; Kuffner, J.J. Randomized Kinodynamic Planning. *Int. J. Robot. Res.* **2001**, *20*, 378–400, doi:10.1177/02783640122067453. [CrossRef]
10. Lavalle, S.M.; Kuffner, J.J., Jr. Rapidly-Exploring Random Trees: Progress and Prospects. In *Algorithmic and Computational Robotics: New Directions*; CRC Press: Boca Raton, FL, USA, 2000; pp. 293–308.
11. Dijkstra, E.W. A Note on Two Problems in Connexion with Graphs. *Numer. Math.* **1959**, *1*, 269–271. [CrossRef]
12. Wang S.X. The Improved Dijkstra's Shortest Path Algorithm and Its Application. *Procedia Eng.* **2012**, *29*, 1186–1190. [CrossRef]
13. Fujita, Y.; Nakamura, Y.; Shiller, Z. Dual Dijkstra Search for paths with different topologies. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '03), Taipei, Taiwan, 14–19 September 2003; Volume 3, pp. 3359–3364. [CrossRef]
14. Rösmann, C.; Hoffmann, F.; Bertram, T. Integrated online trajectory planning and optimization in distinctive topologies. *Robot. Autom. Syst.* **2017**, *88*, 142–153. [CrossRef]
15. Ravankar, A.; Ravankar, A.A.; Kobayashi, Y.; Emaru, T. Hitchhiking Robots: A Collaborative Approach for Efficient Multi-Robot Navigation in Indoor Environments. *Sensors* **2017**, *17*, 1878. [CrossRef]
16. Ravankar, A.; Ravankar, A.A.; Kobayashi, Y.; Hoshino, Y.; Peng, C.C.; Watanabe,M. Hitchhiking Based Symbiotic Multi-Robot Navigation in Sensor Networks. *Robotics* **2018**, *7*, 37. [CrossRef]
17. Rashid, A.T.; Ali, A.A.; Frasca, M.; Fortuna, L. Path planning with obstacle avoidance based on visibility binary tree algorithm. *Robot. Autom. Syst.* **2013**, *61*, 1440–1449. [CrossRef]
18. Ravankar, A.; Ravankar, A.A.; Kobayashi, Y.; Emaru, T. Symbiotic Navigation in Multi-Robot Systems with Remote Obstacle Knowledge Sharing. *Sensors* **2017**, *17*, 1581. [CrossRef] [PubMed]
19. Ravankar, A.; Ravankar, A.A.; Hoshino, Y.; Kobayashi, Y. On Sharing Spatial Data with Uncertainty Integration Amongst Multiple Robots Having Different Maps. *Appl. Sci.* **2019**, *9*, 2753. [CrossRef]
20. Ravankar, A.; Ravankar, A.A.; Kobayashi, Y.; Emaru, T. Can robots help each other to plan optimal paths in dynamic maps? In Proceedings of the 2017 56th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE), Kanazawa, Japan, 19–22 September 2017; pp. 317–320. [CrossRef]
21. Ravankar, A.; Ravankar, A.A.; Kobayashi, Y.; Peng, C.; Emaru, T. Real-time multi-robot path planning revisited as a caching problem. In Proceedings of the 2018 IEEE International Conference on Applied System Invention (ICASI), Tokyo, Japan, 13–17 April 2018; pp. 350–353. [CrossRef]
22. Ravankar, A.; Ravankar, A.A.; Kobayashi, Y.; Emaru, T. Avoiding blind leading the blind: Uncertainty integration in virtual pheromone deposition by robots. *Int. J. Adv. Robot. Syst.* **2016**, *13*, 1–16, [CrossRef]
23. Ravankar, A.; Ravankar, A.A.; Kobayashi, Y.; Emaru, T. On a bio-inspired hybrid pheromone signalling for efficient map exploration of multiple mobile service robots. *Artif. Life Robot.* **2016**, 1–11. [CrossRef]
24. Fraichard, T.; Scheuer, A. From Reeds and Shepp's to continuous-curvature paths. *IEEE Trans. Robot.* **2004**, *20*, 1025–1035.[CrossRef]
25. Liscano, R.; Green, D. Design and Implementation of a Trajectory Generator for an Indoor Mobile Robot. In Proceedings of the IEEE/RSJ International Workshop on Intelligent Robots and Systems '89, The Autonomous Mobile Robots and Its Applications (IROS '89), Tsukuba, Japan, 4–6 September 1989; pp. 380–385. [CrossRef]

26. Dubins, L.E. On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents. *Am. J. Math.* **1957**, *79*, 497–516. [CrossRef]
27. Dubins, L.E. On plane curves with curvature. *Pac. J. Math.* **1961**, *11*, 471–481. [CrossRef]
28. Yang, D.; Li, D.; Sun, H. 2D Dubins Path in Environments with Obstacle. *Math. Probl. Eng.* **2013**, *291372*, 1–6. [CrossRef]
29. Gerlach, A.R.; Kingston, D.; Walker, B.K. UAV navigation using predictive vector field control. In Proceedings of the 2014 American Control Conference, Portland, OR, USA, 4–6 June 2014; pp. 4907–4912. [CrossRef]
30. Komoriya, K.; Tanie, K. Trajectory Design and Control of a Wheel-type Mobile Robot Using B-spline Curve. In Proceedings of the IEEE/RSJ International Workshop on Intelligent Robots and Systems '89, The Autonomous Mobile Robots and Its Applications (IROS '89), Tsukuba, Japan, 4–6 September 1989; pp. 398–405. [CrossRef]
31. Delingette, H.; Hebert, M.; Ikeuchi, K. Trajectory generation with curvature constraint based on energy minimization. In Proceedings of the IEEE/RSJ International Workshop on Intelligent Robots and Systems '91, The Autonomous Mobile Robots and Its Applications (IROS '91), Osaka, Japan, 3–5 November 1991; Volume 1, pp. 206–211. [CrossRef]
32. Piazzi, A.; Guarino Lo Bianco, C.; Bertozzi, M.; Fascioli, A.; Broggi, A. Quintic G2-splines for the iterative steering of vision-based autonomous vehicles. *IEEE Trans. Intell. Transp. Syst.* **2002**, *3*, 27–36. [CrossRef]
33. Schmid, A.J.; Wörn, H. Path planning for a humanoid using NURBS curves. In Proceedings of the IEEE International Conference on Automation Science and Engineering, Edmonton, AB, Canada, 1–2 August 2005; pp. 351–356.
34. Ravari, A.N.; Taghirad, H.D. NURBS-based representation of urban environments for mobile robots. In Proceedings of the 2016 4th International Conference on Robotics and Mechatronics (ICROM), Tehran, Iran, 26–28 October 2016; pp. 20–25. [CrossRef]
35. Belaidi, H.; Hentout, A.; Bouzouia, B.; Bentarzi, H.; Belaidi, A. NURBs trajectory generation and following by an autonomous mobile robot navigating in 3D environment. In Proceedings of the 4th Annual IEEE International Conference on Cyber Technology in Automation, Control and Intelligent, Hong Kong, China, 4–7 June 2014; pp. 168–173. [CrossRef]
36. Choi, J.W.; Curry, R.; Elkaim, G. Path Planning Based on Bezier Curve for Autonomous Ground Vehicles. In Proceedings of the Advances in Electrical and Electronics Engineering—IAENG Special Edition of the World Congress on Engineering and Computer Science 2008, WCECS '08, San Francisco, CA, USA, 22–24 October 2008; pp. 158–166. [CrossRef]
37. Rastelli, J.P.; Lattarulo, R.; Nashashibi, F. Dynamic trajectory generation using continuous-curvature algorithms for door to door assistance vehicles. In Proceedings of the 2014 IEEE Intelligent Vehicles Symposium Proceedings, Dearborn, MI, USA, 8–11 June 2014; pp. 510–515. [CrossRef]
38. Liang, Z.; Zheng, G.; Li, J. Automatic parking path optimization based on Bezier curve fitting. In Proceedings of the 2012 IEEE International Conference on Automation and Logistics, Zhengzhou, China, 15–17 August 2012; pp. 583–587. [CrossRef]
39. Ravankar, A.; Ravankar, A.A.; Kobayashi, Y.; Emaru, T. SHP: Smooth Hypocycloidal Paths with Collision-Free and Decoupled Multi-Robot Path Planning. *Int. J. Adv. Robot. Syst.* **2016**, *13*, 133. [CrossRef]
40. Ravankar, A.; Ravankar, A.A.; Kobayashi, Y.; Emaru, T. Path smoothing extension for various robot path planners. In Proceedings of the 2016 16th International Conference on Control, Automation and Systems (ICCAS), Gyeongju, Korea, 16–19 October 2016; pp. 263–268. [CrossRef]
41. Mathematical Interpolation. Wikipedia 2016. Available online: https://en.wikipedia.org/wiki/Interpolation (accessed on 11 February 2016) .
42. Campana, M.; Lamiraux, F.; Laumond, J.P. A Simple Path Optimization Method for Motion Planning. Available online: https://hal.archives-ouvertes.fr/hal-01137844v2/document (accessed on 10 October 2019).
43. Park, C.; Pan, J.; Manocha, D. ITOMP: Incremental trajectory optimization for real-time replanning in dynamic environments. In Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS 2012), Sao Paulo, Brazil, 25–29 June 2012; pp. 207–215.

44. Garber, M.; Lin, M.C. Constraint-Based Motion Planning Using Voronoi Diagrams. In *Algorithmic Foundations of Robotics V*; Boissonnat, J.D., Burdick, J., Goldberg, K., Hutchinson, S., Eds.; Springer: Berlin/Heidelberg, Germany, 2004; pp. 541–558._32. [CrossRef]
45. Richardson, A.; Olson, E. Iterative path optimization for practical robot planning. In Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, San Francisco, CA, USA, 25–30 September 2011; pp. 3881–3886. [CrossRef]
46. Zhu, Z.; Schmerling, E.; Pavone, M. A convex optimization approach to smooth trajectories for motion planning with car-like robots. In Proceedings of the 2015 54th IEEE Conference on Decision and Control (CDC), Osaka, Japan, 15–18 December 2015; pp. 835–842. [CrossRef]
47. Kogan, D.; Murray, R.M. Optimization-Based Navigation for the DARPA Grand Challenge. In Proceedings of the 45th IEEE Conference on Decision and Control (CDC) in San Diego, CA, USA, 10–14 December 2006.
48. Roesmann, C.; Feiten, W.; Woesch, T.; Hoffmann, F.; Bertram, T. Trajectory modification considering dynamic constraints of autonomous robots. In Proceedings of the 7th German Conference on Robotics ROBOTIK 2012, Munich, Germany, 21–22 May 2012; pp. 1–6.
49. Rösmann, C.; Feiten, W.; Wösch, T.; Hoffmann, F.; Bertram, T. Efficient trajectory optimization using a sparse model. In Proceedings of the 2013 European Conference on Mobile Robots, Barcelona, Spain, 25–27 September 2013; pp. 138–143. [CrossRef]
50. Rösmann, C.; Hoffmann, F.; Bertram, T. Planning of multiple robot trajectories in distinctive topologies. In Proceedings of the 2015 European Conference on Mobile Robots (ECMR), Lincoln, UK, 2–4 September 2015; pp. 1–6. [CrossRef]
51. Rösmann, C.; Hoffmann, F.; Bertram, T. Kinodynamic trajectory optimization and control for car-like robots. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 5681–5686. [CrossRef]
52. Ravankar, A.; Ravankar, A.A.; Hoshino, Y.; Emaru, T.; Kobayashi, Y. On a Hopping-points SVD and Hough Transform Based Line Detection Algorithm for Robot Localization and Mapping. *Int. J. Adv. Robot. Syst.* **2016**, *13*, 98. [CrossRef]
53. Ravankar, A.A.; Hoshino, Y.; Ravankar, A.; Jixin, L.; Emaru, T.; Kobayashi, Y. Algorithms and a Framework for Indoor Robot Mapping in a Noisy Environment Using Clustering in Spatial and Hough Domains. *Int. J. Adv. Robot. Syst.* **2015**, *12*, 27. [CrossRef]
54. Ravankar, A.A.; Ravankar, A.; Emaru, T.; Kobayashi, Y. A hybrid topological mapping and navigation method for large area robot mapping. In Proceedings of the 2017 56th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE), Kanazawa, Japan, 19–22 September 2017; pp. 1104–1107. [CrossRef]
55. Mullane, J.; Vo, B.N.; Adams, M.D.; Vo, B.T. A Random-Finite-Set Approach to Bayesian SLAM. *IEEE Trans. Robot.* **2011**, *27*, 268–282. [CrossRef]
56. Huh, U.Y.; Chang, S.R. A G2 Continuous Path-Smoothing Algorithm Using Modified Quadratic Polynomial Interpolation. *Int. J. Adv. Robot Syst.* **2013**, *11:25.*, doi:10.5772/59463. [CrossRef]
57. Waring, E. Problems concerning Interpolations. *Philoshopical Trans. R. Soc.* **1779**, *69*, 59–67. [CrossRef]
58. Waring, E. *Problems concerning Interpolations*; The Royal Society Publishing: London, UK, 2015. Available online: http://rstl.royalsocietypublishing.org/content/69/59.full.pdf+html (accessed on 15 January 2016).
59. Song, B.; Tian, G.; Zhou, F. A comparison study on path smoothing algorithms for laser robot navigated mobile robot path planning in intelligent space. *J. Inf. Comput. Sci.* **2010**, *7*, 2943–2950.
60. Takahashi, A.; Hongo, T.; Ninomiya, Y.; Sugimoto, G. Local Path Planning and Motion Control for Agv in Positioning. In Proceedings of the IEEE/RSJ International Workshop on Intelligent Robots and Systems '89, The Autonomous Mobile Robots and Its Applications (IROS '89), Tsukuba, Japan, 4–6 September 1989; pp. 392–397. [CrossRef]
61. Pioneer P3-DX. Pioneer P3-DX Robot. 2019. Available online: www.mobilerobots.com/Libraries/Downloads/Pioneer3DX-P3DX-RevA.sflb.ashx (accessed on 2 May 2019).
62. TurtleBot 2. TurtleBot 2 Robot. 2019. Available online: http://turtlebot.com/ (accessed on 2 May 2019).
63. Wikipedia. Microsoft Kinect. 2019. Available online: https://en.wikipedia.org/wiki/Kinect (accessed on 2 May 2019).

64. UHG-08LX Technical Specifications. UHG-08LX Technical Specifications. 2018. Available online: https://autonomoustuff.com/product/hokuyo-uhg-08lx/ (accessed on 2 May 2018).
65. Quigley, M.; Conley, K.; Gerkey, B.P.; Faust, J.; Foote, T.; Leibs, J.; Wheeler, R.; Ng, A.Y. ROS: An Open-Source Robot Operating System. In Proceedings of the ICRA Workshop on Open Source Software, Kobe, Japan, 17 May 2009.