Robotics and Autonomous Systems 100 (2018) 171–185

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

A survey of robotic motion planning in dynamic environments

M.G. Mohanan [*], Ambuja Salgoankar

Department of Computer Science, University of Mumbai, Vidhya Nagari, Santacruz(East), Mumbai 400098, India

ARTICLE INFO

Article history:
Received 13 January 2017
Received in revised form 17 September 2017
Accepted 18 October 2017
Available online 2 December 2017

Keywords:
Robot motion planning
Dynamic environments
Collision avoidance

ABSTRACT

Robot Motion Planning (RMP) has been a thrust area of research in computing due to its complexity, since RMP in dynamic environments for a point robot with bounded velocity is an NP-hard problem. This paper is a critical review of the major contributions to RMP in dynamic environments. Between 1985 and 2015 the focus has changed from the classical approach to a heuristic approach. For velocity based motion planning in dynamic environments, ICS — AVOID (Fraichard and Asama, 2004, also see Section 2.4.4) is the safest approach which means that this method have the capability of for an autonomous robotic system to avoid collision with the obstacles in the environment. Other important approaches include artificial potential field based, artificial intelligence based, probabilistic based RMP and applications in areas of Agent systems and computer geometry. Classification of the RMP literature on the basis of the techniques and their performance has been attempted.

1. Introduction

Robot motion planning in dynamic environments is one of the areas of research in computer science and computational geometry. The fundamental problem of motion planning is obtaining a collision-free path from start to goal for a robot that moves in a static and totally known environment that consists of one or many obstacles [1]. Robot motion planning in dynamic environments [RMPDE] has been studied extensively [1–5]. Motion planning in dynamic environments with moving obstacles and moving targets is another thrust area [6–10]. There is sturdy evidence that a complete planner, i.e., one that finds a path whenever one exists and reports that no one exists otherwise, will take time exponentially with the number of degrees of freedom of the robot. RMPDE problem is NP-complete [1]. It was proved by [4] that dynamic motion planning for a point in the plane, with bounded velocity and arbitrary many obstacles, is not tractable and NP-hard. Three papers from around 1990 [1,4,11], along with an earlier paper [2] form the basis of research in RMP. This area has been comprehensively covered in a 2005 textbook [12].

In the present paper, we analyze the major aspects of the RMPDE and provide a classification of the diverse approaches of research for identifying recent trends and active areas. Approaches based on artificial potential fields, velocity, artificial intelligence and probability appear to be the most active areas of research. Motion planning in dynamic environments is not studied only in robotics but also in Agent systems and Computer Geometry. An important point revealed throughout the course of this study is that in spite of major advances in the area over the past three decades, not much work has been carried out on multi-robot systems. Different look ahead heuristics of the collision avoidance schemes TVDW [7] and NLVO [8], (see Section 2.4.3 for an explanation of these terms) truncate their future model and disregard any information beyond breaking time and heading time, respectively. When supplied with the same amount of information about the future evolution of the environment, ICS-AVOID is the best method when compared with the other two schemes. The paper is organized as follows: Section 2 gives an overview of the diverse approaches of RMPDE (Fig. 1) namely, artificial potential fields approach, velocity based approach, probability based approach, AI based approach, applications in Agent systems & Computer Geometry and also other popular methods for motion planning.

In the next Section

2. RMPDE approaches

2.1. Artificial potential fields (APF)

In this model, the goal is an attractive force field for the robot while obstacles are repulsive forces. A resultant force is calculated for the robot, with the direction of the force denoting the desired direction of motion and the magnitude of the force, the desired speed. The idea of employing time-varying potential field functions as an obstacle avoidance technique for robotic manipulators and mobile robots dates back to 1986 [11]. While the original study dealt solely with static obstacles, it was extended to incorporate dynamic ones.

[*] Correspondence to: 7/Sai Prasad, Yoga Institute Road, Santacruz(East), Mumbai 400055, India.
E-mail addresses: mohanan@udcs.mu.ac.in (M.G. Mohanan), ambujas@udcs.mu.ac.in (A. Salgoankar).

https://doi.org/10.1016/j.robot.2017.10.011
0921-8890/© 2017 Elsevier B.V. All rights reserved.

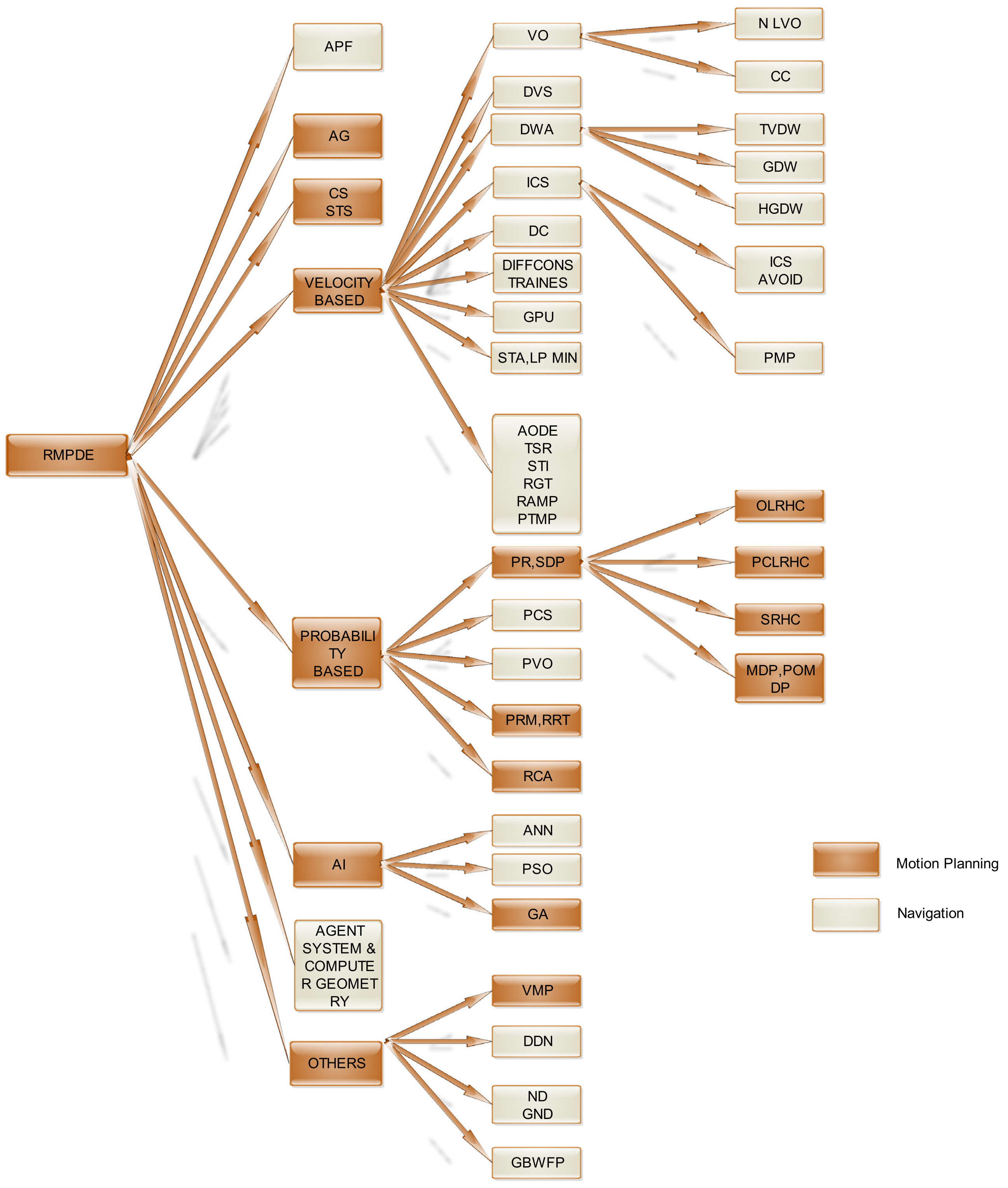

Fig. 1. Diagrammatic representation of diverse research approaches in RMPDE.

The robot is a point mass whose position, velocity and acceleration, target's position and velocity are known. The obstacles are convex polygon whose position and velocity can be measured. At each instant of time, it has been assumed that only one obstacle is close to the robot.

Avoidability measure, a function of the distance between the robot and the object, and the speed of the object relative to the robot, is employed to compute the possibility of a robot colliding with an obstacle [13]. The avoidability measure increases as the distance to the nearest obstacle increases and relative velocity decreases. This study uses a virtual distance function as its avoidability measure, which emphasizes the distance metric over the speed. This function can be tuned so that the robot begins to avoid obstacles closer or further away. It is then mapped to a potential force to be used with the traditional potential field method.

This method was extended for dynamic obstacles and moving targets in a number of studies. In [14] a similar concept was employed for path finding, modeling a robot's attraction towards the target and repulsion from the obstacles. This study addresses local minima issues that arise when an obstacle is between the robot and the target and moves in the same direction as the two, and when the robot is very close to the goal but cannot reach it because of an obstacle in between. Another study [15] addresses the issue of moving obstacles and targets in a robotic soccer scenario, by defining a relative threat function that lets the robot experience a repulsive potential while it is in a specific proximity range of an obstacle. A fractional potential is used in [16] to generate a partial path that considers the danger due to each moving object. Fractional potential ensures a continuous flow of potential among isolated sources that in turn avoids local minima.

2.2. *Accessibility graph (AG)*

Motion planning for a robot was studied in a time varying environment [11,17]. Each obstacle is a polygon that moves in a fixed direction at a constant speed. The destination point to be reached also moves along an identified trajectory. The concept of *accessibility* from a point to a moving object is introduced and is used to define a graph on a set of moving obstacles. If the point robot is able to move faster than any of the obstacles, then the graph shows an important property: a time-optimal motion is given as a series of edges in the graph. The complexities of the algorithm to find the motion is $O(n^2 \log n)$ where n is the number of vertices in the accessibility graph and it is time minimal.

2.3. *Configuration space (CS), state time space (STS)*

Configuration space was introduced by [2] in motion planning. Consider a single, rigid body $\mathcal{A}$ moving in $\mathcal{W}$, represented as a Euclidean space $\mathcal{R}^{\mathrm{d}}$, with $d = 2$ or $d = 3$. $\mathcal{W}$ has a fixed Cartesian coordinate frame, $\mathcal{FW}$. $\mathcal{A}$ is represented at a reference position and orientation as a subset of $\mathcal{R}^{\mathrm{d}}$. A body-fixed frame $\mathcal{FA}$ is attached to $\mathcal{A}$. A *configuration* of $\mathcal{A}$, denoted as q, is a measurement of the position and orientation of $\mathcal{FA}$ with respect to $\mathcal{FW}$. The configuration space, represented as $\mathcal{C}$, is the space of all configurations of the robot. A configuration is a point in this configuration space representation. The subset of the workspace $\mathcal{W}$ that is occupied by a configuration q of $\mathcal{A}$ is denoted as $\mathcal{A}(q)$. In similar fashion, a point a in $\mathcal{A}(q)$ is denoted as $a(q)$.

The task is to find a path π in the form of a continuous sequence of configurations of $\mathcal{A}$, from an initial configuration q_i to a goal configuration q_g, that do not collide or contact with O_i. A collision or contact is defined by mapping the obstacles into the configuration space under consideration. A $\mathcal{W}$-obstacle in $\mathcal{C}$ is called a $\mathcal{C}$-obstacle and is defined as $\mathcal{C}O_i = \{q \in \mathcal{C} | \mathcal{A}(q) \cap O_i \neq \phi\}$. The union of these configuration space obstacles, $\mathcal{C}O = \cup_{i=1}^{nk} \mathcal{C}O_i$ is called the *configuration space obstacle region*. Its complement, $\mathcal{C}_{\text{free}} = \mathcal{C} | \mathcal{C}O$ is called the *free configuration space*.

The basic motion planning problem can now be defined as finding a path from q_i to q_g in $\mathcal{C}_{free}$. A *path* is defined as a continuous function π that maps a path parameter s (usually taken in unit interval [0, 1]) to a curve in $\mathcal{C}_{free}$. So a path is defined as the continuous function $\pi : [0, 1] \rightarrow \mathcal{C}_{free}$. Where $\pi(0) = q_i, \pi(1) = q_g$ and $\pi(s) \in \mathcal{C}_{\text{free}}, \forall s \in [0, 1]$.

The concept of state time space, i.e. the state space of the robot augmented with the time dimension was introduced in [18]. Like configuration space which is a tool to prepare and design path planning problems, state-time space is a tool to prepare and design trajectory planning in dynamic workspace problems. It permits the study of different aspects of dynamic trajectory planning, such as moving obstacles and dynamic constraints, in a integrated way. This method is applied to the case of a four wheeler robot subject to dynamic constraints and moving along a given path in a dynamic workspace. A time-optimal approach that searches the solution trajectory over a restricted set of canonical trajectories is used. These trajectories are defined as having discrete and piecewise constant acceleration.

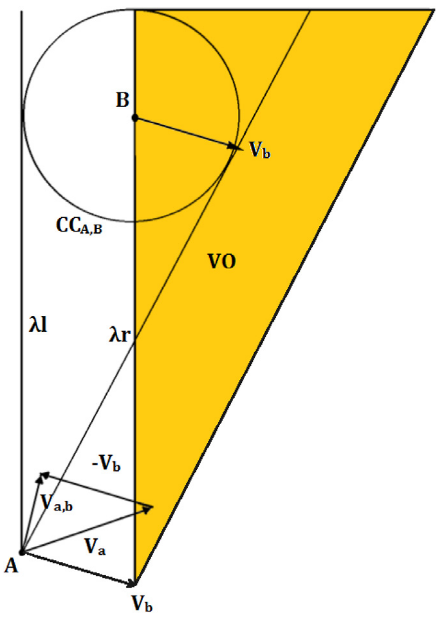

Fig. 2. Velocity obstacles [3].

2.4. *Velocity based motion planning*

2.4.1. *Velocity obstacles (VO)*

The complete problem of motion planning can be divided into two separate problems:
kinematic and dynamic. The kinematic problem consists of finding a trajectory that takes into account the position and velocity of the obstacles as well as an approximation of the dynamic constraints of the robot. The dynamic problem consists of computing an optimal trajectory that satisfies the full set of kinematic and dynamic constraints, and is in a close neighborhood of the solution to the kinematic problem. Therefore the approach to the solution of the complete motion planning problem consists of two steps: the first step computes a kinematic trajectory that solves the kinematic problem, and the second step uses dynamic optimization to optimize motion time subject to the dynamic constraints, using the kinematic trajectory as an initial guess. Ultimately the approach fails if the second step is not able to find the path from the first step.

The trajectories of robots moving in a time-varying environment are computed by using the concept of velocity obstacles (VO), which denote the robot's velocities that would cause a collision with obstacles at some near-future time [3,19–21]. An avoidance maneuver is computed by choosing velocities that are outside of the velocity obstacles. To ensure that the maneuver is dynamically feasible, robot dynamics and actuator constraints are mapped into the velocity space. A trajectory consists of a series of avoidance maneuvers, computed by searching over a tree of avoidance maneuvers created at distinct time intervals. For real time applications, the tree is pruned using a heuristic search designed to attain a prioritized set of objectives, such as avoiding collisions, searching

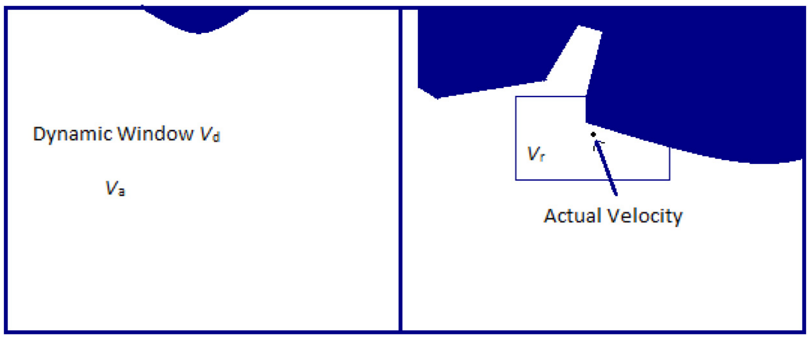

Fig. 3. Dynamic window [22].

the goal, optimizing speed or computing trajectories with desirable topology. To evaluate the quality of these trajectories, they are compared to the trajectories computed using the kinematic trajectory as an initial guess.

Definition (*Using Mayer's Notation [3]*). Mathematically, the planning problem consists of computing the control $u(t) \in U$ in $t_0 \leq t \leq t_f$ that minimize the performance index $J = \Phi(X_{(tf)}, tf) + \int_{t0}^{tf} L(x, u, t)\,dt$ subject to two sets of constraints:

kinematic constraints

initial manifold: $\Gamma(x(t0), t0) = 0$

final manifold: $\Omega(x(tf), tf) = 0$

obstacles: $\psi: \bigcup_{i=1}^{n} [Si(x(t), t) = 0]$

dynamic constraints

robot dynamics:

$$\dot{x} = F(x, u) = f(x) + g(x)u$$

actuator constraints:

$$u_i(\min) \leq u_i \leq u_i(\max)$$

For RMP either the robot's environment could be broken up into spatial states [2,4] or it could be discredited into a finite number of linear and angular velocity pairs. In the latter case, the reactive obstacle avoidance during the path planning has been facilitated by using VO.

The VO is the vector sum of collision cone with the velocity vector of the obstacle (see Fig. 2). The VO represents a region in the velocity space of the robot that would lead to a collision with the obstacle within a time horizon. Obstacle avoidance is carried out by creating a set of reachable avoidance velocities defined by the dynamic constraints of the vehicle. The approach consists of refining the trajectory with a dynamic optimization that, subject to the robot's dynamics, actuator constraints and time-varying obstacle constraints, minimizes motion time. The dynamic optimization is based on Pontryagin's *minimum principle* [3] and uses a gradient descent method. It works for an obstacle that moves with a constant linear velocity.

The advantages of VO approach are (a) proper geometric representation of maneuvers avoiding any types of obstacles (b) simple consideration of robot dynamics and actuator constraints.

2.4.2. Dynamic velocity space (DVS)

DOV is the dynamic object velocity, given by DOV $= \{(\nu, \omega, t) | (\nu, \omega) \in [\mathrm{INT}(V_{\mathrm{DOV}}) \cup \delta V_{\mathrm{DOV}}]\}$, where ν is the linear velocity, ω is the angular velocity, t is the time, INT denotes the set of velocities belonging to V_{DOV} neighborhood and δ the velocities of the outline of V_{DOV}. The dynamic object velocity set DOVS $= \cup_{i=1}^{nk}(\mathrm{DOV}_i)$, $V_{\mathrm{free}} = \{(\nu, \omega) | (\nu, \omega) \in [V_{\mathrm{adm}} \Theta V_{\mathrm{DOVS}}]\}$, where V_{adm} is the set of acceptable velocities limited by v_{max} and w_{max} and Θ the operation of set difference. Dynamic velocity space (DVS) is the velocity time space that includes the DOVS and V_{free} information. DVS is a method for modeling the dynamics of the environment along with the robot constraints. It is obtained by mapping the configuration space to a velocity space by using the idea of *estimated arriving time* to calculate the time to collision and the time to escape from collision with obstacles. An optimization heuristic based on the trajectory with minimum time or the shortest path or both has been proposed to compute velocity commands directly in the velocity space while avoiding static and moving obstacles [23].

2.4.3. Dynamic window approach (DWA)

DWA is an extension of DVS for robots that follow a circular path with translational and rotational velocities. If "look ahead time" is the time taken by the robot to stop and no collisions occur during the interval, then the velocity of the robot at that time is considered to be an admissible velocity V_a [24] (see Fig. 3). In other words, a velocity is admissible if it allows the system to stop before hitting an obstacle.

DWA is a velocity space based local avoidance scheme where search for admissible control is carried out in the space of velocities (VS). The search space is reduced by the system kinematic and dynamic constraints to a set of reachable velocities (V_r) in a short time interval Δt around the current velocity vector.

$$V_r = \{(\nu, \omega) | \nu \in [\nu_c - \dot{\nu}_b \Delta t, \nu_c + \dot{\nu}_b \Delta t] \wedge \omega \in [\omega_c - \dot{\omega}_b \Delta t, \omega_c + \dot{\omega}_b \Delta t]\} \quad (1)$$

where $\dot{\nu}_a$, $\dot{\omega}_a$, $\dot{\nu}_b$ and $\dot{\omega}_b$, are the maximal translational/rotational accelerations and breakage decelerations.

$$V_a = \left\{ (\nu, \omega) / \nu \leq \sqrt{2\rho \min(\nu,\ \omega) \dot{\nu} \mathrm{b}} \wedge \omega \leq \sqrt{2\rho \min(\nu,\ \omega) \dot{\omega} \mathrm{b}} \right\} \quad (2)$$

The first step in DWA is to prune the overall search space by considering only the next steering command and producing a two-dimensional search space of circular trajectories [22]. After that, the search space is reduced to the admissible velocities that allow the robot to stop safely without collision with an obstacle. Finally, the dynamic window reduces the admissible velocities to those that can be reached within a small time interval given the limited accelerations of the robot. The strategy is that the robot picks a trajectory at which it can maximize its translational velocity and the distance to the obstacles and at the same time minimize the angle of its goal relative to its own heading direction by optimizing the objective function [25].

Time varying dynamic window (TVDW), as given in Fig. 4, is a variant of DWA that computes a set of immediate future obstacles trajectories in order to check for collision in the near future.

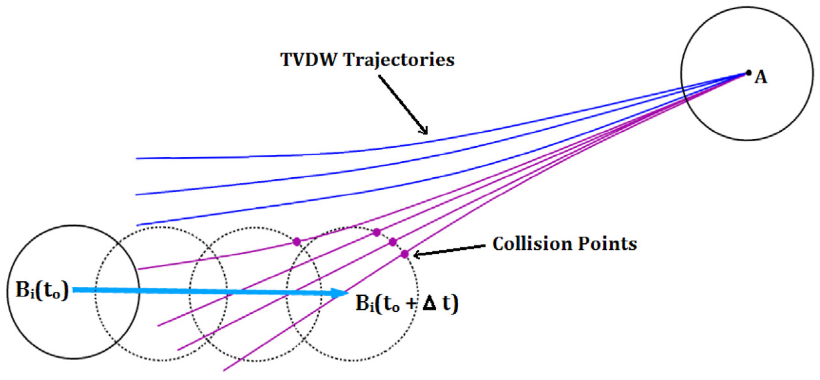

Fig. 4. Time varying dynamic window [8].

The global dynamic window (GDW) approach is a generalization of the DWA which combines methods from motion planning and obstacle avoidance [26–29]. It is a method that allows execution of high velocity, destination oriented motion for a robot in unknown and dynamic environments. The GDW approach and holonomic GDW (HGDW) [30] are extensions of these methods well suited for changing and dynamic environments. Non-linear velocity obstacles have been modeled using a wrap cone in NLVO [7].

Velocities giving a collision after a given time horizon t_H are considered as acceptable. VO was extended by NLVO to consider a known velocity outline for the moving object. NLVO consists of all velocities of $\mathcal{A}$ at t_o that would result in a collision with $\mathcal{B}$ at any time $t_o \le t \le t_H$. NLVO(t) is a scaled $\mathcal{B}$, bounded by the cone formed between $\mathcal{A}$ and $\mathcal{B}(t)$, NLVO is a warped cone with apex at $\mathcal{A}$ and NLVO $= \bigcup_{t_0 \le t \le t_H} \frac{B(t)}{t - t_0}$ as in Fig. 5.

Implementation of this concept is found in [32]. The algorithm estimates time-to-collision for each of the NLVOs, using the A^* algorithm to search through the velocity-space of the robot [33].

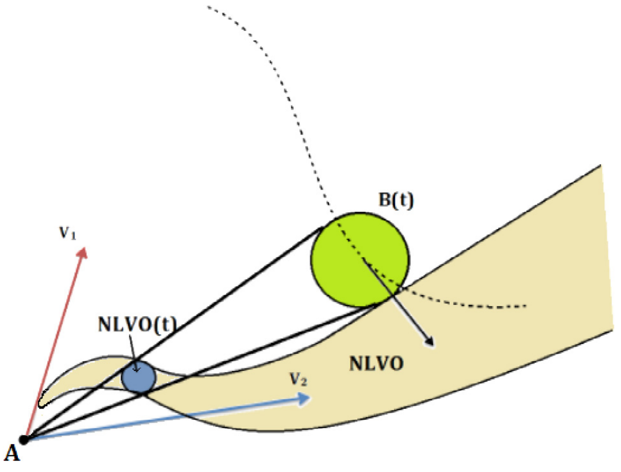

Fig. 5. Non-linear velocity obstacles.

2.4.4. Inevitable collision state (ICS) and ICS-AVOID

The dynamics of $\mathcal{A}$ is described by a state transition equation of the form $\dot{s} = f(s, u)$, where $s \in S$ is a state of $\mathcal{A}$, $\dot{s}$ is time derivative and $u \in \mathcal{U}$ a control, and S and $\mathcal{U}$ denote the state space and the control space of $\mathcal{A}$. Let $s(t)$, $\mathcal{A}$'s state at time t and $\mathcal{A}(s)$ represents the closed subset of $\mathcal{W}$ occupied by $\mathcal{A}$ when in the state s. Let $\tilde{u}$: $[t_0, \infty] \to \mathcal{U}$ denotes a control trajectory, $\tilde{u}(t)$ denote the element of $\tilde{u}$ at time t. All possible control trajectories over $[t_0, \infty]$ is denoted by $\tilde{U}$. Let $\mathcal{B}_i$ denote a work space object for $i = 1, \ldots, n_k$, and $\mathcal{B}$ denote the union of the work space objects, $\mathcal{B} = \cup_{i=1}^{nk} \mathcal{B}_i$.

ICS is a state for which the robotic system is in a collision irrespective of the future trajectory.

ICS Definition [6]:

$$\mathrm{ICS}(\mathcal{B}) = \{s \in S | \forall \tilde{u} \in \tilde{U}, \exists t, \mathcal{A}(\tilde{u}(s,t)) \cap \mathcal{B}(t) \neq \phi\}. \tag{3}$$

It is possible to define the ICS set yielding a collision with an object $\mathcal{B}_i$

$$\mathrm{ICS}(\mathcal{B}_i) = \{s \in S | \forall \tilde{u} \in \tilde{U}, \exists t, \mathcal{A}(\tilde{u}(s,t)) \cap \mathcal{B}_i(t) \neq \phi\}. \tag{4}$$

The ICS set giving a collision with $\mathcal{B}_i$ for a given trajectory $\tilde{u}$, or a particular set of trajectories $I \subset \tilde{U}$ is given by

$$\mathrm{ICS}(\mathcal{B}_i, \tilde{u}) = \{s \epsilon S | \exists t, (\tilde{u}(s,t)) \cap \mathcal{B}(t) \neq \phi\} \tag{5}$$

$$\mathrm{ICS}(\mathcal{B}_i, I) = \{s \epsilon S | \forall \tilde{u} \in I, \exists t, \mathcal{A}(\tilde{u}(s,t)) \cap \mathcal{B}_i(t) \neq \phi\}. \tag{6}$$

By definition a robotic system in a non-ICS state has at least one non-intersecting trajectory that it can use. ICS-AVOID, a method for "collision avoidance by design", has been introduced by [34] and the safety of the robot is guaranteed. When supplied with the same level of information about the future advancement of the environment, ICS-AVOID is the best performing method as compared with the dynamic window (DW) and the velocity obstacle (VO) approaches [8,35]. Two factors are important: the extent to any collision avoidance scheme reasons about the future and its ability to find a safe control if one exists.

2.4.5. Partial motion planning (PMP)

The first step of the PMP is to get the model of the environment which can be given as a priory or built using sensor observations. The iterative PMP algorithm gives significance to the planning time constraints and the validity duration of the model of the environment. The PMP algorithm consists in iteratively exploring the state-time space during a fixed limited time, by building a tree using probabilistic techniques and the optimum trajectory is computed from the tree. During a cycle, a complete trajectory calculation to the goal cannot be guaranteed. At each time-out a PMP algorithm returns the best partial motion (i.e., one computed so far) to the goal. Safety against the system reaching to an inevitable collision has been provided by computing an ICS-free partial motion in which at least one collision free trajectory can be obtained [9].

2.4.6. Directive circle (DC)

A sensor-based online method is presented for generating a collision free path for differential-drive wheeled robots aiming a moving target surrounded by dynamic and static obstacles [31,36,37]. Robot is circular and move in any direction with a maximum speed which is known in the beginning of planning. There is only one target and its velocity is proportional to the robot's velocity. Obstacles can be of polygonal shape and their instantaneous speed vector is unknown to robot. Robot is having detectors or range sensors to get the velocity profile of obstacles.

The velocity of the obstacle is $\overline{VO}$ and the Collision Cone, $CC_{R,OB}$ can be defined as the set of relative velocities between the colliding R and OB

$$CC_{R,OB} = \{V_{R,OB}/\lambda_{R,OB} \cap OB \neq \phi\}$$

where $\overline{V}_{R,OB} = \overline{V}_R - \overline{V}_O$ is the relative velocity of R with respect to OB, and $\lambda_{R,OB}$ is the direction of $\overline{V}_{R,OB}$, λr and λt represent the two tangent to the obstacle, and are determined in visibility scan. For every robot, obstacle pair, there is a unique Collision Cone. In DC, we can obtain the forbidden directions as set of directions in $\lambda_{R,OB}$ lied on $CC_{R,OB}$. For obtaining the Directed Circle, the position of the robot along − VO must be shifted and then draw a circle C around the robot with a radius of maximum velocity of R. Geometrically, one case out of four cases may occur

(i) the circle intersect with λ_r
(ii) the circle intersect with λ_l
(iii) the circle intersect λ_r and λ_l
(iv) the circle intersect none of them.

As shown in Fig. 6, the circle C has intersection with λ_l at he point A and λr with B. If the slope of $\lambda_{R,OB}$ falls between the slope of PA and PB vectors, the robot will collide with the obstacles. C is the Directive Circle. If the circle C does not have any intersection points with the tangent λr and λl, DC does not have any forbidden zone, and robot can freely move along the calculated optimal direction to the target. If whole circle lies in $CC_{R,OB}$, then all possible directions are prohibited and the robot should stop. At each iteration, the set of all collision-free directions are computed using velocity vectors of the robot related to each obstacle, thus forming the directive circle (DC). Then the best visible direction close to the optimal direction to the target is selected from the DC, which prevents the robot from being trapped in local minima. The movements of the robots are governed by the exponential stabilizing control method that provides a proper motion at each step while considering the robot's kinematic constraints so that the robot is able to achieve the target. Comparing with RRT based off line path planning methods this online algorithm take the robot to the near optimum path with more efficiency. The method have safety issues in the case of narrow passage problem and the abrupt change of velocity of obstacles.

2.4.7. Differential constraints

An approach to the problem of differentially constrained mobile robot motion planning in arbitrary time-varying cost fields is presented in [38]. The method is based on discretization of robot state space called *state lattices*. It represents a graph whose vertices are a discretized set of all reachable states of the system, edges are feasible motions and controls which connect these states. The motions represented in the edges of the *state lattices* make a repeated unit that can be copied to every vertex with the property that each edge joins neighboring vertex exactly once. A search space is constructed which is properly suited to the requirements of dynamic environments, including feasible motion plans that satisfy differential constraints, efficient plan repair with high update rates, and deliberative goal-directed behavior. The search space consists of edges which adapt to the state sampling resolution and acquire states in order to permit the use of the dynamic programming principle without any infeasibility. It is a lattice depends

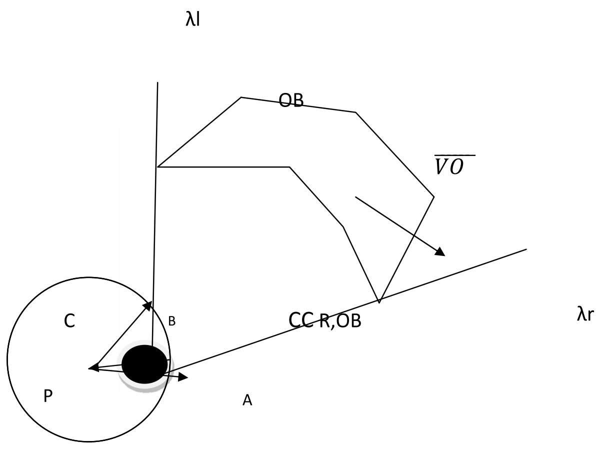

Fig. 6. Directed circle [31].

on a repeating unit of controls which allows calculation of the planner heuristic. It gives simulation of motion and swept volumes associated with each motion. The search space features superior resolution near the vehicle and reduced resolution far away and thus adds efficiency to the RMP. Advantages of this method are fast online performance by managing the fidelity of representations and quick robot reaction to the change environments.

2.4.8. Graphic processor unit (GPU)

An algorithm to compute collision-free trajectories in dynamic environments has been proposed [39]. It employs a re-planning framework that incorporates optimization-based planning with execution and uses parallel techniques to manage dynamic environments. It is a trajectory computation to an optimization problem which minimize the objective function which consist of costs corresponds to static obstacles, dynamic obstacles, problem based constraints and smoothness. No a priori knowledge about the obstacles or their motion is required. It also describes a parallel formulation that exploits a high number of commodity graphics processors to evaluate a good-quality path in a given time interval and derive bounds on parallelization in order to improve the accessibility of the planner, responsiveness and the quality of the trajectory.

2.4.9. Search tree algorithm (STA), LP minimization

A collision avoidance scheme to develop the safety conditions in situations with multiple unmanned aerial vehicles (UAV) sharing the same aerial space with other aircrafts or mobile obstacles is proposed [40]. The method modifies the velocity profile of the UAV under control maintaining the paths initially planned. An heuristic method based on the combination of a Search Tree algorithm which finds a solution if it exists, and the minimization of the cost function, that will use the information obtained by the STA. The Search Tree will obtain a valid order of pass for the vehicles in a given conflict, allowing to formulate the problem to be solved as a LP problem. The approach considers the UAV representation and the distances traveled by the UAVs in each cell. Finally, it will be assumed that the UAVs are initially moving at their maximum speed, as long as it allows to perform the goal in a minimum time.

2.4.10. Atomic obstacles and dynamic envelope (AODE)

AODE involves a sensor based real-time detection of an obstacle. If a robot trajectory is going to be a curve in an unknown configuration-time space, then irrespective of the movements of the obstacles, a collision free path is guaranteed [41]. The process works independent of the obstacle geometry. It does not call for identification of obstacles or prediction of the obstacle movements. Therefore it is useful for guiding mobile robots and manipulators in unknown and unpredictable environments. As far as the robot is moving along a detected collision-free trajectory, its safety is guaranteed.

2.4.11. *Temporal and spatial reshaping (TSR)*

TSR is a planning structure for developing 3-D collision-free-motions that take complex robot dynamics into account [42]. The two stages that are applied iteratively include (i) obtaining a collision-free path through a geometric and kinematic sampling based motion planning and (ii) transforming the path into dynamically executable robot trajectories generated through a set of dedicated dynamic motion generators. Temporal or spatial reshaping methods are used to handle detected collisions depend upon the application requirements. Temporal reshaping regulate the velocity, whereas spatial reshaping changes the path itself.

2.4.12. *Safe time interval (STI)*

STI is a planner that follows the following observation: While the number of safe time steps in any configuration are unbounded, the number of safe time intervals in a configuration is finite [43]. A safe time interval is a duration for a configuration with no collisions that, extended by one time step in every direction, leads to a collision. The planner uses this observation and constructs a search-space with states defined by their configuration and safe interval, resulting in a graph that has only a few states per configuration.

2.4.13. *Rendezvous-guidance technique (RGT)*

RGT provides a simultaneous positional interception and velocity tuning of the target moving in a dynamic environment with static and dynamic obstacles by using the principles of parallel navigation law and line of sight [44]. The generation of rendezvous-guidance algorithm consists of steps: (a) receive the instantaneous state of the obstacles and the robot (b) determine the feasible velocity region of the robot for the current state (c) obtain the Rendezvous Line (RL), a parameterized line obtained from the position vector of robot & velocity of target and hence obtain the maximum closing velocity (d) construct the Rendezvous Set (RS), a set of points within RL (e) find the velocity of robot based on the intersection of RS and feasible velocity region.

The RGT system consists of three modules: (i) Vision modules consists of Image acquisition, Image processing and Transforming to world coordinates which gives the position and velocity state vectors of all objects in the work space and give it to the robot path planner module. (ii) RG-Modified Exact Cell Decomposition Algorithm for the robot path planner. The RG method first finds the maximum desired velocity to be obtained in the next instant of time and then compute the acceleration command to achieve this velocity. The improved acceleration command is obtained in the work through a modified cell decomposition method which divides the workspace into triangles and nodes are placed at the midpoint of the edges of the triangles. A weighting factor is used to combine these acceleration by giving more weightage to RG when the obstacle is far away from the robot and otherwise weightage emphasis will be for MECD acceleration command and hence give to the Controller module. (iii) Robot controller module receives the combined acceleration command to converts into necessary velocities to move the robot to the target.

2.4.14. *Real-time adaptive motion planning (RAMP)*

The difference between AODE and RAMP is that the AODE is for motion planning for robots with low degrees of freedom and the RAMP is for ones with high degrees of freedom [45]. Simultaneous planning of path, trajectory and execution of motion in real time is a distinguishing characteristic of RAMP. It provides real-time optimization of trajectories under various optimal criteria, such as minimizing energy, cost and time and maximizing flexibility. Loose coupling of robot configuration variables has been employed to overcome the redundancy in a redundant robot. The RAMP method has been tested and implemented over different sets of task environments with multiple mobile robots.

2.4.15. *Predictive temporal motion planning (PTMP)*

PTMP is an approach to the problem of obtaining solution trajectories in dynamic environments by connecting the tasks of obstacle identification, prediction, environment mapping and planning [46]. The dynamic environment is represented by grids which are extended into the time dimension by adding time layers to the grid structure each representing a distinct time-step into the future. The time-steps are determined by considering the way the motion planning algorithm calculates its discrete control commands. The future positions of moving obstacles are predicted and displayed in the layer of the temporal grid associated with the prediction times. The predictive temporal grid is used to explore potential control input sequences to make an optimal trajectory. The algorithm evaluates control commands at various futuristic time-steps by evaluating the various temporal layers of the grid structure that corresponds to the distinct control times. The estimated future motions of any obstacles play a vital role in efficient RMP.

All the schemes proposed have two purpose, collision avoidance and reach the goal in an optimum level. The robot moves in real world in dynamic environments, the motion safety is not guaranteed which means that collision can be made even if they have full knowledge of the environments future behavior. Completeness of the algorithm depends upon the assumptions and hypothesis made about the behavior of the obstacles. In the absence of any information about the intensions and performance of the obstacles, there is no guarantee that the planner can obtain a solution even if one exists. In comparison with all the approaches, ICS, VO and DVS have taken more account for the dynamic nature of the environments. With respect to the kinematic and dynamic constraints accountability, the approaches DW, ICS and DVS are prominent. ICS and VO represent the continuous set of velocity and ND, DW and DVS are discretized. ICS is the only approach have the representations of uncertainty in perception system. The complexities of the approaches DW and VO are the lowest and other velocity based methods have higher complexities. VO is based on angular velocities and it does not consider the kinematic constraints of the robot.

2.5. *Probability based motion planning (PMP)*

Robots in dynamic and uncertain environments (DUE) require interpretation about future developments and uncertainties about the states of the dynamic agents and obstacles [47–49]. A solution approach for solving chronological decision models based on inductive computation is the Dynamic programming (DP). DP has been employed here to account for future information assimilation and the quality of that information in the planning process. Dynamic Programming gives a basis for compiling planning results into strategies for control and for learning such strategies when the system being controlled is partially known. Stochastic dynamic programming (SDP) generates an approximate solution to RMP in DUE [50]. The limits of the time horizon in partially closed-loop receding horizon control (PCLRHC) are constrained.

If the most likely measurement, $\tilde{y}_i = E[y_i|I_{i-1}^{PCL}]$, is assumed for future measurements, the limited information set is: $I_i^{PCL} = (y_1, \ldots, y_k, \tilde{y}_{k+1}, \ldots, \tilde{y}_i, u_k, \ldots, u_{i-1})$.

The belief state associated with I^{PCL} is

$$b_i^{PCL} = \mathcal{P}(x_i|I_i^{PCL}) = p(x_i|u_{0:i-1}, y_{1:k}, \tilde{y}_{k+1:i}) \quad (7)$$

And the state transition function is defined as

$$b_{i+1}^{PCL} = f_b^{PCL}(b_i^{PCL}, u_i, \tilde{y}_{i+1}) \tag{8}$$

This belief state is defined in terms of the control sequence. The resulting minimization problem solved by PCLRHC is:

$$\min_{u_{k:M}} c_M(b_M^{PCL}) + \sum_{i=k}^{M-1} c_i(b_{i+1}^{PCL}, u_i) \tag{9}$$

such that

$$b_{i+1}^{PCL} = f_b^{PCL}(b_i^{PCL}, u_i, \tilde{y}_{i+1}) \tag{10}$$

$$P(\mathcal{g}_b(b_i^{PCL}, u_{i-1}) \leq 0) \geq \alpha \quad \forall i = k \dots M. \tag{11}$$

While accounting for chance constraints that occur from the uncertain locations of the robot and obstacles, the solution integrates prediction, assessment and planning. It is reported in [50] that when tested in a simulated environment, PCLRHC in simple static and dynamic scenarios outperforms the open loop receding horizon control (OLRHC). The computational difference between the Open Loop Receding Horizon Control and Partially Closed-Loop Receding Horizon Control approaches lies in the transmission of the belief states. The belief state transition function is obtained from Bayes' rule. For linear systems with Gaussian noise, the belief state development is given by the Kalman filter. For the OLRHC, only the prediction step is executed, which is known to scale as $O(n^3)$, where n is the dimension of the space [50].

For the Partially Closed-Loop Receding Horizon Control, the additional covariance update involves matrix inversion and matrix multiplications, which also scales as $O(n^3)$. Thus, the complexity of this step differs from that of the OLRHC method by a constant factor.

2.5.1. Probabilistic robot (PR)

Probabilistic robotics is a paradigm for robot programming [51]. The probabilistic model underlines the inherent uncertainty in robot observation depending on representations of uncertainty when determining what is to be done in the next instant of time [52]. The main assumption is that probabilistic methods of robotics do better in complex real-world applications than methods that ignore a robot's uncertainty.

A time-variant probabilistic collision state checker system is introduced in [34,53–55,91], which finds a safe route with a minimum collision probability for a robot. A sequential Bayesian model has been employed to approximately estimate the movement patterns of the obstacles. A time-reliant variation of Dijkstra's algorithm has been formulated to compute safe trajectories through a crowded neighborhood.

Localization, i.e., the estimation of a robot's location from sensor data, is a basic problem in mobile robotics. A version of Markov localization presented in [56] which provides position estimates that are adapted to dynamic environments. The principle of Markov localization is to keep a probability density function over the space of all locations of a robot in its environment. Space is a 2-D matrix with a fine-grained grid that approximates probability densities. One of the uncertainties resolved by probabilistic robots is failure of localization in which robot's local coordinates are lost by some mishap like non-functioning of sensors. A probabilistic approach is able to globally localize the robot from scratch to recover the situation. It can manage noisy information of the environment such as occupancy grid maps and ultrasound sensors. This is an iterative procedure which allows a mobile robot to reliably estimate its position even in densely populated environments in which several obstacles block the robot's sensors for extended periods of time.

A parallel and incremental approach for learning and predicting motion patterns based on a growing hidden Markov model has been discussed in [57]. It is a time dependent Hidden Markov Model with continuous observation variables. When a new observation sequence is available, the number of discrete states, structure and the probability parameters such as state prior, transition and observation are updated every time. Structure learning is obtained by constructing a topological map which is represented by a graph whose nodes are region of the space, edges connect neighboring nodes and updating of this map is done through an algorithm. Parameter learning is obtained by using the incremental Expectation–Maximization approach by taking likelihoods are weights to update the state or transition of the model.

2.5.2. Probabilistic collision state (PCS)

In busy environments with many humans or robots, the possible set of inevitable collision states (ICS) is very high, with the result that the robot has to stop and wait in too many situations [58]. For this reason, the idea of ICS is extended to probabilistic collision states (PCS), which evaluates the collision probability for a given state. A planning algorithm is executed if the collision probability is within a certain threshold. The method also employs mechanisms such as controlling the speed of a robot or an obstacle in order to ease the risk of a collision. The results show a considerable difference in interaction behavior of robots and obstacles with the result that the approach is suited for active and non-deterministic dynamic obstacles in the robot workspace. The technique has implementation issues, as it has an infinite number of input trajectories and a limited time horizon. The infinite number of trajectories can be solved by computing only a finite subset of input trajectories. The problem of infinite breaking time can be resolved by applying breaking maneuvers that come to an idle state after a finite time horizon [59].

2.5.3. Probabilistic velocity obstacle (PVO)

PVO is a solution to the several autonomous navigation systems in the dynamic environment that fail to perform satisfactorily because they do not consider dynamics of the obstacles or the limits of the sensor system [60]. A dynamic occupancy grid is created to hold probability values of the VO. Uncertainty in position, shape and velocity of the obstacles, occlusions and limited sensor range, have been resolved. This input is fed into a simple navigation algorithm. As it is a reactive algorithm, the planning solution does not guarantee that robot will reach the goal or it would put in emergency situations. Also the method is not efficient for non-linear obstacles [61,62].

2.5.4. Probabilistic roadmap (PRM) and rapidly-exploring random tree (RRT)

PRM was developed for single car-like robots in a static environment and was later applied to multiple robots [63,64]. First, the random samples are generated uniformly in configuration space and each sample is tested for collision between robot and the obstacles. Only the free samples are stored. Each sample is connected by an edge to its k-nearest collision-free samples. This result in a roadmap represented by a graph-like structure in which path can be found using standard graph-search method. A path from source to a vertex is possible if there exists a direct collision-free path between the two or else if a collision-free path from the source to one or many configurations in the roadmap and a similar path exists from one or many configurations in the roadmap to the target.

For the multi-robot situation, a "super-graph" is built from the individual roadmaps of the robots. When moving along a particular edge in the map, areas swept by the robots are checked against each other and not permitted if they intersect. This method is probabilistically complete, i.e., given enough time, the planner necessarily finds a solution to any query (a feasible path) if one exists. It assumes static environments and roadmaps are generated off-line.

The dynamic environments are taken care of by constructing temporal roadmaps that are connected to some initial and final state in a map [65]. The use of PRM in the state-time space of a robot by introducing the concept of roadmap time space in planning is discussed [10]. It is belief that the trajectories of the dynamic obstacles are known *a priori*. The A* algorithm is employed for a global-level search of a trajectory that is passed on to the local planner that in turn uses a depth-first search to investigate the edges of the roadmap in state-time space. Connecting two disconnected configurations by putting an intermediate artificial configuration in between is termed as expansion [66].

A rapidly exploring random tree (RRT) is an algorithm planned to search non convex high-dimensional spaces by randomly building a tree [67]. The tree is constructed by random samples drawn from the search space and is influenced to grow towards unsearched areas of the problem. The method grows a tree rooted at the starting configuration from the search space by using random samples. A connection is attempted between the sample and the nearest state in the tree, as and when each sample is drawn. If the connection is feasible then addition of the new state to the tree. RRT growth can be influenced by changing the probability of sampling states [68].

An incremental learning algorithm for on-line management of roadmaps for a planning query has been discussed [69]. Unlike conventional PRM it uses a rapidly-exploring random tree (RRT) structure to let the map grow. Efficiency in the creation of a new RRT is because of the learning that has taken place while creating RRTs in past. Different path-finding queries generating different RRTs leads to a forest of RRTs, termed as the reconfigurable random forest (RRF). This structure is kept manageable by pruning out the invalid nodes and branches that are affected by moving obstacles. The issue of PRM-based planners spending most of their time in checking for collisions while constructing the roadmap has been addressed here [69]. The fundamental assumption here is that a moving object only partially changes the workspace of the robot. If an important area of the roadmap is broken by a moving obstacle, the algorithm employs an RRT planner to attempt to locally reconnect the endpoints of the blocked edges. In case this local reconnection solution fails, the extra nodes are placed in the map near the broken edges and then connected to allow for more options. This algorithm also stores past positions of moving objects to reduce the number of collision checks required for a particular edge in the map.

A method by employing a Flexible Anytime Dynamic A* PRM (FADPRM) algorithm has been implemented by allowing the planner to use a roadmap to provide a sub-optimal path quickly [70]. Based on the different degrees of desirability of traversal, robot's workspace is divided into zones. The algorithm then incrementally improves the quality of the solution path if extra time is available. Experiments are conducted by simulating the Space Station remote manipulator system (SSRMS) [70]. The results of these simulations show that the FADPRM algorithm initially requires more time to re-plan but the time gets quickly and significantly reduced.

[71] presents an approach for anytime path planning and replanning in partially-known, dynamic environments. This approach consist of three stages (i) construction of PRM representing the static portion of the planning space. (ii) An initial trajectory is planned over this PRM in state-time space by taking into account any known dynamic obstacles. The trajectory is planned and continuously improved until the time for consideration is exhausted. The planning is done incrementally in a backwards direction from the goal to the start, so that when the agent moves, the stored paths and path costs of all the states in the search space that have already been computed are not affected. Since we do not know in advance at what time the goal will be reached, we start the search with multiple goal states. To improve the efficiency of the search, the approach uses two important heuristic values, the minimum possible time and the minimum possible cost from the current start position to any particular position on the PRM. (iii) When changes are observed, the current trajectory is repaired to reflect these changes in an anytime and a solution can be obtained by using Anytime D*(AD*) algorithm.

[72] gives a method consists of a pre-processing stage that involves three steps. First, a roadmap is constructed using random sampling of the C space. Secondly, we decide the mapping from cells in the robot's workspace to nodes and arcs in the C-space roadmap. The redundancy in this mapping is used to construct a significantly condensed representation. Thirdly, we enhance the C-space roadmap using the minimum cut set of the roadmap and ρ-robustness, a measure that was defined to obtain the robustness of the roadmap to the introduction of obstacles into the robot workspace. For roadmap construction, we have examined the effect of the geometry of the robot on the sampling of its C-space. Construct the graph for an obstacle-free workspace, and code the mapping from workspace cells to nodes and arcs in the graph. This mapping is used to make the suitable modifications to the graph when the environment changes, and plans can be generated by searching the modified graph. For the mapping from the workspace to the roadmap, the approach examined the size of the mapping, the computation time required to generate the mapping, and the improvement in the size of the mapping gained by using a more efficient encoding.

[73] presents a path planner for robots operating in dynamically changing environments with both static and moving obstacles. The planner begins with a pre processing stage that constructs a roadmap of valid paths by considering the static obstacles. To update the roadmap according to the dynamic changes, it uses lazy-evaluation mechanisms along with a single-query technique as local planner. When the solution cannot be found in the updated roadmap, the planner initiates a reinforcement stage that results into the creation of cycles indicating alternative paths that were not already stored in the roadmap. The approach consists of Roadmap labeling and solution path search, query nodes connections, local reconnections by using RRT techniques, nodes insertion & cycles creation and edge labeling.

Simulation results show that this approach gives to efficient global planner capable of solving real-time problems in complex dynamic environments. It has been proved that the probability of failure for a planner to find a solution trajectory, if one exists, converges rapidly to zero as the number of collision-free samples of the workspace increases.

$Pr[(p,q)\,\text{Failure}] \leq \ell e^{-pn}$, $\ell \geq 0$, where $p > 0$ and n is the number of configurations [12]. The issues raised in this method are with the following aspects: sampling in CS (configuration space) and corridors within CS, distance matrix calculation to find closeness of configurations and interpolation issues of CS.

2.5.5. Markov decision processes (MDP) and partially observable Markov decision process (POMDP)

MDP are approaches for sequential decision-making when outcomes are random and uncertain. The MDP model consists of decision period, states, actions, rewards, and evolution probabilities [74]. The process is given briefly below:

(1) At time t, a particular state x of the Markov chain is observed.

(2) After the observation of the state, an action, say u, is taken from a set of possible decisions (different states may have different sets of decisions).

(3) An immediate gain (or loss) $r(x, u)$ is then incurred according to the current state x and the action u taken.

(4) The evolution or transition probability $p\ (x'|u, x)$ is then affected by the action u.

(5) As the time t increases and another transition occurs, all the steps stated above are repeated.

The value iteration is done by the recursive formula

$$V_T = \Upsilon \max_u [r(x,u) + \int V_{T-1}(x')p(x'|u,x)dx'],$$
$$\text{with } V_1 = \Upsilon \max_u r(x,u) \qquad (12)$$

where Υ is a problem-dependent discount factor which takes values in [0 1].

POMDP has been formally given in [75] as follows:

Let $X \subset R^n$ be the space of all likely states x of the robot, $U \subset R^m$ be the space of all likely control inputs u of the robot, and $Z \epsilon R^k$ be the space of all likely sensor measurements z that the robot may receive. General POMDPs accept as input a dynamics and observation model, given here in probabilistic notation:

$$x_{t+1} \sim p[x_{t+1}|x_t, u_t]; \; z_t \sim p[z_t|x_t]$$

where $x_t \subset X$, $u_t \subset U$, and $z_t \subset Z$ are the robot's state, control input and measurement at stage t, respectively.

The belief $b[x_t]$ of the robot is defined as the distribution of the state x_t, given all past control inputs and sensor measurements.

$$b[xt] = p[xt|u0, \ldots, ut-1; z1, \ldots, zt] :$$

Given a control input u_t and a measurement z_{t+1}, the belief is propagated using Bayesian filtering:

$$b[x_{t+1}] = \eta p[z_{t+1}|x_{t+1}] \int p[x_{t+1}|x_t, u_t] b[x_t] dx_t; \qquad (13)$$

where η is a normalizer independent of x_{t+1}. Denoting belief $b[x_t]$ by b_t, and the space of all possible beliefs by $B \subset \{X \to [01]\}$, the belief dynamics defined can be written as a function $\beta: B \times U \times Z \to B$,

$$b_{t+1} = \beta[b_t; u_t; z_{t+1}].$$

The POMDP problem is to find a control policy $\Pi t: B \to U$ for all $0 \le t < l$, where ℓ is the time possibility, such that selecting the controls $u_t = \Pi t\,[b_t]$ minimizes the objective function:

$$\underset{z_1,\ldots,z\ell}{E}\left[c\ell[b\ell] + \sum_{t=0}^{\ell-1} ct[bt, ut]\right]$$

for given immediate cost functions $c\ell$ and c_t. The expectation is taken given the random nature of the measurements.

A common solution approach uses value iteration, a backward recursion procedure, to find the control policy Πt for each stage t:

$$v[b\ell] = c\ell[b\ell]$$

$$vt[b_t] = \min(c_t[b_t, u_t] + {}_{zt+1}{}^{E}[vt+1\,[\beta\,[bt, ut, zt+1]]]) \qquad (14)$$

$$\Pi_t[b_t] = \underset{ut}{\operatorname{argmin}}(c_t[b_t, u_t] + \mathrm{E}_{zt+1}[v_{t+1}[[b_t, u_t, z_{t+1}]]]). \qquad (15)$$

Where $v_t[b_t]: B \to R$ is the value function at time t.

Uncertainty in path planning of the robot systems with non-linear dynamics is estimated on the basis of the sensor inputs, and a locally optimal solution to a continuous partially observable Markov decision process (POMDP) has been computed [75]. Belief, i.e., the distribution of the robot's state estimate, is represented by Gaussian distribution and an extended Kalman filter. As quadratic function, the value function which restricts path's validity to the vicinity of a trajectory through belief space [76].

The linear–quadratic Gaussian motion planning (LQG-MP), an approach based on the linear–quadratic controller with Gaussian models of uncertainty, is discussed [77]. It uses a formulation of the equations of motion and characterizes a priori probability distributions of the state of the robot along its path that could be used to assess the quality of the path.

The main idea of LQG-MP is that the priority knowledge of the sensors and controller that will be used during the execution of the path can be used to optimize the path. The iterative linear–quadratic Gaussian approach develops second-order convergence towards a linear control policy over the belief space that is optimal with respect to a defined cost function. The method assumes neither maximum-likelihood observations, nor fixed estimator or control gains. Given an initial trajectory, the planning algorithm executes in polynomial time $[O(n^7)]$ with the dimension of the state space and converges towards a locally-optimal control policy.

2.5.6. *Reciprocal collision avoidance (RCA)*

The addition of control obstacles such as velocity obstacles and acceleration obstacles to general reciprocal collision avoidance is considered in [78,79] for non-linear, non-homogeneous systems where the robots may have different state spaces and non-linear equations of motion from one another that prohibits us from applying LQG-MP for path planning.

Sampling-based approaches work in a continuous state space, which is used for real world navigation. They are well suited to the dynamic environment in order to handle time-constraints and high dimensional spaces. The main idea of probabilistic approach to robot motion planning is to represent the uncertainties, perceptions, planning and control by using probability distributions over all space of guesses. Probabilistic algorithms are computationally very costly as compared to other approaches since it required to calculate the entire probability densities instead of a single guess.

2.6. *Artificial intelligence based motion planning (AI)*

2.6.1. *Artificial neural networks (ANN) for navigation*

A model-free ANN is discussed in [80,81] that uses back propagation for single time-step prediction of moving obstacles for mobile robot navigation. Obstacles are assumed to follow a rectilinear path with constant velocity. Past sensor readings are the inputs, and the outputs are the predicted readings at the next time instant for that particular sensor. It has been shown that the two most recent sensor readings are enough to predict the next one. This predictor is coupled with a virtual force guidance navigation scheme. The repulsive forces are modeled as obstacles in the APF. The algorithm is later tested on a mobile robot with human obstacles, which has been proved to be more efficient.

ANN is used to solve the navigation problem in [82]. It applies a hierarchical partially observable Markov decision process (POMDP) that in effect models the decision process of the robot, takes a state representation of the environment as input and outputs the optimal control actions. The model is computationally more efficient in large search spaces as it decomposes the problem into multiple linked POMDPs, each with a smaller state space to search.

A mutation-based evolutionary polynomial ANN has been used for short-term prediction of the motion of a robot. The output is integrated into the reward function of the POMDP. A moving obstacle avoidance algorithm has been developed by blending a radial-basis-function neural network (RBFNN) with a regular planning method. The motions of the dynamic obstacles were captured by a camera mounted on the robot and the images were used for training the ANN. A rolling window was obtained to minimize the search space to the sensor's viewing area. If the predicted position of a moving obstacle would obstruct the robot, the obstacle is treated as static at that predicted position and an obstacle avoidance maneuver is generated.

2.6.2. Particle swarm optimization (PSO)

Dynamic obstacle avoidance path planning based on PSO is discussed for soccer robots (i.e., humanoid robots that play football) [83]. An obstacle's avoidance approach with multiple objective optimization by PSO in dynamic environment is given in [84, 85]. Based on the environment map of a mobile robot represented with a series of horizontal and vertical lines, an optimization model of the problem with two indices, the length and the danger degree of a path, is obtained. An improved multi-objective particle swarm optimization algorithm of solving the model is developed. A self-adaptive mutation operation based on the degree of a path blocked by obstacles is considered to improve the feasibility of a new path. A new approach for robot navigation based on PSO and stream functions is addressed in [86]. The stream functions direct the autonomous robot to avoid the obstacles and PSO is applied to generate each optimal step from initial to goal position. A planning algorithm based on PSO of Ferguson splines is developed in [87]. The cubic Ferguson splines are used as model path for robot and path planning means optimization of the parameters of spline. The fitness function is the combinations of the Euclidean distance between actual and desired position of the robot, influence of obstacles and qualities of each particle.

Ferguson spline approach can overcome the local optima in comparison with other optimization methods. In sharp turns, long obstacles, standard situation in robotic soccer the approach will be more effective as compared with Potential field and Visibility Graph methods and drawback in the case of small computational time.

2.6.3. Genetic algorithms (GA)

In [88] GA is used for generating via-points after finding the objects using an imaging system. The fitness value of the path is estimated in terms of the safety from the obstructing dynamic objects and the distance to the goal position. A GA with variable length chromosomes capable of path planning in both static and dynamic environment is presented in [89]. The robot's environment is represented by a discretized grid net, with the property of each cell considered as a gene. GA incorporates the domain knowledge into its specialized operators of mutation, crossover and selection. Local search techniques are employed for obtaining efficiency [90,91].

The advantage of AI based planning is the high speed implementations but it may not guarantee to get a solution.

2.7. Applications in agent systems and computer geometry

Motion planning in dynamic environments is not only studied in robotics but also in areas like Agent systems and Computer Geometry. The global path planning along with local collision avoidance for each virtual agent is the difficult challenges in a large-scale agent-based simulation. Since each character is a dynamic obstacle for other agents, path planning problem become very complex for real-time applications with a large group of moving virtual characters. The methods are limited to static environments to perform local collision avoidance computations which results in abnormal behavior or getting trapped in local minima.

2.7.1. Multi-agent navigation graph (MaNG)

[92] has given a method for accurate path planning and navigation of multiple virtual agents in complex dynamic scenes by constructing a data structure, Multi-agent Navigation Graph (MaNG), by using Voronoi diagrams. The MaNG is used to perform on a real time basis the route planning and proximity computations for each agent. It uses the concepts of the first and second order Voronoi graphs to get a single MaNG which provides global path planning of multiple virtual agents. It gives an algorithm for computing the first and second order Voronoi diagrams of agents and obstacles using GPU. The approach can manage environments with large number of agents and obstacles in real time. The computed path lies along the Voronoi boundary, which is far away from the obstacles and without making any collision.

Given n dynamic agents, the complexity of computing the Voronoi diagram and the MaNG on a discrete grid of resolution $m \times m$ is $O(m^2 + n \log m)$. The MaNG provides paths of better coverage of the agents over the environment and it is efficient.

2.7.2. Heterogeneous crowd simulations using AERO

A heterogeneous crowd is defined as consisting of many different types of groups, independent behavior patterns and goals. For examples, large trade fair halls, festival areas, busy urban street scenes, etc. The "Adaptive Elastic ROadmaps" (AERO), is a connectivity graph structure that is lazily computed using a generalized crowd dynamics model [93]. A reactive roadmap computation algorithm that modifies and updates the links of the roadmap graphs using a particle-based dynamics simulator. The algorithm includes path updating which consists of addition and deletion of links. Dynamics model captures macroscopic mutual interactions among multiple moving groups and microscopic local forces among individual pedestrians or agents. By using the approach, AERO to perform dynamic, global path planning.

2.7.3. Navigation using reciprocal velocity

Reciprocal Velocity Obstacles (RVO) is an extension of the concept of Velocity Obstacle (VO) used for robot navigation. VO was introduced for the navigation among passive moving obstacles. Reciprocal Velocity Obstacle, which is defined mathematically as follows [94]

$$RVO_B^A(v_B, vA) = \{v'A | 2v'A - vA \in VO_B^A(vB)\}.$$

The reciprocal velocity obstacle $RVO_B^A\,(vB, vA)$ of agent B to agent A contains all velocities for agent A that are the average of the current velocity vA and a velocity inside the velocity obstacle $VO_B^A\,(vB)$ of agent B. It has been proved mathematically that RVO is a collision free and oscillation free navigation approach in a multiple and dynamic moving agents in the environment. Reciprocal Velocity Obstacle is a simple and natural method that is commonly applicable and easy to implement.

2.7.4. High-density autonomous crowds (HiDAC) system

The classification of crowd agent motions by using three approaches (i) social forces models (ii) rule based models and (iii) cellular automata models. The High-Density Autonomous Crowds system which is used for simulating the local motion and global way of finding behaviors of crowds moving in a dynamically changing virtual environments [95]. The method allows for diverse crowds where a number of different behaviors can be exhibited simultaneously. A combination of psychological and geometrical rules with a social and physical forces model, HiDAC provides a wide variety of applications such as fast perception of environment, eliminate shaking behavior, natural bidirectional, behavior of queuing and pushing, falling agents becoming new obstacles, panic propagation, crowd impatience, flow and its consequences relative to the current situation, personalities of the individuals etc.

2.7.5. Neural-network-based path planning for a multi robot system

A biologically motivated neural-net work based planner is considered for the coordination of Multi Agent Systems (MASs). A landscape of the neural activities for all neurons of a Coordinated Hybrid Agent (CHA) contains information about the agent's local goal and moving obstacles [96,97]. In the topologically organized neural network, the dynamics of each neuron is characterized by a neural equation. The purpose of building the planner is to plan actions for multiple mobile robots to coordinate with others and accomplish the global goal. The method is able to plan the paths for multiple robots by avoiding moving obstacles and it was simulated using Vortex. In implementation, it executes control commands from the control system module, and provides the outputs giving the vehicle state and terrain information. The results show that the developed intelligent planner of the CHA approach can control complex system so that coordination within agents can be obtained.

2.8. Other methods

2.8.1. Visual motion planning (VMP)

A framework for image-based motion planning called VMP where the computation of motion plans is performed in the image plane using the image details obtained from the visual data is presented in [98]. The method avoid the step of sending image features back to robot pose and makes motion plans directly in the image plane. Vision based sensors and different types of camera are used for images. Given a robotic system with dynamics in the feature space, a set of constraints on the robot motion and features that to be satisfied by robot during the motion, the initial and final configurations, find a sequence of features and inputs that satisfy all the conditions to minimize the cost function and drive the features from the start to goal.

Similar to visual servo control [98], the visual motion planning model takes advantage of the image features to attain a fast motion planning solution. It provides an effective trajectory in the image plane for the robot to follow, with standard visual servo techniques.

2.8.2. Destination driven navigator (DDN)

DDN is a real-time navigating system for a mobile robot working in static and dynamic environments [88]. The technique comprises of the following elements: cross-line obstacle representation method, suggestion of a "work space" to reduce the robot's search space and the environment storage cost, an adaptive regression model to forecast the motion of dynamic obstacles, multi-state path repair rules to quickly translate an infeasible path into a feasible one and finally, the path-planning algorithm to generate a path.

2.8.3. Nearness diagram (ND), global nearness diagram (GND)

The GND makes motion commands to drive a robot between locations with safety and avoiding collisions [90]. Terminology: PND represents the nearness of the obstacles to the central point, and the RND represents the nearness of the obstacles to the robot boundary.

The formal definition of ND is as follows:

Nearness diagram from the central point

$$PND : C_{free} \times A \rightarrow (\mathbb{R}^{+} \cup \{0\})^{n}$$
$$(q, b) \rightarrow (D_1, \ldots, D_n).$$

If $\delta_i(q, b) > 0, D_i = \text{PND}_i(q, b) = d_{\max} + l - \delta_i(q, b)$, otherwise $D_i = \text{PND}_i(q, b) = 0$ where $d_{\max}$ is maximum value of δ, representing the maximum range of the sensor used, I is the maximum distance between two points of the robot (like the diameter for a circular robot).

Nearness Diagram from the Robot

$$\text{RND} : C_{free} \times A \rightarrow (\mathbb{R}^{+} \cup \{0\})^{n}$$
$$(q, b) \rightarrow (D_1, \ldots, D_n).$$

If $\delta_i(q, b) > 0, D_i = \text{RND}_i(q, b) = d_{\max} + E_i(b) - \delta_i(q, b)$, otherwise $D_i = \text{RND}_i(q, b) = 0$, where E is an enlarging function that depends on the robot geometry. The value of this function in each sector $E_i(b)$ is the robot radius for a circular robot.

Motion commands are based on five laws depend upon different situations (i) Navigation in low safety situation in which the obstacles are closer to the security distance on one side of the gap and closer to the goal. Main objective is to push the robot away from the closest obstacle while moving towards the goal. (ii) Navigation in low safety situation in which the obstacles are closer to the security distance on both sides of the gap. The objective is to maintain the robot at the same distance from the two closer obstacles. (iii) Navigation in High Safety Goal Valley in which the goal location is inside the free walking area and navigation directly takes to the goal. (iv) Navigation in High Safety Wide Valley in which the goal sector is not inside the free walking area but it is wide. The navigation gives as a result of a motion along the side of obstacle. (v) Navigation in High Safety Narrow Valley in which the goal sector is not inside the free walking area and it is narrow. This navigation rule gives direction to the robot among the obstacles.

GND has all the advantages of using the scheme of Nearness Diagram (ND) [90], while having the capability to reason and plan globally and reaching global convergence to the navigation problem.

The real advantage of this method is when dealing with dense and complex environments where other methods such as APF and DWA are avoided. This is the best performing method in all respects as it is based on a geometric approach. Computational run time is relatively larger as the size of occupancy grid grows large. The method is not effective in the trap situations like robot reaching to a border, and for Non-holonomic and robots with large shapes.

2.8.4. Grid based wave front propagation (GBWFP)

A reactive obstacle avoidance technique has been proposed that guarantees safe operation, smooth path planning and adaptability with respect to the environment information about the motion of persons and objects [99]. In GBWFP, dynamic path representation and high-fidelity execution are combined.

2.8.5. Focused D^* and D^* Lite algorithms

[100] presents the Focused D^* algorithm for real-time path replanning. The algorithm computes an initial path from the goal to the start and then modifies this path during the traverse as cost of the arc changes. The algorithm gives an optimal traverse by taking all known information at each step. The focused version of D^* performs better than the basic D^* in total time, and gives the choice of distributing the computational load amongst the on- and off-line portions of the operation. The addition of a heuristic focusing function to D^* to the development of a dynamic environments—A^* is the special case of D^* and the arc costs do not change during the traverse of the solution path.

Presented D^* Lite, a fast replanning method for goal-directed navigation in unknown terrain [96,101]. The approach implements the same navigation strategy as (focused) D^*. Both algorithms search from the goal vertex towards the current vertex of the robot, use heuristics to focus the search, and use the same ways to minimize the reordering of the priority queue. D^* Lite builds on Lifelong Planning A^* (LPA*), an incremental version of A^* which is efficient since it does not expand any vertices whose g-values, an estimate of start distance to any vertex were already equal to their respective goal distances. It is shorter, simpler, and hence easier to understand.

3. Conclusions

The scope of the review presented in this paper is restricted to a set of 101 research papers in the domain of RMPDE that were published during 1985 to 2016. From the number of papers published we can infer that compared to the classical approaches, APF, VB, AI and PMP are relatively more active areas of research. Apparently, despite major advancements in RMP during the past three decades, not much work has been carried out in multi robot systems. There is a wide scope for research in coordinated, networked multi robot systems.

Because of its infinite look-ahead-time feature, ICS-AVOID outperforms TVDW and NLVO. The performance of the latter two is restricted by the truncated future model that they follow and therefore disregard the information beyond breaking time and heading time. (Recall that breaking time and heading time denote the time taken by a robot to stop before hitting the obstacle in TVDW NLVO respectively.) ICS-AVOID is the only scheme that reasons over an infinite-time horizon and therefore provides the safest motion to robots in dynamic environments with respect to velocity-based planning. Therefore there is scope for further research with this method for safe motion planning.

References

[1] J.-C. Latombe, Robot Motion Planning, Kluwer Academic Publishers, Boston, MA, 1991.
[2] Tomas Lozano-Perez, Spatial planning: A configuration space approach, IEEE Trans. Comput. c-32 (2) (1983).
[3] Paolo Fiorini, Robot Motion Planning Among Moving Obstacles, (Ph.D. thesis), University of California, Los Angeles, 1995.
[4] J.F. Canny, The Complexity of Robot Motion Planning, MIT Press, Cambridge, MA, 1988.
[5] S.M. LaValle, Planning Algorithms, Cambridge University Press, Cambridge, UK, 2006 Available at http://planning.cs.uiuc.edu/.
[6] Thierry Fraichard, Hajime Asama, Inevitable collision states a step towards safer robots, Advanced Robotics (2004).
[7] Zvi Shiller, Federic Large, Sepanta Sekhavat, Christian Laugier, Motion planning in dynamic environments: Obstacles moving along arbitrary trajectories, in: IEEE International Conference on Robotics and Automation, ICRA'2001May 21-26, 2001 Seoul, Korea.
[8] Zvi Shiller, Oren Gal, Thierry Fraichard, The Nonlinear Velocity Obstacle Revisited: the Optimal Time Horizon, Guaranteeing Safe Navigation in Dynamic Environments Workshop, May 2010, Anchorage, United States.
[9] Stephane Petti, Thierry Fraichard, Safe motion planning in dynamic environments, in: Proc. Of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems, Aug 2005, Edmonton, AB (CA), France. 2005. <inria-00182046>.
[10] Jur Pieter van den Berg, Path Planning in Dynamic Environments, (Ph.D. thesis), The Netherlands, 2007.
[11] Kikuo Fujimura, Motion Planning in Dynamic Environments, Springer-Verlag Berlin and Heidelberg GmbH &Co.k, 1991 ISBN10:3540700838.
[12] H. Choset, K.M. Lynch, S. Hutchinson, G.A. Kantor, W. Burgard, L.E. Kavraki, S. Thrun, Principles of Robot Motion: Theory, Algorithms, and Implementations, MIT Press, 2005 ISBN 0-262-03327-5.
[13] N.Y. Ko, B.H. Lee, Avoidability measure in moving obstacle avoidance problem and its use for robot motion planning, in: Proceedings of the IEEE International Conference on Intelligent Robots and Systems, Osaka, Japan, 1996, November, pp. 1296-1303.
[14] S.S. Ge, Y.J. Cui, Dynamic motion planning for mobile robots using potential field method, Auton. Robots 13 (3) (2002) 207–222.
[15] Q. Cao, Y. Huang, J. Zhou, An evolutionary artificial potential field algorithm for dynamic path planning of mobile robot, in: Proceedings of the IEEEInternational Conference on Intelligent Robots and Systems, Beijing, China, 2006, October pp. 3331–3336.
[16] A. Poty, P. Melchior, A. Oustaloup, Dynamic path planning by fractional potential, in: Proceedings of the IEEE International Conference on Computational Cybernetics, Vienna, Austria, 2004, August, pp. 365–371.
[17] Kikuo Fujimura, Hanan Samet, Planning a time- Minimal Motion Among Moving Obstacles, Algorithmica, Springer Verlag, New York, 1993.
[18] Th. Fraichard, Trajectory planning in a dynamic workspace: A 'State-Time Space' Approach (Fraichard 1999 [18]), Adv. Robot. 13 (1) (1999) 75–94.
[19] Paolo Fiorini, Zvi Shiller, Robot motion planning in dynamic environments, in: International Symposium of Robotic Research, pp. 237–248, Oct-1995.
[20] Paolo Fiorini, Zvi Shiller, Time Optimal Trajectory Planning in Dynamic Environments, in: IEEE International Conference of Robotics and Automation, pages 1553–1558, April-1996.
[21] Paolo Fiorini, Zvi Shiller, Robot motion planning in dynamic environments using velocity obstacles, Int. J. Robot. Res. (1998).
[22] Dieter Fox, Wolfram Burgard, Sebastian Thrun, The dynamic window approach to collision avoidance, IEEE Robot. Autom. Mag. (1997).
[23] Eduardo Owen, Luis Montano, Motion Planning in Dynamic Environments using the velocity space, in: Intelligent Robots and Systems, 2005, IROS 2005, 2005 IEEE/RSJ International Conference on Aug-2005.
[24] Luis Martinez-Gomez, Thierry Fraichard, Collision avoidance in dynamic environments: an ICS-based solution and its comparative evaluation, an ICS-based solution and its comparative evaluation, in: IEEE Int. Conf. on Robotics and Automation, May 2009, Kobe, Japan. 2009 <inria-00361324>.
[25] Marija Seder, Ivan Petrovic, Dynamic window based approach to mobile robot motion control in the presence of moving obstacles, Robotics and Automation, in: 2007 IEEE International Conference on April 2007.
[26] Oliver Brock, Oussama Khatib, High- Speed Navigation using the global dynamic Window approach.
[27] Zvi Shiller, Oren Gal, Thierry Fraichard, The Nonlinear Velocity Obstacle Revisited: the Optimal Time Horizon.
[28] Luis Martinez-Gomez, Thierry Fraichard, Benchmarking collision avoidance schemes for dynamic environment, in: IEEE International Conference on Robotics And Automation, Kobe Japan May 2009.
[29] Oren Gal, Zvi Sriller, Mapping obstacles to collision states for on –line motion planning in dynamic environments, in: IEEE International Conference on Robotics And Automation, Kobe Japan May 2009.
[30] Oliver Brock, Oussama Khatib, High-speed navigation using the global dynamic window approach, in: IEEE International Conference on Robotics and Automation, 1999.
[31] Ellips Masehian, Yalda Katebi, Sensor-based motion planning of wheeled mobile robots in unknown dynamic environments, J. Intell. Robot Sst. 74 (2014) 893–914. http://dx.doi.org/10.1007/s10846-013-9837-3.
[32] F. Large, C. Laugier, Z. Shiller, Navigation among moving obstacles using the NLVO: Principles and applications to intelligent vehicles, Autonom. Robots 19 (2005) 159–171.
[33] Oren Gal, Zvi Shiller, Mapping Obstacles to Collision States for On-line Motion Planning in Dynamic Environments IEEE, ICRA 2009.
[34] Javier Hernández-Aceituno, A. Leopoldo, Environments, application of time dependent probabilistic collision state checkers in highly dynamic environments, PLoS ONE 10 (3) (2015) e0119930. http://dx.doi.org/10.1371/journal.pone.0119930. IEEE 2015 International Conference on Robotics and Automation.
[35] Oren Gal, Zvi Shiller, Elon Rimon, Efficient and safe on-line motion planning in dynamic environments, robotics and automation, 2009. ICRA '09. IEEE International Conference on 12–17 May 2009.
[36] Ellips Masehian, Robot motion planning in dynamic environments with moving obstacles and target, world academy of science, engineering and technology, Int. J. Comput. Electr. Autom. Control Inform. Eng. 1 (5) (2007).
[37] Ellips Masehian, Davound Sedighizadeh, Classic and heuristic approaches in robot motion planning - A chronological review, world academy of science, engineering and technology, Int. J. Mech. Aerosp. Indust. Mech. Manuf. Eng. 1 (5) (2007).
[38] Mihail Pivtoraiko, Alonzo Kelly, Fast and Feasible Deliberative Motion Planner forDynamic Environments, IEEE, ICRA, 2009.
[39] Chonhyon Park, Jia Pan, Dinesh Manocha, Real-time Optimization-based Planning in Dynamic Environments using GPUs.
[40] J.J. Rebollo, I. Maza, A. Ollero, A two step velocity planning method forreal-time collision avoidance of multiple aerial robots in dynamic environments, in: Proceedings of the 17th World Congress The International Federation of Automatic Control Seoul, Korea, July 6–11, 2008.
[41] Rayomand Vatcha, Jing Xiao, Detecting guaranteed collision-free robot trajectories in unknown and unpredictable environments, in: ICRA 2010 Workshop on Guaranteeing Safe Navigation in Dynamic Environments.
[42] Eiichi Yoshida, Member, IEEE, Claudia Esteves, Igor Belousov, Jean-Paul Laumond, Fellow, IEEE, Takeshi Sakaguchi, and Kazuhito Yokoi, Member, IEEE, Planning 3-D collision -free dynamic robotic motion planning through iterative reshapping, IEEE Trans. Robot., 24(5) (2008).
[43] Mike Phillips, Maxim Likhachev, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, SIPP: Safe Interval Path Planning for Dynamic Environments, ICRA 2011.
[44] F. Kunwar, F. Wong, R. Ben Mrad, B. Benhabib, Guidance-based on-line robot motion planning for the interception of mobile targets in dynamic environments, J. Intell. Robot. Syst. (2006). http://dx.doi.org/10.1007/s10846-006-9080-2.
[45] John Vannoy, Jing Xiao, Real-time adaptive motion planning(RAMP) of mobile manipuiators in dynamic environments with unforseen changes, IEEE Trans. Robot. 24 (5) (2008).
[46] Eric L. Thorn Jr., Autonomous Motion Planning Using a Predictive Temporal Method, (Ph.D. thesis), 2009.
[47] Noel E. Du Toit, Joel W. Burdick, IEEE robot motion planning in dynamic, Uncertain Environ. IEEE Transs. Robotics 28 (1) (2012).
[48] Noel E. Du Toit, Joel W. Burdick, Probailistic collision checking with chance constraints, IEEE Trans. Robot. 27 (4) (2011).

[49] NoelE. Du Toit, W. Joel Burdick, Robotic motion planning in dynamic, cluttered, uncertain environments, in: 2010 IEEE International Conference on Robotics and Automation, USA.
[50] Noel E. Du Toit, Robot Motion Planning in Dynamic, Cluttered, and Uncertain Environments: the Partially Closed-Loop Receding Horizon Control Approach, (Ph.D. thesis), California Institute of Technology, California, 2009.
[51] Sabastian Thrun, Wolfram Burgard, Dieter Fox, Probabilistic Robotics, MIT Press, Cambridge, London, 2005.
[52] Sebastin Thrun, Probabilistic Algorithms in Robotics Sebastin Thrun, CMU-CS-2000-126.
[53] Konstantinos I. Tsianos, Ioan A. Sųcan, Lydia E. Kavraki, Sampling-based robot motion planning: Towards realistic application, Comput. Sci. Rev. 1 (1) (2007) 2–11.
[54] Jur P. van den Berg, Mark H. Overmars, Roadmap- based motion planning in dynamic environments, IEEE Trans. Robot. 21 (5) (2005).
[55] Cyrill Stachniss, Wolfram BurgardUniversity of Freiburg, Department of Computer Science, D-79110 Germany, An Integrated Approach to Goal-directed Obstacle Avoidance under Dynamic Constraints for Dynamic Environments.
[56] Dieter Fox, Wolfram Burgard, Sebastin Thrun, Markov localization for mobile robots in dynamic environment, J. Artificial Intelligence Res. 11 (1999) 391–427.
[57] Dizan Alejandro Vasquez Govea, Thierry Fraichard, Christian Laugier, Incremental Learning of Statistical Motion Patterns with Growing Hidden Markov Models, Transactions on intelligent transportation systems. IEEE 2009.
[58] Daniel Althoff, Matthias Althoff, Dirk Wollherr, Martiin Buss, Probabilistic Collision State checker for Crowded Environment, Robotics and Automation(ICRA) May 2010, IEEE International Conference.
[59] Daniel. Althoff, Safety Assessment for Motion Planning in Uncertain and Dynamic Environments, (Ph.D. thesis), 2013.
[60] Chiara Fulgenzi, Anne Spalanzani, Christian Laugier, Dynamic obstacle avoidance in uncertain environment combining PVOs and occupancy grid, in: Proceedings of the IEEE Int. Conf. on Robotics and Automation, Apr 2007, Rome, France. 2007. <inria-00181443>.
[61] Federic Large, Sepantha Shekavat, Z. Shiller, C. Laugier, Using Non Linear Velocity Obstacles to plan Motion in a Dynamic Environment.
[62] Antoine Bautin, Luis Martinez-Gomez, Thierry Fraichard, Inevitablec ollision states: A probabilistic perspective, in: IEEE International Conference on Robotics and Automation - ICRA 2010, May 2010, Anchorage, AK, United States.
[63] Svestka & Overmars, Motion Planning for Carlike robots using a Probabilistic learning approach (Technical report UU-CS-1994-33), Utrech.
[64] Svestka & Overmars, Coordinated motion planning for multiple car-like robots using Probabilistic roadmaps, in: Proceeding of the IEEE Int. Conference on Robotic & Automation, Nagoya, Japan, 1995.
[65] Kindel Hsu, Latombe & Rock, Kinodynamic motion planning amidst moving obstacles, in: Proceeding of the IEEE Int. Conference on Robotic & Automation, Sanfransisco, CA, 2002, pp 537–543.
[66] Safaa H. Shwail, Karim IraqAlia, Turner Scott, Probabilistic multi robot path planning in dynamic environments: A comparison between A* and DFS, Int. J. Comput. Appl. 82 (7) (2013) 0975–8887.
[67] S.M. LaValle, Rapidly- Exploring Random Trees: A New Tool for Path Planning, IEEE, ICRA, 1998.
[68] Chiara Fulgenzi, Anne Spalanzani, Christian Laugier, Probabilistic Rapidly-exploring Random Trees for Autonomous Navigation among moving pedestrians, IEEE, ICRA, 2009.
[69] Jaillet Simeon, A PRM- based Motion planner for dynamically changing environments, in: Proceeding of the IEEE Int. Conference on Int, Robots & Systems, Sendai, Japan, 2004, pp. 1606–1611.
[70] Belghith, Kabanza, Hartman, Nkanbou, Anytime dynamic path planning with flexible probabilistic roadmaps, in: Proceeding of the IEEE Int. Conference on Robotic & Automation, Orlando, 2006, pp. 2372–2377.
[71] J. Van Den Berg, D. Ferguson, J. Kuffner, Anytime path planning and replanning in dynamic environments, in: Robotics and Automation, ICRA 2006 Proceedings 2006 IEEE International Conference on, IEEE, 2006, pp. 2366–2371.
[72] P. Leven, S. Hutchinson, A framework for real-time path planning in changing environments, Int. J. Robot. Res. 21 (12) (2002) 999–1030.
[73] L. Jailet, T. Simeon, A PRM-based motion planner for dynamically changing environments, in: Proceeding of the IEEE International Conference on Intelligent Robots and Systems, IROS, 2004.
[74] Yoshihiko Nakamura, Incremental learning, clustering and hierarchy formation of whole body motion patterns using adaptive Hidden Markov Chains, Int. J. Robot. Res. 27 (7) (2008) 761–784.
[75] Jur Van Den Berg, Sachin Patil, Ron Alterovitz, Motion planning under uncertainty using iterative local optimisation in belief space, Int. J. Robot. Res. 31 (11) (2012) 1263–1278.
[76] Mohamad Ali Movafaghpour, Ellips Masehian, Optimal probabilistic robot planning with missing information, intelligent robots and systems, IROS, in: 2011 IEEE/RSJ International Conference on 25–30 Sep 2011.
[77] Jur van den Berg, Pieter Abbeel, Ken Goldberg, LQG-MP: Optimized path planning forrobots with motion uncertainty and imperfect state information, 2011.
[78] Daman Bareiss, Jur van den Berg, Generalized reciprocal collision avoidance, Int. J. Robot. Res. (2015) 1–14.
[79] O. Khatib, Real –time obstacle avoidance for manipulators and mobile robots, Int J. Robot. Res. 5 (1986) 90–98.
[80] C.C. Chang, K.-T. Song, Environment prediction for a mobile robot in a dynamic environment, IEEE Trans. Robot. Autom. 13 (6) (1997) 862–872.
[81] C.C. Chang, K.-T. Song, Dynamic motion planning based on real-time obstacle prediction, in: Proceedings of the IEEE International Conference on Robotics and Automatio Minneapolis, MN, 1996, May, pp. 2402–2407.
[82] Y. Li, C. Li, R. Song, A new hybrid algorithm of dynamic obstacle avoidance based on dynamic rolling planning and RBFNN, in: Proceedings of the IEEE International Conference on Robotics and Biomimetics, Bangkok, Thailand, 2008, December, pp. 2064-2068.
[83] L. Wang, Y. Liu, H. Deng, Obstacle-avoidance path planning for soccer robots using particle swarm optimization, in: Proc. IEEE Int. Conf. on Robotic and Biometrics, 2006, pp. 1233–1238.
[84] M. Hua-Quing, Z. Jin-Hui, Z. Xi-Jing, Obstacle avoidance with multi-objective optimization by PSO in dynamic environment, in: Proc. IEEE Int. Conf. in Machine Learning and Cybernetics, 2005, pp. 2950–2956.
[85] N. Lv, Z.F. eng, Numerical Potential Field and Ant Colony Optimisation Based path planning in Dynamic Environment, in: Proc. IEEE 6th world congress on Intelligent control and Automation, 2006, pp. 8966–8970.
[86] C. Hu, X. Wu, Q. Liang, Y. Wang, Autonomous Robot Path Planning Based on Swarm Intelligence and Stream Functions, in: Lecture Notes in Computer Science, vol. 4684, Springer, 2007, pp. 277–284.
[87] M. Saka, M. Macas, L. Preucil, L. Lhotska, Robot Path Planning using Particle Swarm Optimization of Ferguson.
[88] Huiming Yu, Tong Su, Destination driven motion planning via obstacle motion prediction and multi-state path repair, J. Intell. Robot. Syst. 36 (2003) 149–173 ©2003 Kluwer Academic Publishers. Printed in the Netherlands.
[89] Eric Demeester, Marnix Nuttin, Dirk Vanhooydonck, Gerolf Vanacker, Hendrik Van Brussel, Global dynamic window approach for arbitrarily shaped holonomic and non-holonomic mobile robots, in: IEEE/RSJ Int. Conference on Intelligent Robots and Systems (IROS 2005), August 2–6, Edmonton, Canada, pp. 2357 –2362.
[90] J. Minguez, L. Montano, T. Simeony, R. Alamiy, Global Nearness Diagram Navigation (GND) IEEE, International Conference on Robotic & Autamation, Seoul, May 2001.
[91] Soheil Keshmiri, Shahram Payandeh, Experimental Robotics Laboratory, School of Engineering Science, Simon Fraser University, in: Proceeding of the 14th IASTED International Conference Robotics and Applications(RA 2009), Nov 2-4, 2009, USA.
[92] A. Sud, E. Andersen, S. Curtis, M.C. Lin, D. Manocha, Real-time path planning in dynamic virtual environments using multiagent navigation graphs, IEEE Trans. Vis. Commput. Graph. 14 (3) (2008) 526–538.
[93] A. Sud, E. Andersen, S. Curtis, M. Lin, D. Manocha, Real-time path planning for virtual agents in dynamic environments, in: ACM SIGGRAPH 2008 classes, ACM, 2008, p. 55.
[94] J. Van den Berg, M. Lin, D. Manocha, Reciprocal velocity obstacles for real-time multi-agent navigation, in: Robotics and Automation, 2008 ICRA 2008 IEEE International Conference on, IEEE, 2008, pp. 1928–1935.
[95] N. Pelechano, J.M. Allbeck, N.I. Badler, Controlling individual agents in high-density crowd simulation, in: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation, Eurographics Association, 2007, pp. 99–108.
[96] S. Koenig, M. Likhachev, Improved fast replanning for robot navigation in unknown terrain, in: Proceedings of the IEEE International Conference on Robotics and Automation, ICRA, 2002.
[97] H. Li, S.X. Yang, M.L. Seto, Neural-network-based path planning for a multi-robot system with moving obstacles, IEEE Trans. Syst. Man Cybernet. Part C 39 (4) (2009) 410–419.
[98] H. Zhang, Robot. Autom., IEEE Trans. 18 (2) (2002) Dept. of Mech. Eng., Rowan Univ., Glassboro, NJ, USA; Ostrowski, JP Visual Motion Planning for Mobile Robots.
[99] Roland Philippsen, Motion Planning and Obstacle Avoidance for MobileRobots in Highly Cluttered Dynamic Environments, (Ph.D. thesis), 2004.
[100] A. Stentz, The Focussed D* Algorithm for Real-Time Replanning, in: Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI, 1995.
[101] Kindel Hsu, Latombe Rock, D*lite, in: Proceeding of the Int. Conference on Artificial Intelligence, Edmonton, 2002.